LITERATURE

To order Intel Literature or obtain literature pricing information in the U.S. and Canada call or write Intel Literature Sales. In Europe and other international locations, please contact your local sales office or distributor.

INTEL LITERATURE SALES
P.O. BOX 58130
SANTA CLARA, CA 95052-8130

In the U.S. and Canada
call toll free
(800) 548-4725

CURRENT HANDBOOKS

Product line handbooks contain data sheets, application notes, article reprints and other design information.

<table>
<thead>
<tr>
<th>TITLE</th>
<th>LITERATURE ORDER NUMBER</th>
</tr>
</thead>
<tbody>
<tr>
<td>COMPLETE SET OF HANDBOOKS</td>
<td>231003</td>
</tr>
<tr>
<td>(Available in U.S. and Canada only)</td>
<td></td>
</tr>
<tr>
<td>AUTOMOTIVE PRODUCTS HANDBOOK</td>
<td>231792</td>
</tr>
<tr>
<td>(Not included in handbook set)</td>
<td></td>
</tr>
<tr>
<td>COMPONENTS QUALITY / RELIABILITY HANDBOOK</td>
<td>210997</td>
</tr>
<tr>
<td>EMBEDDED CONTROL APPLICATIONS HANDBOOK</td>
<td>270648</td>
</tr>
<tr>
<td>8-BIT EMBEDDED CONTROLLER HANDBOOK</td>
<td>270645</td>
</tr>
<tr>
<td>16-BIT EMBEDDED CONTROLLER HANDBOOK</td>
<td>270646</td>
</tr>
<tr>
<td>32-BIT EMBEDDED CONTROLLER HANDBOOK</td>
<td>270647</td>
</tr>
<tr>
<td>MEMORY COMPONENTS HANDBOOK</td>
<td>210830</td>
</tr>
<tr>
<td>MICROCOMMUNICATIONS HANDBOOK</td>
<td>231658</td>
</tr>
<tr>
<td>MICROCOMPUTER PROGRAMMABLE LOGIC HANDBOOK</td>
<td>296083</td>
</tr>
<tr>
<td>MICROPROCESSOR AND PERIPHERAL HANDBOOK (2 volume set)</td>
<td>230843</td>
</tr>
<tr>
<td>MILITARY PRODUCTS HANDBOOK</td>
<td>210461</td>
</tr>
<tr>
<td>(2 volume set. Not included in handbook set)</td>
<td></td>
</tr>
<tr>
<td>OEM BOARDS AND SYSTEMS HANDBOOK</td>
<td>280407</td>
</tr>
<tr>
<td>PRODUCT GUIDE</td>
<td>210846</td>
</tr>
<tr>
<td>(Overview of Intel's complete product lines)</td>
<td></td>
</tr>
<tr>
<td>SYSTEMS QUALITY / RELIABILITY HANDBOOK</td>
<td>231762</td>
</tr>
<tr>
<td>INTEL PACKAGING OUTLINES AND DIMENSIONS (Packaging types, number of leads, etc.)</td>
<td>231369</td>
</tr>
<tr>
<td>LITERATURE PRICE LIST (U.S. and Canada) (Comprehensive list of current Intel Literature)</td>
<td>210620</td>
</tr>
<tr>
<td>INTERNATIONAL LITERATURE GUIDE</td>
<td>E00029</td>
</tr>
</tbody>
</table>

About Our Cover:

Intel is committed to providing microcommunications solutions for local and wide area networking at all levels of integration. Our technology gives simple wires the astounding ability for elaborate communications. Today, complex thoughts and ideas are easily exchanged, many of which were made possible by Intel's advanced solutions.
# U.S. and CANADA LITERATURE ORDER FORM

**NAME:** ______________________________________________________

**COMPANY:** ____________________________________________________

**ADDRESS:** _____________________________________________________

**CITY:** _____________________________ **STATE:** _______ **ZIP:** ______

**COUNTRY:** _____________________________________________________

**PHONE NO.:** (____) ___________

<table>
<thead>
<tr>
<th>ORDER NO.</th>
<th>TITLE</th>
<th>QTY.</th>
<th>PRICE</th>
<th>TOTAL</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Subtotal** __________

Postage: add 10% of subtotal

**Postage Tax** __________

**Total** __________

Pay by check, money order, or include company purchase order with this form ($100 minimum). We also accept VISA, MasterCard or American Express. Make payment to Intel Literature Sales. Allow 2-4 weeks for delivery.

- [ ] VISA  - [ ] MasterCard  - [ ] American Express  Expiration Date __________________________

**Account No.** __________________________

**Signature** __________________________

**Mail To:** Intel Literature Sales  
P.O. Box 58130  
Santa Clara, CA 95052-8130

**International Customers** outside the U.S. and Canada should use the International order form or contact their local Sales Office or Distributor.

**For phone orders in the U.S. and Canada**  
**Call Toll Free:** (800) 548-4725

Prices good until 12/31/89.

Source HB
INTERNATIONAL LITERATURE ORDER FORM

NAME: ________________________________________________________

COMPANY: ______________________________________________

ADDRESS: ______________________________________________________

CITY: __________________________ STATE: _______ ZIP: __________

COUNTRY: _________________________________________________

PHONE NO.: (______) __________________________

<table>
<thead>
<tr>
<th>ORDER NO.</th>
<th>TITLE</th>
<th>QTY.</th>
<th>PRICE</th>
<th>TOTAL</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Subtotal __________

Must Add Your Local Sales Tax __________

Total __________

PAYMENT

Cheques should be made payable to your local Intel Sales Office (see inside back cover.)

Other forms of payment may be available in your country. Please contact the Literature Coordinator at your local Intel Sales Office for details.

The completed form should be marked to the attention of the LITERATURE COORDINATOR and returned to your local Intel Sales Office.
Intel the Microcomputer Company:

When Intel invented the microprocessor in 1971, it created the era of microcomputers. Whether used as microcontrollers in automobiles or microwave ovens, or as personal computers or supercomputers, Intel's microcomputers have always offered leading-edge technology. In the second half of the 1980s, Intel architectures have held at least a 75% market share of microprocessors at 16 bits and above. Intel continues to strive for the highest standards in memory, microcomputer components, modules, and systems to give its customers the best possible competitive advantages.

MICROCOMMUNICATIONS HANDBOOK

1989
Intel Corporation makes no warranty for the use of its products and assumes no responsibility for any errors which may appear in this document nor does it make a commitment to update the information contained herein.

Intel retains the right to make changes to these specifications at any time, without notice.

Contact your local sales office to obtain the latest specifications before placing your order.

The following are trademarks of Intel Corporation and may only be used to identify Intel Products:

Above, BITBUS, COMMputer, CREDIT, Data Pipeline, ETOX, FASTPATH, Genius, i, i, ICE, iCEL, iCS, iDBP, iDIS, i2ICE, iLBX, iM, iMDDX, iMMX, Inboard, Insite, Intel, intel, Intel376, Intel386, Intel486, intelBOS, Intel Certified, Intelevision, intelligent Identifier, intelligent Programming, Intelliec, Intelliink, iOSP, iPDS, iPSC, iRMK, iRMX, iSBC, iSBX, iSDM, iSXM, KEPROM, Library Manager, MAPNET, MCS, Megachassis, MICROMAINFRAME, MULTIBUS, MULTICHLANNEL, MULTIMODULE, ONCE, OpenNET, OTP, PC BUBBLE, Plug-A-Bubble, PROMPT, Promware, QUEST, QueX, Quick-Erase, Quick-Pulse Programming, Ripplemode, RMX/80, RUPI, Seamless, SLD, SugarCube, UPI, and VLSICEL, and the combination of ICE, iCS, iRMX, iSBC, iSBX, iSXM, MCS, or UPI and a numerical suffix, 4-SITE, 376, 386, 486.

MDS is an ordering code only and is not used as a product name or trademark. MDS® is a registered trademark of Mohawk Data Sciences Corporation.

*MULTIBUS is a patented Intel bus.

Additional copies of this manual or other Intel literature may be obtained from:

Intel Corporation
Literature Sales
P.O. Box 58130
Santa Clara, CA 95052-8130

© INTEL CORPORATION 1988
CUSTOMER SUPPORT

INTEL'S COMPLETE SUPPORT SOLUTION WORLDWIDE

Customer Support is Intel's complete support service that provides Intel customers with hardware support, software support, customer training, consulting services and network management services. For detailed information contact your local sales offices.

After a customer purchases any system hardware or software product, service and support become major factors in determining whether that product will continue to meet a customer's expectations. Support requires an international support organization and a breadth of programs to meet a variety of customer needs. As you might expect, Intel's customer support is quite extensive. It can start with assistance during your development effort to network management. 100 Intel sales and service offices are located worldwide — in the U.S., Canada, Europe and the Far East. So wherever you're using Intel technology, our professional staff is within close reach.

HARDWARE SUPPORT SERVICES

Intel's hardware maintenance service, starting with complete on-site installation will boost your productivity from the start and keep you running at maximum efficiency. Support for system or board level products can be tailored to match your needs, from complete on-site repair and maintenance support economical carry-in or mail-in factory service.

Intel can provide support service for not only Intel systems and emulators, but also support for equipment in your development lab or provide service on your product to your end-user/customer.

SOFTWARE SUPPORT SERVICES

Software products are supported by our Technical Information Phone Service (TIPS) that has a special toll free number to provide you with direct, ready information on known, documented problems and deficiencies, as well as work-arounds, patches and other solutions.

Intel's software support consists of two levels of contracts. Standard support includes TIPS (Technical Information Phone Service), updates and subscription service (product-specific troubleshooting guides and; COMMENTS Magazine). Basic support consists of updates and the subscription service. Contracts are sold in environments which represent product groupings (e.g., iRMX® environment).

CONSULTING SERVICES

 Intel provides field system engineering consulting services for any phase of your development or application effort. You can use our system engineers in a variety of ways ranging from assistance in using a new product, developing an application, personalizing training and customizing an Intel product to providing technical and management consulting. Systems Engineers are well versed in technical areas such as microcommunications, real-time applications, embedded microcontrollers, and network services. You know your application needs; we know our products. Working together we can help you get a successful product to market in the least possible time.

CUSTOMER TRAINING

 Intel offers a wide range of instructional programs covering various aspects of system design and implementation. In just three to ten days a limited number of individuals learn more in a single workshop than in weeks of self-study. For optimum convenience, workshops are scheduled regularly at Training Centers worldwide or we can take our workshops to you for on-site instruction. Covering a wide variety of topics, Intel's major course categories include: architecture and assembly language, programming and operating systems, BITBUS™ and LAN applications.

NETWORK MANAGEMENT SERVICES

Today's networking products are powerful and extremely flexible. The return they can provide on your investment via increased productivity and reduced costs can be very substantial.

Intel offers complete network support, from definition of your network's physical and functional design, to implementation, installation and maintenance. Whether installing your first network or adding to an existing one, Intel's Networking Specialists can optimize network performance for you.
Table of Contents

Alphanumeric Index ........................................................................................................ xii
Microcommunications Overview AP-302 ................................................................. xi

SECTION ONE—DATA COMMUNICATIONS COMPONENTS

CHAPTER 1
Local Area Networks
CSMA/CD Access Method

DATA SHEETS
- 82586 Local Area Network Coprocessor .................................................. 1-1
- 82C501 Ethernet Serial Interface ................................................................. 1-38
- Twisted Pair Ethernet Chip Set ................................................................. 1-54
- 82560 Host Interface and Memory Controller ................................................. 1-56
- 82590 Advanced CSMA/CD LAN Controller with 8-Bit Data Path ..................... 1-86
- 82592 Advanced CSMA/CD LAN Controller with 16-Bit Data Path ................... 1-103
- 82588 High Integration LAN Controller .................................................... 1-120

APPLICATION NOTES
- An 82586 Data Link Driver AP-235 .......................................................... 1-176
- Implementing Ethernet/Cheapernet with the Intel 82586 AP-274 ....................... 1-256
- Implementing StarLAN with the Intel 82588 AP-236 .................................... 1-345
- Using the Intel 82592 to Integrate a Low-Cost Ethernet Solution into a PC Motherboard AP-320 ................................................................. 1-419

CSMA/CD ACCESS METHOD EVALUATION TOOLS
- PC586E CSMA/CD LAN Evaluation Board .................................................. 1-464

CHAPTER 2
Wide Area Networks

DATA SHEETS
- 8251A Programmable Communication Interface .......................................... 2-1
- 82050 Asynchronous Communications Controller ........................................ 2-26
- 82510 Asynchronous Serial Controller ........................................................ 2-40
- 8273 Programmable HDLC/SDLC Protocol Controller ................................... 2-80
- 8274 Multi-Protocol Serial Controller (MPSC) ............................................. 2-112
- 82530/82530-6 Serial Communications Controller (SCC) ............................. 2-150

APPLICATION NOTES
- Designing with the 82510 Asynchronous Serial Controller AP-401 .................. 2-182
- High Performance Driver for 82510 AP-310 ................................................. 2-262
- Using the 8273 SDLC/HDLC Protocol Controller AP-36 .............................. 2-293
- Asynchronous Communication with the 8274 Multiple Protocol Serial Controller AP-134 ................................................................. 2-345
- Synchronous Communication with the 8274 Multiple Protocol Serial Controller AP-145 ................................................................. 2-383
- Asynchronous and SDLC Communications with the 82530 AP-222 .................. 2-421

CHAPTER 3
Other Components

DATA SHEETS
- 8291A GPIB Talker/Listener ................................................................. 3-1
- 8292 GPIB Controller .............................................................................. 3-33
- 8294A Data Encryption/Decryption Unit .................................................... 3-52

APPLICATION NOTES
- Using the 8291A GPIB Talker/Listener AP-166 ........................................... 3-65
- Using the 8292 GPIB Controller AP-66 .................................................... 3-95
Table of Contents (Continued)

SECTION TWO—TELECOMMUNICATION COMPONENTS

CHAPTER 4
Modem Products
DATA SHEETS
89024 2400 BPS Intelligent Modem Chip Set ........................................ 4-1
89C024XE High Performance 2400 BPS Intelligent Modem Chip Set ............. 4-22
MNP Application Package
MNP Class 5 Software for the 89C024XE Modem Chip Set ...................... 4-44
EVALUATION KITS
89024 MEKPC, PC Card Modem Evaluation Kit for 89024 ....................... 4-46
89024 MEK2, 89024 Enhanced Modem Evaluation Kit .......................... 4-48
MEK3 Modem Evaluation Kit .......................................................... 4-50

CHAPTER 5
iATC Advanced Telecommunications Components for Analog and ISDN Applications
SLD Interface Specification ............................................................. 5-1
DATA SHEETS
iATC 29C48 Feature Control Combo .............................................. 5-5
iATC 29C53AA Digital Loop Controller ........................................ 5-25
89151 T-Link Communications Controller ................................... 5-50
ISP188 ISDN Software Package for the 80188 .................................. 5-53
PC53 ISDN Board ........................................................................ 5-59
IDK 29C53 ISDN Development Kit for 29C53 .................................. 5-67
LEK 29C53 Line Card Evaluation Kit ........................................... 5-73
TEK 29C53 Terminal Evaluation Kit .............................................. 5-75
APPLICATION NOTES
29C53 Transceiver Line Interfacing AP-282 .................................... 5-78
ISDN Applications with 29C53 and 80188 AP-400 .......................... 5-92

CHAPTER 6
PCM Codec/Filter and Combo
DATA SHEETS
2910A PCM Codec-μ Law 8-Bit Companded A/D and D/A Converter ........ 6-1
2911A-1 PCM Codec-A Law 8-Bit Companded A/D and D/A Converter .... 6-19
2912A PCM Transmit/Receive Filter .............................................. 6-34
2913 and 2914 Combined Single-Chip PCM Codec and Filter .............. 6-46
2916/2917 HMOS Combined Single-Chip PCM Codec and Filter .......... 6-67
APPLICATION NOTES
Applications Information 2910A/2911A/2912A ................................ 6-85
Designing Second-Generation Digital Telephony Systems Using the Intel 2913/2914 Codec/Filter Combochip AP-142 ....................... 6-88

SECTION THREE—SYSTEM PRODUCTS

CHAPTER 7
Local Area Networking Boards and Software
OpenNET™ Local Area Network Family .......................................... 7-1

CHAPTER 8
Serial Communication Boards
iSBC® 88/45 Advanced Data Communications Processor Board ............. 8-1
iSBC® 188/56 Advanced Communicating Computer .......................... 8-10
iSBC® 534 Four Channel Communications Expansion Board .................. 8-19
iSBC® 544 Intelligent Communications Controller ............................ 8-24
Table of Contents (Continued)

<table>
<thead>
<tr>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>iSBC® 548/549 Terminal Controllers</td>
<td>8-33</td>
</tr>
<tr>
<td>iSBC® 561 SOEMI Controller Board</td>
<td>8-37</td>
</tr>
<tr>
<td>iSBX™ 351 Serial I/O MULTIMODULE™ Board</td>
<td>8-42</td>
</tr>
<tr>
<td>iSBX™ 354 Dual Channel Serial I/O MULTIMODULE™ Board</td>
<td>8-48</td>
</tr>
<tr>
<td>iSBC® 186/410 MULTIBUS® II Serial Communications Computer</td>
<td>8-53</td>
</tr>
</tbody>
</table>
## Alphanumeric Index

<table>
<thead>
<tr>
<th>Reference</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2910A PCM Codec-μ Law 8-Bit Companded A/D and D/A Converter</td>
<td>6-1</td>
</tr>
<tr>
<td>2911A-1 PCM Codec-μ Law 8-Bit Companded A/D and D/A Converter</td>
<td>6-19</td>
</tr>
<tr>
<td>2912A PCM Transmit/Receive Filter</td>
<td>6-34</td>
</tr>
<tr>
<td>2913 and 2914 Combined Single-Chip PCM Codec and Filter</td>
<td>6-46</td>
</tr>
<tr>
<td>2916/2917 HMOS Combined Single-Chip PCM Codec and Filter</td>
<td>6-67</td>
</tr>
<tr>
<td>29C53 Transceiver Line Interfacing AP-282</td>
<td>5-78</td>
</tr>
<tr>
<td>82050 Asynchronous Communications Controller</td>
<td>2-26</td>
</tr>
<tr>
<td>82510 Asynchronous Serial Controller (SCC)</td>
<td>2-40</td>
</tr>
<tr>
<td>8251A Programmable Communication Interface</td>
<td>2-1</td>
</tr>
<tr>
<td>82530/82530-6 Serial Communications Controller (SCC)</td>
<td>2-150</td>
</tr>
<tr>
<td>82560 Host Interface and Memory Controller</td>
<td>1-56</td>
</tr>
<tr>
<td>82586 Local Area Network Coprocessor</td>
<td>1-1</td>
</tr>
<tr>
<td>82588 High Integration LAN Control</td>
<td>1-120</td>
</tr>
<tr>
<td>82590 Advanced CSMA/CD LAN Controller with 8-Bit Data Path</td>
<td>1-86</td>
</tr>
<tr>
<td>82592 Advanced CSMA/CD LAN Controller with 16-Bit Data Path</td>
<td>1-103</td>
</tr>
<tr>
<td>8273 Programmable HDLC/SDLC Protocol Controller</td>
<td>2-80</td>
</tr>
<tr>
<td>8274 Multi-Protocol Serial Controller (MPSC)</td>
<td>2-112</td>
</tr>
<tr>
<td>8291A GPIB Talker/Listener</td>
<td>3-1</td>
</tr>
<tr>
<td>8292 GPIB Controller</td>
<td>3-33</td>
</tr>
<tr>
<td>8294A Data Encryption/Decryption Unit</td>
<td>3-52</td>
</tr>
<tr>
<td>82C501 Ethernet Serial Interface</td>
<td>1-38</td>
</tr>
<tr>
<td>89024 2400 BPS Intelligent Modem Chip Set</td>
<td>4-1</td>
</tr>
<tr>
<td>89024 MEK2, 89024 Enhanced Modem Evaluation Kit</td>
<td>4-48</td>
</tr>
<tr>
<td>89024 MEKPC, PC Card Modem Evaluation Kit for 89024</td>
<td>4-46</td>
</tr>
<tr>
<td>89151 T-Link Communications Controller</td>
<td>5-50</td>
</tr>
<tr>
<td>89C024XE High Performance 2400 BPS Intelligent Modem Chip Set</td>
<td>4-22</td>
</tr>
<tr>
<td>An 82586 Data Link Driver AP-235</td>
<td>1-176</td>
</tr>
<tr>
<td>Applications Information 2910A/2911A/2912A</td>
<td>6-85</td>
</tr>
<tr>
<td>Asynchronous and SDLC Communications with the 82530 AP-222</td>
<td>2-421</td>
</tr>
<tr>
<td>Asynchronous Communication with the 8274 Multiple Protocol Serial Controller AP-134</td>
<td>2-345</td>
</tr>
<tr>
<td>Designing Second-Generation Digital Telephony Systems Using the Intel 2913/2914 Codec/Filter Combochip AP-142</td>
<td>6-88</td>
</tr>
<tr>
<td>Designing with the 82510 Asynchronous Serial Controller AP-401</td>
<td>2-182</td>
</tr>
<tr>
<td>High Performance Driver for 82510 AP-310</td>
<td>2-262</td>
</tr>
<tr>
<td>iATC 29C48 Feature Control Combo</td>
<td>5-5</td>
</tr>
<tr>
<td>iATC 29C53AA Digital Loop Controller</td>
<td>5-25</td>
</tr>
<tr>
<td>IDK 29C53 ISDN Development Kit for 29C53</td>
<td>5-67</td>
</tr>
<tr>
<td>Implementing Ethernet/Cheapernet with the Intel 82586 AP-274</td>
<td>1-256</td>
</tr>
<tr>
<td>Implementing StarLAN with the Intel 82588 AP-236</td>
<td>1-345</td>
</tr>
<tr>
<td>iSBC® 186/410 MULTIBUS® II Serial Communications Computer</td>
<td>8-53</td>
</tr>
<tr>
<td>iSBC® 188/56 Advanced Communicating Computer</td>
<td>8-10</td>
</tr>
<tr>
<td>iSBC® 534 Four Channel Communications Expansion Board</td>
<td>8-19</td>
</tr>
<tr>
<td>iSBC® 544 Intelligent Communications Controller</td>
<td>8-24</td>
</tr>
<tr>
<td>iSBC® 548/549 Terminal Controllers</td>
<td>8-33</td>
</tr>
<tr>
<td>iSBC® 561 SOEMI Controller Board</td>
<td>8-37</td>
</tr>
<tr>
<td>iSBC® 88/45 Advanced Data Communications Processor Board</td>
<td>8-1</td>
</tr>
<tr>
<td>iSBX™ 351 Serial I/O MULTIMODULE™ Board</td>
<td>8-42</td>
</tr>
<tr>
<td>iSBX™ 354 Dual Channel Serial I/O MULTIMODULE™ Board</td>
<td>8-48</td>
</tr>
<tr>
<td>ISDN Applications with 29C53 and 80188 AP-400</td>
<td>5-92</td>
</tr>
<tr>
<td>ISP188 ISDN Software Package for the 80188</td>
<td>5-53</td>
</tr>
<tr>
<td>LEK 29C53 Line Card Evaluation Kit</td>
<td>5-73</td>
</tr>
<tr>
<td>MEK3 Modem Evaluation Kit</td>
<td>4-50</td>
</tr>
<tr>
<td>MNP Application Package</td>
<td>4-44</td>
</tr>
<tr>
<td>MNP Class 5 Software for the 89C024XE Modem Chip Set</td>
<td>4-44</td>
</tr>
</tbody>
</table>
# Alphanumeric Index (Continued)

<table>
<thead>
<tr>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>OpenNET™ Local Area Network Family</td>
<td>7-1</td>
</tr>
<tr>
<td>PC53 ISDN Board</td>
<td>5-59</td>
</tr>
<tr>
<td>PC586E CSMA/CD LAN Evaluation Board</td>
<td>1-464</td>
</tr>
<tr>
<td>SLD Interface Specification</td>
<td>5-1</td>
</tr>
<tr>
<td>Synchronous Communication with the 8274 Multiple Protocol Serial Controller</td>
<td>2-383</td>
</tr>
<tr>
<td>TEK 29C53 Terminal Evaluation Kit</td>
<td>5-75</td>
</tr>
<tr>
<td>Twisted Pair Ethernet Chip Set</td>
<td>1-54</td>
</tr>
<tr>
<td>Using the 8273 SDLC/HDLC Protocol Controller AP-36</td>
<td>2-293</td>
</tr>
<tr>
<td>Using the 8291A GPIB Talker/Listener AP-166</td>
<td>3-65</td>
</tr>
<tr>
<td>Using the 8292 GPIB Controller AP-66</td>
<td>3-95</td>
</tr>
<tr>
<td>Using the Intel 82592 to Integrate a Low-Cost Ethernet Solution into a PC Motherboard</td>
<td>1-419</td>
</tr>
<tr>
<td>Intel Corporation Product Codes/Part Numbers</td>
<td>Intel Puerto Rico, Inc., Product Codes/Part Numbers</td>
</tr>
<tr>
<td>---------------------------------------------</td>
<td>-----------------------------------------------</td>
</tr>
<tr>
<td>SBC28616 pSBC28616</td>
<td>SBCMEM310 pSBCMEM310</td>
</tr>
<tr>
<td>SBC300 pSBC300</td>
<td>SBCMEM312 pSBCMEM312</td>
</tr>
<tr>
<td>SBC301 pSBC301</td>
<td>SBCMEM320 pSBCMEM320</td>
</tr>
<tr>
<td>SBC302 pSBC302</td>
<td>SBCMEM340 pSBCMEM340</td>
</tr>
<tr>
<td>SBC304 pSBC304</td>
<td>SBE96 pSBE96</td>
</tr>
<tr>
<td>SBC307 pSBC307</td>
<td>SBX217 SBX217</td>
</tr>
<tr>
<td>SBC314 pSBC314</td>
<td>SBX218 SBX218</td>
</tr>
<tr>
<td>SBC322 pSBC322</td>
<td>SBX270 SBX270</td>
</tr>
<tr>
<td>SBC324 pSBC324</td>
<td>SBX311 SBX311</td>
</tr>
<tr>
<td>SBC337 pSBC337</td>
<td>SBX328 SBX328</td>
</tr>
<tr>
<td>SBC341 pSBC341</td>
<td>SBX331 SBX331</td>
</tr>
<tr>
<td>SBC386 pSBC386 sSBC386</td>
<td>SBC386 SBC386</td>
</tr>
<tr>
<td>SBC386116 pSBC386116</td>
<td>SBC386116 SBC386116</td>
</tr>
<tr>
<td>SBC386120 pSBC386120</td>
<td>SBC386120 SBC386120</td>
</tr>
<tr>
<td>SBC38621 pSBC38621</td>
<td>SBC38621 SBC38621</td>
</tr>
<tr>
<td>SBC38622 pSBC38622</td>
<td>SBC38622 SBC38622</td>
</tr>
<tr>
<td>SBC38624 pSBC38624</td>
<td>SBC38624 SBC38624</td>
</tr>
<tr>
<td>SBC38628 pSBC38628</td>
<td>SBC38628 SBC38628</td>
</tr>
<tr>
<td>SBC38631 pSBC38631</td>
<td>SBC38631 SBC38631</td>
</tr>
<tr>
<td>SBC38632 pSBC38632</td>
<td>SBC38632 SBC38632</td>
</tr>
<tr>
<td>SBC38634 pSBC38634</td>
<td>SBC38634 SBC38634</td>
</tr>
<tr>
<td>SBC38638 pSBC38638</td>
<td>SBC38638 SBC38638</td>
</tr>
<tr>
<td>SBC428 pSBC428 sSBC428</td>
<td>SBC428 SBC428</td>
</tr>
<tr>
<td>SBC464 pSBC464 sSBC464</td>
<td>SBC464 SBC464</td>
</tr>
<tr>
<td>SBC517 pSBC517</td>
<td>SBC517 SBC517</td>
</tr>
<tr>
<td>SBC519 pSBC519 sSBC519</td>
<td>SBC519 SBC519 sSBC519</td>
</tr>
<tr>
<td>SBC534 pSBC534 sSBC534</td>
<td>SBC534 SBC534</td>
</tr>
<tr>
<td>SBC548 pSBC548</td>
<td>SBC548 SBC548</td>
</tr>
<tr>
<td>SBC550 pTSBC550 TSBC550</td>
<td>SBC550 pTSBC550 TSBC550</td>
</tr>
<tr>
<td>SBC550 pSBC550</td>
<td>SBC550 pSBC550</td>
</tr>
<tr>
<td>SBC552 pSBC552</td>
<td>SBC552 pSBC552</td>
</tr>
<tr>
<td>SBC556 pSBC556 sSBC556</td>
<td>SBC556 pSBC556 sSBC556</td>
</tr>
<tr>
<td>SBC569 pSBC569</td>
<td>SBC569 pSBC569</td>
</tr>
<tr>
<td>SBC589 pSBC589</td>
<td>SBC589 pSBC589</td>
</tr>
<tr>
<td>SBC604 pSBC604</td>
<td>SBC604 pSBC604</td>
</tr>
<tr>
<td>SBC608 pSBC608</td>
<td>SBC608 pSBC608</td>
</tr>
<tr>
<td>SBC614 pSBC614</td>
<td>SBC614 pSBC614</td>
</tr>
<tr>
<td>SBC618 pSBC618</td>
<td>SBC618 pSBC618</td>
</tr>
<tr>
<td>SBC655 pSBC655</td>
<td>SBC655 pSBC655</td>
</tr>
<tr>
<td>SBC6611 pSBC6611</td>
<td>SBC6611 pSBC6611</td>
</tr>
<tr>
<td>SBC8010 pSBC8010</td>
<td>SBC8010 pSBC8010</td>
</tr>
<tr>
<td>SBC8020 pSBC8020</td>
<td>SBC8020 pSBC8020</td>
</tr>
<tr>
<td>SBC8024 pSBC8024</td>
<td>SBC8024 pSBC8024</td>
</tr>
<tr>
<td>SBC8030 pSBC8030</td>
<td>SBC8030 pSBC8030</td>
</tr>
<tr>
<td>SBC8605 pSBC8605</td>
<td>SBC8605 pSBC8605</td>
</tr>
<tr>
<td>SBC8612 pSBC8612</td>
<td>SBC8612 pSBC8612</td>
</tr>
<tr>
<td>SBC8614 pSBC8614</td>
<td>SBC8614 pSBC8614</td>
</tr>
<tr>
<td>SBC8630 pSBC8630</td>
<td>SBC8630 pSBC8630</td>
</tr>
<tr>
<td>SBC8635 pSBC8635</td>
<td>SBC8635 pSBC8635</td>
</tr>
<tr>
<td>SBC86C38 pSBC86C38</td>
<td>SBC86C38 pSBC86C38</td>
</tr>
<tr>
<td>SBC8825 pSBC8825</td>
<td>SBC8825 pSBC8825</td>
</tr>
<tr>
<td>SBC8840 pSBC8840</td>
<td>SBC8840 pSBC8840</td>
</tr>
<tr>
<td>SBC8845 pSBC8845</td>
<td>SBC8845 pSBC8845</td>
</tr>
<tr>
<td>SBC905 pSBC905</td>
<td>SBC905 pSBC905</td>
</tr>
<tr>
<td>SBCLNK001 pSBCLNK001</td>
<td>SBCLNK001 pSBCLNK001</td>
</tr>
</tbody>
</table>
Any of the following products may appear in this publication. If so, it must be noted that such products have counterparts manufactured by Intel Puerto Rico, Inc., Intel Puerto Rico II, Inc., and/or Intel Singapore, Ltd. The product codes/part numbers of these counterpart products are listed below next to the corresponding Intel Corporation product codes/part numbers.

<table>
<thead>
<tr>
<th>Intel Corporation Product Codes/Part Numbers</th>
<th>Intel Puerto Rico, Inc. Product Codes/Part Numbers</th>
<th>Intel Singapore, Ltd. Product Codes/Part Numbers</th>
<th>Intel Corporation Product Codes/Part Numbers</th>
<th>Intel Puerto Rico, Inc. Product Codes/Part Numbers</th>
<th>Intel Singapore, Ltd. Product Codes/Part Numbers</th>
</tr>
</thead>
<tbody>
<tr>
<td>376SKIT</td>
<td>903</td>
<td>904</td>
<td>KM2</td>
<td>KM4</td>
<td>KM8</td>
</tr>
<tr>
<td></td>
<td>913</td>
<td>914</td>
<td>KNLAN</td>
<td>KT60</td>
<td>pKNLAN</td>
</tr>
<tr>
<td></td>
<td>923</td>
<td>924</td>
<td>KW140</td>
<td>KW140</td>
<td>kW140</td>
</tr>
<tr>
<td></td>
<td>952</td>
<td>953</td>
<td>KW40</td>
<td>KW40</td>
<td>kW40</td>
</tr>
<tr>
<td></td>
<td>954</td>
<td></td>
<td>KW80</td>
<td>KW80</td>
<td>kW80</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>M1</td>
<td>M1</td>
<td>M1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>M2</td>
<td>M2</td>
<td>M2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>M4</td>
<td>M4</td>
<td>M4</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>M8</td>
<td>M8</td>
<td>M8</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>MDS610</td>
<td>MDS610</td>
<td>MDS610</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>MDX3015</td>
<td>MDX3015</td>
<td>MDX3015</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>MDX3015</td>
<td>MDX3015</td>
<td>MDX3015</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>MDX3016</td>
<td>MDX3016</td>
<td>MDX3016</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>MDX457</td>
<td>MDX457</td>
<td>MDX457</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>MDX457</td>
<td>MDX457</td>
<td>MDX457</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>MDX458</td>
<td>MDX458</td>
<td>MDX458</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>MDX458</td>
<td>MDX458</td>
<td>MDX458</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>MSA96</td>
<td>MSA96</td>
<td>MSA96</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>NLAN</td>
<td>NLAN</td>
<td>NLAN</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>PCX344A</td>
<td>pPCX344A</td>
<td>sPCX344A</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>R286ASM</td>
<td>R286ASM</td>
<td>R286ASM</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>R286ED1</td>
<td>R286ED1</td>
<td>R286ED1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>R286PLM</td>
<td>R286PLM</td>
<td>R286PLM</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>R286SSC</td>
<td>R286SSC</td>
<td>R286SSC</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>R86FOR</td>
<td>R86FOR</td>
<td>R86FOR</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>RC84410</td>
<td>RC84410</td>
<td>RC84410</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>RCX920</td>
<td>RCX920</td>
<td>RCX920</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>RMX286</td>
<td>RMX286</td>
<td>RMX286</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>RMXNET</td>
<td>RMXNET</td>
<td>RMXNET</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>S301</td>
<td>S301</td>
<td>S301</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>S386</td>
<td>S386</td>
<td>S386</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>SBC010</td>
<td>SBC010</td>
<td>SBC010</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>SBC012</td>
<td>SBC012</td>
<td>SBC012</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>SBC020</td>
<td>SBC020</td>
<td>SBC020</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>SBC028</td>
<td>SBC028</td>
<td>SBC028</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>SBC040</td>
<td>SBC040</td>
<td>SBC040</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>SBC056</td>
<td>SBC056</td>
<td>SBC056</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>SBC108</td>
<td>SBC108</td>
<td>SBC108</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>SBC116</td>
<td>SBC116</td>
<td>SBC116</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>SBC18603</td>
<td>SBC18603</td>
<td>SBC18603</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>SBC18604</td>
<td>SBC18604</td>
<td>SBC18604</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>SBC18651</td>
<td>SBC18651</td>
<td>SBC18651</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>SBC186530</td>
<td>SBC186530</td>
<td>SBC186530</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>SBC18678</td>
<td>SBC18678</td>
<td>SBC18678</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>SBC18848</td>
<td>SBC18848</td>
<td>SBC18848</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>SBC18856</td>
<td>SBC18856</td>
<td>SBC18856</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>SBC208</td>
<td>SBC208</td>
<td>SBC208</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>SBC214</td>
<td>SBC214</td>
<td>SBC214</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>SBC215</td>
<td>SBC215</td>
<td>SBC215</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>SBC220</td>
<td>SBC220</td>
<td>SBC220</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>SBC221</td>
<td>SBC221</td>
<td>SBC221</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>SBC28610</td>
<td>SBC28610</td>
<td>SBC28610</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>SBC28610</td>
<td>SBC28610</td>
<td>SBC28610</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>SBC28612</td>
<td>SBC28612</td>
<td>SBC28612</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>SBC28614</td>
<td>SBC28614</td>
<td>SBC28614</td>
</tr>
</tbody>
</table>

OVERVIEW

Imagine for a moment a world where all electronic communications were instantaneous. A world where voice, data, and graphics could all be transported via telephone lines to a variety of computers and receiving systems. A world where the touch of a finger could summon information ranging from stock reports to classical literature and bring it into environments as diverse as offices and labs, factories and living rooms.

Unfortunately, these promises of the Information Age still remain largely unfulfilled. While computer technology has accelerated rapidly over the last twenty years, the communications methods used to tie the wide variety of electronic systems in the world together have, by comparison, failed to keep pace. Faced with a tangle of proprietary offerings, high costs, evolving standards, and incomplete technologies, the world is still waiting for networks that are truly all-encompassing, the missing links to today's communications puzzle.

Enter microcommunications—microchip-based digital communications products and services. A migration of the key electronics communications functions into silicon is now taking place, providing the vital interfaces that have been lacking among the various networks now employed throughout the world. Through the evolution of VLSI (Very Large Scale Integration) technology, microcommunications now can offer the performance required to effect these communications interfaces at affordable costs, spanning the globe with silicon to eradicate the troublesome bottleneck that has plagued information transfer during recent years.

“There are three parts to the communications puzzle,” says Gordon Moore, Intel Chairman and CEO. “The first incorporates the actual systems that communicate with each other, and the second is the physical means to connect them—such as cables, microwave technology, or fiber optics. It is the third area, the interfaces between the systems and the physical links, where silicon will act as the linchpin. That, in essence, is what microcommunications is all about.”

THE COMMUNICATIONS BOTTLENECK

Visions of global networks are not new. Perhaps one of the most noteworthy of these has been espoused by Dr. Koji Kobayashi, chairman of NEC Corporation. His view of the future, developed over the nearly fifty years of his association with NEC, is known as C&C (Computers and Communications). It defines the marriage of passive communications systems and computers as processors and manipulators of information, providing the foundation for a discipline that is changing the basic character of modern society.

Kobayashi’s macro vision hints at the obstacles confronting the future of C&C. When taken to the micro level, to silicon itself, one begins to understand the complexities that are involved. When Intel invented the microprocessor fifteen years ago, the first seeds of the personal computer revolution were sown, marking an era that over the last decade has dramatically influenced the way people work and live. PCs now proliferate in the office, in factories, and throughout laboratory environments. And their “intimidation” factor has lessened to where they are also becoming more and more prevalent in the home, beginning to penetrate a market that to date has remained relatively untapped.

Thanks to semiconductor technology, the personal computer has raised the level of productivity in our society. But most of that productivity has been gained by individuals at isolated workstations. Group productivity, meanwhile, still leaves much to be desired. The collective productivity of organizations can only be enhanced through more sophisticated networking technology. We are now faced with isolated “islands of automation” that must somehow be developed into networks of productivity.

But no amount of computing can meet these challenges if the corresponding communications technology is not sufficiently in step. The Information Age can only grow as fast as the lowest common denominator—which in this case is the aggregate communications bandwidth that continues to lag behind our increased computing power. Such is the nature of the communications bottleneck, where the growing amounts of information we are capable of generating can only flow as fast as the limited and incompatible communications capabilities now in place. Clearly, a crisis is at hand.

BREAKING UP THE BOTTLENECK

Three factors have contributed to this logjam: lack of industry standards, an insufficient cost/performance ratio, and the incomplete status of available communications technology to date.

- Standards—One look at the tangle of proprietary systems now populating office, factory, and laboratory environments gives a good indication of the inherent difficulty in hooking these diverse systems together. And these systems do not merely feature different architectures—they also represent completely different levels of computing, ranging from giant mainframes at one end of the scale down to individual microcontrollers on the other.

The market has simply grown too fast to effectively accommodate the changes that have occurred. Suppliers face the dilemma of meshing product differentiation issues with industry-wide compatibility as
they develop their strategies; opting for one in the past often meant forsaking the other. And while some standards have coalesced, the industry still faces a technological Tower of Babel, with many proprietary solutions vying to be recognized in leadership positions.

- Cost/Performance Ratio—While various communications technologies struggle toward maturity, the industry has had to cope with tremendous costs associated with interconnectivity and interoperability. Before the shift to microelectronic interfaces began to occur, these connections often were prohibitively expensive.

Says Ron Whittier, Intel Vice President and Director of Marketing: “Mainframes offer significant computing and communications power, but at a price that limits the number of users. What is needed is cost-effective communications solutions to hook together the roughly 16 million installed PCs in the market, as well as the soon-to-exist voice/data terminals. That’s the role of microcommunications—bringing cost-effective communications solutions to the microcomputer world.”

- Incomplete Technology—Different suppliers have developed many networking schemes, but virtually all have been fragmented and unable to meet the wide range of needs in the marketplace. Some of these approaches have only served to create additional problems, making OEMs and systems houses loathe to commit to suppliers who they fear cannot provide answers at all of the levels of communications that are now funneled into the bottleneck.

THE NETWORK TRINITY

Three principal types of networks now comprise the electronic communications marketplace: Wide Area Networks (WANs), Local Area Networks (LANs), and Small Area Networks (SANs). Each in its own fashion is turning to microcommunications for answers to its networking problems.

WANs—known by some as Global Area Networks (GANs)—are most commonly associated with the worldwide analog telephone system. The category also includes a number of other segments, such as satellite and microwave communications, traditional networks (like mainframe-to-mainframe connections), modems, statistical multiplexers, and front-end communications processors. The lion’s share of nodes—electronic network connections—in the WAN arena, however, resides in the telecommunications segment. This is where the emerging ISDN (Integrated Services Digital Network) standard comes into focus as the most visible portion of the WAN marketplace.

The distances over which information may be transmitted via a WAN are essentially unlimited. The goal of ISDN is to take what is largely an analog global system and transform it into a digital network by defining the standard interfaces that will provide connections at each node.

These interfaces will allow basic digital communications to occur via the existing twisted pair of wires that comprise the telephone lines in place today. This would bypass the unfeasible alternative of installing completely new lines, which would be at cross purposes with the charter of ISDN: to reduce costs and boost performance through realization of an all-digital network.

The second category, Local Area Networks, represents the most talked-about link provided by microcommunications. In their most common form, LANs are comprised of—but not limited to—PC-to-PC connections. They incorporate information exchange over limited distances, usually not exceeding five kilometers, which often takes place within the same building or between adjacent work areas. The whole phenomenon surrounding LAN development, personal computing, and distributed processing essentially owes its existence to microcomputer technology, so it is not surprising that this segment of networking has garnered the attention it has in microelectronic circles.

Because of that, progress is being made in this area. The most prominent standard—which also applies to WANs and SANs—is the seven-layer Open Systems Interconnection (OSI) Model, established by the International Standards Organization (ISO). The model provides the foundation to which all LAN configurations must adhere if they hope to have any success in the marketplace. Interconnection protocols determining how systems are tied together are defined in the first five layers. Interoperation concepts are covered in the upper two layers, defining how systems can communicate with each other once they are tied together.

In the LAN marketplace, a large number of networking products and philosophies are available today, offering solutions at various price/performance points. Diverse approaches such as StarLAN, Token Bus and Token Ring, Ethernet, and PC-NET, to name a few of the more popular office LAN architectures, point to many choices for OEMs and end users.

A similar situation exists in the factory. While the Manufacturing Automation Protocol (MAP) standard is coalescing around the leadership of General Motors,
Boeing, and others, a variety of proprietary solutions also abound. The challenge is for a complete set of interfaces to emerge that can potentially tie all of these networks together in—and among—the office, factory, and lab environments.

The final third of the network trinity is the Small Area Network (SAN). This category is concerned with communications over very short distances, usually not exceeding 100 meters. SANs most often deal with chip-to-chip or chip-to-system transfer of information; they are optimized to deal with real-time applications generally managed by microcontrollers, such as those that take place on the factory floor among robots at various workstations.

SANs incorporate communications functions that are undertaken via serial backplanes in microelectronic equipment. While they represent a relatively small market in 1986 when compared to WANs and LANs, a tenfold increase is expected through 1990. SANs will have the greatest number of nodes among network applications by the next decade, thanks to their preponderance in many consumer products.

While factory applications will make up a large part of the SAN marketplace probably the greatest contributor to growth will be in automotive applications. Microcontrollers are now used in many dashboards to control a variety of engine tasks electronically, but they do not yet work together in organized and efficient networks. As Intel's Gordon Moore commented earlier this year to the New York Society of Security Analysts, when this technology shifts into full gear during the next decade, the total automobile electronics market will be larger than the entire semiconductor market was in 1985.

MARKET OPPORTUNITIES

Such growth is also mirrored in the projections for the WAN and LAN segments, which, when combined with SANs, make up the microcommunications market pie. According to Intel analysts, the total silicon microcommunications market in 1985 amounted to $522 million. By 1989, Intel predicts this figure will have expanded to $1290 million, representing a compounded annual growth rate of 25%.

And although the WAN market will continue to grow at a comfortable rate, the SAN and LAN pieces of the pie will increase the most dramatically. Whereas SANs represented only about 12.5% ($65 million) in 1985, they could explode to 22.5% ($290 million) of the larger pie by 1989. This growth is paralleled by increases in the LAN segment, which should grow from 34.5% of the total silicon microcommunications market in 1985 to 44.5% of the expanded pie in 1989.

Opportunities abound for microcommunications suppliers as the migration to silicon continues. And perhaps no VLSI supplier is as well-positioned in this marketplace as Intel, which predicts that 50% of its products will be microcommunications-related by 1990. The key here is the corporation's ability to bridge the three issues that contribute to the communications bottleneck: standards, cost-performance considerations, and the completeness of microcomputer and microcommunications product offerings.

INTEL AND VLSI: THE MICROCOMMUNICATIONS MATCH

Intel innovations helped make the microcomputer revolution possible. Such industry “firsts” include the microprocessor, the EPROM, the E2PROM, the microcontroller, development systems, and single board computers. Given this legacy, it is not surprising that the corporation should come to the microcommunications marketplace already equipped with a potent arsenal of tools and capabilities.

The first area centers on industry standards. As a VLSI microelectronic leader, Intel has been responsible for driving many of the standards that are accepted by the industry today. And when not actually initiating these standards, Intel has supported other existing and emerging standards through its longtime “open systems” philosophy. This approach protects substantial customer investments and ensures easy upgradability by observing compatibility with previous architectures and industry-leading standards.

Such a position is accentuated by Intel's technology relationships and alliances with many significant names in the microcommunications field. Giants like AT&T in the ISDN arena, General Motors in factory networking, and IBM in office automation all are working closely with Intel to further the standardization of the communications interfaces that are so vital to the world’s networking future.

Cost/performance considerations also point to Intel's strengths. As a pioneer in VLSI technology, Intel has been at the forefront of achieving greater circuit densities and performance on single pieces of silicon: witness the 275,000 transistors housed on the 32-bit 80386, the highest performance commercial microprocessor ever built. As integration has increased, cost-per-bit has decreased steadily, marking a trend that remains consistent in the semiconductor industry. And one thing is
certain: microcommunications has a healthy appetite for transistors, placing it squarely in the center of the VLSI explosion.

But it is in the final area—completeness of technology and products—where Intel is perhaps the strongest. No other microelectronic vendor can point to as wide an array of products positioned across the various segments that comprise the microelectronic marketplace. Whether it be leadership in the WAN marketplace as the number one supplier of merchant telecommunications components, strength in SANs with world leadership in microcontrollers, or overall presence in the LAN arena with complete solutions in components, boards, software, and systems, Intel is a vital presence in the growing microcommunications arena.

That leadership extends beyond products. Along with its own application software, Intel is promoting expansion through partnerships with many different independent software vendors (ISVs), ensuring that the necessary application programs will be in place to fuel the gains provided by the silicon "engines" residing at the interface level. And finally, the corporation's commitment to technical support training, service, and its strong force of field applications engineers guarantees that it will back up its position and serve the needs that will continue to spring up as the microcommunications evolution becomes a reality.

Together, all the market segment alluded to in this article comprise the world of microcommunications, a world coming closer together every day as the web of networking solutions expands—all thanks to the technological ties that bind, reaching out to span the globe with silicon.
Local Area Networks
82586
LOCAL AREA NETWORK COPROCESSOR

- Performs Complete CSMA/CD Medium Access Control Functions Independently of CPU — High Level Command Interface
- Supports Established and Emerging LAN Standards — IEEE 802.3/Ethernet (10BASE5) — IEEE 802.3/Cheapernet (10BASE2) — IBM PC Network — IEEE 802.3/StarLAN (1BASE5) — Proprietary CSMA/CD Networks up to 10 Mbps
- On-Chip Memory Management — Automatic Buffer Chaining — Buffer Reclaim After Receipt of Bad Frames — Save Bad Frames, Optionally
- Interfaces to 8-bit and 16-bit Microprocessors
- 48 Pin DIP and 68 Pin PLCC (see "Intel Packaging" Document, Order Number: 231369-001)

- Supports Minimum Component Systems — Shared Bus Configuration — Interface to iAPX 186 and 188 Microprocessors without Glue
- Supports High Performance Systems — Bus Master, with On-Chip DMA — 5 MBytes/Sec Bus Bandwidth — Compatible with Dual Port Memory — Back to Back Frame Reception at 10 Mbps
- Network Management — CRC Error Tally — Alignment Error Tally — Location of Cable Faults
- Self-Test Diagnostics — Internal Loopback — External Loopback — Internal Register Dump — Backoff Timer Check

Figure 1. 82586 Functional Block Diagram

*IBM is a trademark of International Business Machines Corp.
NOTE:
The symbols in parentheses correspond to minimum mode.

Figure 2. 82586 Pinout Diagrams
The 82586 is an intelligent, high performance Local Area Network coprocessor, implementing the CSMA/CD access method (Carrier Sense Multiple Access with Collision Detection). It performs all time-critical functions independently of the host processor, which maximizes performance and network efficiency.

The 82586 performs the full set of IEEE 802.3 CSMA/CD media access control and channel interface functions including: framing, preamble generation and stripping, source address generation, destination address checking, CRC generation and checking, short frame detection. Any data rate up to 10 Mb/s can be used.

The 82586 features a powerful host system interface. It automatically manages memory structures with command chaining and bidirectional data chaining. An on-chip DMA controller manages 4 channels transparently to the user. Buffers containing errored or collided frames can be automatically recovered. The 82586 can be configured for 8-bit or 16-bit data path, with maximum burst transfer rate of 2 or 4 Mbyte/sec. respectively. Memory address space is 16 Mbyte maximum.

The 82586 provides two independent 16 byte FIFOs, one for receiving and one for transmitting. The threshold for block transfer to/from memory is programmable, enabling the user to optimize bus overhead for a given worst case bus latency.

The 82586 provides a rich set of diagnostic and network management functions including: internal and external loopbacks, exception condition tallies, channel activity indicators, optional capture of all frames regardless of destination address, optional capture of errored or collided frames, and time domain reflectometry for locating faults in the cable.

The 82586 can be used in either baseband or broadband networks. It can be configured for maximum network efficiency (minimum contention overhead) for any length network operating at any data rate up to 10 Mbps. The controller supports address field lengths of 1, 2, 3, 4, 5, or 6 bytes. It can be configured for either the IEEE 802.3/Ethernet or HDLC method of frame delineation. Both 16-bit and 32-bit CRC are supported.

The 82586 is fabricated in Intel's reliable HMOS II 5V technology and is available in a 48 pin DIP or 68 pin PLCC package.

Table 1. 82586 Pin Description

<table>
<thead>
<tr>
<th>Symbol</th>
<th>48 Pin DIP Pin No.</th>
<th>68 Pin PLCC Pin No.</th>
<th>Type Level</th>
<th>Name and Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>VCC, VCC</td>
<td>48, 36</td>
<td>8, 9, 10, 11, 61, 62</td>
<td></td>
<td>System Power: +5V Power Supply.</td>
</tr>
<tr>
<td>VSS, VSS</td>
<td>12, 24</td>
<td>26, 27, 41, 42, 43, 44</td>
<td></td>
<td>System Ground.</td>
</tr>
<tr>
<td>RESET</td>
<td>34</td>
<td>13</td>
<td>I TTL</td>
<td>RESET is an active HIGH internally synchronized signal, causing the 82586 to terminate present activity immediately. The signal must be HIGH for at least four clock cycles. The 82586 will execute RESET within ten system clock cycles starting from RESET HIGH. When RESET returns LOW, the 82586 waits for the first CA to begin the initialization sequence.</td>
</tr>
<tr>
<td>TxD</td>
<td>27</td>
<td>22</td>
<td>0 TTL</td>
<td>Transmitted Serial Data output signal. This signal is HIGH when not transmitting.</td>
</tr>
<tr>
<td>TxC</td>
<td>26</td>
<td>23</td>
<td>I *</td>
<td>Transmit Data Clock. This signal provides timing information to the internal serial logic, depending upon the mode of data transfer. For NRZ mode of operation, data is transferred to the TxD pin on the HIGH to LOW clock transition.</td>
</tr>
<tr>
<td>RxD</td>
<td>25</td>
<td>24</td>
<td>I TTL</td>
<td>Received Data Input Signal.</td>
</tr>
<tr>
<td>RxC</td>
<td>23</td>
<td>28</td>
<td>I *</td>
<td>Received Data Clock. This signal provides timing information to the internal shifting logic depending upon the mode of data transfer. For NRZ data, the state of the RxD pin is sampled on the HIGH to LOW clock transition.</td>
</tr>
</tbody>
</table>

*See D.C. Characteristics.
<table>
<thead>
<tr>
<th>Symbol</th>
<th>48 Pin DIP Pin No.</th>
<th>68 Pin PLCC Pin No.</th>
<th>Type Level</th>
<th>Name and Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>RTS</td>
<td>28</td>
<td>21</td>
<td>0 TTL</td>
<td>Request To Send signal. When LOW, notifies an external interface that the 82586 has data to transmit. It is forced HIGH after a Reset and while the Transmit Serial Unit is not sending data.</td>
</tr>
<tr>
<td>CTS</td>
<td>29</td>
<td>20</td>
<td>I TTL</td>
<td>Active LOW Clear To Send input enables the 82586 transmitter to actually send data. It is normally used as an interface handshake to RTS. This signal going inactive stops transmission. It is internally synchronized. If CTS goes inactive, meeting the setup time to TxC negative edge, transmission is stopped and RTS goes inactive within, at most, two TxC cycles.</td>
</tr>
<tr>
<td>CRS</td>
<td>31</td>
<td>18</td>
<td>I TTL</td>
<td>Active LOW Carrier Sense input used to notify the 82586 that there is traffic on the serial link. It is used only if the 82586 is configured for external Carrier Sense. When so configured, external circuitry is required for detecting serial link traffic. It is internally synchronized. To be accepted, the signal must stay active for at least two serial clock cycles.</td>
</tr>
<tr>
<td>CDT</td>
<td>30</td>
<td>19</td>
<td>I TTL</td>
<td>Active LOW Collision Detect input is used to notify the 82586 that a collision has occurred. It is used only if the 82586 is configured for external Collision Detect. External circuitry is required for detecting the collision. It is internally synchronized. To be accepted, the signal must stay active for at least two serial clock cycles. During transmission, the 82586 is able to recognize a collision one bit time after preamble transmission has begun.</td>
</tr>
<tr>
<td>INT</td>
<td>38</td>
<td>6</td>
<td>0 TTL</td>
<td>Active HIGH interrupt request signal.</td>
</tr>
<tr>
<td>CLK</td>
<td>32</td>
<td>15</td>
<td>I MOS</td>
<td>The system clock input from the 80186 or another symmetrical clock generator.</td>
</tr>
<tr>
<td>MN/MX</td>
<td>33</td>
<td>14</td>
<td>I TTL</td>
<td>When HIGH, MN/MX selects RD, WR, ALE, DEN, DT/R (Minimum Mode). When LOW, MN/MX selects A22, A23, READY, S0, S1 (Maximum Mode). Note: This pin should be static during 82586 operation.</td>
</tr>
<tr>
<td>AD0–AD15</td>
<td>6–11, 13–22</td>
<td>29–33, 36–40, 45, 48, 49, 50, 53, 54</td>
<td>I/O TTL</td>
<td>These lines form the time multiplexed memory address (t1) and data (t2, t3, tW, t4) bus. When operating with an 8-bit bus, the high byte will output the address only during T1. AD0–AD15 are floated after a RESET or when the bus is not acquired.</td>
</tr>
<tr>
<td>A16–A18</td>
<td>1, 3–5</td>
<td>55–57, 59, 63–65</td>
<td>0 TTL</td>
<td>These lines constitute 7 out of 8 most significant address bits for memory operation. They switch during t1 and stay valid during the entire memory cycle. The lines are floated after RESET or when the bus is not acquired. Address lines A22 and A23 are not available for use in minimum mode.</td>
</tr>
<tr>
<td>A20–A23</td>
<td>45–47</td>
<td>58</td>
<td>0 TTL</td>
<td>During t1 it forms line 19 of the memory address. During t2 through t4 it is used as a status indicating that this is a Master peripheral cycle, and is HIGH. Its timing is identical to that of AD0–AD15 during write operation.</td>
</tr>
</tbody>
</table>
Table 1. 82586 Pin Description (Continued)

<table>
<thead>
<tr>
<th>Symbol</th>
<th>48 Pin DIP Pin No.</th>
<th>68 Pin PLCC Pin No.</th>
<th>Type Level</th>
<th>Name and Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>HOLD</td>
<td>43</td>
<td>67</td>
<td>0 TTL</td>
<td>HOLD is an active HIGH signal used by the 82586 to request local bus mastership at the end of the current CPU bus transfer cycle, or at the end of the current DMA burst transfer cycle. In normal operation, HOLD goes inactive before HLDA. The 82586 can be forced off the bus by HLDA going inactive. In this case, HOLD goes inactive within four clock cycles in word mode and eight clock cycles in byte mode.</td>
</tr>
<tr>
<td>HLDA</td>
<td>42</td>
<td>68</td>
<td>1 TTL</td>
<td>HLDA is an active HIGH Hold Acknowledge signal indicating that the CPU has received the HOLD request and that bus control has been relinquished to the 82586. It is internally synchronized. After HOLD is detected as LOW, the processor drives HLDA LOW. Note, CONNECTING VCC TO HLDA IS NOT ALLOWED because it will cause a deadlock. Users wanting to give permanent bus access to the 82586 should connect HLDA with HOLD.</td>
</tr>
<tr>
<td>CA</td>
<td>35</td>
<td>12</td>
<td>1 TTL</td>
<td>The CA pin is a Channel Attention input used by the CPU to initiate the 82586 execution of memory resident Command Blocks. The CA signal is synchronized internally. The signal must be HIGH for at least one system clock period. It is latched internally on HIGH to LOW edge and then detected by the 82586.</td>
</tr>
<tr>
<td>BHE</td>
<td>44</td>
<td>66</td>
<td>0 TTL</td>
<td>The Bus High Enable signal (BHE) is used to enable data onto the most significant half of the data bus. Its timing is identical to that of A16–A23. With a 16-bit bus it is LOW and with an 8-bit bus it is HIGH. Note: after RESET, the 82586 is configured to 8-bit bus.</td>
</tr>
<tr>
<td>READY</td>
<td>39</td>
<td>5</td>
<td>1 TTL</td>
<td>This active HIGH signal is the acknowledgement from the addressed memory that the transfer cycle can be completed. While LOW, it causes wait states to be inserted. This signal must be externally synchronized with the system clock. The Ready signal internal to the 82586 is a logical OR between READY and SRDY/ARDY.</td>
</tr>
<tr>
<td>ARDY/SRDY</td>
<td>37</td>
<td>7</td>
<td>1 TTL</td>
<td>This active HIGH signal performs the same function as READY. If it is programmed at configure time to SRDY, it is identical to READY. If it is programmed to ARDY, the positive edge of the Ready signal is internally synchronized. Note, the negative edge must still meet setup and hold time specifications, when in ARDY mode. The ARDY signal must be active for at least one system clock HIGH period for proper strobing. The Ready signal internal to the 82586 is a logical OR between READY (in Maximum Mode only) and SRDY/ARDY. Note that following RESET, this pin assumes ARDY mode.</td>
</tr>
</tbody>
</table>
### Table 1. 82586 Pin Description (Continued)

<table>
<thead>
<tr>
<th>Symbol</th>
<th>48 Pin DIP Pin No.</th>
<th>68 Pin PLCC Pin No.</th>
<th>Type Level</th>
<th>Name and Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>S0, ST</td>
<td>40, 41</td>
<td>4, 3</td>
<td>0</td>
<td>TTL</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Maximum mode only. These status pins define the type of DMA transfer during the current memory cycle. They are encoded as follows:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>S1  S0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0   0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0   1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1   0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1   1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Status is active from the middle of t4 to the end of t2. They return to the passive state during t3 or during tW when READY or ARDY is HIGH. These signals can be used by the 8288 Bus Controller to generate all memory control and timing signals. Any change from the passive state, signals the 8288 to start the next t1 to t4 bus cycle. These pins are pulled HIGH and floated after a system RESET and when the bus is not acquired.</td>
</tr>
<tr>
<td>RD</td>
<td>46</td>
<td>64</td>
<td>0</td>
<td>TTL</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Used in minimum mode only. The read strobe indicates that the 82586 is performing a memory read cycle. RD is active LOW during t2, t3 and tW of any read cycle. This signal is pulled HIGH and floated after a RESET and when the bus is not acquired.</td>
</tr>
<tr>
<td>WR</td>
<td>45</td>
<td>65</td>
<td>0</td>
<td>TTL</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Used in minimum mode only. The write strobe indicates that the 82586 is performing a write memory cycle. WR is active LOW during t2, t3 and tW of any write cycle. It is pulled HIGH and floats after RESET and when the bus is not acquired.</td>
</tr>
<tr>
<td>ALE</td>
<td>39</td>
<td>5</td>
<td>0</td>
<td>TTL</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Used in minimum mode only. Address Latch Enable is provided by the 82586 to latch the address into the 8282/8283 address latch. It is a HIGH pulse, during t1 (‘clock low’) of any bus cycle. Note that ALE is never floated.</td>
</tr>
<tr>
<td>DEN</td>
<td>40</td>
<td>4</td>
<td>0</td>
<td>TTL</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Used in minimum mode only. Data ENable is provided as output enable for the 8286/8287 transceivers in a stand-alone (no 8288) system. DEN is active LOW during each memory access. For a read cycle, it is active from the middle of t2 until the beginning of t4. For a write cycle, it is active from the beginning of t2 until the middle of t4. It is pulled HIGH and floats after a system RESET or when the bus is not acquired.</td>
</tr>
<tr>
<td>DT/RI</td>
<td>41</td>
<td>3</td>
<td>0</td>
<td>TTL</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Used in minimum mode only. DT/RI is used in non-8288 systems using an 8286/8287 data bus transceiver. It controls the direction of data flow through the Transceiver. Logically, DT/RI is equivalent to ST. It becomes valid in the t4 preceding a bus cycle and remains valid until the final t4 of the cycle. This signal is pulled HIGH and floated after a RESET or when the bus is not acquired.</td>
</tr>
</tbody>
</table>

**NOTE:**

*8288 does not support 10 MHz operation.*
82586/HOST CPU INTERACTION

Communication between the 82586 and the host is carried out via shared memory. The 82586's on-chip DMA capability allows autonomous transfer of data blocks (buffers, frames) and relieves the CPU of byte transfer overhead. The 82586 is optimized to interface the iAPX 186, but due to the small number of hardware signals between the 82586 and the CPU, the 82586 can operate easily with other processors. The 82586/host interaction is explained separately in terms of the logical interface and the hardware bus interface.

The 82586 consists of two independent units: Command Unit (CU) and Receive Unit (RU). The CU executes commands from shared memory. The RU handles all activities related to frame reception. The CU and RU enable the 82586 to engage in the two types of activities simultaneously: the CU may be fetching and executing commands out of memory, and the RU may be storing received frames in memory. CPU intervention is only required after the CU executes a sequence of commands or the RU stores a sequence of frames.

The only hardware signals that connect the CPU and the 82586 are INTERRUPT and CHANNEL ATTENTION (see Figure 3). Interrupt is used by the 82586 to draw the CPU's attention to a change in the contents of the SCB. Channel Attention is used by the CPU to draw the 82586's attention.

82586 SYSTEM MEMORY STRUCTURE

The Shared Memory structure consists of four parts: Initialization Root, System Control Block (SCB), Command List, and Receive Frame Area (RFA) (see Figure 4).

The Initialization Root is at a predetermined location in the memory space, (0FFFFF6H), known to both the host CPU and the 82586. The root is accessed at initialization and points to the System Control Block.

The System Control Block (SCB) functions as a bidirectional mail drop between the host CPU, CU and RU. It is the central element through which the CPU and the 82586 exchange control and status information. The SCB consists of two parts, the first of which entails instructions from the CPU to the 82586. These include: control of the CU and RU (START, ABORT, SUSPEND, RESUME), a pointer to the list of commands for the CU, a pointer to the receive frame area, and a set of Interrupt acknowledge bits. The second entails status information keyed by the 82586 to the CPU, including: state of the CU and RU (e.g. IDLE, ACTIVE READY, SUSPENDED, NO RECEIVE RESOURCES), interrupts bits (command completed, frame received, CU not ready, RU not ready), and statistics (see Figure 4).

The Command List serves as a program for the CU. Individual commands are placed in memory units called a Command Block, or CB. CB's contain command specific parameters and command specific statuses. Specifically, these high level commands are called Action Commands (e.g. Transmit, Configure).

A specific command, Transmit, causes transmission of a frame by the 82586. The Transmit command block includes Destination Address, Length Field, and a pointer to a list of linked buffers that holds the frame to be constructed from several buffers scattered in memory. The Command Unit performs with-
out the CPU intervention, the DMA of each buffer and the prefetching of references to new buffers in parallel. The CPU is notified only after successful transmission or retransmission.

The Receive Frame Area is a list of Free Frame Descriptors (Descriptors not yet used) and a list of buffers prepared by the user. It is conceptually distinct from the Command List. Frames arrive without being solicited by the 82586. The 82586 must be prepared to receive them even if it is engaged in other activities and to store them in the Free Frame Area. The Receive Unit fills the buffers upon frame reception and reformats the Free Buffer List into received frame structures. The frame structure is virtually identical to the format of the frame to be transmitted. The first frame descriptor is referenced by SCB. A Frame Descriptor and the associated Buffer Descriptor wasted upon receiving a Bad Frame (CRC or Alignment errored, Receive DMA overrun errored, or Collision fragmented frame) are automatically reclaimed and returned to the Free Buffer List, unless the chip is configured to Save Bad Frames.

Receive buffer chaining (i.e. storing incoming frames in a linked list of buffers) improves memory utilization significantly. Without buffer chaining, the user must allocate consecutive blocks of the maximum frame size (1518 bytes in Ethernet) for each frame. Taking into account that a typical frame size may be about 100 bytes, this practice is very inefficient. With buffer chaining, the user can allocate small buffers and the 82586 uses only as many as needed.

In the past, the drawback of buffer chaining was the CPU processing overhead and the time involved in the buffer switching (especially at 10 Mb/s). The 82586 overcomes this drawback by performing buffer management on its own for both transmission and reception (completely transparent to the user).

The 82586 has a 22-bit memory address range in minimum mode and 24-bit memory address range in maximum mode. All memory structures, the System Control Block, Command List, Receive Descriptor List, and all buffer descriptors must reside within one 64K-byte memory segment. The Data Buffers can be located anywhere in the memory space.

Figure 4. 82586 Shared Memory Structure
TRANSMITTING FRAMES

The 82586 executes high level action commands from the Command List in external memory. Action commands are fetched and executed in parallel with the host CPU’s operation, thereby significantly improving system performance. The general action commands format is shown in Figure 5.

Figure 5. Action Command Format

Message transmission is accomplished by using the Transmit command. A single Transmit command contains, as part of the command-specific parameters, the destination address and length field for the transmitted frame along with a pointer to a buffer area in memory containing the data portion of the frame. (See Figure 15.) The data field is contained in a memory data structure consisting of a Buffer Descriptor (BD) and Data Buffer (or a linked list of buffer descriptors and buffers) as shown in Figure 6. The BD contains a Link Field which points to the next BD on the list and a 24-bit address pointing to the Data Buffer itself. The length of the Data Buffer is specified by the Actual Count field of the BD.

Using the BD’s and Data Buffers, multiple Data Buffers can be ‘chained’ together. Thus, a frame with a long Data Field can be transmitted using multiple (shorter) Data buffers chained together. This chaining technique allows the system designer to develop efficient buffer management policies.

The 82586 automatically generates the preamble (alternating 1’s and 0’s) and start frame delimiter, fetches the destination address and length field from the Transmit command, inserts its unique address as the source address, fetches the data field from buffers pointed to by the Transmit command, and computes and appends the CRC at the end of the frame. See Figure 7.

The 82586 can be configured to generate either the Ethernet or HDLC start and end frame delimiters. In the Ethernet mode, the start frame delimiter is 10101011 and the end frame delimiter indicated by the lack of a signal after transmitting the last bit of the frame check sequence field. When in the HDLC mode, the 82586 will generate the 01111110 ‘flag’ for the start and end frame delimiters and perform the standard ‘bit stuffing/stripping’. In addition, the 82586 will optionally pad frames that are shorter than the specified minimum frame length by appending the appropriate number of flags to the end of the frame.

In the event of a collision (or collisions), the 82586 manages the entire jam, random wait and retry process, reinitializing DMA pointers without CPU intervention. Multiple frames can be sent by linking the appropriate number of Transmit commands together. This is particularly useful when transmitting a message that is larger than the maximum frame size (1518 bytes for Ethernet).

RECEIVING FRAMES

In order to minimize CPU overhead, the 82586 is designed to receive frames without CPU supervision. The host CPU first sets aside an adequate

<table>
<thead>
<tr>
<th>PREAMBLE</th>
<th>START FRAME DELIMITER</th>
<th>DEST ADDR</th>
<th>SOURCE ADDR</th>
<th>LENGTH FIELD</th>
<th>DATA FIELD</th>
<th>FRAME CHECK SEQUENCE</th>
<th>END FRAME DELIMITER</th>
</tr>
</thead>
</table>

Figure 7. Frame Format
amount of receive buffer space and then enables
the 82586's Receive Unit. Once enabled, the RU
'watches' for any of its frames which it automatically
stores in the Receive Frame Area (RFA). The RFA
consists of a Receive Descriptor List (RDL) and a list
of free buffers called the Free Buffer List (FBL) as
shown in Figure 8. The individual Receive Frame
Descriptors that make up the RDL are used by the
82586 to store the destination and source address,
length field and status of each frame that is re­
ceived. (Figure 9.)

The 82586, once enabled, checks each passing
frame for an address match. The 82586 will recog­
nize its own unique address, one or more multicast
addresses or the broadcast address. If a match oc­
curs, it stores the destination and source address
and length field in the next available RFD. It then
begins filling the next free Data Buffer on the FBL
(which is pointed to by the current RFD) with the
data portion of the incoming frame. As one DB is
filled, the 82586 automatically fetches the next DB
on the FBL until the entire frame is received. This
buffer chaining technique is particularly memory ef­
ficient because it allows the system designer to set
aside buffers that fit a frame size that may be much
shorter than the maximum allowable frame.

Once the entire frame is received without error, the
82586 performs the following housekeeping tasks:
• Updates the Actual Count field of the last Buffer
Descriptor used to hold the frame just received
with the number of bytes stored in its associated
Data Buffer.
• Fetches the address of the next free Receive Frame Descriptor.
• Writes the address of the next free Buffer Descriptor into the next free Receive Frame Descriptor.
• Posts a ‘Frame Received’ interrupt status bit in the SCB.
• Interrupts the CPU.

In the event of a frame error, such as a CRC error, the 82586 automatically reinitializes its DMA pointers and recovers any data buffers containing the bad frame. As long as Receive Frame Descriptors and data buffers are available, the 82586 will continue to receive frames without further CPU help.

82586 NETWORK MANAGEMENT AND DIAGNOSTIC FUNCTIONS

The behavior of data communication networks is typically very complex due to their distributed and asynchronous nature. It is particularly difficult to pinpoint a failure when it occurs. The 82586 was designed in anticipation of these problems and includes a set of features for improving reliability and testability.

The 82586 reports on the following events after each frame transmitted:
• Transmission successful.
• Transmission unsuccessful; lost Carrier Sense.
• Transmission unsuccessful; lost Clear-to-Send.
• Transmission unsuccessful; DMA underrun because the system bus did not keep up with the transmission.
• Transmission unsuccessful; number of collisions exceeded the maximum allowed.

The 82586 checks each incoming frame and reports on the following errors, (if configured to ‘Save Bad Frame’):
• CRC error: incorrect CRC in a well aligned frame.
• Alignment error: incorrect CRC in a misaligned frame.
• Frame too short: the frame is shorter than the configured value for minimum frame length.
• Overrun: the frame was not completely placed in memory because the system bus did not keep up with incoming data.
• Out of buffers: no memory resources to store the frame, so part of the frame was discarded.

NETWORK PLANNING AND MAINTENANCE

To perform proper planning, operation, and maintenance of a communication network, the network management entity must accumulate information on network behavior. The 82586 provides a rich set of network-wide diagnostics that can serve as the basis for a network management entity.

Network Activity information is provided in the status of each frame transmitted. The activity indicators are:
• Number of collisions: number of collisions the 82586 experienced in attempting to transmit this frame.
• Deferred transmission: indicates if the 82586 had to defer to traffic on the link during the first transmission attempt.

Statistics registers are updated after each received frame that passes the address filtering, and is longer than the Minimum Frame Length configuration parameter.
• CRC errors: number of frames that experienced a CRC error and were properly aligned.
• Alignment errors: number of frames that experienced a CRC error and were misaligned.
• No-resources: number of correct frames lost due to lack of memory resources.
• Overrun errors: number of frame sequences lost due to DMA overrun.

The 82586 can be configured to Promiscuous Mode. In this mode it captures all frames transmitted on the Network without checking the Destination Address. This is useful in implementing a monitoring station to capture all frames for analysis.

The 82586 is capable of determining if there is a short or open circuit anywhere in the Network using the built in Time Domain Reflectometer (TDR) mechanism.

STATION DIAGNOSTICS

The chip can be configured to External Loopback. The transmitter to receiver interconnection can be placed anywhere between the 82586 and the link to locate faults, for example: the 82586 output pins, the Serial Interface Unit, the Transceiver cable, or in the Transceiver.
The 82586 has a mechanism recognizing the transceiver ‘heart beat’ signal for verifying the correct operation of the Transceiver's collision detection circuitry.

82586 SELF TESTING

The 82586 can be configured to Internal Loopback. It disconnects itself from the Serial Interface Unit, and any frame transmitted is received immediately. The 82586 connects the Transmit Data to the Receive Data signal and the Transmit Clock to the Receive Clock.

The Dump Command causes the chip to write over 100 bytes of its internal registers to memory.

The Diagnose command checks the exponential Backoff random number generator internal to the 82586.

CONTROLLING THE 82586

The CPU controls operation of the 82586’s Command Unit (CU) and Receive Unit (RU) of the 82586 via the System Control Block.

THE COMMAND UNIT (CU)

The Command Unit is the logical unit that executes Action Commands from a list of commands very similar to a CPU program. A Command Block (CB) is associated with each Action Command.

The CU can be modeled as a logical machine that takes, at any given time, one of the following states:

- **IDLE**—CU is not executing a command and is not associated with a CB on the list. This is the initial state.
- **SUSPENDED**—CU is not executing a command but (different from IDLE) is associated with a CB on the list.
- **ACTIVE**—CU is currently executing an Action Command, and points to its CB.

The CPU may affect the CU operation in two ways: issuing a CU control Command or setting bits in the COMMAND word of the Action Command.

THE RECEIVE UNIT (RU)

The Receive Unit is the logical unit that receives frames and stores them in memory.

The RU is modeled as a logical machine that takes, at any given time, one of the following states:

- **IDLE**—RU has no memory resources and is discarding incoming frames. This is the initial RU state.
- **NO-RESOURCES**—RU has no memory resources and is discarding incoming frames. This state differs from the IDLE state in that RU accumulates statistics on the number of frames it had to discard.
- **SUSPENDED**—RU has free memory resources to store incoming frames but discard them anyway.

```
15 ODD BYTE

<table>
<thead>
<tr>
<th>STAT</th>
<th>CUS</th>
<th>RUS</th>
<th>RUC</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACK</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

CBL OFFSET

RFA OFFSET

CRCERRS

ALNERRS

RSCERRS

OVRNERRS

SCB (STATUS)

SCB + 2 (COMMAND)

SCB + 4

SCB + 6

SCB + 8

SCB + 10

SCB + 12

SCB + 14

Figure 10. System Control Block (SCB) Format
```
• READY—RU has free memory resources and stores incoming frames.

The CPU may affect RU operation in three ways: issuing an RU Control Command, setting bits in Frame Descriptor, FD, COMMAND word of the frame currently being received, or setting EL bit of Buffer Descriptor, BD, of the buffer currently being filled.

SYSTEM CONTROL BLOCK (SCB)

The System Control Block is the communication mailbox between the 82586 and the host CPU. The SCB format is shown in Figure 10.

The host CPU issues Control Commands to the 82586 via the SCB. These commands may appear at any time during routine operation, as determined by the host CPU. After the required Control Command is setup, the CPU sends a CA signal to the 82586.

SCB is also used by the 82586 to return status information to the host CPU. After inserting the required status bits into SCB, the 82586 issues an Interrupt to the CPU.

The format is as follows:

STATUS word: Indicates the status of the 82586. This word is modified only by the 82586. Defined bits are:

<table>
<thead>
<tr>
<th>Bit 15</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>CX</td>
<td>A command in the CBL having its 'I' (interrupt) bit set has been executed.</td>
</tr>
<tr>
<td>FR</td>
<td>A frame has been received.</td>
</tr>
<tr>
<td>CNR</td>
<td>The Command Unit left the Active state.</td>
</tr>
<tr>
<td>RNR</td>
<td>The Receive Unit left the Ready state.</td>
</tr>
<tr>
<td>CUS</td>
<td>(3 bits) this field contains the status of the Command Unit. Valid values are:</td>
</tr>
<tr>
<td></td>
<td>0 — Idle</td>
</tr>
<tr>
<td></td>
<td>1 — Suspended</td>
</tr>
<tr>
<td></td>
<td>2 — Active</td>
</tr>
<tr>
<td></td>
<td>3—7 — Not Used</td>
</tr>
<tr>
<td>RUS</td>
<td>(3 bits) this field contains the status of the Receive Unit. Valid values are:</td>
</tr>
<tr>
<td></td>
<td>0 — Idle</td>
</tr>
<tr>
<td></td>
<td>1 — Suspended</td>
</tr>
<tr>
<td></td>
<td>2 — No Resources</td>
</tr>
<tr>
<td></td>
<td>3 — Not Used</td>
</tr>
<tr>
<td></td>
<td>4 — Ready</td>
</tr>
<tr>
<td></td>
<td>5—7 — Not Used</td>
</tr>
</tbody>
</table>

COMMAND word: Specifies the action to be performed as a result of the CA. This word is set by the CPU and cleared by the 82586. Defined bits are:

<table>
<thead>
<tr>
<th>Bit 15</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACK-CX</td>
<td>Acknowledges the command executed event.</td>
</tr>
<tr>
<td>ACK-FR</td>
<td>Acknowledges the frame received event.</td>
</tr>
<tr>
<td>ACK-CNA</td>
<td>Acknowledges that the Command Unit became not ready.</td>
</tr>
<tr>
<td>ACK-RNR</td>
<td>Acknowledges that the Receive Unit became not ready.</td>
</tr>
<tr>
<td>CUC</td>
<td>(3 bits) this field contains the command to the Command Unit. Valid values are:</td>
</tr>
<tr>
<td></td>
<td>0 — NOP (doesn't affect current state of the unit).</td>
</tr>
<tr>
<td></td>
<td>1 — Start execution of the first command on the CBL. If a command is in execution, then complete it before starting the new CBL. The beginning of the CBL is in CBL OFFSET.</td>
</tr>
<tr>
<td></td>
<td>2 — Resume the operation of the command unit by executing the next command. This operation assumes that the command unit has been previously suspended.</td>
</tr>
<tr>
<td></td>
<td>3 — Suspend execution of commands on CBL after current command is complete.</td>
</tr>
<tr>
<td></td>
<td>4 — Abort execution of commands on CBL after current command is complete.</td>
</tr>
<tr>
<td></td>
<td>5—7 — Reserved, illegal for use.</td>
</tr>
<tr>
<td>RUC</td>
<td>(3 bits) This field contains the command to the receive unit. Valid values are:</td>
</tr>
<tr>
<td></td>
<td>0 — NCP (does not alter current state of unit).</td>
</tr>
<tr>
<td></td>
<td>1 — Start reception of frames. If a frame is being received, then complete reception before starting. The beginning of the RFA is contained in the RFA OFFSET.</td>
</tr>
<tr>
<td></td>
<td>2 — Resume frame receiving (only when in suspended state.)</td>
</tr>
<tr>
<td></td>
<td>3 — Suspend frame receiving. If a frame is being received, then complete its reception before suspending.</td>
</tr>
<tr>
<td></td>
<td>4 — Abort receiver operation immediately.</td>
</tr>
<tr>
<td></td>
<td>5—7 — Reserved, illegal for use.</td>
</tr>
<tr>
<td>RESET</td>
<td>Reset chip (logically the same as hardware RESET).</td>
</tr>
</tbody>
</table>
CBL-OFFSET:

Gives the 16-bit offset address of the first command (Action Command) in the command list to be executed following CU-START. Thus, the 82586 reads this word only if the CUC field contained a CU-START Control Command.

RFA-OFFSET:

Points to the first Receive Frame Descriptor in the Receive Frame Area.

CRCERRS:

CRC Errors - contains the number of properly aligned frames received with a CRC error.

ALNERRS:

Alignment Errors - contains the number of misaligned frames received with a CRC error.

RSCERRS:

Resource Errors - records the number of correct incoming frames discarded due to lack of memory resources (buffer space or received frame descriptors).

OVRNERRS:

Overrun Errors - counts the number of received frame sequences lost because the memory bus was not available in time to transfer them.

ACTION COMMANDS

The 82586 executes a 'program' that is made up of action commands in the Command List. As shown in Figure 5, each command contains the command field, status and control fields, link to the next action command in the CL, and any command-specific parameters. This command format is called the Command Block.

The 82586 has a repertoire of 8 commands:

- NOP
- Setup Individual Address
- Configure
- Setup Multicast Address
- Transmit
- TDR
- Diagnose
- Dump

NOP

This command results in no action by the 82586, except as performed in normal command processing. It is present to aid in Command List manipulation.

NOP command includes the following fields:

STATUS word (written by 82586):

<table>
<thead>
<tr>
<th>Bit 15</th>
<th>Bit 14</th>
<th>Bit 13</th>
<th>Bit 12</th>
</tr>
</thead>
<tbody>
<tr>
<td>C</td>
<td>B</td>
<td>OK</td>
<td>CMD</td>
</tr>
</tbody>
</table>

- C (Bit 15) - Command Completed
- B (Bit 14) - Busy Executing Command
- OK (Bit 13) - Error Free Completion
- CMD (Bits 0–2) - NOP

COMMAND word:

<table>
<thead>
<tr>
<th>Bit 15</th>
<th>Bit 14</th>
<th>Bit 13</th>
<th>Bits 0–2</th>
</tr>
</thead>
<tbody>
<tr>
<td>EL</td>
<td>S</td>
<td>I</td>
<td>CMD</td>
</tr>
</tbody>
</table>

- EL (Bit 15) - End of Command List
- S (Bit 14) - Suspend After Completion
- I (Bit 13) - Interrupt After Completion
- CMD (Bits 0–2) - NOP = 0

LINK OFFSET: Address of next Command Block

Figure 11. The NOP Command Block
IA-SETUP

This command loads the 82586 with the Individual Address. This address is used by the 82586 for recognition of Destination Address during reception and insertion of Source Address during transmission.

The IA-SETUP command includes the following fields:

![Diagram of IA-SETUP Command Block]

STATUS word (written by 82586):

<table>
<thead>
<tr>
<th>Bit</th>
<th>Field Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>C</td>
<td>Command Completed</td>
</tr>
<tr>
<td>B</td>
<td>Busy Executing Command</td>
</tr>
<tr>
<td>OK</td>
<td>Error Free Completion</td>
</tr>
<tr>
<td>A</td>
<td>Command Aborted</td>
</tr>
</tbody>
</table>

COMMAND word:

<table>
<thead>
<tr>
<th>Bit</th>
<th>Field Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>EL</td>
<td>End of Command List</td>
</tr>
<tr>
<td>S</td>
<td>Suspend After Completion</td>
</tr>
<tr>
<td>I</td>
<td>Interrupt After Completion</td>
</tr>
<tr>
<td>CMD</td>
<td>IA-SETUP = 1</td>
</tr>
</tbody>
</table>

LINK OFFSET: Address of next Command Block

INDIVIDUAL ADDRESS: Individual Address parameter

The least significant bit of the Individual Address parameter must be zero for IEEE 802.3/Ethernet. However, no enforcement of 0 is provided by the 82586. Thus, an Individual Address with least significant bit 1, is possible.

CONFIGURE

The CONFIGURE command is used to update the 82586 operating parameters.

STATUS word (written by 82586):

<table>
<thead>
<tr>
<th>Bit</th>
<th>Field Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>C</td>
<td>Command Completed</td>
</tr>
<tr>
<td>B</td>
<td>Busy Executing Command</td>
</tr>
<tr>
<td>OK</td>
<td>Error Free Completion</td>
</tr>
<tr>
<td>A</td>
<td>Command Aborted</td>
</tr>
</tbody>
</table>

COMMAND word:

<table>
<thead>
<tr>
<th>Bit</th>
<th>Field Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>EL</td>
<td>End of Command List</td>
</tr>
<tr>
<td>S</td>
<td>Suspend After Completion</td>
</tr>
<tr>
<td>I</td>
<td>Interrupt After Completion</td>
</tr>
<tr>
<td>CMD</td>
<td>Configure = 2</td>
</tr>
</tbody>
</table>

LINK OFFSET: Address of next Command Block

Byte 6–7:

<table>
<thead>
<tr>
<th>Bit</th>
<th>Field Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BYTE CNT</td>
<td>Byte Count, Number of bytes including this one, holding the parameters to be configured. A number smaller than 4 is interpreted as 4. A number greater than 12 is interpreted as 12.</td>
</tr>
<tr>
<td>Byte 8–9:</td>
<td>Description</td>
</tr>
<tr>
<td>---------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td><strong>SRDY/ARDY</strong></td>
<td>(Bit 6) 0 – SRDY/ARDY pin operates as ARDY (internal synchronization).</td>
</tr>
<tr>
<td></td>
<td>1 – SRDY/ARDY pin operates as SRDY (external synchronization).</td>
</tr>
<tr>
<td><strong>SAV-BF</strong></td>
<td>(Bit 7) 0 – Received bad frames are not saved in memory.</td>
</tr>
<tr>
<td></td>
<td>1 – Received bad frames are saved in memory.</td>
</tr>
<tr>
<td><strong>ADD-LEN</strong></td>
<td>(Bits 8–10) Number of address bytes. NOTE: 7 is interpreted as 0.</td>
</tr>
<tr>
<td><strong>AL-LOC</strong></td>
<td>(Bit 11) 0 – Address and Length Fields separated from data and associated</td>
</tr>
<tr>
<td></td>
<td>with Transmit Command Block or Receive Frame Descriptor. For transmitted</td>
</tr>
<tr>
<td></td>
<td>Frame, Source Address is inserted by the 82586.</td>
</tr>
</tbody>
</table>

**Figure 13. The CONFIGURE Command Block**

<table>
<thead>
<tr>
<th>Byte 10–11:</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>PREAM-LEN</strong></td>
<td>(Bits 12–13) Address and Length Fields are part of the Transmit/Receive data</td>
</tr>
<tr>
<td></td>
<td>buffers, including Source Address (which is not inserted by the 82586).</td>
</tr>
<tr>
<td><strong>INT-LPBCK</strong></td>
<td>(Bit 14) Preamble Length including Beginning of Frame indicator:</td>
</tr>
<tr>
<td></td>
<td>00 - 2 bytes</td>
</tr>
<tr>
<td></td>
<td>01 - 4 bytes</td>
</tr>
<tr>
<td></td>
<td>10 - 8 bytes</td>
</tr>
<tr>
<td></td>
<td>11 - 16 bytes</td>
</tr>
<tr>
<td><strong>EXT-LPBCK</strong></td>
<td>(Bit 15) Internal Loopback</td>
</tr>
<tr>
<td></td>
<td>External Loopback.</td>
</tr>
<tr>
<td></td>
<td>NOTE: Bits 14 and 15 configured to 1, cause Internal Loopback.</td>
</tr>
<tr>
<td><strong>LIN-PRIOR</strong></td>
<td>(Bits 0–2) Linear Priority</td>
</tr>
<tr>
<td><strong>ACR</strong></td>
<td>(Bits 4–6) Accelerated Contention Resolution (Exponential Priority)</td>
</tr>
<tr>
<td><strong>BOF-MET</strong></td>
<td>(Bit 7) Exponential Backoff Method</td>
</tr>
<tr>
<td></td>
<td>0 - IEEE 802.3/Ethernet</td>
</tr>
<tr>
<td></td>
<td>1 - Alternate Method</td>
</tr>
</tbody>
</table>
**CONFIGURATION DEFAULTS**

The default values of the configuration parameters are compatible with the IEEE 802.3/Ethernet Standards. RESET configures the 82586 according to the defaults shown in Table 2.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Default Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Preamble Length (Bytes)</td>
<td>8</td>
</tr>
<tr>
<td>Address Length (Bytes)</td>
<td>6</td>
</tr>
<tr>
<td>Broadcast Disable</td>
<td>0</td>
</tr>
<tr>
<td>CRC-16/CRC-32</td>
<td>0</td>
</tr>
<tr>
<td>No CRC Insertion</td>
<td>0</td>
</tr>
<tr>
<td>Bitstuffing/EOC</td>
<td>0</td>
</tr>
<tr>
<td>Padding</td>
<td>0</td>
</tr>
<tr>
<td>Min-Frame-Length (Bytes)</td>
<td>64</td>
</tr>
<tr>
<td>Interframe Spacing (Bits)</td>
<td>96</td>
</tr>
<tr>
<td>Slot Time (Bits)</td>
<td>512</td>
</tr>
<tr>
<td>Number of Retries</td>
<td>15</td>
</tr>
<tr>
<td>Linear Priority</td>
<td>0</td>
</tr>
<tr>
<td>Accelerated Contention Resolution</td>
<td>0</td>
</tr>
<tr>
<td>Exponential Backoff Method</td>
<td>0</td>
</tr>
<tr>
<td>Manchester/NRZ</td>
<td>0</td>
</tr>
<tr>
<td>Internal CRS</td>
<td>0</td>
</tr>
<tr>
<td>Internal COT</td>
<td>0</td>
</tr>
<tr>
<td>Crs Filter</td>
<td>0</td>
</tr>
<tr>
<td>CDT Filter</td>
<td>0</td>
</tr>
<tr>
<td>Transmit On No CRS</td>
<td>0</td>
</tr>
<tr>
<td>FIFO THRESHOLD</td>
<td>8</td>
</tr>
<tr>
<td>SRDY/ARDY</td>
<td>0</td>
</tr>
<tr>
<td>Save Bad Frame</td>
<td>0</td>
</tr>
<tr>
<td>Address/Length Location</td>
<td>0</td>
</tr>
<tr>
<td>INT Loopback</td>
<td>0</td>
</tr>
<tr>
<td>EXT Loopback</td>
<td>0</td>
</tr>
<tr>
<td>Promiscuous Mode</td>
<td>0</td>
</tr>
</tbody>
</table>
MC-SETUP

This command sets up the 82586 with a set of Multicast Addresses. Subsequently, incoming frames with Destination Addresses from this set are accepted.

The MC-SETUP command includes the following fields:

**STATUS word (written by 82586):**

<table>
<thead>
<tr>
<th>Bit 15</th>
<th>Bit 14</th>
<th>Bit 13</th>
<th>Bit 12</th>
</tr>
</thead>
<tbody>
<tr>
<td>C</td>
<td>B</td>
<td>OK</td>
<td>A</td>
</tr>
</tbody>
</table>

- **C** (Bit 15) • Command Completed
- **B** (Bit 14) • Busy Executing Command
- **OK** (Bit 13) • Error Free Completion
- **A** (Bit 12) • Command Aborted

**COMMAND word:**

<table>
<thead>
<tr>
<th>Bit 15</th>
<th>Bit 14</th>
<th>Bit 13</th>
<th>Bits 0–2</th>
</tr>
</thead>
<tbody>
<tr>
<td>EL</td>
<td>S</td>
<td>I</td>
<td>CMD</td>
</tr>
</tbody>
</table>

- **EL** (Bit 15) • End of Command List
- **S** (Bit 14) • Suspend After Completion
- **I** (Bit 13) • Interrupt After Completion
- **CMD** (Bits 0–2) • MC-SETUP = 3

**LINK OFFSET:** Address of next Command Block

**MC-CNT:** A 14-bit field indicating the number of bytes in the MC-LIST field. MC-CNT is truncated to the nearest multiple of Address Length (in bytes).

Issuing a MC-SETUP command with MC-CNT = 0 disables reception of any incoming frame with a Multicast Address.

**MC-LIST:** A list of Multicast Addresses to be accepted by the 82586. Note that the most significant byte of an address is followed immediately by the least significant byte of the next address. Note also that the least significant bit of each Multicast Address in the set must be a one.

The Transmit-Byte-Machine maintains a 64-bit HASH table used for checking Multicast Addresses during reception.

An incoming frame is accepted if it has a Destination Address whose least significant bit is a one, and after hashing points to a bit in the HASH table whose value is one. The hash function is selecting bits 2 to 7 of the CRC register. RESET causes the HASH table to become all zeros.

After the Transmit-Byte-Machine reads a MC-SETUP command from TX-FIFO, it clears the HASH table and reads the bytes in groups whose length is determined by the ADDRESS length. Each group is hashed using CRC logic and the bit in the HASH table to which bits 2–7 of the CRC register point is set to one. A group that is not complete has no effect on the HASH table. Transmit-Byte-Machine notifies CU after completion.
TRANSMIT

The TRANSMIT command causes transmission (and if necessary retransmission) of a frame.

TRANSMIT CB includes the following fields:

**STATUS word (written by 82586):**

<table>
<thead>
<tr>
<th>Bit</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>C (Bit 15)</td>
<td>Command Completed</td>
</tr>
<tr>
<td>B (Bit 14)</td>
<td>Busy Executing Command</td>
</tr>
<tr>
<td>OK (Bit 13)</td>
<td>Error Free Completion</td>
</tr>
<tr>
<td>A (Bit 12)</td>
<td>Command Aborted</td>
</tr>
<tr>
<td>S10 (Bit 10)</td>
<td>No Carrier Sense signal during transmission (between beginning of Destination Address and end of Frame Check Sequence).</td>
</tr>
<tr>
<td>S9 (Bit 9)</td>
<td>Transmission unsuccessful (stopped) due to loss of Clear-to-Send signal.</td>
</tr>
<tr>
<td>S8 (Bit 8)</td>
<td>Transmission unsuccessful (stopped) due to DMA underrun, (i.e. data not supplied from the system for transmission).</td>
</tr>
<tr>
<td>S7 (Bit 7)</td>
<td>Transmission had to Defer to traffic on the link.</td>
</tr>
</tbody>
</table>

**MAX-COLL** (Bits 3-0) • Number of Collisions experienced by this frame. S5 = 1 and MAX-COLL = 0 indicates that there were 16 collisions.

**COMMAND word:**

<table>
<thead>
<tr>
<th>Bit</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>EL (Bit 15)</td>
<td>End of Command List</td>
</tr>
<tr>
<td>S (Bit 14)</td>
<td>Suspend After Completion</td>
</tr>
<tr>
<td>I (Bit 13)</td>
<td>Interrupt After Completion</td>
</tr>
<tr>
<td>CMD (Bits 0-2)</td>
<td>TRANSMIT = 4</td>
</tr>
</tbody>
</table>

**LINK OFFSET:** Address of next Command Block

**TBD OFFSET:** Address of list of buffers holding the information field. TBD-OFFSET = 0FFFFH indicates that there is no information field.

**DESTINATION ADDRESS:** Destination Address of the frame.

**LENGTH FIELD:** Length field of the frame.
STATUS word:

- **EOF**
  - Indicates that this is the Buffer Descriptor of the last buffer of this frame's Information Field.
- **ACT-COUNT** (Bits 0–13)
  - Actual number of data bytes in buffer (can be even or odd).

**NEXT BD OFFSET**: points to next Buffer Descriptor in list. If EOF is set, this field is meaningless.

**BUFFER ADDRESS**: 24-bit absolute address of buffer.

**TIME DOMAIN REFLECTOMETER - TDR**

This command performs a Time Domain Reflectometer test on the serial link. By performing the command, the user is able to identify shorts or opens and their location. Along with transmission of 'All Ones,' the 82586 triggers an internal timer. The timer measures the time elapsed from transmission start until 'echo' is obtained. 'Echo' is indicated by Collision Detect going active or Carrier Sense signal drop.

TDR command includes the following fields:

**STATUS word (written by 82586):**

<table>
<thead>
<tr>
<th>Field</th>
<th>Bit 15</th>
<th>Bit 14</th>
<th>Bit 13</th>
</tr>
</thead>
<tbody>
<tr>
<td>C</td>
<td>Command Completed</td>
<td></td>
<td></td>
</tr>
<tr>
<td>B</td>
<td>Busy Executing Command</td>
<td></td>
<td></td>
</tr>
<tr>
<td>OK</td>
<td>Error Free Completion</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**COMMAND word:**

<table>
<thead>
<tr>
<th>Field</th>
<th>Bit 15</th>
<th>Bit 14</th>
<th>Bit 13</th>
</tr>
</thead>
<tbody>
<tr>
<td>EL</td>
<td>End of Command List</td>
<td></td>
<td></td>
</tr>
<tr>
<td>S</td>
<td>Suspend After Completion</td>
<td></td>
<td></td>
</tr>
<tr>
<td>I</td>
<td>Interrupt After Completion</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CMD</td>
<td>TDR = 5</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

![Figure 16. The Transmit Buffer Description](image16.png)

![Figure 17. The TDR Command Block](image17.png)
LINK OFFSET: Address of next Command Block

RESULT word:

<table>
<thead>
<tr>
<th>Field</th>
<th>Bit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>LNK-OK</td>
<td>15</td>
<td>• No Link Problem Identified</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Transceiver Cable Problem identified (valid only in the case of a</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Transceiver that does not return Carrier Sense during transmission).</td>
</tr>
<tr>
<td>XCVR-PRB</td>
<td>14</td>
<td>• Open on the link identified (valid only in the case of a Transceiver that</td>
</tr>
<tr>
<td></td>
<td></td>
<td>returns Carrier Sense during transmission).</td>
</tr>
<tr>
<td>ET-OPN</td>
<td>13</td>
<td>• Short on the link identified (valid only in the case of a Transceiver that</td>
</tr>
<tr>
<td></td>
<td></td>
<td>returns Carrier Sense during transmission).</td>
</tr>
<tr>
<td>ET-SRT</td>
<td>12</td>
<td>• Specifying the distance to a problem on the link (if one exists) in transmit</td>
</tr>
<tr>
<td>TIME</td>
<td>0-10</td>
<td>Clock cycles.</td>
</tr>
</tbody>
</table>

DUMP

This command causes the contents of over a hundred bytes of internal registers to be placed in memory. It is supplied as a self diagnostic tool, as well as to supply registers of interest to the user.

DUMP command includes the following fields:

STATUS word (written by 82586):

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>Command Completed</td>
</tr>
<tr>
<td>14</td>
<td>Busy Executing Command</td>
</tr>
<tr>
<td>13</td>
<td>Error Free Completion</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

COMMAND word:

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>End of Command List</td>
</tr>
<tr>
<td>14</td>
<td>Suspend After Completion</td>
</tr>
<tr>
<td>13</td>
<td>Interrupt After Completion</td>
</tr>
<tr>
<td>0-2</td>
<td>DUMP = 6</td>
</tr>
</tbody>
</table>

BUFFER OFFSET: This word specifies the offset portion of the memory address which points to the top of the buffer allocated for the dumped registers contents. The length of the buffer is 170 bytes.

DUMP AREA FORMAT

Figure 18 shows the format of the DUMP area. The fields are as follows:

Bytes 00H to 0AH: These bytes correspond to the 82586 CONFIGURE command field.

Bytes 0CH to 11H: The Individual Address Register content. IARO is the Individual Address least significant byte.

Bytes 12H to 13H: Status word of last command block (only bits 0-13).

![Figure 18. The DUMP Command Block](231246-18)
Bytes 14H to 17H: Content of the Transmit CRC generator. TXCRCRO is the least significant byte. The contents are dependent on the activity before the DUMP command:

After RESET - 'All Ones.'

After successful transmission - 'All Zeros'.

After MC-SETUP command - Generated CRC value of the last MC address, on MC-LIST.

After unsuccessful transmission, depends on where it stopped.

**NOTE:**

For 16-bit CRC only TXCRCRO and TXCRCR1 are valid.
Bytes 18H to 1BH: Contents of Receive CRC Checker. RXCRCRO is the least significant byte. The contents are dependent on the activity performed before the DUMP command:

After RESET - 'All Ones.'

After good frame reception—
1. For CRC-CCITT - 010DFH
2. For CRC-Autodin-II - C704DD7BH

After Bad Frame reception - corresponds to the received information.

After reception attempt, i.e. unsuccessful check for address match, corresponds to the CRC performed on the frame address.

NOTE:
Any frame on the serial link modifies this register contents.

Bytes 1CH to 21H: Temporary Registers.

Bytes 22H to 23H: Receive Status Register. Bits 6, 7, 8, 10, 11 and 13 assume the same meaning as corresponding bits in the Receive Frame Descriptor Status field.

Bytes 24H to 2BH: HASH TABLE.

Bytes 2CH to 2DH: Status bits of the last time TDR command that was performed.

NXT-RB-SIZE: Let N be the last buffer of the last received frame, then NXT-RB-SIZE is the number of bytes of available in the N + 1 buffer. EL - The EL bit of the Receive Buffer Descriptor.

NXT-RB-ADR: Let N be the last Receive Buffer used, then NXT-RB-ADR is the BUFFER-ADDRESS field in the N + 1 Receive-Buffer Descriptor, i.e. the pointer to the N + 1 Receive Buffer.

CUR-RB-SIZE: The number of bytes in the last buffer of the last received frame. EL - The EL bit of the last buffer in the last received frame.

LA-RBD-ADR: Look Ahead Buffer Descriptor, i.e. the pointer to N + 2 Receiver Buffer Descriptor.

NXT-RBD-ADR: Next Receive Buffer Descriptor Address. Similar to LA-RBD-ADR but points to N + 1 Receive Buffer Descriptor.

CUR-RBD-ADR: Current Receive Buffer Descriptor Address. Similar to LA-RBD-ADR, but point to Nth Receive Buffer Descriptor.

CUR-RB-EBC: Current Receive Buffer Empty Byte Count. Let N be the currently used Receive Buffer. Then CUR-RB-EBC indicates the empty part of the buffer, i.e. the ACT-COUNT of buffer N is given by the difference between its SIZE and the CUR-RB-EBC.

NXT-FD-ADR: Next Frame Descriptor Address. Define N as the last Receive Frame Descriptor with bits C = 1 and B = 0, then NXT-FD-ADR is the address of N + 2 Receive Frame Descriptor (with B = C = 0) and is equal to the LINK-ADDRESS field in N + 1 Receive Frame Descriptor.

CUR-FD-ADR: Current Frame Descriptor Address. Similar to next NXT-FD-ADR but refers to N + 1 Receive Frame Descriptor (with B = 1, C = 0).

Bytes 54H to 55H: Temporary register.

NXT-TB-CNT: Next Transmit Buffer Count. Let N be the last transmitted buffer of the TRANSMIT command executed recently, the NXT-TB-CNT is the ACT-COUNT field in the Nth Transmit Buffer Descriptor. EOF - Corresponds to the EOF bit of the Nth Transmit Buffer Descriptor. EOF = 1 indicates that the last buffer accessed by the 82586 during Transmit was the last Transmit Buffer in the data buffer chain associated with the Transmit Command.

BUF-ADR: Buffer Address. The BUF-PTR field in the DUMP-STATUS Command Block.

NXT-TB-ADR: Next Transmit Buffer Address Low. Let N be the last Transmit Buffer in the transmit buffer chain of the TRANSMIT Command performed recently, then NXT-TB-ADR-L are the two least significant bytes of the Nth buffer address.

LA-TB-ADR: Look Ahead Transmit Buffer Descriptor Address. Let N be the last Transmit Buffer in the transmit buffer chain of the TRANSMIT Command performed recently, then LA-TBD-ADR is the NEXT-BD-ADDRESS field of the Nth Buffer Descriptor.

NXT-TBD-ADR: Next Transmit Buffer Descriptor Address. Similar in function to LA-TBD-ADR but related to Transmit Buffer Descriptor N-1. Actually, it is the address of Transmit Buffer Descriptor N.

Bytes 60H, 61H: This is a copy of the 2nd word in the DUMP-STATUS command presently executing.

NXT-CB-ADR: Next Command Block Address. The LINK-ADDRESS field in the DUMP Command Block presently executing. Points to the next command.

CUR-CB-ADR: Current Command Block Address. The address of the DUMP Command Block currently executing.
SCB-ADR: Offset of the System Control Block (SCB).

Bytes 7EH, 7FH:
RU-SUS-RQ (Bit 4) - Receive Unit Suspend Request.

Bytes 80H, 81H:
CU-SUS-RQ (Bit 4) - Command Unit Suspend Request.
END-OF-CBL (Bit 5) - End of Command Block List. If "1" indicates that DUMP-STATUS is the last command in the command chain.
ABRT-IN-PROG (Bit 6) - Command Unit Abort Request.
RU-SUS-FD (Bit 12) - Receive Unit Suspend Frame Descriptor Bit. Assume N is the Receive Frame Descriptor used recently, then RU-SUS-FD is equivalent to the S bit of N + 1 Receive Frame Descriptor.

Bytes 82H, 83H:
RU-SUS (Bit 4) - Receive Unit in SUSPENDED state.
RU-NRSRC (Bit 5) - Receive Unit in NO RESOURCES state.
RU-RDY (Bit 6) - Receive Unit in READY state.
RU-IDL (Bit 7) - Receive Unit in IDLE state.
RNR (Bit 12) - RNR Interrupt in Service bit.
CNA (Bit 13) - CNA Interrupt in Service bit.
FR (Bit 14) - FR Interrupt in Service bit.
CX (Bit 15) - CX Interrupt in Service bit.

Bytes 90H to 93H:
BUF-ADR-PTR - Buffer pointer is the absolute address of the bytes following the DUMP Command block.

Bytes 94H to 95H:
RCV-DMA-BC - Receive DMA Byte Count. This field contains number of bytes to be transferred during the next Receive DMA operation. The value depends on AL-LOCation configuration bit.

1. If AL-LOCation = 0 then RCV-DMA-BC = (2 times ADDR-LEN plus 2) if the next Receive Frame Descriptor has already been fetched.
2. If AL-LOCation = 1 then it contains the size of the next Receive Buffer.

BR + BUF-PTR + 96H - Sum of Base Address plus BUF-PTR field and 96H.

RCV-DMA-ADR - Receive DMA absolute Address. This is the next RCV-DMA start address. The value depends on AL-LOCation configuration bit.
1. If AL-LOCation = 0, then RCV-DMA-ADR is the Destination Address field located in the next Receive Frame Descriptor.
2. If AL-LOCation = 1, then RCV-DMA-ADR is the next Receive Data Buffer Address.

The following nomenclature has been used in the DUMP table:

<table>
<thead>
<tr>
<th></th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>The 82586 writes zero in this location.</td>
</tr>
<tr>
<td>1</td>
<td>The 82586 writes one in this location.</td>
</tr>
<tr>
<td>X</td>
<td>The 82586 writes zero or one in this location.</td>
</tr>
<tr>
<td>///</td>
<td>The 82586 copies this location from the corresponding position in the memory structure.</td>
</tr>
</tbody>
</table>

DIAGNOSE

The DIAGNOSE Command triggers an internal self test procedure of backoff related registers and counters.

The DIAGNOSE command includes the following:

STATUS word (written by 82586):

<table>
<thead>
<tr>
<th>C</th>
<th>(Bit 15)</th>
<th>Command Completed</th>
</tr>
</thead>
<tbody>
<tr>
<td>B</td>
<td>(Bit 14)</td>
<td>Busy Executing Command</td>
</tr>
<tr>
<td>OK</td>
<td>(Bit 13)</td>
<td>Error Free Completion</td>
</tr>
<tr>
<td>FAIL</td>
<td>(Bit 11)</td>
<td>Indicates that the Self Test Procedure Failed</td>
</tr>
</tbody>
</table>

COMMAND word:

| EL  | (Bit 15) | End of Command List |
| S   | (Bit 14) | Suspend After Completion |
| I   | (Bit 13) | Interrupt After Completion |
| CMD | (Bits 0–2) | DIAGNOSE = 7 |

LINK OFFSET: Address of next Command Block.
**RECEIVE FRAME AREA (RFA)**

The Receive Frame Area, RFA, is prepared by the host CPU, data is placed into the RFA by the 82586 as frames are received. RFA consists of a list of Receive Frame Descriptors (FD), each of which is associated with a frame. RFA-OFFSET field of SCB points to the first FD of the chain; the last FD is identified by the End-of-Listing flag (EL). See Figure 21.

**FRAME DESCRIPTOR (FD) FORMAT**

The FD includes the following fields:

**STATUS word (set by the 82586):**

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>C (Bit 15)</td>
<td>• Completed Storing Frame.</td>
</tr>
<tr>
<td>B (Bit 14)</td>
<td>• FD was Consumed by RU.</td>
</tr>
</tbody>
</table>
**Figure 22. The Frame Descriptor (FD) Format**

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OK</td>
<td>Frame received successfully. If this bit is set, then all others will be reset; if it is reset, then the other bits will indicate the nature of the error.</td>
</tr>
<tr>
<td>S11</td>
<td>Received Frame Experienced CRC Error.</td>
</tr>
<tr>
<td>S10</td>
<td>Received Frame Experienced an Alignment Error.</td>
</tr>
<tr>
<td>S9</td>
<td>RU ran out of resources during reception of this frame.</td>
</tr>
<tr>
<td>S8</td>
<td>RCV-DMA Overrun.</td>
</tr>
<tr>
<td>S7</td>
<td>Received frame had fewer bits than configured Minimum Frame Length.</td>
</tr>
<tr>
<td>S6</td>
<td>No EOF flag detected (only when configured to Bitstuffing).</td>
</tr>
</tbody>
</table>

**COMMAND word:**

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>EL</td>
<td>Last FD in the List.</td>
</tr>
<tr>
<td>S</td>
<td>RU should be suspended after receiving this frame.</td>
</tr>
</tbody>
</table>

**LINK OFFSET:** Address of next FD in list.

**RBD-OFFSET:** (initially prepared by the CPU and later may be updated by 82586): Address of the first RBD that represents the Information Field. RBD-OFFSET = 0FFFFH means there is no Information Field.

**DESTINATION ADDRESS (written by 82586):** Contains Destination Address of received frame. The length in bytes, it is determined by the Address Length configuration parameter.

**SOURCE ADDRESS (written by 82586):** Contains Source Address of received frame. Its length is the same as DESTINATION ADDRESS.

**LENGTH FIELD (written by 82586):** Contains the 2 byte Length or Type Field of received frame.

**RECEIVE BUFFER DESCRIPTOR FORMAT**

The Receive Buffer Descriptor (RBD) holds information about a buffer; size and location, and the means for forming a chain of RBDs, (forward pointer and end-of-frame indication).

The Buffer Descriptor contains the following fields.
Figure 23. The Receive Buffer Descriptor (RBD) Format

**STATUS word (written by the 82586).**

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>EOF</td>
<td>(Bit 15) - Last buffer in received frame.</td>
</tr>
<tr>
<td>F</td>
<td>(Bit 14) - ACT COUNT field is valid.</td>
</tr>
<tr>
<td>ACT COUNT</td>
<td>(Bits 0–13) - Number of bytes in the buffer that are actually occupied.</td>
</tr>
</tbody>
</table>

**BUFFER ADDRESS:** 24-bit absolute address of buffer.

**EL/SIZE:**

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>EL</td>
<td>(Bit 15) - Last BD in list.</td>
</tr>
<tr>
<td>SIZE</td>
<td>(Bits 0–13) - Number of bytes the buffer is capable of holding.</td>
</tr>
</tbody>
</table>

**NEXT RBD OFFSET:** Address of next BD in list of BD's.
ABSOLUTE MAXIMUM RATINGS*  
Ambient Temperature Under Bias ........... 0°C to 70°C  
Storage Temperature ...................... −65°C to 150°C  
Voltage on Any Pin with Respect to Ground ........... −1.0V to +7V  
Power Dissipation ................. 3.0 Watts

*Notice: Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions above those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

D.C. CHARACTERISTICS  
$T_A = 0°C$ to $70°C$, $T_C = 0°C$ to $105°C$, $V_{CC} = 5V ±10\%$, CLK has MOS levels (See $V_{MIL}$, $V_{MIH}$, $V_{MOL}$, $V_{MOH}$). $T_xC$ and $R_xC$ have 82C501 compatible levels ($V_{MIL}$, $V_{TIH}$, $V_{RIH}$). All other signals have TTL levels (see $V_{IL}$, $V_{IH}$, $V_{OL}$, $O_H$).

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Parameter</th>
<th>Min</th>
<th>Max</th>
<th>Units</th>
<th>Test Conditions</th>
</tr>
</thead>
<tbody>
<tr>
<td>$V_{IL}$</td>
<td>Input Low Voltage (TTL)</td>
<td>$-0.5$</td>
<td>$+0.8$</td>
<td>V</td>
<td></td>
</tr>
<tr>
<td>$V_{IH}$</td>
<td>Input High Voltage (TTL)</td>
<td>$2.0$</td>
<td>$V_{CC} + 0.5$</td>
<td>V</td>
<td></td>
</tr>
<tr>
<td>$V_{OL}$</td>
<td>Output Low Voltage (TTL)</td>
<td>$0.45$</td>
<td>V</td>
<td>$I_{OL} = 2.5$ mA</td>
<td></td>
</tr>
<tr>
<td>$V_{OH}$</td>
<td>Output High Voltage (TTL)</td>
<td>$2.4$</td>
<td>V</td>
<td>$I_{OH} = 400$ μA</td>
<td></td>
</tr>
<tr>
<td>$V_{MIL}$</td>
<td>Input Low Voltage (MOS)</td>
<td>$-0.5$</td>
<td>$0.6$</td>
<td>V</td>
<td></td>
</tr>
<tr>
<td>$V_{MIH}$</td>
<td>Input High Voltage (MOS)</td>
<td>$3.9$</td>
<td>$V_{CC} + 0.5$</td>
<td>V</td>
<td></td>
</tr>
<tr>
<td>$V_{TIH}$</td>
<td>Input High Voltage ($T_xC$)</td>
<td>$3.3$</td>
<td>$V_{CC} + 0.5$</td>
<td>V</td>
<td></td>
</tr>
<tr>
<td>$V_{RIH}$</td>
<td>Input High Voltage ($R_xC$)</td>
<td>$3.0$</td>
<td>$V_{CC} + 0.5$</td>
<td>V</td>
<td></td>
</tr>
<tr>
<td>$V_{MOL}$</td>
<td>Output Low Voltage (MOS)</td>
<td>$0.45$</td>
<td>V</td>
<td>$I_{OL} = 2.5$ mA</td>
<td></td>
</tr>
<tr>
<td>$V_{MOH}$</td>
<td>Output High Voltage (MOS)</td>
<td>$V_{CC} - 0.5$</td>
<td>V</td>
<td>$I_{OH} = 400$ μA</td>
<td></td>
</tr>
<tr>
<td>$I_{LI}$</td>
<td>Input Leakage Current</td>
<td>$±10$</td>
<td>μA</td>
<td>$0 \leq V_{IN} \leq V_{CC}$</td>
<td></td>
</tr>
<tr>
<td>$I_{LO}$</td>
<td>Output Leakage Current</td>
<td>$±10$</td>
<td>μA</td>
<td>$0.45 \leq V_{OUT} \leq V_{CC}$</td>
<td></td>
</tr>
<tr>
<td>$C_{IN}$</td>
<td>Capacitance of Input Buffer</td>
<td>$10$</td>
<td>pF</td>
<td>$F_C = 1$ MHz</td>
<td></td>
</tr>
<tr>
<td>$C_{OUT}$</td>
<td>Capacitance of Output Buffer</td>
<td>$20$</td>
<td>pF</td>
<td>$F_C = 1$ MHz</td>
<td></td>
</tr>
<tr>
<td>$I_{CC}$</td>
<td>Power Supply Current</td>
<td>$550$</td>
<td>mA</td>
<td>$T_A = 0°C$</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>$450$</td>
<td></td>
<td>$T_A = 70°C$</td>
<td></td>
</tr>
</tbody>
</table>
SYSTEM INTERFACE A.C. TIMING CHARACTERISTICS

$T_A = 0°C$ to $70°C$, $T_C = 0°C$ to $105°C$, $V_{CC} = 5V \pm 10\%$. Figures 24 and 25 define how the measurements should be done.

**INPUT AND OUTPUT WAVEFORMS FOR A.C. TESTS**

AC Testing Inputs are Driven at 2.4V for a Logic 1 and 0.45 for a Logic 0. Timing measurements are made at 1.5V for both a Logic 1 and 0.

**Figure 24. TTL Input/Output Voltage Levels for Timing Measurements**

MOS I/O measurements are taken at 0.1 and 0.9 of the voltage swing.

**Figure 25. System Clock CMOS Input Voltage Levels for Timing Measurements**
## INPUT TIMING REQUIREMENTS*

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Parameter</th>
<th>82586-6 (6 MHz)</th>
<th>82586-6 (8 MHz)</th>
<th>82586-10 (10 MHz)</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Min</td>
<td>Max</td>
<td>Min</td>
<td>Max</td>
</tr>
<tr>
<td>T1</td>
<td>CLK Cycle Period</td>
<td>166</td>
<td>2000</td>
<td>125</td>
<td>2000</td>
</tr>
<tr>
<td>T2</td>
<td>CLK Low Time at 1.5V</td>
<td>73</td>
<td>1000</td>
<td>55</td>
<td>1000</td>
</tr>
<tr>
<td>T3</td>
<td>CLK Low Time at 0.9V</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T4</td>
<td>CLK High Time at 1.5V</td>
<td>73</td>
<td>55</td>
<td>44</td>
<td></td>
</tr>
<tr>
<td>T5</td>
<td>CLK High Time at 3.6V</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T6</td>
<td>CLK Rise Time</td>
<td>15</td>
<td>15</td>
<td>12</td>
<td></td>
</tr>
<tr>
<td>T7</td>
<td>CLK Fall Time</td>
<td>15</td>
<td>15</td>
<td>12</td>
<td></td>
</tr>
<tr>
<td>T8</td>
<td>Data in Setup Time</td>
<td>20</td>
<td>20</td>
<td>15</td>
<td></td>
</tr>
<tr>
<td>T9</td>
<td>Data in Hold Time</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>T10</td>
<td>Async RDY Active Setup Time</td>
<td>20</td>
<td>20</td>
<td>15</td>
<td></td>
</tr>
<tr>
<td>T11</td>
<td>Async RDY Inactive Setup Time</td>
<td>35</td>
<td>35</td>
<td>25</td>
<td></td>
</tr>
<tr>
<td>T12</td>
<td>Async RDY Hold Time</td>
<td>15</td>
<td>15</td>
<td>15</td>
<td></td>
</tr>
<tr>
<td>T13</td>
<td>Synchronous Ready/Active Setup</td>
<td>35</td>
<td>35</td>
<td>35</td>
<td></td>
</tr>
<tr>
<td>T14</td>
<td>Synchronous Ready Hold Time</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>T15</td>
<td>HLDA Setup Time</td>
<td>20</td>
<td>20</td>
<td>20</td>
<td></td>
</tr>
<tr>
<td>T16</td>
<td>HLDA Hold Time</td>
<td>10</td>
<td>10</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>T17</td>
<td>Reset Setup Time</td>
<td>20</td>
<td>20</td>
<td>20</td>
<td></td>
</tr>
<tr>
<td>T18</td>
<td>Reset Hold Time</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>T19</td>
<td>CA Pulse Width</td>
<td>1</td>
<td>T1</td>
<td>1</td>
<td>T1</td>
</tr>
<tr>
<td>T20</td>
<td>CA Setup Time</td>
<td>20</td>
<td>20</td>
<td>20</td>
<td></td>
</tr>
<tr>
<td>T21</td>
<td>CA Hold Time</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td></td>
</tr>
</tbody>
</table>

## OUTPUT TIMINGS**

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Parameter</th>
<th>Min</th>
<th>Max</th>
<th>Min</th>
<th>Max</th>
<th>Min</th>
<th>Max</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>T22</td>
<td>DT/R Valid Delay</td>
<td>0</td>
<td>60</td>
<td>0</td>
<td>60</td>
<td>0</td>
<td>44</td>
<td></td>
</tr>
<tr>
<td>T23</td>
<td>WR, DEN Active Delay</td>
<td>0</td>
<td>70</td>
<td>0</td>
<td>70</td>
<td>0</td>
<td>56</td>
<td></td>
</tr>
<tr>
<td>T24</td>
<td>WR, DEN Inactive Delay</td>
<td>10</td>
<td>65</td>
<td>10</td>
<td>65</td>
<td>10</td>
<td>45</td>
<td></td>
</tr>
<tr>
<td>T25</td>
<td>Int. Active Delay</td>
<td>0</td>
<td>85</td>
<td>0</td>
<td>85</td>
<td>0</td>
<td>70</td>
<td>Note 4</td>
</tr>
<tr>
<td>T26</td>
<td>Int. Inactive Delay</td>
<td>0</td>
<td>85</td>
<td>0</td>
<td>85</td>
<td>0</td>
<td>70</td>
<td>Note 4</td>
</tr>
<tr>
<td>T27</td>
<td>Hold Active Delay</td>
<td>0</td>
<td>85</td>
<td>0</td>
<td>85</td>
<td>0</td>
<td>70</td>
<td>Note 4</td>
</tr>
<tr>
<td>T28</td>
<td>Hold Inactive Delay</td>
<td>0</td>
<td>85</td>
<td>0</td>
<td>85</td>
<td>0</td>
<td>70</td>
<td>Note 4</td>
</tr>
<tr>
<td>T29</td>
<td>Address Valid Delay</td>
<td>0</td>
<td>55</td>
<td>0</td>
<td>55</td>
<td>0</td>
<td>50</td>
<td>Note 4</td>
</tr>
<tr>
<td>T30</td>
<td>Address Float Delay</td>
<td>0</td>
<td>50</td>
<td>0</td>
<td>50</td>
<td>12</td>
<td>50</td>
<td></td>
</tr>
<tr>
<td>T31</td>
<td>Data Valid Delay</td>
<td>0</td>
<td>55</td>
<td>0</td>
<td>55</td>
<td>0</td>
<td>50</td>
<td>Note 7</td>
</tr>
<tr>
<td>T32</td>
<td>Data Hold Time</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T33</td>
<td>Status Active Delay</td>
<td>10</td>
<td>60</td>
<td>10</td>
<td>60</td>
<td>10</td>
<td>45</td>
<td></td>
</tr>
</tbody>
</table>
### OUTPUT TIMINGS** (Continued)

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Parameter</th>
<th>82582-6 (6 MHz)</th>
<th>82586 (8 MHz)</th>
<th>82586-10 (10 MHz)</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Min</td>
<td>Max</td>
<td>Min</td>
<td>Max</td>
</tr>
<tr>
<td>T34</td>
<td>Status Inactive Delay</td>
<td>10</td>
<td>70</td>
<td>10</td>
<td>70</td>
</tr>
<tr>
<td>T35</td>
<td>ALE Active Delay</td>
<td>0</td>
<td>45</td>
<td>0</td>
<td>45</td>
</tr>
<tr>
<td>T36</td>
<td>ALE Inactive Delay</td>
<td>0</td>
<td>45</td>
<td>0</td>
<td>45</td>
</tr>
<tr>
<td>T37</td>
<td>ALE Width</td>
<td>T2-10</td>
<td>T2-10</td>
<td>T2-10</td>
<td>Note 5</td>
</tr>
<tr>
<td>T38</td>
<td>Address Valid to ALE Low</td>
<td>T2-40</td>
<td>T2-30</td>
<td>T2-25</td>
<td></td>
</tr>
<tr>
<td>T39</td>
<td>Address Hold to ALE Inactive</td>
<td>T4-10</td>
<td>T4-10</td>
<td>T4-10</td>
<td></td>
</tr>
<tr>
<td>T40</td>
<td>RD Active Delay</td>
<td>10</td>
<td>95</td>
<td>10</td>
<td>95</td>
</tr>
<tr>
<td>T41</td>
<td>RD Inactive Delay</td>
<td>10</td>
<td>70</td>
<td>10</td>
<td>70</td>
</tr>
<tr>
<td>T42</td>
<td>RD Width</td>
<td>2T1-50</td>
<td>2T1-50</td>
<td>2T1-46</td>
<td></td>
</tr>
<tr>
<td>T43</td>
<td>Address Float to RD Active</td>
<td>10</td>
<td>10</td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>T44</td>
<td>RD Inactive to Address Active</td>
<td>T1-40</td>
<td>T1-40</td>
<td>T1-34</td>
<td></td>
</tr>
<tr>
<td>T45</td>
<td>WR Width</td>
<td>2T1-40</td>
<td>2T1-40</td>
<td>2T1-34</td>
<td></td>
</tr>
<tr>
<td>T46</td>
<td>Data Hold After WR</td>
<td>T2-25</td>
<td>T2-25</td>
<td>T2-25</td>
<td></td>
</tr>
<tr>
<td>T47</td>
<td>Control Inactive After Reset</td>
<td>0</td>
<td>60</td>
<td>0</td>
<td>60</td>
</tr>
</tbody>
</table>

*All units are in ns.

**CL on all outputs is 20–200 pF unless otherwise specified.

#### NOTES:
1. 1.0V to 3.5V
2. 3.5V to 1.0V
3. To guarantee recognition at next clock
4. CL = 50 pF
5. CL = 100 pF
6. Affects:
   - MIN MODE: RD, WR, DT/R, DEN
   - MAX MODE: S6, S1
7. High address lines (A16–A24, BHE) become valid one clock before T1 only on first memory cycle after the 82586 acquired the bus.
8. S6, S0 go inactive just prior to T4.

---

**Figure 26. INT Output Timing**

**Figure 27. CA Input Timing**

**Figure 28. RESET Timing**
Figure 29. ARDY and SRDY Timings Relative to CLK

Figure 30. HOLD/HLDA Timing Relative to CLK
Figure 31. Read Cycle Timing

Figure 32. Write Cycle Timing
### Serial Interface A.C. Timing Characteristic

**CLOCK SPECIFICATION**

- Applies for TxC, RxC for NRZ:
  - \( f_{\text{min}} = 100 \, \text{kHz} \pm 100 \, \text{ppm} \)
  - \( f_{\text{max}} = 10 \, \text{MHz} \pm 100 \, \text{ppm} \)

- for Manchester, symmetry is needed:
  - \( f_{\text{min}} = 500 \, \text{kHz} \pm 100 \, \text{ppm} \)
  - \( f_{\text{max}} = 10 \, \text{MHz} \pm 100 \, \text{ppm} \)
  - \( T_{51}, T_{52} = \frac{1}{2f} \pm 5\% \)

### A.C. Characteristics

**Transmit and Receive Timing Parameter Specification**

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Parameter</th>
<th>Min</th>
<th>Max</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>TRANSMIT CLOCK PARAMETERS</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T48</td>
<td>TxC Cycle</td>
<td>100</td>
<td>1000</td>
<td>Notes 14, 2</td>
</tr>
<tr>
<td>T48</td>
<td>TxC Cycle</td>
<td>100</td>
<td>1000</td>
<td>Notes 14, 3</td>
</tr>
<tr>
<td>T49</td>
<td>TxC Rise Time</td>
<td>5</td>
<td></td>
<td>Note 14</td>
</tr>
<tr>
<td>T50</td>
<td>TxC Fall Time</td>
<td>5</td>
<td></td>
<td>Note 14</td>
</tr>
<tr>
<td>T51</td>
<td>TxC High Time @ 3.0V</td>
<td>40</td>
<td>1000</td>
<td>Note 14</td>
</tr>
<tr>
<td>T52</td>
<td>TxC Low Time @0.9V</td>
<td>40</td>
<td></td>
<td>Notes 14, 4</td>
</tr>
<tr>
<td><strong>TRANSMIT DATA PARAMETERS</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T53</td>
<td>TxD Rise Time</td>
<td>10</td>
<td></td>
<td>Notes 5, 13</td>
</tr>
<tr>
<td>T54</td>
<td>TxD Fall Time</td>
<td>10</td>
<td></td>
<td>Notes 5, 13</td>
</tr>
<tr>
<td>T55</td>
<td>TxD Transition-Transition</td>
<td>Min (T51, T52) − 7</td>
<td></td>
<td>Notes 2, 5</td>
</tr>
<tr>
<td>T56</td>
<td>TxC Low to TxD Valid</td>
<td>40</td>
<td></td>
<td>Notes 3, 5</td>
</tr>
<tr>
<td>T57</td>
<td>TxC Low to TxD Transition</td>
<td>40</td>
<td></td>
<td>Notes 2, 5</td>
</tr>
<tr>
<td>T58</td>
<td>TxC High to TxD Transition</td>
<td>40</td>
<td></td>
<td>Notes 2, 5</td>
</tr>
<tr>
<td>T59</td>
<td>TxC Low to TxD High at the Transmission End</td>
<td>40</td>
<td></td>
<td>Note 5</td>
</tr>
<tr>
<td><strong>REQUEST TO SEND/CLEAR TO SEND PARAMETERS</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T60</td>
<td>TxC Low to RTS Low. Time to Activate RTS</td>
<td>40</td>
<td></td>
<td>Note 6</td>
</tr>
<tr>
<td>T61</td>
<td>CTS Valid to TxC Low. CTS Setup Time</td>
<td>45</td>
<td></td>
<td></td>
</tr>
<tr>
<td>T62</td>
<td>TxC Low to CTS Invalid. CTS Hold Time</td>
<td>20</td>
<td></td>
<td>Note 7</td>
</tr>
<tr>
<td>T63</td>
<td>TxC Low to RTS High, time to Deactivate RTS</td>
<td>40</td>
<td></td>
<td>Note 6</td>
</tr>
<tr>
<td><strong>RECEIVE CLOCK PARAMETERS</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T64</td>
<td>RxC Clock Cycle</td>
<td>100</td>
<td></td>
<td>Notes 15, 3</td>
</tr>
<tr>
<td>T65</td>
<td>RxC Rise Time</td>
<td>5</td>
<td></td>
<td>Note 15</td>
</tr>
<tr>
<td>T66</td>
<td>RxC Fall Time</td>
<td>5</td>
<td></td>
<td>Note 15</td>
</tr>
<tr>
<td>T67</td>
<td>RxC High Time @ 2.7V</td>
<td>36</td>
<td>1000</td>
<td>Note 15</td>
</tr>
<tr>
<td>T68</td>
<td>RxC Low Time @0.9V</td>
<td>40</td>
<td></td>
<td>Note 15</td>
</tr>
</tbody>
</table>

*All units are in ns.*
A.C. CHARACTERISTICS (Continued)

TRANSMIT AND RECEIVE TIMING PARAMETER SPECIFICATION* (Continued)

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Parameter</th>
<th>Min</th>
<th>Max</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>RECEIVE DATA PARAMETERS</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T69</td>
<td>RxD Setup Time</td>
<td>30</td>
<td></td>
<td>Note 1</td>
</tr>
<tr>
<td>T70</td>
<td>RxD Hold Time</td>
<td>30</td>
<td></td>
<td>Note 1</td>
</tr>
<tr>
<td>T71</td>
<td>RxD Rise Time</td>
<td></td>
<td>10</td>
<td>Note 1</td>
</tr>
<tr>
<td>T72</td>
<td>RxD Fall Time</td>
<td></td>
<td>10</td>
<td>Note 1</td>
</tr>
<tr>
<td></td>
<td>CARRIER SENSE/COLLISION DETECT PARAMETERS</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T73</td>
<td>CDT Valid to TxC High Ext. Collision Detect Setup Time</td>
<td>30</td>
<td></td>
<td>Note 12</td>
</tr>
<tr>
<td>T74</td>
<td>TxC High to CDT Inactive. CDT Hold Time</td>
<td>20</td>
<td></td>
<td>Note 12</td>
</tr>
<tr>
<td>T75</td>
<td>CDT Low to Jamming Start</td>
<td></td>
<td></td>
<td>Note 8</td>
</tr>
<tr>
<td>T76</td>
<td>CRS Valid to TxC High Ext. Carrier Sense Setup Time</td>
<td>30</td>
<td></td>
<td>Note 12</td>
</tr>
<tr>
<td>T77</td>
<td>TxC High to CRS Inactive. CRS Hold Time</td>
<td>20</td>
<td></td>
<td>Note 12</td>
</tr>
<tr>
<td>T78</td>
<td>CRS Low to Jamming Start</td>
<td></td>
<td></td>
<td>Note 9</td>
</tr>
<tr>
<td>T79</td>
<td>Jamming Period</td>
<td></td>
<td></td>
<td>Note 10</td>
</tr>
<tr>
<td>T80</td>
<td>CRS Inactive Setup Time to RxC High End of Receive Frame</td>
<td>60</td>
<td></td>
<td></td>
</tr>
<tr>
<td>T81</td>
<td>CRS Active Hold Time from RxC High</td>
<td></td>
<td>3</td>
<td></td>
</tr>
<tr>
<td></td>
<td>INTERFRAME SPACING PARAMETER</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T82</td>
<td>Inter Frame Delay</td>
<td></td>
<td></td>
<td>Note 11</td>
</tr>
</tbody>
</table>

*All units are in ns.

NOTES:
1. TTL levels
2. Manchester only
3. NRZ only
4. Manchester requires 50% duty cycle
5. 1 TTL load + 50 pF
6. 1 TTL load + 100 pF
7. Abnormal end of transmission. CTS expires before RTS
8. Programmable value:
   \[ T75 = NCDF \times T48 + (12.5 \text{ to } 23.5) \times T48 \text{ if collision occurs after preamble} \]
   NCDF—The collision detection filter configuration value
9. Programmable value:
   \[ T78 = NCSF \times T48 + (12.5 \text{ to } 23.5) \times T48 \]
   NCSF—The carrier sense filter configuration value
   TBD is a function of internal/external carrier sense bit
10. \[ T79 = 32 \times T48 \]
11. Programmable value:
    \[ T82 = NIFS \times T48 \]
    NIFS—the IFS configuration value
12. To guarantee recognition on the next clock
13. Applies to TTL levels
14. 82C501 compatible levels, see Figure 34
15. 82C501 compatible levels, see Figure 35
A.C. TIMING CHARACTERISTICS

Input and Output Waveforms for AC Tests

AC testing inputs are driven at 2.4V for a Logic 1 and 0.45 for a Logic 0. Timing measurements are made at 1.5V for both a Logic 1 and 0.

Figure 33. TTL Input/Output Voltage Levels for Timing Measurements

Figure 34. TxC Input Voltage Levels for Timing Measurements

Figure 35. RxC Input Voltage Levels for Timing Measurements
Figure 36. Transmit and Control and Data Timing

Figure 37. RxD Timing Relative to RxC

Figure 38. CRS Timing Relative to RxC
82C501AD ETHERNET SERIAL INTERFACE

- CHMOS Replacement for Intel 82C501, 82501 or SEEQ 8023A
- Conforms to IEEE 802.3 10BASE5 (Ethernet) and 10BASE2 (Cheapernet) Specifications
- Direct Interface to the Intel LAN Controller and the Attachment Unit Interface (Transceiver) Cable
- 10 Mbps Operation
- Manchester Encoding/Decoding and Receive Clock Recovery

The 82C501AD Ethernet Serial Interface (ESI) chip is designed to work directly with the Intel LAN Controller in IEEE 802.3 (10BASE5 and 10BASE2), 10 Mbps, Local Area Network applications. The major functions of the 82C501AD are to generate the 10 MHz transmit clock for the Intel LAN Controller, perform Manchester encoding/decoding of the transmitted/received frames, and provide the electrical interface to the Ethernet transceiver cable (AUI). Diagnostic loopback control enables the 82C501AD to route the signal to be transmitted from the Intel LAN Controller through its Manchester encoding and decoding circuitry and back to the Intel LAN Controller. The combined loopback capabilities of the Intel LAN Controller and 82C501AD result in highly effective fault detection and isolation through sequential testing of the communications interface. A (defeatable) on-chip watchdog timer circuit prevents the station from locking up in a continuous transmit mode. The 82C501AD is pin compatible with the 82C501 and functionally compatible with the 82501 and SEEQ 8023A.

Figure 1. 82C501AD Functional Block Diagram

Figure 2. Pin Configuration
### Table 1. Pin Description

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Pin No.</th>
<th>Type</th>
<th>Name and Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>ENETV1</td>
<td>1</td>
<td>I</td>
<td><strong>ETHERNET VERSION 1.0</strong>: An active low, MOS-level input. When ENETV1 is asserted, the TRMT/ITRMT pair remains at high differential voltage at the end of transmission. This operation is compatible with the Ethernet Version 1.0 specification. If the ENETV1 pin is left floating, an internal pull-up resistor biases the input inactive high. When ENETV1 is high, the TRMT/ITRMT differential voltage gradually approaches 0V at the end of transmission.</td>
</tr>
<tr>
<td>NOOR</td>
<td>2</td>
<td>I</td>
<td><strong>CRS 'OR'</strong>: An active low, MOS-level input. When NOOR is low, only the presence of a valid signal on the RCV/RCV pair will force CRS active. If the NOOR pin is floating, an internal pull-up resistor biases the input inactive high. When NOOR is in active high, either the presence of a valid signal on CLSN/CLSN or on RCV/RCV will force CRS active.</td>
</tr>
<tr>
<td>LPBK/ WDTD</td>
<td>3</td>
<td>I</td>
<td><strong>LOOPBACK/WATCHDOG TIMER DISABLE</strong>: An active low, TTL-level control signal that enables the loopback mode. In loopback mode serial data on the TXD input is routed through the 82C501AD internal circuits and back to the RXD output without driving the TRMT/ITRMT output pair to the transceiver cable. During loopback CDT is asserted at the end of each transmission to simulate the SQE test. <strong>WATCHDOG TIMER DISABLE</strong>: An input voltage of 10V to 16V through a 1 kΩ resistor will disable the on-chip watchdog timer.</td>
</tr>
<tr>
<td>RCV</td>
<td>4</td>
<td>I</td>
<td><strong>RECEIVE PAIR</strong>: A differentially driven input pair which is tied to the receive pair of the Ethernet transceiver cable. The first transition on RCV will be negative-going to indicate the beginning of a frame. The last transition should be positive-going to indicate the end of the frame. The received bit stream is assumed to be Manchester encoded.</td>
</tr>
<tr>
<td>RCV</td>
<td>5</td>
<td>I</td>
<td><strong>RECEIVE PAIR</strong>: A differentially driven input pair which is tied to the receive pair of the Ethernet transceiver cable. The first transition on RCV will be negative-going to indicate the beginning of a frame. The last transition should be positive-going to indicate the end of the frame. The received bit stream is assumed to be Manchester encoded.</td>
</tr>
<tr>
<td>CRS</td>
<td>6</td>
<td>O</td>
<td><strong>CARRIER SENSE</strong>: An active low, MOS-level output to notify the Intel LAN Controller that there is activity on the coaxial cable. The signal is asserted when a valid signal on RCV/RCV is present. If the NOOR input is inactive high, then CRS is also asserted when a valid signal on CLSN/CLSN is present. It is deasserted at the end of a frame: or when the end of the collision-presence signal is detected, synchronous to RXC. After transmission, once deasserted, CRS will not be reasserted again for a period of 5 μs minimum or 7 μs maximum, regardless of any activity on the collision-presence signal (CLSN/CLSN) and RCV/RCV inputs.</td>
</tr>
<tr>
<td>CDT</td>
<td>7</td>
<td>O</td>
<td><strong>COLLISION DETECT</strong>: An active-low, MOS-level signal which drives the CDT input of the Intel LAN Controller. It is asserted as long as there is activity on the collision pair (CLSN/CLSN), and during SQE (heartbeat) test in loopback.</td>
</tr>
<tr>
<td>RXC</td>
<td>8</td>
<td>O</td>
<td><strong>RECEIVE CLOCK</strong>: A 10 MHz MOS level clock output with 5 ns rise and fall times. This output is connected to the Intel LAN Controller receive clock input RXC. There is a maximum 1.4 μs delay at the beginning of a frame reception before the clock recovery circuit gains lock. During idle (no incoming frames) RXC is forced low.</td>
</tr>
<tr>
<td>RXD</td>
<td>9</td>
<td>O</td>
<td><strong>RECEIVE DATA</strong>: A MOS-level output tied directly to the RXD input of the Intel LAN Controller and sampled by the Intel LAN Controller at the negative edge of RXC. The bit stream received from the transceiver cable is Manchester decoded prior to being transferred to the controller. This output remains high during idle.</td>
</tr>
</tbody>
</table>
Table 1. Pin Description (Continued)

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Pin No.</th>
<th>Type</th>
<th>Name and Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>GND</td>
<td>10</td>
<td></td>
<td>GROUND: Reference.</td>
</tr>
<tr>
<td>CLSN</td>
<td>12</td>
<td>I</td>
<td>COLLISION PAIR: A differentially driven input pair tied to the collision-presence pair of the Ethernet transceiver cable. The collision-presence signal is a 10 MHz square wave. The first transition at CLSN is negative-going to indicate the beginning of the signal; the last transition is positive-going to indicate the end of the signal.</td>
</tr>
<tr>
<td>CLSN</td>
<td>11</td>
<td>I</td>
<td></td>
</tr>
<tr>
<td>X1</td>
<td>14</td>
<td>I</td>
<td>CLOCK CRYSTAL: 20 MHz crystal inputs. When X2 is floated, X1 can be used as an external MOS level input clock.</td>
</tr>
<tr>
<td>X2</td>
<td>13</td>
<td>I</td>
<td></td>
</tr>
<tr>
<td>TEN</td>
<td>15</td>
<td>I</td>
<td>TRANSMIT ENABLE: An active low, TTL level signal synchronous to TXC that enables data transmission to the transceiver cable and starts the watchdog timer. TEN can be driven by the RTS from the Intel LAN Controller.</td>
</tr>
<tr>
<td>TXC</td>
<td>16</td>
<td>O</td>
<td>TRANSMIT CLOCK: A 10 MHz MOS level clock output with 5 ns rise and fall times. This clock is connected directly to the TXC input of the Intel LAN Controller.</td>
</tr>
<tr>
<td>TXD</td>
<td>17</td>
<td>I</td>
<td>TRANSMIT DATA: A TTL-level input signal that is directly connected to the serial data output, TXD, of the Intel LAN Controller.</td>
</tr>
<tr>
<td>TRMT</td>
<td>19</td>
<td>O</td>
<td>TRANSMIT PAIR: A differential output driver pair that drives the transmit pair of the transceiver cable. The output bit stream is Manchester encoded. Following the last transmission, which is always positive at TRMT, the differential voltage is slowly reduced to zero volts in a series of steps. If ENETV1 is asserted this voltage stepping is disabled.</td>
</tr>
<tr>
<td>TRMT</td>
<td>18</td>
<td>O</td>
<td></td>
</tr>
<tr>
<td>VCC</td>
<td>20</td>
<td></td>
<td>POWER: 5V ± 10%.</td>
</tr>
</tbody>
</table>

FUNCTIONAL DESCRIPTION

Clock Generation

A 20 MHz parallel resonant crystal is used to control the clock generation oscillator which provides the basic 20 MHz clock source. An internal divide-by-two counter generates the 10 MHz ± 0.01% clock required by the IEEE 802.3 specification.

It is recommended that a crystal meeting the following specifications be used:

- Quartz Crystal
- 20.00 MHz ± 0.002% @ 25°C
- Accuracy ± 0.005% Over Full Operating Temperature, 0 to 70°C
- Parallel resonant with 20 pF Load Fundamental Mode

Several vendors have these crystals available; either off the shelf or custom made. Two possible vendors are:

1. M-Tron Industries, Inc
   Yankton, SD 57078

2. Crystek Corporation
   100 Crystal Drive
   Ft Myers, FL 33907

For best operation, the total crystal load capacitance should not exceed 20 pF. The total length of the line on each side of the crystal (between X1 and X2, the crystal, and the capacitor) should be less than 2.5 cm.

An external, 20 MHz, MOS-level clock may be applied to pin X1 while pin X2 is left floating.

TRANSMIT SECTION

Manchester Encoder and Transceiver Cable Driver

The 20 MHz clock is used to Manchester encode data on the TXD input line. The clock is also divided by two to produce the 10 MHz clock required by the
Intel LAN Controller for synchronizing its RTS and TXD signals. See Figure 3. (Note that the 82586 RTS is tied to the 82C501AD TEN input as shown in Figure 4.)

Data encoding and transmission begins with TEN going low. Since the first bit is a '1', the first transition on the transmit output TRMT is always negative. Transmission ends with the TEN going high. The last transition is always positive at TRMT and may occur at the center of the bit cell (last bit = 1) or at the boundary of the bit cell (last bit = 0). A 1.5-bit delay is introduced by the 82C501AD between its TXD input and TRMT/TRMT output as shown in Figure 3. If the signal applied to the ENETV1 input is inactive high, the TRMT differential output is kept at high differential for 200 ns, then it is gradually reduced. The TRMT/TRMT differential voltage will become less than 40 mV within 8 μs after the last data transition. The undershoot for return to idle is less than 100 mV differentially. This mode of operation is compatible with the IEEE 802.3 transceiver specifications. See Figure 4.

If an active signal is present at the ENETV1 input at the end of transmission, the TRMT/TRMT pair output will remain a high differential voltage. As a result there will be a positive differential voltage during the entire transmit idle time. This mode of operation is compatible with the Ethernet Version 1.0 specification.

Immediately after the end of a transmission all signals on the receive pair are inhibited for 5 μs minimum to 7 μs maximum. This dead time is required for proper operation of the SQE (heartbeat) test.

An internal watchdog timer is started when TEN is asserted low at the beginning of the frame. The duration of the watchdog timer is 25 ms ± 15%. If the transmission terminates (by deasserting the TEN) before the timer expires, the timer is reset (and ready for the next transmission). If the timer expires before the transmission ends the frame is aborted. The frame is aborted by disabling the output driver for the TRMT/TRMT pair. RXD and RXC are not affected. The watchdog timer is reset only when the TEN is deasserted.
The cable driver is a differential circuit requiring external pulldown resistors of 240Ω ± 5%. In addition, high-voltage protection of +10V maximum, and short circuit protection to ground is provided.

To provide additional high voltage protection if the cable is shorted, an isolation transformer can be used to isolate the TRMT and TRMT outputs from the transceiver cable. Transmit circuit inductance (including the IEEE 802.3 transceiver transformers) should be a minimum of 27 µH. We recommend that the transformer at the 82C501AD end have a minimum inductance of 75 µH for Ethernet applications.

RECEIVE SECTION

Cable Interface

The 82C501AD input circuits can be driven directly from the Ethernet transceiver cable receive pair. In this case the cable is terminated with a resistor of 78Ω ± 6% for proper impedance matching. See Figure 4.

The signal received on the RCV/RCV pair from the transceiver defines both the RXC and RXD outputs to the Intel LAN Controller. The RXC and RXD signals are recovered from the encoded RCV/RCV pair signal by Manchester decode circuitry.

NOTE:
C1 = C2 = 20 pF ± 10%

Figure 4. LAN Controller/82C501AD/Transceiver Interface
The input circuits can also be driven with ECL voltage levels. In either case, the input common mode voltage must be in the range of 0-V_{CC} volts to allow for wide driver supply variation at the transceiver. To provide additional high voltage protection, if the cable is shorted, an isolation transformer can be used to isolate the RCV and RCV inputs from the cable.

**Manchester Decoder and Clock Recovery**

The Manchester-encoded data stream is decoded to separate the Receive Clock (RxC) and the Receive Data (RxD) from the stream. The 82C501AD uses an advanced digital technique to perform the decoding function. The use of digital circuitry instead of analog circuitry (e.g., a phase-lock loop) to perform the decoding ensures that the decoding function is less sensitive to variations in operating conditions.

A simplified diagram of the decoder appears in Figure 5. A high-resolution phase reference is used to digitize the phase of the incoming data bit-center transition. The digitizer has a phase resolution of 1/32 bit time.

The digitized phase is filtered by a digital low-pass filter to remove rapid phase variations; i.e., phase jitter. Slow phase variations, such as those caused by small differences between the data frequency and the clock frequency, are passed unfiltered by the low-pass filter.

The RxC generator digitally sets the phases of the two transitions to respectively lead and lag the bit-center transition by 1/4 bit time. RxC is used to recover RxD by sampling the incoming data with an edge-triggered flip-flop.

The Frame_Detect signal informs the decoder that the first valid negative transition of a new frame has been detected. This signal is used to initiate the lock-on sequence of the decoder. Lock is achieved by reducing the time constant of the digital filter to zero at the start of a new frame. With a time constant of zero, the filter immediately outputs the phase of the second bit-center transition. Any uncertainty in the bit-center phase of the first transition that is caused by jitter is subsequently removed by gradually increasing the filter time constant during the following preamble. By that time, the exact phase of the bit center is output by the filter, and the lock is achieved. Lock is achieved within the first 14 bit times as seen by the RCV/RCV inputs. The maximum bit-cell timing distortion (jitter) tolerated by the Manchester Decoder Circuitry is ± 12 ns for the preamble and ± 18 ns for the data.

![Figure 5. Manchester Decoder](image_url)
COLLISION-PRESENCE SECTION

The CLSN/CLSN input signal is a 10 MHz ± 15% square-wave generated by the transceiver whenever two or more data frames are superimposed on the coaxial cable. The maximum asymmetry in the CLSN/CLSN signal is 60/40% for low-to-high or high-to-low levels.

The common-mode voltage and external termination are identical to the RCV/RCV input. (See Figure 4.)

A valid collision presence signal will assert the 82C501AD CDT output, which can be directly tied to the CDT input of the Intel LAN Controller. During normal operation the 82C501AD logically "ORs" the collision presence signal with an internal signal, indicating valid data reception on the RCV/RCV pair to generate CRS output. If, however, the NOOR input is asserted low, this "OR" function is removed and CRS is only asserted by the presence of valid data on the RCV/RCV pair. This mode of operation is required for repeater design.

During the time that valid collision-presence transitions are present on the CLSN/CLSN input, invalid data transitions may be present on the receive data pair due to the superposition of signals from two or more stations transmitting simultaneously. It is possible for RCV/RCV to lose transitions for a few bit times due to perfect cancellation of the signals; this may cause the 82C501AD to abort the reception.

The CRS signal is asserted low (along with CDT) whenever a valid collision-presence signal is present. If this collision-presence signal arrives within 5 μs to 7 μs after the last transmission, only CDT is generated. This ensures that the LAN Controller recognizes the active CDT as a valid SQE (heartbeat) test signal.

NOISE FILTERING ON RCV AND CLSN PAIRS

Both the receive and collision pairs have the following characteristics.

- At idle, the noise filter is turned on.
- Any pulse narrower than 5 ns or with an amplitude of less than 140 mV is rejected by the noise filter.
- The filter is turned off by the first valid negative pulse on the RCV or CLSN pair. A negative pulse wider than 30 ns and having an amplitude greater than 285 mV is considered a valid pulse.
- The filter is turned on again when no positive transition is observed on the RCV or CLSN pair for 160 ns.

Internal Loopback

When asserted, LPBK causes the 82C501AD to route serial data from its TXD input through its transmit logic (retiming and Manchester encoding); returning it through the receive logic (Manchester decoding and receive clock generation) to RXD output. The internal routing prevents the data from passing through the output drivers and onto the transmit output pair TRMT/TRMT. When in loopback mode all of the transmit and receive circuits, are tested except for the transceiver cable output driver and input receivers. Also, at the end of each frame transmitted in loopback mode the 82C501AD generates the SQE test (heartbeat) signal within 1 μs after the end of the frame. Thus, the collision circuits, are also tested in loopback mode.

The watchdog timer remains enabled in loopback mode, terminating test frames that exceed its timeout period. The watchdog can be inhibited by connecting LPBK to a 1 kΩ resistor connected to 10 to 16 Volts. The loopback feature can still be used to test the integrity of the 82C501AD by using the circuit shown in Figure 6.

<table>
<thead>
<tr>
<th>LPBK</th>
<th>WTD</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>X</td>
<td></td>
<td>LPBK mode</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>Normal mode</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>Normal mode with watchdog timer disabled</td>
</tr>
</tbody>
</table>

Figure 6. Watchdog Timer Disable

The 82C501AD operates as a full-duplex device, being able to transmit and receive simultaneously. By combining the internal and external loopback modes of the Intel LAN Controller, and the internal loopback and normal modes of the 82C501AD, incremental testing of an 82586/82C501AD-based interface can be performed under program control for systematic fault detection and fault isolation.
**Interface Example**

The 82C501AD is designed to work directly with the 82586 controller in IEEE 802.3 10 Mbps, as well as other 10 Mbps LAN applications. The control and data signals connect directly between the two devices without the need for additional external logic. The complete 82586/82C501AD Ethernet Transceiver interface is shown in Figure 4. The 82C501AD provides the driver and receivers needed to directly connect to the transceiver cable; requiring only terminating resistors on each input signal pair and 2400 pull-down resistors.

The Transmit, Receive, and Collision pairs have a maximum 10V overvoltage protection.

If additional high voltage protection is desired, a pulse transformer should be included for Ethernet applications. IEEE 802.3 10BASE5 (Ethernet) specifications require at least 16V protection for the Transmit, Receive, and Collision pairs. In 10BASE2 (Cheapernet) a pulse transformer is required to be inserted between the DTE (82586/82C501AD) and the transceiver. Through the use of jumpers, the same transformer can be used for an Ethernet connection at minimal additional cost.

The pulse transformer should have the following characteristics:
1. A minimum inductance of 50 μH (75 μH is preferable for Ethernet applications).
3. 2000V isolation between primaries of separate transformers.

Since Ethernet Version 1.0 transceivers may require a positive differential on the TRMT pair during idle, check with the transceiver vendor before including the pulse transformer.
**ABSOLUTE MAXIMUM RATING**

Case Temperature Under Bias ........... 0°C to +85°C  
Storage Temperature ................. -65°C to +140°C  
All Output and Supply Voltages ........... -0.5V to + 7V  
All Input Voltages ....................... -1.0V to + 6.0V(1)  
Operating Power Dissipation .......... 0.75W  

*Notice: Stresses above those listed under “Absolute Maximum Ratings” may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions above those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.  

**NOTICE:** Specifications contained within the following tables are subject to change.

<table>
<thead>
<tr>
<th>D.C. CHARACTERISTICS</th>
<th>$T_C = 0°C$ to $+85°C$, $V_{CC} = 5V$ ± 10%</th>
</tr>
</thead>
<tbody>
<tr>
<td>Symbol</td>
<td>Parameter</td>
</tr>
<tr>
<td>$V_{IL}$</td>
<td>Input Low Voltage</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>$V_{IH}$</td>
<td>Input High Voltage</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>$V_{IDF}$</td>
<td>Input Differential Voltage</td>
</tr>
<tr>
<td>$V_{CM}$</td>
<td>Input Common Mode Voltage</td>
</tr>
<tr>
<td>$V_{OCM}$</td>
<td>Common Mode Output(2)</td>
</tr>
<tr>
<td>$V_{OL}$</td>
<td>Output Low Voltage @ $I_{OL} = 4$ mA</td>
</tr>
<tr>
<td>$V_{OH}$</td>
<td>Output High Voltage (MOS) @ $I_{OH} = -500$ μA</td>
</tr>
<tr>
<td>$V_{ODF}$</td>
<td>Differential Output Swing(3)</td>
</tr>
<tr>
<td>$I_{LI}$</td>
<td>Input Leakage Current(4) @ $V_{IN} = 0V$ to $V_{CC}$</td>
</tr>
<tr>
<td>$C_{IN}$</td>
<td>Input Capacitance @ $f_c = 1$ MHz</td>
</tr>
<tr>
<td>$C_{OUT}$</td>
<td>Output Capacitance @ $f_c = 1$ MHz(6)</td>
</tr>
<tr>
<td>$I_{CC}$</td>
<td>Power Supply Current @ $T_C = 85°C$(5)</td>
</tr>
<tr>
<td>$I_{SP}$</td>
<td>Short Protection Activation Current</td>
</tr>
<tr>
<td>$I_{DI}$</td>
<td>Input Current into/out of Differential Input(7)</td>
</tr>
<tr>
<td>$V_{U}$</td>
<td>Differential Return to Zero Undershoot</td>
</tr>
<tr>
<td>$V_{DI}$</td>
<td>Differential Idle Voltage</td>
</tr>
</tbody>
</table>

**NOTES:**
1. The voltage levels for $CLS$/$CLS$, $RCV$/$RCV$ inputs are $-0.75V$ to $+10V$.  
2. The load is a $78Ω$ ± 6% resistor in parallel with a $27 μH$ ± 1% inductor.  
3. DC measurement values.  
4. Applies to TXD and TEN pins.  
5. Part of the power is dissipated through the pulldown resistors connected to TRMT/TRMT outputs.  
6. With the exception of TRMT/TRMT.  
7. Applies to $RCV$/$RCV$, $CLS$/$CLS$, $LPBK$/$WDTD$, $NOOR$, and $ENETVI$ inputs for input voltages from $0V$ to $V_{CC}$.  

---

1-46
A.C. MEASUREMENT CONDITIONS

1. \( T_C = 0^\circ C \) to \(+85^\circ C\), \( V_{CC} = 5V \pm 10\%\).
2. The AC MOS and TTL measurement levels are referred to in Figures 7, 8, 9, 10 and 10A.
3. AC Loads:
   a) MOS: a 20 pF total capacitance to ground.
   b) Differential: a 10 pF total capacitance from each terminal to ground, and a load resistor of \( 78\Omega \pm 6\% \) in parallel with a 27 \( \mu H \pm 1\% \) inductor between terminals.

Clock Timing\(^{(1)}\)

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Parameter</th>
<th>Min</th>
<th>Max</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>t1</td>
<td>( X_1 ) Cycle Time(^{(2)})</td>
<td>49.995</td>
<td>50.005</td>
<td>ns</td>
</tr>
<tr>
<td>t2</td>
<td>( X_1 ) Fall Time(^{(3)})</td>
<td>5</td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>t3</td>
<td>( X_1 ) Rise Time(^{(3)})</td>
<td>5</td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>t4</td>
<td>( X_1 ) Low Time (at 0.9V)(^{(3)})</td>
<td>15</td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>t5</td>
<td>( X_1 ) High Time (at 3.0V)(^{(3)})</td>
<td>15</td>
<td>ns</td>
<td></td>
</tr>
</tbody>
</table>

NOTES:
1. Refer to Figure 9.
2. Applies to crystal based inputs.
3. Applies to external clock inputs.

TRANSMIT TIMING\(^{(1)}\)

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Parameter</th>
<th>Min</th>
<th>Max</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>t6</td>
<td>( TXC ) Cycle Time(^{(2)})</td>
<td>99.99</td>
<td>100.01</td>
<td>ns</td>
</tr>
<tr>
<td>t7</td>
<td>( TXC ) Fall Time</td>
<td></td>
<td>5</td>
<td>ns</td>
</tr>
<tr>
<td>t8</td>
<td>( TXC ) Rise Time</td>
<td></td>
<td>5</td>
<td>ns</td>
</tr>
<tr>
<td>t9</td>
<td>( TXC ) Low Time (at 0.9V)</td>
<td>40</td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>t10</td>
<td>( TXC ) High Time (at 3.0V)</td>
<td>40</td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>t11</td>
<td>Transmit Enable/Disable to ( TXC ) Low</td>
<td>45</td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>t12</td>
<td>TXD Stable to ( TXC ) Low</td>
<td>45</td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>t13</td>
<td>Bit Cell Center to Bit Cell Center of Transmit Pair Data(^{(3)})</td>
<td>99.5</td>
<td>100.5</td>
<td>ns</td>
</tr>
<tr>
<td>t14</td>
<td>Transmit Pair Data Fall Time</td>
<td>1.0</td>
<td>5.0</td>
<td>ns</td>
</tr>
<tr>
<td>t15</td>
<td>Transmit Pair Data Rise Time</td>
<td>1.0</td>
<td>5.0</td>
<td>ns</td>
</tr>
<tr>
<td>t16</td>
<td>Bit Cell Center to Bit Cell Boundary of Transmit Pair Data(^{(3)})</td>
<td>49.5</td>
<td>50.5</td>
<td>ns</td>
</tr>
<tr>
<td>t17</td>
<td>TRMT held low from Last Positive Transition of the Transmit Pair at the End of Frame</td>
<td>200</td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>t18</td>
<td>From Last Positive Transition of Transmit Pair Differential Output Approaches Within 40 mV of zero volts.</td>
<td>8000</td>
<td>ns</td>
<td></td>
</tr>
</tbody>
</table>

NOTES:
1. Refer to Figure 11.
2. This parameter is determined by the crystal.
3. Characterized not tested.
### RECEIVE TIMING(1)

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Parameter</th>
<th>Min</th>
<th>Max</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>t19</td>
<td>Duration which the RXC is held at Low State</td>
<td>1400</td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>t20</td>
<td>Receive Pair Signal Rise/Fall Time at 0.285V</td>
<td></td>
<td>10</td>
<td>ns</td>
</tr>
<tr>
<td>t21</td>
<td>Receive Pair Bit Cell Center from Crossover Timing Distortion:(2)</td>
<td>± 12</td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td></td>
<td>In Preamble</td>
<td>± 18</td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td></td>
<td>In Data</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>t22</td>
<td>Receive Pair Bit Cell Boundary Allowing for Timing Distortion In Data(2)</td>
<td></td>
<td>± 18</td>
<td>ns</td>
</tr>
<tr>
<td>t23</td>
<td>Receive Idle Time Before the Next Reception can Begin in a Transmitting</td>
<td></td>
<td>8</td>
<td>μs</td>
</tr>
<tr>
<td></td>
<td>Station (as Measured from the Deassertion of CRS)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>t24</td>
<td>Receive Pair Signal Return to Zero Level from Last Valid Positive Transition</td>
<td>160</td>
<td></td>
<td>ns</td>
</tr>
<tr>
<td>t25</td>
<td>CRS Assertion Delay from the First Received Valid Negative Transition of Receive Pair Signal</td>
<td>100</td>
<td></td>
<td>ns</td>
</tr>
<tr>
<td>t26</td>
<td>CRS Deassertion Delay from the Last Valid Positive Transition Received (when no Collision-Presence Signal Exists on the Transceiver Cable)(3)</td>
<td>300</td>
<td></td>
<td>ns</td>
</tr>
<tr>
<td>t27</td>
<td>RXC Cycle Time</td>
<td>96</td>
<td>104</td>
<td>ns</td>
</tr>
<tr>
<td>t28</td>
<td>RXC Rise/Fall Time</td>
<td>5.0</td>
<td></td>
<td>ns</td>
</tr>
<tr>
<td>t29</td>
<td>RXC Low Time (at 0.9V)</td>
<td>40</td>
<td></td>
<td>ns</td>
</tr>
<tr>
<td>t30</td>
<td>RXC High Time (at 3.0V)</td>
<td>36</td>
<td></td>
<td>ns</td>
</tr>
<tr>
<td>t31</td>
<td>Receive Data Stable Before the Negative Edge of RXC</td>
<td>30</td>
<td></td>
<td>ns</td>
</tr>
<tr>
<td>t32</td>
<td>Receive Data Held Valid Past the Negative Edge of RXC</td>
<td>30</td>
<td></td>
<td>ns</td>
</tr>
<tr>
<td>t33</td>
<td>Carrier Sense Active → Inactive Hold Time from RXC High</td>
<td>10</td>
<td>40</td>
<td>ns</td>
</tr>
<tr>
<td>t34</td>
<td>Receive Data Rise/Fall Time</td>
<td>10</td>
<td></td>
<td>ns</td>
</tr>
<tr>
<td>t35</td>
<td>CRS Inhibit Time After Frame Transmission(4)</td>
<td>5</td>
<td>7</td>
<td>μs</td>
</tr>
</tbody>
</table>

**NOTES:**

1. Refer to Figures 12 and 13.
2. Measured per 802.3 Para B1.1.4.2 recommendations.
3. CRS is deasserted synchronously with the RXC. This condition is not specified in the IEEE 802.3 specification.
4. Required for SQE test.
### COLLISION TIMING

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Parameter</th>
<th>Min</th>
<th>Max</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>t36</td>
<td>CLSN/CLSN Cycle Time</td>
<td>85</td>
<td>118</td>
<td>ns</td>
</tr>
<tr>
<td>t37</td>
<td>CLSN/CLSN Rise/Fall Time at ±0.285V</td>
<td>15</td>
<td></td>
<td>ns</td>
</tr>
<tr>
<td>t38</td>
<td>CLSN/CLSN Transition Time</td>
<td>35</td>
<td>70</td>
<td>ns</td>
</tr>
<tr>
<td>t39</td>
<td>CLSN Pair Return to Zero from Last Positive Transition</td>
<td>160</td>
<td></td>
<td>ns</td>
</tr>
<tr>
<td>t40</td>
<td>CDT Assertion from the First Valid Negative Edge of Collision Pair Signal</td>
<td>75</td>
<td></td>
<td>ns</td>
</tr>
<tr>
<td>t41</td>
<td>CDT Deassertion from the Last Positive Edge of CLSN/CLSN Signal</td>
<td>300</td>
<td></td>
<td>ns</td>
</tr>
<tr>
<td>t42</td>
<td>CRS Deassertion from the Last Positive Edge of CLSN/CLSN Signal (if no Post-Collision Signal Remains on the Receive Pair)</td>
<td>950</td>
<td></td>
<td>ns</td>
</tr>
</tbody>
</table>

**NOTE:**
1. Refer to Figure 14.

### LOOPBACK TIMING

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Parameter</th>
<th>Min</th>
<th>Max</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>t43</td>
<td>LPBK asserted before the first attempted transmission (2)</td>
<td>500</td>
<td></td>
<td>ns</td>
</tr>
<tr>
<td>t44</td>
<td>Simulated collision test delay from the end of each attempted transmission</td>
<td>0.5</td>
<td>1.5</td>
<td>μs</td>
</tr>
<tr>
<td>t45</td>
<td>Simulated collision test duration(3)</td>
<td>0.6</td>
<td>1.6</td>
<td>μs</td>
</tr>
<tr>
<td>t46</td>
<td>LPBK deasserted after the last attempted transmission</td>
<td>5</td>
<td></td>
<td>μs</td>
</tr>
</tbody>
</table>

**NOTES:**
1. Refer to Figure 15.
2. In Loopback mode, RXC and CRS function in the same manner as a normal Receive.
3. SQE test (heartbeat) signal

### NOISE FILTER

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Parameter</th>
<th>Min</th>
<th>Max</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>t47</td>
<td>RCV/RCV Noise Filter Pulse Width Rejected</td>
<td>5</td>
<td></td>
<td>ns</td>
</tr>
<tr>
<td>t48</td>
<td>RCV/RCV Noise Filter Pulse Width Accepted</td>
<td>30</td>
<td></td>
<td>ns</td>
</tr>
<tr>
<td>t49</td>
<td>CLSN/CLSN Noise Filter Pulse Width Rejected</td>
<td>5</td>
<td></td>
<td>ns</td>
</tr>
<tr>
<td>t50</td>
<td>CLSN/CLSN Noise Filter Pulse Width Accepted</td>
<td>30</td>
<td></td>
<td>ns</td>
</tr>
<tr>
<td>V_reject</td>
<td>Differential Reject Voltage(2)</td>
<td>-140</td>
<td></td>
<td>mV</td>
</tr>
<tr>
<td>V_accept</td>
<td>Differential Accept Voltage(2)</td>
<td>-285</td>
<td></td>
<td>mV</td>
</tr>
</tbody>
</table>

**NOTES:**
1. Refer to Figure 16.
2. DC parameters.
A.C. TIMING CHARACTERISTICS

Input and Output Waveforms for AC Tests

<table>
<thead>
<tr>
<th>Voltage</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.4V</td>
<td>1.5 TEST POINTS 1.5</td>
</tr>
<tr>
<td>0.45V</td>
<td></td>
</tr>
</tbody>
</table>

A.C. Testing inputs are driven at 2.4V for a Logic 1 and 0.45V for a Logic 0. Timing measurements are made at 1.5V for both a Logic 1 and 0.

Figure 7. TTL Input/Output Voltage Levels for Timing Measurements

Figure 8. Voltage Levels for MOS Level Output-Timing Measurements (TXC, RXC, CRS, CDI, and RXD).

Figure 9. X1 Input Voltage Levels for Timing Measurements

Figure 10. Voltage Levels for Differential-Input Timing Measurements (RCV/RCV, CLSN/CLS and CLSN).

Figure 10A. Voltage Levels for TRMT/TRMT Output-Timing Measurements
TRANSMIT TIMING

Figure 11

RECEIVE TIMING: START OF FRAME

Figure 12
**RECEIVE TIMING: END OF FRAME**

![Diagram of receive timing](image)

*NOTE:*  
CRS can be triggered on again by the collision-presence signal.

**Figure 13**

**COLLISION TIMING**

![Diagram of collision timing](image)

**Notes:**  
1. CRS will be deasserted for a period up to 7 \( \mu \)s maximum when RCV/RCV or CLSN/CLSN terminates, whichever occurs later.  
2. CRS will remain asserted after the CLSN/CLSN signal terminates if RCV/RCV signals continue.

**Figure 14**
LOOBACK TIMING

![Diagram showing Loopback Timing](image)

**NOTE:**
1. During Loopback, the 82C501AD receive circuitry uses 14 bit times while the Manchester Decoder locks on the data. As a result, the first 14 bits are lost and RXC is held low during that time.

**NOISE FILTER TIMING**

![Diagram showing Noise Filter Timing](image)

**TESTABILITY**

**NOTE:**
1. All AC Parameters become valid after the High Resolution Phase Reference has stabilized: 100 μs after the application of power.
Twisted Pair Ethernet — Chip Set Overview

- Complete Twisted Pair Ethernet solution
  - Transceiver Serial Interface (TSI)
    - Interface from all Intel Ethernet LAN Controllers to twisted pair link segment
    - Provides data and clock signal recovery from incoming 10 Mb/s Manchester data
    - Converts Manchester data into NZR format
    - Detects Manchester code violations and end-of-packet delimiter (IDL)
  - Multiport Repeater Controller (MPR)
    - Complies with IEEE 802.3 Ethernet standard for Repeaters (Std 802.3 c-1988)
    - Allows up to 11 twisted pair ports and 1 AUI port
    - Automatic preamble generation
    - Minimum frame length enforcement (96 bits)
    - Provides Manchester encoding of transmitted data
    - Pin selective FIFO fill level

- Twisted Pair Media Access Unit (TP MAU)
  - Direct interface to AUI and twisted pair isolation transformers
  - Generates internal predistortion signal
  - Resetable Jabber function
  - Signal Quality Error (SQE) testing
  - Uses low power CMOS technology

Chip Set Description

The Twisted Pair Ethernet chip set provides the complete component solution for running 10 Mb/s Ethernet over low cost unshielded twisted pair wire. It is designed to support both new Twisted Pair Ethernet LAN designs as well as retrofitting existing Ethernet systems for twisted pair. The set includes the Transceiver Serial Interface Component (TSI), the Multiport Repeater Controller (MPR), and the Twisted Pair Media Access Unit (TP MAU).

TPE SYSTEM TOPOLOGY

---

MAU = Medium Access Unit
TSI = Transceiver Serial Interface
MPR = Multiport Repeater
AUI = Attachment Unit Interface
Component Description

TRANSCEIVER SERIAL INTERFACE COMPONENT (TSI)
The Transceiver Serial Interface component (TSI) is intended for all Twisted Pair Ethernet LAN applications using 10 Mbps Manchester-coded data, such as client stations, file servers, and repeaters. It reduces design time by providing a direct serial interface from the twisted pair wire to any of Intel's Ethernet LAN Controllers. The TSI chip performs clock recovery and Manchester Decoding of 10 Mbps data, and produces NRZ data and clock signals for the LAN controller. In addition, the TSI supports a predistortion method to prevent line over-charge. It is fabricated using CMOS processing technology and is available in 24-lead plastic DIP and 28-lead SOJ packages.

MULTIPORT REPEATER CONTROLLER (MPR)
The Multiport Repeater Controller component (MPR) is designed for use in Twisted Pair Ethernet repeater applications. The MPR combines with a single TSI chip to provide all necessary repeater functions including automatic preamble generation, Manchester encoding of transmitted data, Jabber function, Jam signal generation and minimal frame length enforcement. The MPR/TSI set supports up to eleven twisted pair ports and one AUI port. The MPR offers pin selectable FIFO fill level and LED output control of Traffic status, Jam status, port Jabber status, and FIFO error status. The MPR Controller is fabricated using CMOS processing technology and is available in a 68-lead plastic lead chip carrier package (PLCC).

TWISTED PAIR MEDIA ACCESS UNIT (TP MAU)
The Twisted Pair Media Access Unit (TP MAU) is designed for Ethernet Node Adapter (ENA) applications to interface the Ethernet AUI cable directly to the twisted pair wire. It offers LAN designers a cost effective, integrated solution to the problem of upgrading existing standard Ethernet networks to twisted pair. The TP MAU generates internal predistortion signals to eliminate line over-charge. It provides selectable diagnostics features including selectable Signal Quality Error (SQE) test and manual or automatic Jabber Reset. In addition, the TP MAU supports LED control for Transmit, Receive, Jabber and Collision. It is fabricated using CMOS processing technology and is available in 28-lead plastic DIP and 28-lead SOJ packages.
82560 HOST INTERFACE AND MEMORY CONTROLLER

- Host Interface to the IBM PC/XT/AT and PS/2™ Buses for 82590, 82592, and 82588 LAN Controllers
- Allows 32-, 16-, and 8-Bit Data Transfers
- Supports Local Static RAM
  - Up to 32 Kilobytes
  - Programmable Access Time
- Zero-Wait-State Host Interface Option
- Dual-Channel DMA Controller with Ring Buffer Management Scheme
- Implements Tightly Coupled Interface Mode to 82590/82592
  - Automatic Retransmission upon Collision
  - Transmit Chaining
  - Back-to-Back Frame Reception
  - Automatic Buffer Reclamation
  - Address PROM or Other Peripheral Support
  - Interfaces Memory-Mapped or I/O-Mapped Adapters
- CHMOS III Technology
- 68-Lead PLCC Package

The 82560 Host Interface and Memory Controller is a companion chip for the Intel 82590 and 82592 Advanced CSMA/CD LAN Controllers as well as the Intel 82588. The 82560 interfaces these controllers to IBM PC/XT/AT and PS/2 systems. It integrates all the interface functions required to implement a nonintelligent, locally buffered LAN solution. The zero wait state and 32-bit data transfers improve the system performance by minimizing the LAN's requirement for Host bandwidth. The 82560's DMA performs data transfers between the LAN controller and the ring-configured local memory. Ring buffer implementation results in highly efficient use of the local memory. The 82560 supports the 82590 and 82592 in their Tightly Coupled Interface (TCI) mode. Without CPU intervention, the 82560 performs transmit chaining, automatic retransmission, back-to-back frame reception, and frame reclamation. The TCI support reduces the software and hardware overhead between frame transfers, and increases the average sustained transfer rate. Combined with the 82590 or 82592, the 82560 provides a high-performance LAN solution for industry standard or custom CSMA/CD networks.
### Table 1. 82560 Pin Description

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Pin No.</th>
<th>Type</th>
<th>Name and Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>VCC</td>
<td>5, 23, 57</td>
<td>I</td>
<td>POWER: Connected to +5V power supply.</td>
</tr>
<tr>
<td>VSS</td>
<td>10, 29, 43, 63</td>
<td>I</td>
<td>GROUND: Ground connection.</td>
</tr>
<tr>
<td>CLK</td>
<td>11</td>
<td>I</td>
<td>CLOCK INPUT: This is the system clock input for the 82560. It controls the internal operations of the 82560 and its cycle timing.</td>
</tr>
<tr>
<td>RESET</td>
<td>42</td>
<td>I</td>
<td>RESET: Active high. When active it resets the 82560 to a known passive state.</td>
</tr>
<tr>
<td>D0–D7</td>
<td>40, 41, 44–49</td>
<td>I/O</td>
<td>82560 DATA BUS: Tri-state bus. Used for programmatic access to the 82560 registers. They are also used in the tightly coupled interface (TCI) mode.</td>
</tr>
<tr>
<td>A0–A12</td>
<td>26–39</td>
<td>I</td>
<td>ADDRESS LINES: The 13 address lines select either an 82560 register, or an address in the Local Memory.</td>
</tr>
<tr>
<td>HF0, HF1</td>
<td>25, 24</td>
<td>I</td>
<td>HOST FUNCTION SELECT: These two inputs indicate the type of access requested by the host. These signals are generated by external decode logic and are completely asynchronous to the 82560 system clock. The proper combinations for each access type are shown below:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Host Function</td>
<td>HF1</td>
<td>HF0</td>
<td>Access Type</td>
</tr>
<tr>
<td>Idle (No Access Being Requested)</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>Request to Access Shared Portion of Local Memory</td>
<td>1</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>Request to Access 82560 Registers or the Slave Controller (SCS)</td>
<td>0</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>Request to Access External PROMs or Latches (GCS)</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>RDI</td>
<td>17</td>
<td>I</td>
<td>READ: Active low. This signal is used to indicate the direction of the host transfer. When active, data is being read from the destination (RAM, 560, or GCS port).</td>
</tr>
<tr>
<td>HRDY</td>
<td>20</td>
<td>O</td>
<td>HOST READY: Active high. This signal from the 82560 is activated when the device on the Local Bus of the LAN adapter is ready to accept data (write cycle) or to output data (read cycle). When no access is being requested by the host (i.e., both HF0 and HF1 are high), this signal is tri-stated in the normal mode, and is driven high in the pipeline mode.</td>
</tr>
<tr>
<td>XCV1</td>
<td>22</td>
<td>O</td>
<td>TRANSCEIVER ENABLE 1: Enables the transceiver that connects the lower byte of the host and Local data buses. In pipeline mode it enables the transceiver during non-memory host cycles.</td>
</tr>
<tr>
<td>XCV2/PCS</td>
<td>21</td>
<td>O</td>
<td>TRANSCEIVER ENABLE 2: Enables the transceiver that connects the upper byte of the host and Local data buses. In pipeline mode it enables the latch during memory host cycles.</td>
</tr>
<tr>
<td>INT</td>
<td>50</td>
<td>O</td>
<td>INTERRUPT OUT: This signal is a logical OR of all enabled interrupt requests. When active it indicates an interrupt request to the CPU. This signal is tristated after reset.</td>
</tr>
<tr>
<td>GCS</td>
<td>59</td>
<td>O</td>
<td>GENERAL CHIP SELECT: Active low. This signal is asserted by the 82560 when the host requests access to external ROMs or latches.</td>
</tr>
</tbody>
</table>
### Table 1. 82560 Pin Description (Continued)

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Pin No.</th>
<th>Type</th>
<th>Name and Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>BE0</td>
<td>18</td>
<td>I</td>
<td><strong>BYTE ENABLE</strong>: Active low. This signal is asserted in 16- or 32-bit-wide host memory cycles to select the lower memory bank. It may be connected to the processor’s A0 pin.</td>
</tr>
<tr>
<td>BE1</td>
<td>19</td>
<td>I</td>
<td><strong>BYTE ENABLE 1</strong>: Active low. This signal is asserted in 16- or 32-bit-wide host memory cycles to select the upper memory bank. It may be connected to the processor’s BHE signal. These two signals are connected as follows:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td><strong>Host Bus</strong></td>
</tr>
<tr>
<td>8-Bit</td>
<td>8-Bit</td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>8-Bit</td>
<td>16-Bit</td>
<td></td>
<td>SA0</td>
</tr>
<tr>
<td>16-Bit</td>
<td>16-Bit</td>
<td></td>
<td>SA0</td>
</tr>
<tr>
<td>16-Bit</td>
<td>32-Bit</td>
<td></td>
<td>SA1</td>
</tr>
<tr>
<td>32-Bit*</td>
<td>32-Bit</td>
<td></td>
<td>BE0 + BE1</td>
</tr>
</tbody>
</table>

*80386 address pins
+ stands for logical OR

<p>| DRQ0   | 54      | I    | <strong>DMA REQUEST CHANNEL 0</strong>: Active high. This is an input from the LAN controller or other peripherals, it requests DMA service. The DMA cycles are run on an on-demand basis, and are prioritized between themselves (two channels) and with the host cycles on an alternating basis. In 82590 Tightly Coupled mode this signal is sampled by the 82560 at the last clock of the Read or Write signal along with DACK1/EOP to determine the state of the transmit or receive process (see Tightly Coupled Interface for more details). |
| DRQ1   | 52      | I    | <strong>DMA REQUEST CHANNEL 1</strong>: Active high. This is an input from the LAN controller or other peripherals, requesting DMA service. The DMA cycles are run on an on-demand basis, and are prioritized between themselves (two channels) and with the host cycles on an alternating basis. In Tightly Coupled mode this signal is sampled by the 82560 at the last clock of the Read or Write signal (see Tightly Coupled Interface for more details). |
| DACK0/DACK | 55 | O    | <strong>Dual Function</strong>: This is a dual function pin which serves as DACK0, DMA acknowledge for Channel 0, in all modes except the Tightly Coupled Interface mode. It serves as DACK, DMA acknowledge for both channels, in Tightly Coupled Interface mode. <strong>DMA ACKNOWLEDGE0</strong>: Active low. Acknowledge DMA requests on channel 0. During special chip select cycles, this signal is controlled by the CPU. <strong>DMA ACKNOWLEDGE</strong>: Active low. Acknowledge DMA requests on either channel 0, or channel 1. It operates in this mode only when programmed for Tightly Coupled Interface with the 82590 or 82592. This pin can be directly connected to the DACK0/DACK pin of the 82590 or 82592 LAN controllers. |
| DACK1/EOP | 53 | I/O  | <strong>Dual Function</strong>: This is a dual function, bidirectional pin which serves as DACK1, DMA acknowledge for channel 1, in all modes except 8259X Tightly Coupled Interface mode. It serves as EOP, End of Process indicator, an input, during this Tightly Coupled Interface mode. <strong>DMA ACKNOWLEDGE1</strong>: Output. Active low. DMA acknowledge for channel 1. During Special Chip Select (SCS) cycles this signal is controlled by the CPU and can be used for accessing the 8259X port 1. The output level is determined by the address of the SCS. |</p>
<table>
<thead>
<tr>
<th>Symbol</th>
<th>Pin No.</th>
<th>Type</th>
<th>Name and Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>DACK1/EOP</td>
<td>53</td>
<td>I/O</td>
<td>END OF PROCESS: Input. In the Tightly Coupled Interface mode, this input, along with the DRQ pin, is sampled by the 82560 at the last clock of the Read or Write signal. The combination of the two pins indicates the status of the Transmit or Receive process. When low, the EOP signal indicates that the active DMA service should be terminated.</td>
</tr>
<tr>
<td>IOWR</td>
<td>56</td>
<td>O</td>
<td>I/O WRITE. Active low. This is the write strobe to the LAN controller or I/O device. It is asserted when data is being written to the LAN controller by either the Host CPU or the 82560 internal DMA. During pipeline read transfers it is the write control signal to the buffer.</td>
</tr>
<tr>
<td>IORD/MWR</td>
<td>58</td>
<td>O</td>
<td>Dual Function: Active low. This signal is used for two different operations. It is a control signal during read cycles from the LAN controller or another I/O device. It is a write strobe during write cycles to the local memory. I/O READ: Active low. It is asserted when data is being read from the LAN controller by either the host CPU or the 82560 internal DMA. During pipeline write transfers it is the read control signal to the buffer. MEMORY WRITE: Active low. It is asserted when data is being written to local memory.</td>
</tr>
<tr>
<td>INTR</td>
<td>51</td>
<td>I</td>
<td>INTERRUPT REQUEST: This signal when active indicates an interrupt request. It is usually connected to the interrupt output of the LAN controller. It may be programmed as active high or low, level or edge triggered, and it can also be masked.</td>
</tr>
<tr>
<td>MA0–12</td>
<td>9–1, 68</td>
<td>O</td>
<td>MEMORY ADDRESS 0–12: These 13 address lines can support two 8-kilobyte or 8-kiloword banks of static memory.</td>
</tr>
<tr>
<td>CSL</td>
<td>62</td>
<td>O</td>
<td>RAM CHIP SELECT (LOW BANK): Active low. This signal is activated during all static-RAM accesses in 8-bit mode, even-byte accesses in 16-bit mode, and even-word accesses in 32-bit mode.</td>
</tr>
<tr>
<td>CSH</td>
<td>61</td>
<td>O</td>
<td>RAM CHIP SELECT (HIGH BANK): Active low. This signal is activated during odd-byte accesses in 16-bit mode or odd-word accesses in 32-bit mode.</td>
</tr>
<tr>
<td>MOE</td>
<td>60</td>
<td>O</td>
<td>MEMORY OUTPUT ENABLE: Active low. This signal is used to enable the memory array’s output buffers during memory read cycles.</td>
</tr>
<tr>
<td>GPI</td>
<td>16</td>
<td>I</td>
<td>GENERAL PURPOSE: Input. This is a general purpose input pin, its state may be read by the CPU.</td>
</tr>
<tr>
<td>CS</td>
<td>12</td>
<td>O</td>
<td>CHIP SELECT: Active low. This pin is normally connected to the Chip Select input of the LAN controller or other peripherals. It is activated during non-DMA accesses to the LAN controller. The CPU activates this signal when it accesses addresses 0, 1, 2, or 3 in the Special Chip Select address space of the 82560.</td>
</tr>
<tr>
<td>RSV1, RSV2</td>
<td>13, 14</td>
<td>I</td>
<td>These pins are reserved and should be tied to VCC.</td>
</tr>
</tbody>
</table>
Figure 2. 82560 PLCC Pinout

Figure 3. Nonintelligent, Buffered Adapter Architecture
FUNCTIONAL OVERVIEW

The 82560 is a dual-port memory controller using interrupt logic and DMA to implement a nonintelligent, buffered LAN adapter for the IBM PC/XT/AT bus. This type of adapter uses on-board memory as a buffer to store frames during transmission and reception. It also uses on-board DMA to transfer data between its local memory and the LAN controller. A block diagram of the buffered, nonintelligent LAN adapter is shown in Figure 3. The architecture is termed nonintelligent because it does not use an on-board CPU to process the transmit or receive frames. The host CPU processes the frames and programs the DMA and LAN controllers. The host interface logic, arbitration logic, and the bus transceivers connect the host bus to the adapter’s local bus. They also control all host accesses to the local bus. The local memory is shared by the host and the LAN controller. It stores information that the host wishes to transfer to the LAN controller, and information received by the LAN controller which should be read by the host. The memory can be shared in two ways—mapping into the host memory space, or mapping into the host I/O space. The DMA controller transfers data between the local buffer memory and the LAN controller. The host CPU may use either string move instructions or a system DMA channel to move data into the buffer memory. The host also accesses the LAN controller registers, the DMA controller registers the boot ROM, and the address ROM through the local bus.

The 82560 integrates the host interface, arbitration logic, memory control logic, interrupt logic, and DMA into one component. It replaces 20–30 MSI and SSI components (see Figure 1). It also provides a Tightly Coupled Interface to the 82588, 82590, and 82592 LAN controllers, and an efficient buffer management scheme, which allows the 82588/82560, 82590/82560, and 82592/82560 combinations to handle time-critical processes such as retransmission, buffer reclamation, and continuous back-to-back frame reception without host CPU intervention. This improves the overall data throughput in the network; and, consequently, system performance. The following discussion describes the 82560 interface to the PC/XT/AT bus, its support of locally buffered memory, and the operation of DMA and the Tightly Coupled Interface (including the buffer management scheme).

HOST INTERFACE

The host interface port connects the 82560 to the PC-bus through external decode logic and bus transceivers. The external decode logic generates the HF0 and HF1 signals indicating the kind of access the host desires. When the request is detected by the 82560 (non-pipeline mode) it deasserts the HRDY signal, thereby suspending the host cycle. HRDY is reactivated when the local device being accessed by the host is ready to accept (Write cycle) or output (Read cycle) data. HRDY reactivation time is programmable as mentioned in the register section; it is described in detail in the 82560 Reference Manual. The request undergoes arbitration, and, if granted, the 82560 activates the XCV1 and XCV2 signals. The XCV signals control the transceiver(s) which interfaces the host data bus to the local data bus. By using one or both transceiver control signals the 82560 can support an 8-, 16-, or 32-bit-wide bus. Once the arbiter grants the host access, the 82560 begins the local bus cycle by generating the appropriate address and control signals.

The host CPU can access the internal registers of the 82560, the local memory controlled by the 82560, or other devices—such as Boot ROM or external Latch—that share the same bus as the 82560. Table 2 lists the various access types that can be requested by the host.

<table>
<thead>
<tr>
<th>Access Type</th>
<th>HF1</th>
<th>HF0</th>
<th>Address</th>
<th>Cycle Status Indications</th>
</tr>
</thead>
<tbody>
<tr>
<td>82560 Registers</td>
<td>0</td>
<td>1</td>
<td>Between 8h and 3Fh</td>
<td>HRDY</td>
</tr>
<tr>
<td>Local Memory Access</td>
<td>1</td>
<td>0</td>
<td>User Defined</td>
<td>HRDY, Memory Control Signals, XCV Signals</td>
</tr>
<tr>
<td>GCS Access (Boot ROM) (General Chip Select)</td>
<td>0</td>
<td>0</td>
<td>User Defined</td>
<td>HRDY, GCS, IOWR, IORD/MWR</td>
</tr>
<tr>
<td>Special Chip Select</td>
<td>0</td>
<td>1</td>
<td>Less Than 8h</td>
<td>HRDY, DACK Lines, CS, IOWR, IORD/MWR</td>
</tr>
</tbody>
</table>
The 82560 provides eight semaphore ports to resolve contention in a shared resource system. Only the most significant bit of these ports is used. The CPU writes all 0's to the port to clear it. When the port is read, its current value is reported and the most significant bit becomes a 1 at the end of the cycle. The 82560 also supports devices on the local bus other than memory and the LAN controller. These devices can be accessed in two ways: by using the General Chip Select (GCS) signal, or by using the Special Chip Select (SCS) addresses in the 82560 register space. The first method typically supports EPROMs, external latches, and similar devices. The second method is used for accessing the registers of controllers which use the 82560 DMA channels; e.g., the 82590, 82592, or 82588. Each address in the SCS port provides a unique combination of the DACK0-, DACK1-, and CS-pin output states. The CPU activates the chip select of the device being accessed by asserting or deasserting the appropriate signals.

The host CPU can access the 82560 registers and other devices on the local bus at any time. However, local memory can only be accessed by the host after the 82560 memory control registers are initialized. The host accesses local memory in two ways: Page Access or Sequential (I/O-mapped or pipeline) Access. After reset, the memory access is I/O-mapped mode but host access to local memory is disabled. The 82560 must be configured for the appropriate memory access mode before local memory can be accessed by the host.

The 82560 memory control logic provides the signals required to interface to static memory. The 82560 can address up to 32 kB of local memory. Each memory address can refer to a byte, a word, or a double word of local memory. Thus the 82560 with its two memory chip selects (low to high bank) and its MOE and MWR outputs, can support 8-, 16-, or 32-bit-wide local buses.

In Page Access mode the local memory is mapped into the host memory space. In this mode the host can directly access local memory through a fixed size window which can be moved around in local memory space. This window is referred to as a "page". Figure 5 shows the paging scheme. The page size can vary from 1 kilobyte to 8 kilowords, and can be located anywhere in local memory. The exact location of the page in the local memory is defined by a page register. By reprogramming the page register the user can relocate the page in local memory.

In I/O-mapped Access mode the memory is mapped into the host I/O address space. Data is transferred between host and local memory using host DMA or string I/O instructions. The 82560 can be programmed to support memory accesses through a single I/O port. The I/O port is defined by an address programmed into an 82560 register. The 82560 maintains the current address, which is updated each time a memory cycle is run. The host does not directly access the local memory. It outputs the I/O address onto the A0–A12 address lines, with the HF lines indicating a memory access. If the I/O address matches the address programmed into the 82560, then the 82560 executes the local memory cycle by outputting the current address onto the memory address lines MA0–12.

The 82560 can be configured to interface with the host in a pipeline mode. In this mode, transparent or edge-triggered latches are needed to isolate the host and local bus during memory cycles. Data is written to the latch (from the host bus) and copied (from the latch) to the local memory. In the host read cycles, data is copied from the latch to the host bus. In anticipation of the next host memory request (sequential), the 82560 then copies the next byte or word from local memory to the latch. Thus the host CPU can operate with 0 wait states by reading from and writing to the latch.

**ARBITER**

All requests for access to devices on the local bus, whether by the host or by the 82560 DMA, undergo arbitration. The host requests are indicated on the HF lines; the DMA requests are indicated on the DRQ lines. Figure 4 shows the basic arbitration cycle of the 82560. Arbitration for the local bus is pipelined. It can take place at any time when the 82560 is idle, or one clock before the end of the current local bus cycle. All requests are sampled on the falling edge of the 82560 clock. Arbitration is completed within one clock cycle. The resultant local bus cycle is started on the falling edge of the next clock. If more than one request is active, arbitration is resolved on an alternating priority basis.

The 82560 deactivates its HRDY line when a host request is detected; the request is synchronized and then arbitrated. If the request is granted, the appropriate local bus cycle begins. After a programmable number of clock cycles HRDY will be reactivated, and the handshake with the host will be complete. DMA requests are synchronized and acknowledged once DMA has been granted access to the local bus. The acknowledge lines are kept inactive until the DMA is granted access to the local bus.
**DMA MACHINE**

The 82560 provides two DMA channels. Each channel can access 16 kb of memory address, and has request and acknowledge lines and address registers. The DMA normally operates in the Demand mode, and becomes active in response to a DMA request being granted. The requests come in on the DRQ lines and, if granted, are acknowledged by the DACK lines becoming active. Each channel has a control register that includes an enable bit, a direction bit, and output enable bits (CS, DACK0, and DACK1 are active low signals that can be enabled/disabled during DMA cycles). Each channel also has a base, current, stop, lower-limit, and upper-limit register. The current address register (CAR) is incremented after every DMA transfer except when in double host bus mode. The lower-limit register points to the beginning of the ring buffer; the upper-limit register points to the end of the ring buffer. The 82560 performs the wraparound (lower limit to CAR), each time the CAR equals the upper limit. When the contents of the CAR equal those of the stop register, DMA transfer stops and the 82560 generates an interrupt to the CPU. When the double host bus mode is invoked, the DMA machine will alternately activate low and high banks of memory and will increment the address after each high-bank transfer.
LOOSELY COUPLED MODE

The 82560 performs flyby DMA transfers (read from slave and write to memory or vice versa). The operation continues until the current address register equals the stop register or until the DRQ is removed. When the stop register is reached, the 82560 generates an interrupt.

82590 TIGHTLY COUPLED MODE

The Tightly Coupled Interface is a hardware interface between the 82560 and the 82590. This interface allows transmission and reception events to be processed without CPU intervention. It allows the implementation of the time-critical CSMA/CD processes: automatic retransmission, buffer reclamation, and continuous frame reception and transmission. The basic interface is a two-signal DMA handshake between the 82560 and the 82590; this occurs over the DRQ and EOP pins. The 82590 provides the status of the current transmit or receive process, or requests another DMA cycle at the end of each DMA cycle. When configured for the Tightly Coupled Interface, the 82560 and the 82590 use a specific interrupt scheme to minimize CPU overhead and to improve data throughput. The 82590 will not generate interrupts when events occur that can be handled by the 82560 without CPU intervention. Figure 5 illustrates the Tightly Coupled Interface mechanism. Table 3 lists the various combination of the DRQ and EOP signals, and the events they represent.

If both DRQ and EOP are sampled high, the Current Address Register of the channel is incremented and another DMA cycle begins. If a frame is transmitted or received without errors, both DRQ and EOP are low at the end of the DMA cycle and the 82560 will generate an interrupt. If DRQ is high and EOP is low, a collision occurred during transmission, or an error occurred during reception. In this case the Current Address Register will be reloaded with the value in the Base Address Register; and, once again, it will point to the beginning of the frame structure in memory.

The DACK1/CS1.EOP pin of the 82590 is multiplexed and requires external logic to derive the EOP and CS1 signals (see 82590 data sheet). Because the 82560 integrates this logic, its DACK1/EOP pin can be connected (with a pullup resistor) directly to the DACK1/CS1.EOP pin of the 82590, and its DACK0/DACK pin can be connected directly to the DACK pin of the 82590. For more details, see the 8259X Users Manual.

<table>
<thead>
<tr>
<th>DRQ</th>
<th>EOP</th>
<th>Event Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>Operation Done</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>Idle</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>Retry Request</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>New DMA Transfer Request</td>
</tr>
</tbody>
</table>

Table 3. DMA Handshake Encoding

Figure 6. 82560, 82590 DMA Handshake
Transmit

The 82590 can transmit consecutive frames without using the CPU to issue the Transmit command each time. This improves data throughput during transmission and eliminates CPU overhead. The CPU can place multiple transmit frames in memory, with each frame separated from the next by a Transmit Command byte. (For further information see 82590 and 82592 user manuals.) The 82560 supports transmit chaining. It also supports automatic retransmit on collision (provided that the maximum number of collisions is not reached). In this case the current address register is reloaded with the value of the base address register, and the DMA transfer is resumed without CPU involvement. If the maximum number of collisions has been reached, or if transmit failed for any other reason, the 82560 will need CPU intervention. Thus it will generate an interrupt to the CPU. At the end of transmission of each frame, the 82560 updates the status byte (indicating the number of collisions) in the memory.

Figure 7. Example of a 4-kB Transmit Ring Buffer
Receive

Immediately after a channel is enabled for receive, the 82560 will write FFh into the first two bytes of the frame (pointed to by the base register). The current address register is loaded with the contents of the base register and is incremented twice (past the two reserved bytes). If an error occurs during reception, and the save bad frame bit is 0, the CAR is reloaded with the content of the BAR and incremented past the two reserved bytes; however, if the save bad frame bit is set, the CAR is incremented for the next frame and the 82560 generates an interrupt to the CPU. If no error occurs, the last two bytes received (which are always stored in 82560 internal registers) are copied back to the first two bytes of the frame. These are the byte counts. If the 82590 generates an interrupt on each frame reception the 82560 will relay that interrupt to the CPU. At this time the value of the CAR will be copied into BAR, FF will be written into the next two bytes, and CAR will be incremented as before to point to the new frame reception area.

Figure 8. Example of a 4-kB Receive Ring Buffer
Table 4. Address Map

<table>
<thead>
<tr>
<th>Address</th>
<th>2Fh</th>
<th>3Fh</th>
<th>2Eh</th>
<th>3Eh</th>
<th>2Dh</th>
<th>3Dh</th>
<th>2Ch</th>
<th>3Ch</th>
</tr>
</thead>
<tbody>
<tr>
<td>RESERVED</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>HOST MODE REGISTER</td>
<td></td>
<td></td>
<td>2Fh</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>STOP 0</td>
<td></td>
<td></td>
<td>2Eh</td>
<td></td>
<td>3Eh</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RESERVED</td>
<td></td>
<td></td>
<td>2Bh</td>
<td></td>
<td>3Bh</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>UPPER LIMIT REGISTER 0</td>
<td></td>
<td></td>
<td>2Ah</td>
<td></td>
<td>3Ah</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RECEIVE TEMP. REGISTERS</td>
<td></td>
<td></td>
<td>27h</td>
<td></td>
<td>37h</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LOWER LIMIT REGISTER 0</td>
<td></td>
<td></td>
<td>26h</td>
<td></td>
<td>36h</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DMA CONTROL REGISTER 0</td>
<td></td>
<td></td>
<td>23h</td>
<td></td>
<td>33h</td>
<td></td>
<td></td>
<td>33h</td>
</tr>
<tr>
<td>BASE ADDRESS</td>
<td></td>
<td></td>
<td>22h</td>
<td></td>
<td>32h</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CURRENT ADDRESS</td>
<td></td>
<td></td>
<td>21h</td>
<td></td>
<td>31h</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DMA MODE REGISTER</td>
<td></td>
<td></td>
<td>1Fh</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>HOST ADDRESS</td>
<td></td>
<td></td>
<td>1Eh</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SELECT REGISTER</td>
<td></td>
<td></td>
<td>1Ch</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RESERVED</td>
<td></td>
<td></td>
<td>1Bh</td>
<td></td>
<td>1Ah</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>INT MASK REGISTER</td>
<td></td>
<td></td>
<td>19h</td>
<td></td>
<td>18h</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>INT CONTROL/STATUS REGISTER</td>
<td></td>
<td></td>
<td>17h</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>82588 STATUS 2 REGISTER</td>
<td></td>
<td></td>
<td>16h</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>82588 STATUS 1 REGISTER</td>
<td></td>
<td></td>
<td>15h</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RESERVED</td>
<td></td>
<td></td>
<td>14h</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CONTROL REGISTER</td>
<td></td>
<td></td>
<td>13h</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>IDENTIFICATION REGISTER</td>
<td></td>
<td></td>
<td>12h</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MASTER MODE REGISTER</td>
<td></td>
<td></td>
<td>11h</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SEMAPHORES</td>
<td></td>
<td></td>
<td>10h</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### SCS PORTS§

<table>
<thead>
<tr>
<th></th>
<th>CS</th>
<th>DACK1</th>
<th>DACK0</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1 1 1</td>
<td>07h</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1 1 0</td>
<td>06h</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1 0 1</td>
<td>05h</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1 0 0</td>
<td>04h</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0 1 1</td>
<td>03h</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0 1 0</td>
<td>02h</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0 0 1</td>
<td>01h</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0 0 0</td>
<td>00h</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**NOTE:**
When writing to 3-byte registers, the most significant byte (higher address) should be written last. The value written into the most significant bytes should be 0. The third bytes are reserved for possible future use.
82588 TIGHTLY COUPLED MORE

The 82560 supports a Tightly Coupled Interface (TCI) with the 82588. This interface allows transmit and receive events to be processed without CPU intervention. It allows the combination of the 82588 and 82560 to implement time-critical, CSMA/CD events: automatic retransmission, buffer reclamation, and continuous frame reception (see 82588 Reference Manual). When configured for the 82588 TCI mode, the 82560 uses the 82588 INT pin to determine if an event has occurred. The 82560 then reads the 82588 status register(s) to determine the cause of the interrupt. If the interrupt is due to a collision during transmission, a good frame reception, or errors during frame reception then the 82560 will update its DMA address registers and issue the 82588 the commands necessary for minimizing CPU intervention. The 82560 will regenerate all 82588 interrupts except those generated when a collision occurs during transmission (with the maximum retry count not exceeded). Because transmit and receive interrupts are time-critical processes the 82560 automatically acknowledges such interrupts to reduce dependency on the CPU. It will regenerate the interrupt on its INOUT pin unless the interrupt is due to a transmit collision.

If the 82588 issues an interrupt due to a collision during transmission, and the maximum retry count has not been exceeded, the 82560 will automatically reload the Current Address Register with the value in the Base Address Register, acknowledge the interrupt, and issue a retransmit command to the 82588. If the interrupt is due to the reception of a good frame, the 82560 will update its Base and Current Address Registers and prepare for a new incoming frame. If the interrupt is due to a receive frame error, the 82560 will reclaim the buffer by resetting the Current Address Register to the beginning of the frame buffer.

If the 82588 is unable to transmit due to having exceeded the maximum retry count or a Lost-CTS condition or a Lost-CRS condition, an interrupt is generated; the 82560 will not update its DMA address registers. It will, however, acknowledge the 82588 interrupt and regenerate the interrupt on its INOUT pin.

PROGRAMMING

The 82560 registers may be logically grouped into Device Configuration registers, Status registers and DMA address registers. Table 4 shows all of the 82560 registers and their addresses. All registers except receive temporary registers, 82588 status 1 and 2 registers, and the identification register, which are read only, are read/write registers.

The registers can be accessed by the host CPU. The RD signal indicates the direction of data transfer between the 82560 and the CPU. The actual data transfer takes place over the 82560's 8-bit data bus lines (D7–D0). The address of the register being accessed is taken from the address lines A5–A0.

Since the 82560's data bus is 8-bits wide, all access to its registers is on a byte basis. If a register is longer than 1 byte, each byte has to be accessed individually through its unique address in the 82560 register space.

On power-up or reset, the 82560 registers are set to a default configuration. The user must initialize the 82560 for the proper system configuration.

The SCS ports occupy eight addresses in the 82560 register space. The SCS ports should not be thought of as registers. They are merely addresses in the register space which, when addressed, activate a combination of the DACK0, DACK1 or CS pins. The particular combination of these pins signal levels depends on the SCS port address being accessed. The semaphore ports allow resource sharing in a dual processor (intelligent adapter) environment. Each port can be used as a semaphore to implement mutual exclusion.

CONFIGURATION REGISTERS

By programming these registers, the 82560 can be tailored to support different PCs, slaves and memories. The memory access mode (I/O or memory mapped) and the type of DMA support (loosely or tightly coupled) can also be programmed.
MASTER MODE REGISTER (10h)

- D7: 0 → Host bus interface
- D5, D4, D3: 0 → equal data bus width
- D5, D4, D3: 01 → double data bus width
- D5, D4, D3: 01 → double data bus width with special receive
- D5, D4, D3: 10 → reserved
- D5, D4, D3: Reserved
- D2: 0 → Base/Current select (1/0)
- D1: 0 → HRDY delay
  - 00: no delay
  - 01: 0.5 clock delay
  - 10: 1.5 clock delay
  - 11: 2.5 clock delay
- D0: Reserved
- D7: 1 → Host/DMA Idle priority (1 or 0)

* IN SOME VERSIONS OF 82560, THIS MODE IS NOT TESTED.

CONTROL REGISTER (12h)

- D4, D3: 00 → I/O access delay (0 to 3)
- D2, D1: 0 → Early/Late write option (1/0)
- D0 → Memory access delay (0 to 3)
- D7: 0 → Reserved

HOST MODE REGISTER (2Fh)

- D1, D0: 00 → Enable/Disable pipeline mode (1/0)
- D7: 0 → Pipeline direction read/write (1/0)
- D5, D4, D3, D2, D1: Reserved
- D6: 0 → General purpose input
INTERRUPT MASK REGISTER (17h)

- \( D_7 \ D_6 \ D_5 \ D_4 \ D_3 \ D_2 \ D_1 \ D_0 \)
- Low byte of the host-selected address
  (I/O-mapped memory access)

SELECT REGISTER, HIGH BYTE (1Bh)

- \( D_7 \ D_6 \ D_5 \ D_4 \ D_3 \ D_2 \ D_1 \ D_0 \)
- High bits of the host-selected address
- HRDY delay reference source
- Memory or I/O mapped (1/0)
- Enable/Disable memory access (1/0)

DMA MODE REGISTER (1Fh)

- \( D_7 \ D_6 \ D_5 \ 0 \ 0 \ 0 \ 0 \ 0 \)
- Reserved
- Save/Discard bad frame (1/0)
- DMA Mode
  - 00 loosely coupled (regular)
  - 10 8259X Tightly Coupled Interfaced
  - 01 82588 TCI
  - 11 reserved
INTERRUPT MASK REGISTER (17h)

<table>
<thead>
<tr>
<th>D7</th>
<th>D6</th>
<th>D5</th>
<th>D4</th>
<th>D3</th>
<th>D2</th>
<th>D1</th>
<th>D0</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Level/Edge sensitive (1/0)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>High/Low true assert (1/0)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>00</td>
<td>00 no change*</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>01</td>
<td>enable slave interrupt</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>10</td>
<td>disable slave interrupt</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>11</td>
<td>reserved</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Disable/Enable Tx chain (1/0)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Enable/Disable interrupt tri-state (1/0)</td>
</tr>
</tbody>
</table>

*Whenever one of these bits is “1” while writing to this register, other bits are not affected.

Host Address Registers

Contain the initial memory address when the host accesses memory in I/O mapped or pipeline mode.

Identification/Software Reset Register

Writing to this address will reset the chip. Reading from it will provide the user with 82560 stepping information.

MASTER MODE REGISTER (10h) DMA Control Register* (23h or 33h)

<table>
<thead>
<tr>
<th>D7</th>
<th>D6</th>
<th>D5</th>
<th>D4</th>
<th>D3</th>
<th>D2</th>
<th>D1</th>
<th>D0</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Disable/Enable DACKO during DMA cycles (1/0)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Disable/Enable, DACK1/EOP during DMA cycles (1/0)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Disable CS during DMA cycles (1/0)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Direction bit: memory read/write (1/0)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Enable/Disable DMA channel (1/0)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Receive/Execution channel (1/0)†</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Reserved</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>D5</th>
<th>D3</th>
<th>DMA Channel Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>Transmit</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>Dump (588 or 590/592)</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>Reserved (Do Not Use)</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>Receive</td>
</tr>
</tbody>
</table>

*Each DMA channel has its own control register.
†The following table shows the encoding of bits 3 and 5:
INTERRUPTS

In the non-tightly coupled mode, the 82560 will generate an interrupt when the Current Address register equals the Stop register or when the interrupt input pin is active.

In the tightly coupled modes, the conditions for generating Stop register interrupts are the same however, the 82560 will generate 82590 interrupts only if its source was one of the following.
- Transmission of every frame, or last frame, in the chain is completed (programmable).
- Transmission failed because of a collision, and the maximum number of Transmit retries is reached.
- Transmission failed for a reason other than collision; e.g., lost CRS/CTS.
- Reception failed, and the Save Bad Frame bit is set.
- Reception completed.

The interrupt control register is read by the interrupt routines to determine the exact source of the interrupt.

INTERRUPT STATUS READ REGISTER (16h)

The interrupt control register is written to acknowledge and reset the interrupt.

INTERRUPT CONTROL WRITE REGISTER (16h)
SYSTEM INTERACTION
A typical 82560 system interaction is described below.
1. The CPU configures the 82560 by writing to configuration registers.
2. The CPU accesses the local memory (through the 82560) and prepares a block of transmit frames.
3. The CPU writes the proper addresses into the 82560's DMA address registers, (base, current, lower limit upper limit and stop).
4. The CPU writes to the 82560's DMA control registers to configure and enable the channels.
5. The CPU issues a transmit command to the 82590.
6. The 82560 responds to the 82590's DMA request by transferring data from memory to the 82590.
7. Upon completion of transmission, the 82560 sends an interrupt to the CPU.
8. The CPU reads the 82560 interrupt control register to find the source of the interrupt.
9. The CPU issues a command to the 82590 to clear its interrupt. (If the source of the interrupt was the 82590.)
10. The CPU acknowledges the 82560 interrupt by writing a "1" into the corresponding interrupt control register bit(s).

APPLICATIONS
Figure 9 shows a buffered, nonintelligent StarLAN adapter for the IBM PC bus (using the 82560 and the 82590). Figure 10 shows a buffered, nonintelligent Ethernet adapter for the IBM PC bus (using the 82560, 82592, 82C501 and the 82502).

82560 MACHINE CYCLE
The 82560 machine cycle can be broken down into three basic cycles: Idle (T_{IDLE}), Arbitration (T_A) and Transfer (T_{TSF}). The machine cycle begins when a request (HF or DRQ) becomes active and the 82560 is in the idle state (T_{IDLE}). The requests are synchronized and then undergo arbitration (T_A). Once arbitration is completed, the transfer cycle (T_{TSF}) begins.

Synchronization (T_S) is completed on the falling edge of the clock. If the previous cycle was non-idle, arbitration begins and is completed within one clock period (by the next falling edge of the clock).

The Transfer cycle consists of the following sequential states: the first transfer state (T_1), memory or I/O wait states (T_w), and the second transfer state (T_2). There may be another transfer state, T_{wh} (wait host), during host read or pipeline cycles. When no requests are pending, and the 82560 is not in the transfer or arbitration cycle, it is said to be in the idle state (T_{IDLE}). If the previous cycle was non-idle, the arbitration period (T_A and T_2 of the previous cycle will be done in parallel. (See Figure 4.)

T_w is the programmable portion of the transfer cycle. It can be zero to three clocks long depending on the programmed memory or I/O access delays. If the programmed delay is zero, then there will be no T_w; the first state of the transfer cycle will be T_1. During T_2 the transfer cycle is completed unless the cycle is a host read cycle. In that case the cycle will be extended by inserting T_{wh}. The 82560 will remain in T_{wh} until the HF lines are deasserted. Once HF lines are deasserted, T_2 will begin and one clock period later the bus cycle is complete.
ABSOLUTE MAXIMUM RATINGS*

Case Temperature (TC) under Bias ...................... 0°C to +85°C
Storage Temperature ............ -65°C to +150°C
Voltage on any Pin with Respect to Ground ............ -0.5V to VCC + 0.5V

*Notice: Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions above those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

NOTICE: Specifications contained within the following tables are subject to change.

D.C. CHARACTERISTICS TC = 0°C to +85°C, VCC = +5V ±10%
CLK pin has MOS levels (see \textit{V\textsubscript{MIL}}, \textit{V\textsubscript{MIH}}) All other signals have TTL levels (see \textit{VIL}, \textit{VIH}, \textit{VOL}, \textit{VOH}).

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Parameter</th>
<th>Min</th>
<th>Max</th>
<th>Units</th>
<th>Test Conditions</th>
</tr>
</thead>
<tbody>
<tr>
<td>\textit{VIL}</td>
<td>Input Low Voltage (TTL)</td>
<td>-0.5</td>
<td>+0.8</td>
<td>V</td>
<td></td>
</tr>
<tr>
<td>\textit{VIH}</td>
<td>Input High Voltage (TTL)</td>
<td>2.0</td>
<td>VCC + 0.5</td>
<td>V</td>
<td></td>
</tr>
<tr>
<td>\textit{VOL}</td>
<td>Output Low Voltage (TTL)</td>
<td>0.45</td>
<td>V</td>
<td></td>
<td>\textit{IOL} = 3.2 mA</td>
</tr>
<tr>
<td>\textit{VOH}</td>
<td>Output High Voltage (TTL)</td>
<td>2.4</td>
<td>VCC</td>
<td>V</td>
<td>\textit{IOH} = -400 \mu A</td>
</tr>
<tr>
<td>\textit{V\textsubscript{MIL}}</td>
<td>Input Low Voltage (MOS)</td>
<td>-0.5</td>
<td>0.6</td>
<td>V</td>
<td></td>
</tr>
<tr>
<td>\textit{V\textsubscript{MIH}}</td>
<td>Input High Voltage (MOS)</td>
<td>VCC - 0.6</td>
<td>VCC + 0.5</td>
<td>V</td>
<td></td>
</tr>
<tr>
<td>\textit{I\textsubscript{IL}}</td>
<td>Input Leakage Current</td>
<td>±10</td>
<td>\mu A</td>
<td></td>
<td>0 = \textit{VIN} = VCC - 0.45</td>
</tr>
<tr>
<td>\textit{I\textsubscript{IO}}</td>
<td>I/O Leakage Current</td>
<td>±10</td>
<td>\mu A</td>
<td></td>
<td>0.45 = \textit{VOUT} = VCC - 0.45</td>
</tr>
<tr>
<td>\textit{\textsubscript{CIN}}</td>
<td>Capacitance of Input Buffer</td>
<td>10</td>
<td>pF</td>
<td>FC = 1 MHz</td>
<td></td>
</tr>
<tr>
<td>\textit{\textsubscript{COUT}}</td>
<td>Capacitance of Input/Output Buffer</td>
<td>20</td>
<td>pF</td>
<td>FC = 1 MHz</td>
<td></td>
</tr>
<tr>
<td>\textit{\textsubscript{ICC}}</td>
<td>Power Supply Current</td>
<td>50</td>
<td>mA</td>
<td>10 MHz</td>
<td></td>
</tr>
</tbody>
</table>

A.C. CHARACTERISTICS \textsubscript{CL} on all outputs is 50 pF. The user should add 0.2 ns/pF up to 100 pF

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Parameter</th>
<th>Min</th>
<th>Max</th>
<th>Test Conditions</th>
</tr>
</thead>
<tbody>
<tr>
<td>\textit{T1}</td>
<td>CLK Cycle Period</td>
<td>100</td>
<td></td>
<td>(Note 1)</td>
</tr>
<tr>
<td>\textit{T2}</td>
<td>CLK Low Time</td>
<td>45</td>
<td></td>
<td>(Note 1)</td>
</tr>
<tr>
<td>\textit{T3}</td>
<td>CLK High Time</td>
<td>45</td>
<td></td>
<td>(Note 1)</td>
</tr>
<tr>
<td>\textit{T4}</td>
<td>CLK Rise Time</td>
<td>5</td>
<td></td>
<td>(Note 2)</td>
</tr>
<tr>
<td>\textit{T5}</td>
<td>CLK Fall Time</td>
<td>5</td>
<td></td>
<td>(Note 3)</td>
</tr>
<tr>
<td>\textit{T6}</td>
<td>HF or DREQ Setup Time</td>
<td>10</td>
<td></td>
<td></td>
</tr>
<tr>
<td>\textit{T7}</td>
<td>HF Active Time (Low)</td>
<td>2*T1 + 10</td>
<td></td>
<td>(Note 5)</td>
</tr>
<tr>
<td>\textit{T8}</td>
<td>HF Inactive Time (High)</td>
<td>T1 + 10</td>
<td></td>
<td></td>
</tr>
<tr>
<td>\textit{T9}</td>
<td>HF to HRDY Low</td>
<td>50</td>
<td></td>
<td></td>
</tr>
<tr>
<td>\textit{T10}</td>
<td>HF Active to HDRY High</td>
<td>2*T1 + 50</td>
<td></td>
<td>(Note 4)</td>
</tr>
<tr>
<td>\textit{T11}</td>
<td>HRDY High to HF Inactive</td>
<td>0</td>
<td></td>
<td>(Note 5)</td>
</tr>
</tbody>
</table>
### A.C. CHARACTERISTICS

C\textsubscript{L} on all outputs is 50 pF. The user should add 0.2 ns/pF up to 100 pF (Continued)

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Parameter</th>
<th>Min</th>
<th>Max</th>
<th>Test Conditions</th>
</tr>
</thead>
<tbody>
<tr>
<td>HOST ACCESS CYCLE—NON PIPELINE MODE PARAMETERS (Continued)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T12</td>
<td>HF Inactive to HRDY Float</td>
<td>75</td>
<td></td>
<td></td>
</tr>
<tr>
<td>T13</td>
<td>HF Active to XCVR Lines Low</td>
<td>T1 + T2</td>
<td>2*T1 + T2 + T2 + 75</td>
<td>(Note 6)</td>
</tr>
<tr>
<td>T14</td>
<td>HF Inactive to XCVR Lines High</td>
<td>(Note 7)</td>
<td>75</td>
<td></td>
</tr>
<tr>
<td>T15</td>
<td>HF Active to RD Low</td>
<td>T1 + T2 + 10</td>
<td></td>
<td></td>
</tr>
<tr>
<td>T16</td>
<td>RD Hold after HF Inactive</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>T17</td>
<td>HF Active to Input Add. Valid</td>
<td>-20</td>
<td></td>
<td></td>
</tr>
<tr>
<td>T18</td>
<td>Address Hold after HF Inactive</td>
<td>0</td>
<td></td>
<td>(Note 8)</td>
</tr>
<tr>
<td>T19</td>
<td>HF Active to 82560 Data Valid</td>
<td>3*T1 + 80</td>
<td></td>
<td>(Note 9)</td>
</tr>
<tr>
<td>T20</td>
<td>Data Hold after HF Inactive</td>
<td>T1 + T2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>T21</td>
<td>HF Active to 82560 Add Valid (MAn).</td>
<td>2*T1 + T2 + 75</td>
<td></td>
<td>(Note 9)</td>
</tr>
<tr>
<td>T22</td>
<td>Add Valid or Chip Select Active Time</td>
<td>2*T1</td>
<td></td>
<td>(Note 10)</td>
</tr>
<tr>
<td>T23</td>
<td>HF Active to CS Active</td>
<td>2*T1 + T2 + 50</td>
<td></td>
<td>(Note 9)*</td>
</tr>
<tr>
<td>T24</td>
<td>CS Enveloping Controls</td>
<td>20</td>
<td></td>
<td></td>
</tr>
<tr>
<td>T25</td>
<td>Control Active Time</td>
<td>(Note 11)</td>
<td>(Note 11)</td>
<td></td>
</tr>
<tr>
<td>T26</td>
<td>HF to Data Valid</td>
<td>3*T1-30</td>
<td></td>
<td></td>
</tr>
<tr>
<td>T27</td>
<td>Data Hold after HRDY High</td>
<td>(Note 12)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### HOST ACCESS CYCLE—PIPELINE MODE PARAMETERS

| T28    | HF Active Time | T1 + 10 | (Note 13) | (Note 9) |
| T29    | HF Active to Port CS Active | 2*T1 + 75 |          | (Note 6) |
| T30    | HF Inactive to HRDY Low | 75      |          |                |
| T31    | HRDY Low to HRDY High | (Note 14) |          |                |
| T32    | Port CS Active Time | 2*T1 |          | (Note 14) |
| T33    | HF Inactive to Buffer Write | 10      |          |                |
| T34    | Write Active Time | T1-10 | T1 + 10 |          |                |

### DMA PARAMETERS

| T35    | DRQn High or INTR to Clock Low Setup Time | 50      |          | (Note 15) |
| T36    | DRQn Low to Clock Low, Hold Time | 10      |          |          |
| T37    | EOP Pulse Width | T1      |          |          |
| T38    | Address Delay Time | T2 + 75 |          |          |
| T39    | CS, CSn, DAKn Delay Time | T2 + 50 |          |          |
| T40    | CSn Delay Time (Slave to SRAM Flyby) | 50      |          |          |
| T41    | IORD—MWR, IOWR Delay Time | 45      |          |          |
| T42    | IORD—MWR, IOWR Active Time | (Note 16) | (Note 16) |          |
A.C. CHARACTERISTICS

C_L on all outputs is 50 pF. The user should add 0.2 ns/pF up to 100 pF (Continued)

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Parameter</th>
<th>Min</th>
<th>Max</th>
<th>Test Conditions</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>INTERRUPT PARAMETERS</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T43</td>
<td>Interrupt Delay Time</td>
<td>75</td>
<td></td>
<td></td>
</tr>
<tr>
<td>T44</td>
<td>Interrupt Gap</td>
<td>3*T1-10</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>RESET PARAMETERS</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T45</td>
<td>Reset Setup Time</td>
<td>50</td>
<td></td>
<td></td>
</tr>
<tr>
<td>T46</td>
<td>Reset Active Time (High)</td>
<td>4*T1</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

*For pin HRDY 4 mA.

NOTES:

1. Measured at VCC/2.
2. 3.2V to 1.8V.
3. 1.8V to 3.2V.
4. The following configuration affect the HRDY output going active (high).

Legend:
- TID—The configuration of HRDY delay (master mode register, TID = 0,1,2,3).
- TIO—The configuration of I/O access delay (Control register, TIO = 0,1,2,3).
- TMEM—The configuration of MEM access delay (Control register, TMEM = 0,1,2,3).
- T25—The configuration of 82560 control (Register 25).
- T10—The configuration of 82560 control (Register 10).
- T25—The configuration of 82560 control (Register 25).

4.1 If bit 5 of Register at Address 1BH then 
TID = TID + (greater of TIO and TMEM) + 2 + T1 + 75
else TID = TID + (greater of TIO and TMEM) + 2 + T1 + 75

5. The user should not that the XCVR lines goes inactive immediately after HF inactivation.
6. Provided that the HOST wins arbitration.
7. In the case of HOST write cycle the XCVR lines will go high at the end of the 82560 cycle even if HF lines are still active. In the case of HOST read cycles, the 82560 will terminate the local cycle after HF lines are inactivated.
8. Address lines are latched at the end of T1 of 82560 HOST bus cycles.
9. The maximum time specified assumes that the HOST wins the arbitration. If the HOST loses the arbitration to a DMA request two possible scenarios are:
   a) Arbitration lost to a single DMA cycle. In this case [(greater of TIO and TMEM) + 2] + T2 should be added to the max. time.
   b) Arbitration lost to a DMA cycle which is followed by four locked DMA cycles. In this case [(greater of TIO and TMEM) + 5 + 10] + T2 should be added to the max. time. This might happen in the rare case when the HOST request coincides with the last receive or transmit transfer, in the TCl mode.
   In the case of long (HF) HOST memory read requests, it would be extended until the request is removed.
12. This parameter depends on T10. In terms of machine states, data remains valid until the end of the cycle (end of state T2).
13. (TMEM + 1)*T1 + 75 + Tsystem.
    Tsystem = delay from HRDY to HF inactive.
    This maximum time refers to a second memory request immediately following a first one, assuming that the first one was not delayed by a DMA cycle.
14. [TIO(or TMEM) + 2]*T1 + 75.
15. This is an asynchronous signal (DRQn only in its leading edge). It is internally synchronized. Meeting this parameter, assures recognition on the next clock.
16. Min = [(greater of TIO and TMEM) + 1] + T1 + T2 + 10
    Max = [(greater of TIO and TMEM) + 1] + T1 + T2 + 10
A.C. Testing Inputs are Driven at 2.4V for a Logic "1" and 0.45V for a Logic "0". Timing Measurements are made at 1.5V for both a Logic "1" and "0".
WAVEFORMS (Continued)

HOST WRITE CYCLE—NON PIPELINE MODE

[Diagram showing timing relationships between signals like CLK, HFm, HRDY, XCv, RD, A0:A12, BMn, Data Valid, MA0:12, CSn, GCS, Cs, DACKn, IORD/MWR, IOWR, and their relationship to T0, T1, T2, T3, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14, T17, T18, T21, T22, T23, T24, T25, T26, and T27.

Note: The diagram illustrates the timing for a host write cycle in non-pipeline mode, showing the sequence of events and signal interactions.]
WAVEFORMS (Continued)

HOST READ CYCLE—PIPELINE MODE

- CLK
- HF
- HRDY
- XCl
- RD
- BMn
- MAO-12,
- CSn
- XCV2/PCS
- IOWR
- MOE
WAVEFORMS (Continued)

HOST WRITE CYCLE—PIPELINE MODE

DMA FLYBY CYCLE—SLAVE TO SRAM
WAVEFORMS (Continued)

DMA FLYBY CYCLE—SRAM TO SLAVE

[Diagram showing waveforms with labeled time periods such as T5, T6, T1, etc.]

290180-28
WAVEFORMS (Continued)

INTERRUPT

```
<table>
<thead>
<tr>
<th>CLK</th>
<th>T43</th>
<th>T43</th>
<th>T44</th>
</tr>
</thead>
<tbody>
<tr>
<td>INT</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>INTR</td>
<td>T35</td>
<td></td>
<td>T35</td>
</tr>
</tbody>
</table>
```

RESET

```
<table>
<thead>
<tr>
<th>CLK</th>
<th>T46</th>
</tr>
</thead>
<tbody>
<tr>
<td>RESET</td>
<td>T45</td>
</tr>
</tbody>
</table>
```
82590
ADVANCED CSMA/CD LAN CONTROLLER WITH 8-BIT DATA PATH

- Supports Industry Standard LANs
  - Ethernet and Cheapernet (IEEE 802.3 10BASE5 and 10BASE2)
  - StarLAN (IEEE 802.3 1BASE5)
  - IBM™ PC Network—Baseband and Broadband
- Integrates Physical and Data Link Layers of OSI Model
  - Complete CSMA/CD Medium Access Control (MAC) Functions
  - Manchester, Differential Manchester, or NRZI Encoding/Decoding
  - On-Chip, Logic-Based Collision Detection
  - IEEE 802.3 or HDLC Frame Delimiting
  - Broadband Ethernet (IEEE 802.3 10BROAD36)
- Two Modes of Operation
  - Bit Rates up to 4 Mb/s with On-Chip Encoder/Decoder (High-Integration Mode)
  - Bit Rates up to 20 Mb/s with External Encoder/Decoder (High-Speed Mode)
- High-Performance System Interface
  - 16-MHz Clock, 2 Clocks per Transfer
  - 64 Bytes of Configurable FIFO
- Efficient Memory Use via Buffer and Frame Chaining
- DMA Interface for Retransmission and Continuous Reception without CPU Intervention
  - EOP Signal Generation for 8237 and 82380
  - Tightly Coupled Interface to 82560 Host Interface and Memory Manager
- 82588 Pin- and Software-Compatible Mode
- Local and Remote Power-Down Modes
- Deterministic Collision Resolution
- 24-Bit General Purpose Timer
- On-Chip Jabber Inhibit Function
- Network Management and Diagnostics
  - Monitor Mode
  - CRC, Alignment, and Short Frame Error Detection
  - Three 16-Bit Event Counters
  - Short or Open Circuit Localization
  - Self-Test Diagnostics
  - Internal and External Loopback Operation
  - Internal Register Dump
- High-Speed CHMOS III Technology

**Figure 1. 82590 Block Diagram**

*IBM, PC, PCAT, PCXT are trademarks of International Business Machines.*
The 82590 is a second-generation, 8-bit data path CSMA/CD controller. Its system interface enables efficient operation with a wide variety of Intel microprocessors (such as iAPX 188, 186, 286, or 386) and industry standard buses (such as the IBM PC I/O channel or Personal System/2™ Micro Channel™). The 82590 can be configured to support a wide variety of industry standard networks, including StarLAN and Ethernet/Cheapernet.

The 82590 provides a natural upgrade path for existing 82588 applications, since it is pin and software compatible with its predecessor. Its rich incremental functionality compared to the 82588 can be utilized by selectively modifying existing software drivers.

Together with the 82560 (Host Interface and Memory Manager) the 82590 offers a complete solution for CSMA/CD LAN adapters oriented to the IBM PC environment. The 82590 fully conforms to existing IEEE 802.3 standards (1BASE5, 10BASE5, 10BASE2, and 10BROAD36). Intel also offers the 82592, a 16-bit data path version of the 82590, for higher performance applications.

The 82590 is available in a 28-pin Plastic DIP or a 44-pin PLCC package. It is fabricated with Intel's reliable CHMOS III technology.
Table 1. 82590 Pin Description

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Pin No. (DIP)</th>
<th>Pin No. (PLCC)</th>
<th>Type</th>
<th>Name and Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>D7</td>
<td>6</td>
<td>10</td>
<td>I/O</td>
<td>DATA BUS—The Data Bus lines are bidirectional, three-state lines connected to the CPU’s Data Bus for transfers of data, commands, status, and parameters.</td>
</tr>
<tr>
<td>D6</td>
<td>7</td>
<td>11</td>
<td></td>
<td>READ—Together with CS0, CS1, DACK0, or DACK1, Read controls data or status transfers out of the 82590.</td>
</tr>
<tr>
<td>D5</td>
<td>8</td>
<td>12</td>
<td></td>
<td>WRITE—Together with CS0, CS1, DACK0, or DACK1, Write controls data or command transfers into the 82590.</td>
</tr>
<tr>
<td>D4</td>
<td>9</td>
<td>13</td>
<td></td>
<td>CHIP SELECT (PORT 0)—When LOW, the 82590 is selected by the CPU for command or status transfer through PORT 0.</td>
</tr>
<tr>
<td>D3</td>
<td>10</td>
<td>14</td>
<td></td>
<td>RESET—A HIGH signal on this pin causes the 82590 to terminate current activity. This signal is internally synchronized and must be held HIGH for at least four Clock (CLK) cycles. When the Clock signal is provided internally (CLKSRC is strapped HIGH), the RESET signal must be held HIGH for at least 50 μs. (PLCC version only.)</td>
</tr>
<tr>
<td>D2</td>
<td>11</td>
<td>18</td>
<td></td>
<td>DMA REQUEST (CHANNEL 0)—This pin is used by the 82590 to request DMA transfer. DRQ0 remains HIGH as long as the 82590 requires DMA transfers. Burst transfers are thus possible. When the 82590 is programmed for Tightly Coupled Interface, the 82590 notifies the DMA controller of the status of transmission or reception, using this pin together with EOP.</td>
</tr>
<tr>
<td>D1</td>
<td>12</td>
<td>19</td>
<td></td>
<td>DMA REQUEST (CHANNEL 1)—This pin is used by the 82590 to request DMA transfer. DRQ1 remains HIGH as long as the 82590 requires DMA transfers. Burst transfers are thus possible. When the 82590 is programmed for Tightly Coupled Interface, the 82590 notifies the DMA controller of the status of transmission or reception, using this pin together with EOP.</td>
</tr>
<tr>
<td>D0</td>
<td>13</td>
<td>20</td>
<td></td>
<td>DMA ACKNOWLEDGE (CHANNEL 0)—When LOW, this input signal from the DMA controller notifies the 82590 that the requested DMA cycle is in progress. This signal acts similarly to Chip Select for data and parameter transfers, using DMA channel 0.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>I</td>
<td>DMA ACKNOWLEDGE (CHANNELS 0 AND 1)—When the DACK1/CS1/EOP pin is programmed to CS1/EOP, this pin provides a DMA acknowledge for both channels 0 and 1. Two DMA acknowledge signals from the DMA controller, DACK0 and DACK1, must be externally ANDed in this mode of operation.</td>
</tr>
</tbody>
</table>
### Table 1. 82590 Pin Description (Continued)

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Pin No. (DIP)</th>
<th>Pin No. (PLCC)</th>
<th>Type</th>
<th>Name and Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>DACK1 CST/EOP</td>
<td>27</td>
<td>42</td>
<td>I</td>
<td>This is a multifunction, bidirectional pin which can be programmed to DACK1 or CST/EOP during configuration. When it is configured for EOP, it provides an open-drain output.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>I/O</td>
<td><strong>DMA ACKNOWLEDGE (CHANNEL 1)</strong>—When LOW, this input signal from the DMA controller notifies the 82590 that the requested DMA cycle is in progress. This signal acts similarly to Chip Select for data and parameter transfers, using DMA channel 1.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>I/O</td>
<td><strong>CHIP SELECT (PORT 1)</strong>—When LOW, the 82590 is selected by the CPU for command or status transfer through PORT 1.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td><strong>END OF PROCESS</strong>—A LOW output signal requests the DMA controller to terminate the active DMA service.</td>
</tr>
<tr>
<td>CLK</td>
<td>4</td>
<td>5</td>
<td>I</td>
<td><strong>CLOCK</strong>—In the 28-pin DIP, this is only an input pin. A TTL-compatible clock input to this pin provides the timing for the 82590 parallel subsystem.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>I/O</td>
<td>In the 44-pin PLCC, this pin can be a clock input or output, depending on the state of CLKSRC. If CLKSRC is strapped LOW, this pin is a clock input which provides timing for the 82590 parallel subsystem. If CLKSRC is strapped HIGH, the clock for the 82590 parallel subsystem is generated from the internal clock generator. The CLK pin is then a clock output and provides a clock signal whose frequency can be one-half of or identical to, the frequency of the internally generated parallel subsystem clock, depending on the state of FREQ. Note that the maximum frequency of the clock signal supplied by the CLK pin is 8 MHz.</td>
</tr>
<tr>
<td>CLKSRC FREQ</td>
<td>Clock for the Parallel Subsystem</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0 (LOW)</td>
<td>Don't Care</td>
<td>0</td>
<td>O</td>
<td><strong>Internal Parallel Subsystem Clock Divided by Two</strong></td>
</tr>
<tr>
<td>1 (HIGH)</td>
<td>1</td>
<td>Internal Parallel Subsystem Clock</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0</td>
<td>Prescaled Clock Generated from the Internal Clock Generator</td>
</tr>
<tr>
<td>FREQ</td>
<td></td>
<td></td>
<td>0</td>
<td>Prescaled Clock Generated from the Internal Clock Generator</td>
</tr>
</tbody>
</table>

**CLOCK SOURCE**—When strapped LOW, a clock signal on the CLK pin provides timing for the parallel subsystem. When strapped HIGH, timing for the parallel subsystem is internally generated from the clock generator provided in the serial subsystem. The internal prescaler is programmed during configuration to determine the frequency of the clock for the parallel subsystem.

**FREQUENCY**—When strapped LOW, CLK has an output frequency equal to that of the internal parallel subsystem clock. When strapped HIGH, CLK has an output frequency one-half that of the internal parallel subsystem clock. The state of this pin is relevant only when CLKSRC is strapped HIGH.
## Table 1. 82590 Pin Description (Continued)

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Pin No. (DIP)</th>
<th>Pin No. (PLCC)</th>
<th>Type</th>
<th>Name and Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>X1/X2</td>
<td>15/16</td>
<td>24/25</td>
<td>I</td>
<td><strong>High Integration Mode</strong> &lt;br&gt;<strong>OSCILLATOR INPUTS</strong>—These inputs may be used to connect a quartz crystal which controls the internal clock generator for the serial subsystem. When CLKSRC is strapped HIGH, the clock generator also provides a clock for the parallel subsystem. X1 may also be driven by a MOS-level clock whose frequency is 8, 10, 16, or 18 times the bit rate of Transmit/Receive data. X2 must be left floating if X1 is connected to an external MOS clock.</td>
</tr>
<tr>
<td>TxC</td>
<td>15</td>
<td>24</td>
<td>I</td>
<td><strong>High Speed Mode</strong> &lt;br&gt;<strong>TRANSMIT CLOCK</strong>—This signal provides the fundamental timing for the serial subsystem. The clock is also used to transmit data synchronously on the TxD pin. For NRZ encoding, data is transferred to the TxD pin on the HIGH to LOW clock transition. For Manchester encoding, the transmitted bit center is aligned with the LOW to HIGH transition.</td>
</tr>
<tr>
<td>RxC</td>
<td>16</td>
<td>25</td>
<td>I</td>
<td><strong>RECEIVE CLOCK</strong>—This clock is used to synchronously sample data on the RxD pin. Only NRZ data format is supported for reception. The state of the RxD pin is sampled on the HIGH to LOW transition.</td>
</tr>
<tr>
<td>TCLK/CRS</td>
<td>24</td>
<td>36</td>
<td>I</td>
<td><strong>CARRIER SENSE</strong>—In High Speed Mode this pin is Carrier Sense, CRS, and is used to notify the 82590 that the serial link is active &lt;br&gt;<strong>TRANSMIT CLOCK</strong>—In High Integration Mode this pin is Transmit Clock, TCLK.</td>
</tr>
<tr>
<td>CDT</td>
<td>23</td>
<td>35</td>
<td>I</td>
<td><strong>COLLISION DETECT</strong>—This input notifies the 82590 that a collision has occurred. In High Speed Mode a collision is sensed by this pin only when the 82590 is configured for external Collision Detect (external means are then required for collision detection). In High Integration Mode collisions are sensed by this pin regardless of the internal or external Collision Detect configuration of the 82590.</td>
</tr>
<tr>
<td>RxD</td>
<td>19</td>
<td>31</td>
<td>I</td>
<td><strong>RECEIVE DATA</strong>—This pin receives serial data. It must be HIGH when not receiving.</td>
</tr>
<tr>
<td>TxD</td>
<td>20</td>
<td>32</td>
<td>O</td>
<td><strong>TRANSMIT DATA</strong>—This pin transmits data to the serial link. It is HIGH when not transmitting.</td>
</tr>
<tr>
<td>RTS</td>
<td>21</td>
<td>33</td>
<td>O</td>
<td><strong>REQUEST TO SEND</strong>—When this signal is LOW the 82590 notifies the channel that it has data to transmit. It is forced HIGH after a reset or when transmission is stopped.</td>
</tr>
<tr>
<td>CTS/LPBK</td>
<td>22</td>
<td>34</td>
<td>I/O</td>
<td><strong>CLEAR TO SEND</strong>—An active LOW signal which enables the 82590 to start transmitting data. Asserting this signal HIGH stops the transmission.  &lt;br&gt;<strong>LOOPBACK</strong>—This pin, in conjunction with a pull-down resistor, can be programmed to provide an active HIGH loopback signal to the external interface device.</td>
</tr>
<tr>
<td>VCC</td>
<td>28</td>
<td>1</td>
<td>POWER: +5V ±10%</td>
<td></td>
</tr>
<tr>
<td>Vss</td>
<td>14</td>
<td>21/22/23</td>
<td>GROUND: 0V</td>
<td></td>
</tr>
</tbody>
</table>
FUNCTIONAL DESCRIPTION

Internal Architecture

The 82590 consists of a parallel subsystem, a serial subsystem, and a FIFO subsystem (see Figure 1).

Parallel Subsystem

The parallel subsystem consists of a bus interface unit (BIU), command and status registers, a 24-bit general purpose timer, and three 16-bit event counters.

The BIU provides an 8-bit data bus interface to the external system bus. It handles all data transfers to and from memory (at speeds up to 8 Mbytes/sec.), accepts commands from the CPU, and provides status to the CPU. There are two separate I/O ports, Port 0 and Port 1; and two separate DMA channels, Channel 0 and Channel 1. Port 0 is the 82588-compatible I/O port through which the CPU issues commands such as Transmit and Receive Enable. The 82590's enhanced features, such as the general purpose timer and event counters, are accessed through Port 1. The two DMA channels are independent of each other and can be used for high-performance operations such as simultaneous transmission and reception.

The 24-bit timer consists of a 24-bit maximum count register, a 24-bit count register, and associated control bits in the command registers. Its clock source can be the transmit clock or the parallel subsystem clock. The timer can be programmed to halt or continue on a terminal count with or without causing an interrupt.

The three 16-bit event counters can be programmed to count valid frames, collided frames, and errored (CRC or Alignment) frames. When these event counters are used in Monitor mode, the 82590 is capable of maintaining the network statistics by itself; i.e., without requesting DMA services or causing interrupts to the CPU.

Serial Subsystem

The serial subsystem consists of a CSMA/CD unit, a data encoder and decoder, collision detect and carrier sense logic, and a clock generator.

The 82590's CSMA/CD unit is highly flexible in implementing the CSMA/CD protocol. It can operate in a variety of IEEE 802.3 and other CSMA/CD LAN environments, including 1BASE5 (StarLAN, 10BASE5 (Ethernet), 10BASE2 (Cheapernet), and the IBM™ PC Network (Baseband and Broadband). The programmable parameters include:

- Framing (IEEE 802.3 Framing or HDLC Framing)
- Address Field Length
- Station Priority
- Interframe Spacing
- Slot Time
- CRC-32 or CRC-16

The CSMA/CD unit also has a mode of operation which implements deterministic collision resolution (DCR). The DCR algorithm is fully compatible with the MULTIBUS™ II Serial System Bus (SSB) specifications.

The encoder and decoder in the serial subsystem is capable of NRZI, Manchester, and Differential Manchester encoding and decoding at bit rates up to 4 Mb/s in High-Integration Mode, and Manchester encoding at bit rates up to 20 Mb/s in High-Speed Mode. A digital phase-lock loop is used in High-Integration Mode to decode the receive data and to generate the synchronous receive clock.

The collision detect and carrier sense logic generate the internal collision detect and carrier sense signals for the CSMA/CD unit.

The 82590 implements several different internal, logic-based collision detect mechanisms. Two of these, Code Violation and Bit Comparison, are also available with the 82588. The Code Violation method defines a collision where a transition edge occurs outside the area of normal transitions (as specified by the data encoding method). For example, if there are no mid-bit cell transitions in the Manchester encoded data, this method interprets that condition as a collision. The Bit Comparison method compares the signature of the transmitted frame to the signature of the received frame. If the signatures are different, a collision is assumed to have occurred. Two other internal collision detect methods implemented in the 82590 are Source Address Comparison and StarLAN CPS (Collision Presence Signal) Recognition. The Source Address Comparison compares the source address field of the transmitted frame to the source address field of the received frame. If the source addresses are different, it assumes that a collision has occurred resulting in data corruption in the source address field. The StarLAN CPS Recognition method looks for the specific collision presence signal defined by the IEEE 802.3 1BASE5 standard. Other abnormal circumstances, such as no carrier for more than one-half slot time in the receive channel during transmission, are interpreted as collisions by the 82590.
In addition to these internal, logic-based collision detection methods, an external means of collision detection can be used in parallel by using the CDT input pin.

The clock generator in the serial subsystem is available only in High-Integration Mode and provides timing for the serial subsystem. The clock signal can also be routed to the parallel subsystem, if so desired. The oscillator circuit is designed for use with an external, parallel resonant, fundamental mode crystal. The crystal frequency should be selected at $8 \times$, $10 \times$, $16 \times$, or $18 \times$ the required serial bit rate.

**FIFO Subsystem**

The FIFO subsystem is located between the parallel subsystem and the serial subsystem. It consists of a transmit FIFO, a receive FIFO, and FIFO control logic. The transmit and receive FIFOs are independent of each other and individually provide optimal interfaces between the two subsystems which may have different speeds. There is a total of 64 bytes that can be used for the two separate FIFOs. During configuration these 64 bytes can be divided into one of four possible combinations: 16 and 16 bytes, 16 and 48 bytes, 32 and 32 bytes, or 48 and 16 bytes for the transmit and receive FIFO respectively. The FIFO threshold is also programmed during configuration.

**PROGRAMMING MODEL—REGISTER OVERVIEW**

Figure 4 shows the 82590 internal registers that are directly accessible through the 8-bit I/O ports: Port 0 and Port 1. The registers enclosed in darker lines are 82588-compatible registers and are accessible only through Port 0.

Figure 5 shows the Port 0 commands. All of the Port 0 commands are compatible with the 82588 except for the NOP command with the channel bit set to 1. If the NOP command is executed with the channel bit set to 1, the active port is switched to Port 1. Port 0, which is selected by CS0 in hardware, logically becomes Port 1. When the hardware does not support the second chip select, CST, this software port switch command is used. Figure 6 shows the Port 1 commands. When the SWT-TO-PORT-0 command is executed, the active port is switched back to Port 0.

The 82590 can be configured to have 4 or 6 bytes of status registers in Port 0 (see Figures 4 and 7). When configured to 4 bytes of status registers, formats of these registers are identical to those of the 82588. The first three status registers (STATUS 0 through 2) contain the information about the last command executed or the last frame received. The last status register, STATUS 3, contains the state of the 82590. When the 82590 is configured to 6 bytes of status registers, the two additional bytes are used to report a more complete status of the most recently received frame.

Status of the timer and event counters is available in the Port 1 status registers as shown in Figure 8.

**82590 AND HOST INTERACTION**

The CPU interacts with the 82590 through the system's memory and the 82590's on-chip registers. The CPU creates a data structure in memory, programs the external DMA controller with the start address and byte count of the memory block, and issues a command to the 82590.

The chip select and interrupt lines are used to communicate between the 82590 and the CPU as shown in Figure 9. The interrupt signal is used by the 82590 to attract the CPU's attention. The chip select signal is used by the CPU to attract the 82590's attention. Note that the 82590 does not have any address lines.

There are two kinds of transfers over the bus: command/status and data transfers. The command/status transfers are always performed by the CPU. The data transfers are requested by the 82590, and are usually performed by a DMA controller. Table 2 shows the command/status and data transfer control signals. The CPU writes commands to the 82590 using the CS0 (or CST) and WR signals, and reads status using the CS0 (or CST) and RD signals. When data transfers are performed, DACK0 or DACK1 must be asserted by the DMA controller instead of the Chip Select.
Figure 4. Programming Model—Directly Accessible Registers
(Accessible Through 8-Bit I/O Port[s])
Port 0 Command

7 6 5 4 3 2 1 0

**OPCODE**

- NOP
- IA-SETUP
- CONFIGURE
- MC-SETUP
- TRANSMIT
- TDR
- DUMP
- DIAGNOSE
- RETRANSMIT
- ABORT
- RCV - ENABLE
- RCV - DISABLE
- ASSIGN - ALT - BUF
- FIX - PTR
- RLS - PTR
- RESET

**CHNL**

- CHANNEL 0
- CHANNEL 1

**PTR**

- STATUS 0
- STATUS 1
- STATUS 2
- STATUS 3

**IN = ACK**

- NO ACKNOWLEDGE
- ACKNOWLEDGE

---

**Figure 5. Port 0 Commands**

Port 1 Command

7 6 5 4 3 2 1 0

**TC/GP**

- GENERAL PURPOSE
- TIMER/COUNTERS

**OPCODE**

- NOP
- SWT-TO-PORT-0
- SET-TS
- RST-TS
- LCL-PWR-DWN
- RMT-PWR-DWN
- FIX-PTR
- RLS-PTR
- RESET
- SEL - RST
- NOP
- START
- STOP
- RESUME
- LD & START
- ACK - INT
- COUNT
- START-ALL-COUNTERS
- SET - VAL
- SET-CONF
- RD - MAX - COUNT - VAL
- RD - COUNT - VAL
- RESET

**PTR**

- STATUS0
- STATUS1
- STATUS2
- STATUS3

**INT = ACK**

- NO ACKNOWLEDGE
- ACKNOWLEDGE

---

**Figure 6. Port 1 Commands**
Status Registers—6 Bytes

<table>
<thead>
<tr>
<th>INT</th>
<th>RCV</th>
<th>EXEC</th>
<th>CHNL</th>
<th>EVENT</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0</td>
</tr>
</tbody>
</table>

RESULT 0

RESULT 1

RESULT 2

RECEIVE BYTE COUNT (LOW)/FRAME COUNTER

RECEIVE BYTE COUNT (HIGH)

RCV CHNL RCV STATE BUF. CHAIN’G NO. OF BUF. EXEC CHNL EXEC STATE

STATUS 0

STATUS 1-0

STATUS 1-1

STATUS 2-0

STATUS 2-1

STATUS 3

Status Registers—4 Bytes (82588 Compatible Modes)

<table>
<thead>
<tr>
<th>INT</th>
<th>RCV</th>
<th>EXEC</th>
<th>CHNL</th>
<th>EVENT</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0</td>
</tr>
</tbody>
</table>

RESULT 0

RESULT 1

RESULT 2

RCV CHNL RCV STATE BUF. CHAIN’G NO. OF BUF. EXEC CHNL EXEC STATE

STATUS 0

STATUS 1

STATUS 2

STATUS 3

*Available only after Hardware or Software Reset

Figure 7. Port 0 Status Registers

<table>
<thead>
<tr>
<th>Events</th>
<th>Value (Status 0)</th>
<th>Events</th>
<th>Value (Status 0)</th>
</tr>
</thead>
<tbody>
<tr>
<td>CMOS*</td>
<td>0 (CHNL = 1)</td>
<td>Diagnose-Passed</td>
<td>7</td>
</tr>
<tr>
<td>IA-Setup-Done</td>
<td>1</td>
<td>End-Of-Frame</td>
<td>8</td>
</tr>
<tr>
<td>Configure-Done</td>
<td>2</td>
<td>Request-Next-Buffer</td>
<td>9</td>
</tr>
<tr>
<td>MC-Setup-Done</td>
<td>3</td>
<td>Reception-Aborted</td>
<td>10</td>
</tr>
<tr>
<td>Transmit-Done</td>
<td>4</td>
<td>Retransmit-Done</td>
<td>12</td>
</tr>
<tr>
<td>TDR-Done</td>
<td>5</td>
<td>Execution-Aborted</td>
<td>13</td>
</tr>
<tr>
<td>Dump-Done</td>
<td>6</td>
<td>Diagnose-Failed</td>
<td>15</td>
</tr>
</tbody>
</table>

*The 82590 may have more than one EVENT bit set by the time the CPU reads the status register.

Figure 8. Port 1 Status Registers
Table 2. Data Bus Control Signals and Functions

<table>
<thead>
<tr>
<th>Pin Name</th>
<th>(\text{CS0}^{*})</th>
<th>(\text{RD})</th>
<th>(\text{WR})</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>(\text{CS0})</td>
<td>X</td>
<td>X</td>
<td></td>
<td>No Transfer To/From Command/Status</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td></td>
<td>Illegal</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
<td>Read from Status Register</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td></td>
<td>Write to Command Register</td>
</tr>
<tr>
<td>(\text{DACK0})</td>
<td>X</td>
<td>X</td>
<td></td>
<td>No DMA Transfer</td>
</tr>
<tr>
<td>(\text{DACK1})</td>
<td>1</td>
<td>1</td>
<td></td>
<td>Data Read from DMA Channel 0 (or 1)</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
<td>Illegal</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td></td>
<td>Data Write to DMA Channel 0 (or 1)</td>
</tr>
</tbody>
</table>

*Only one of \(\text{CS0}\), \(\text{CS1}\), \(\text{DACK0}\), or \(\text{DACK1}\) may be active at any time.

To initiate an operation such as Transmit or Configure (see Figure 5), the command from the CPU must first be written to the 82590. Any parameters or data associated with the command are transferred from memory to the 82590 using DMA. Upon completion of the operation, the 82590 updates the appropriate status registers and sends an interrupt to the CPU.

**FRAME TRANSMISSION**

To transmit a frame, the CPU prepares a Transmit Data Block in memory as shown in Figure 10. Its first two bytes specify the length of the rest of the block. The next few bytes (up to six) contain the destination address of the station the frame is being sent to. The rest of the block is the data field. The CPU programs the DMA controller with the start address of the block, length of the block, and other control information and then issues a Transmit command to the 82590. Upon receiving this command, the 82590 fetches the first two bytes of the block to determine its length. If the link is free and the first data byte was fetched, the 82590 begins transmitting the preamble and concurrently fetches more bytes from the Transmit Data Block and loads them into the transmit FIFO to keep them ready for transmission.
The destination address is transmitted after the preamble. This is followed by the source or the station individual address, which was previously stored in the 82590 by the IA-Setup command. After this, the entire information field is transmitted, followed by a CRC field calculated by the 82590. If a collision is encountered during transmission of the frame, then the transmission is aborted after a jam pattern is sent. If the collision is detected during preamble or SFD (Start Frame Delimiter) transmission, the 82590 transmits the jam pattern after the SFD is transmitted. An interrupt is then generated to inform the CPU of the unsuccessful transmission due to a collision. The CPU reinitializes the DMA controller and issues a Retransmit command to the 82590. Retransmission is done by the CPU exactly as the Transmit command is done, except the Retransmit command keeps track of the number of collisions encountered. When the 82590 gets the Retransmit command and the backoff timer is expired, it transmits the frame again. Retransmission is repeated until the attempt is successful, or until the preprogrammed retry number expires.

If the 82590 is programmed to generate the EOP signal to the 8237 or 82380 DMA controller, or if it is used with a DMA controller which implements the Tightly Coupled Interface, retransmission is performed without CPU intervention.

**FRAME RECEPTION**

The 82590 can receive frames when its receiver has been enabled. The 82590 checks for an address match for an individual address, a Multicast address, or a Broadcast address. In the Promiscuous mode the 82590 receives all frames. When the address match is successful, the 82590 transfers the frame to memory using the DMA controller. Before enabling the receiver, it is the CPU's responsibility to make a memory buffer area available to the receiver and to properly program the starting address of the DMA controller. The received frame is transferred to the memory buffer in the format shown in Figure 11. This method of reception is called Single Buffer reception; the entire frame is contained in one continuous buffer. Upon completion of reception, the status of the reception is appended at the end of the received frame in the memory buffer, and the total number of bytes transferred to the memory buffer is loaded into the internal status registers 1 and 2. An interrupt is then generated to inform the CPU of the frame reception.
If the frame size is unknown, memory usage can be optimized by using Multiple Buffer reception. In this mode of operation, the CPU and DMA Controller can dynamically allocate memory space as it receives frames. This method requires both DMA channels to receive the frame alternately. As frame reception begins, the 82590 interrupts the CPU and automatically requests assignment of the next available buffer. The CPU does this and loads the second DMA channel with the next buffers information so the 82590 can immediately switch to the other channel when the current buffer is full. When the 82590 switches from the first to the second buffer it again interrupts the CPU and requests another buffer to be allocated on the previous channel. This process continues until the entire frame is received. The received frame is spread over multiple memory buffers. The link between the buffers is easily maintained by the CPU, using a buffer chain descriptor structure in memory as shown in Figure 12. This dynamic allocation of memory buffers results in efficient use of available storage when handling frames of widely differing sizes.

If the 82590 is programmed to generate the EOP signal to the 8237 or 82380 DMA controller, or if it is used with a DMA controller which implements the Tightly Coupled Interface, buffer reclamation and more advanced data structures for the buffer area can significantly improve system performance.

**EOP SIGNAL TO THE DMA CONTROLLER**

The 82590 can be programmed to assert the EOP signal to the 8237 or 82380 DMA controller when one or more of the following occurs:

- A collision during transmission
- An error (CRC or alignment) during reception
- A good frame reception

If the 8237 or 82380 is programmed for Auto-initialize mode and if the 82590 is programmed to assert the EOP signal on a collision during transmission, the retransmission following a collision is done automatically by the 8237 and the 82590. The 8237 will reinitialize itself automatically and the 82590 will retransmit the same frame from the same memory area without CPU intervention. When the 82590 is programmed for this mode it does not interrupt the CPU upon a collision, and the CPU does not need to issue a Retransmit command to the 82590. The CPU is interrupted only after a successful transmission or retransmission, or after a transmission failure, such as DMA underrun.
If the 82590 is programmed to assert the EOP signal when an error occurs during reception, the 8237 or the 82380 in Auto-initialize mode will be able to reclaim the memory area which would otherwise be wasted for the errored frame reception. If the 82590 is programmed to assert EOP at the end of a frame reception, automatic buffer switching can be accomplished by alternating the DMA channels with the 8237 or the 82380. When the 82380 is used, the buffer switching can be done with only one DMA channel.

The EOP signal must be derived from the DACK1/CS1/EOP pin using external logic (see Figure 13).

**82590/82560 TIGHTLY COUPLED INTERFACE**

The 82590 has a mode of operation called “Tightly Coupled Interface.” In this mode the 82590 provides a tightly coupled interface to a DMA controller in order to execute some of the time-critical processes of the CSMA/CD protocol without any CPU intervention. By using the 82590’s companion chip, the 82560, or by implementing the Tightly Coupled Interface in a DMA controller, operations such as automatic retransmission, continuous back-to-back frame reception, and transmit and/or receive buffer chaining can be accomplished.

The 82590 provides the status of the current active transmission or reception to the DMA controller by using the DRQ and EOP signals at the end of every DMA cycle. The status is encoded according to Table 3. As long as the 82590 generates DRQ High and EOP Floating at the rising edge of RD or WR, the DMA controller repeats DMA transfers. If the transmission is completed without collisions or if the reception is good (no collision, no CRC, or no Alignment error), then DRQ and EOP both become Low at the end of a DMA transfer which follows the last DMA data transfer. If the transmission encountered a collision or if the reception had an error, DRQ becomes High and EOP becomes Low. The DMA controller must decode these signals appropriately and must reinitialize the DMA channel so it can retransmit the same frame or reclaim the otherwise wasted buffer. It is the DMA controller’s responsibility to reprogram itself for the next appropriate operation.

The 82560 fully implements the Tightly Coupled Interface and provides very high-performance DMA services for the 82590 with minimal CPU involvement.

**NETWORK MANAGEMENT AND DIAGNOSTICS**

The 82590 provides a large set of diagnostic and network management functions including: internal and external loopback, monitor mode, optional capture of all frames regardless of destination address (Promiscuous mode), and time domain reflectometry for locating fault points in the network cable. The 82590 Dump command ensures software reliability by dumping the contents of the 82590 internal registers into the system memory.
Table 3. Transmit/Receive Status Encoding on DRQ and EOP

<table>
<thead>
<tr>
<th>DRQ</th>
<th>EOP</th>
<th>Status Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Hi-Z</td>
<td>Idle</td>
</tr>
<tr>
<td>1</td>
<td>Hi-Z</td>
<td>DMA Transfer</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>Transmission or Reception Terminated OK</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>Transmission or Reception Aborted</td>
</tr>
</tbody>
</table>

OTHER ENHANCEMENTS

Compared to the 82588 the 82590 has a number of functional and performance enhancements. This section lists some of these enhancements which are not covered in other sections.

1. **Multi-IA**—The 82590 implements multiple-individual address (Multi-IA) filtering. It can receive more than one IA frame in this mode.

2. **Power Down Modes**—Two power down modes, Local Power Down and Remote Power Down, are available. When the 82590 is in Remote Power Down mode, it can be powered up remotely by sending a special frame to it.

3. **Automatic Padding and IEEE 802.3 Length Field**—If a frame to be transmitted is shorter than the configured Slot Time, the 82590 automatically appends pad bytes up to the shortest frame greater than the Slot Time. If the data field of a received frame is longer than the byte count indicated in the Length field, the extra bytes are stripped automatically according to the Length field. Erroneous conditions are detected and reported by the 82590. An example of such conditions is reception of a frame which is shorter than the byte count indicated in the Length field.

4. **Automatic Retransmission on Collision During Preamble**—The 82590 can be programmed to retransmit automatically if it detects a collision during transmission of the preamble.

5. **On-Chip Jabber Inhibit Function**—The 82590 can be programmed to provide an on-chip jaber inhibit function.
CRC Transfer to Memory—The 82590 can be programmed to transfer the CRC field of a received frame into memory.

Loopback Signal to the 82C501—The 82590 can be programmed to provide an active High loopback signal to the 82C501 (see Figure 14).

StarLAN—The 82590 can be configured to recognize the IEEE 802.3 1BASE5 Collision Presence Signal (CPS). In this mode it also delays deactivation of the RTS signal at the end of a frame transmission in order to insert an end-of-frame marker according to the standard.

APPLICATIONS

The 82590 can be used in a variety of applications. When it is used in High-Integration Mode, it implements most of the Data Link and Physical Layer functions required by the IEEE 802.3 1BASE5 (StarLAN) and the IBM PC Network—Baseband and Broadband. When it is used in High-Speed Mode, it can work with the 82C501 and a standard transceiver for IEEE 802.3 10BASE5 (Ethernet) and 10BASE2 (Cheapernet) implementations.

If the desired network requires determinism, the 82590's Deterministic Collision Resolution (DCR) method can be used.

Figure 15 shows a block diagram of an 82590/82560 High Integration adapter board. The 82560 provides the following functions: DMA for the 82590 with Tightly Coupled Interface and dual-port memory control for the static RAM. The 82590 is configured to High-Integration mode to minimize the serial interface logic.

Figure 14. Loopback Output to the 82C501
**82592 ADVANCED CSMA/CD LAN CONTROLLER WITH 16-BIT DATA PATH**

- **Supports Industry Standard LANs**
  - Ethernet and Cheapernet (IEEE 802.3 10BASE5 and 10BASE2)
  - Broadband Ethernet (IEEE 802.3 10BROAD36)
  - StarLAN (IEEE 802.3 1BASE5)
  - IBM* PC Network—Baseband and Broadband

- **Integrates Physical and Data Link Layers of OSI Model**
  - Complete CSMA/CD Medium Access Control (MAC) Functions
  - Manchester, Differential Manchester, or NRZI Encoding/Decoding
  - On-Chip, Logic-Based Collision Detection
  - IEEE 802.3 or HDLC Frame Delimiting

- **Two Modes of Operation**
  - Bit Rates Up to 4 Mb/s with On-Chip Encoder/Decoder (High-Integration Mode)
  - Bit Rates Up to 20 Mb/s with External Encoder/Decoder (High-Speed Mode)

- **High-Performance System Interface**
  - 16-MHz Clock, 2 Clocks per Transfer
  - 64 Bytes of Configurable FIFO

- **Efficient Memory Use via Buffer and Frame Chaining**
- **DMA Interface for Retransmission and Continuous Reception Without CPU Intervention**
  - EOP Signal Generation for 8237 and 82380
  - Tightly Coupled Interface to 82560 Host Interface and Memory Manager
- **Supports 8- or 16-Bit DMA Transfers**
- **Local and Remote Power-Down Modes**
- **Deterministic Collision Resolution**
- **24-Bit General Purpose Timer**
- **On-Chip Jabber Inhibit Function**
- **Network Management and Diagnostics**
  - Monitor Mode
  - CRC, Alignment, and Short Frame Error Detection
  - Three 16-Bit Event Counters
  - Short or Open Circuit Localization
  - Self-Test Diagnostics
  - Internal and External Loopback Operation
  - Internal Register Dump
- **High-Speed CHMOS III Technology**

---

*IBM is a trademark of International Business Machines Corporation.*
The 82592 is a second-generation, 16-bit data path CSMA/CD controller. Its system interface enables efficient operation with a wide variety of Intel microprocessors (e.g., 80186, 80286, or 80386) and industry standard buses (such as the IBM PC I/O channel or Personal System/2 Micro Channel). The 82592 can be configured to support a wide variety of industry standard networks, including StarLAN, IBM PC Network, and Ethernet/CheaperNet.

The 82592 is ideal for integrated LAN on motherboard solutions. The 82592 architecture offers low cost, high performance and minimal real estate requirements. The 82592’s Tightly Coupled Interface mode allows it to use host DMA without local buffering. An integrated 82592 Ethernet motherboard LAN will occupy less than five percent of the total motherboard area. The CHMOS 82592 can be used in low power or no-fan systems such as diskless workstations and laptop PCs. The 82592 provides two power-down modes for these environments.

Together with the 82560 (Host Interface and Memory Manager) the 82592 offers a complete solution for CSMA/CD LAN adapters oriented to the IBM PC environment. The 82592 fully conforms to existing IEEE-802.3 standards (1BASE5, 10BASE5, 10BASE2, and 10BROAD36). Intel also offers the 82590, an 8-bit data path version of the 82592. The 82590 is pin and software compatible with the 82588.

The 82592 is available in a 40-pin Plastic DIP or a 44-pin PLCC package. It is fabricated with Intel’s reliable CHMOS III technology.
## Table 1. Pin Description

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Pin No. (DIP)</th>
<th>Pin No. (PLCC)</th>
<th>Type</th>
<th>Name and Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>D15</td>
<td>22</td>
<td>36</td>
<td>I/O</td>
<td><strong>DATA BUS</strong>: The Data Bus lines are bidirectional, three-state lines connected to the CPU's Data Bus for transfers of data, commands, status, and parameters.</td>
</tr>
<tr>
<td>D14</td>
<td>23</td>
<td>37</td>
<td></td>
<td></td>
</tr>
<tr>
<td>D13</td>
<td>24</td>
<td>38</td>
<td></td>
<td></td>
</tr>
<tr>
<td>D12</td>
<td>25</td>
<td>39</td>
<td></td>
<td></td>
</tr>
<tr>
<td>D11</td>
<td>37</td>
<td>9</td>
<td></td>
<td></td>
</tr>
<tr>
<td>D10</td>
<td>38</td>
<td>10</td>
<td></td>
<td></td>
</tr>
<tr>
<td>D9</td>
<td>39</td>
<td>11</td>
<td></td>
<td></td>
</tr>
<tr>
<td>D8</td>
<td>40</td>
<td>12</td>
<td></td>
<td></td>
</tr>
<tr>
<td>D7</td>
<td>1</td>
<td>13</td>
<td></td>
<td></td>
</tr>
<tr>
<td>D6</td>
<td>2</td>
<td>14</td>
<td></td>
<td></td>
</tr>
<tr>
<td>D5</td>
<td>3</td>
<td>15</td>
<td></td>
<td></td>
</tr>
<tr>
<td>D4</td>
<td>4</td>
<td>16</td>
<td></td>
<td></td>
</tr>
<tr>
<td>D3</td>
<td>5</td>
<td>17</td>
<td></td>
<td></td>
</tr>
<tr>
<td>D2</td>
<td>6</td>
<td>18</td>
<td></td>
<td></td>
</tr>
<tr>
<td>D1</td>
<td>7</td>
<td>19</td>
<td></td>
<td></td>
</tr>
<tr>
<td>D0</td>
<td>8</td>
<td>20</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RD</td>
<td>36</td>
<td>8</td>
<td>I</td>
<td><strong>READ</strong>: Together with CS0, CS1, DACK0, or DACK1, Read controls data or status transfers out of the 82592.</td>
</tr>
<tr>
<td>WR</td>
<td>34</td>
<td>4</td>
<td>I</td>
<td><strong>WRITE</strong>: Together with CS0, CS1, DACK0, or DACK1, Write controls data or command transfers into the 82592.</td>
</tr>
<tr>
<td>CS0</td>
<td>33</td>
<td>3</td>
<td>I</td>
<td><strong>CHIP SELECT (PORT 0)</strong>: When LOW, the 82592 is selected by the CPU for command or status transfer through PORT 0.</td>
</tr>
<tr>
<td>RESET</td>
<td>26</td>
<td>40</td>
<td>I</td>
<td><strong>RESET</strong>: A HIGH signal on this pin causes the 82592 to terminate current activity. This signal is internally synchronized and must be held HIGH for at least four Clock (CLK) cycles. When the Clock signal is provided internally (CLKSRC is strapped HIGH), the RESET signal must be held HIGH for at least 50 μs. (PLCC version only.)</td>
</tr>
<tr>
<td>INT</td>
<td>27</td>
<td>41</td>
<td>O</td>
<td><strong>INTERRUPT</strong>: A HIGH signal on this pin notifies the CPU that the 82592 is requesting an interrupt.</td>
</tr>
<tr>
<td>DRQ0</td>
<td>14</td>
<td>27</td>
<td>O</td>
<td><strong>DMA REQUEST (CHANNEL 0)</strong>: This pin is used by the 82592 to request DMA transfer. DRQ0 remains HIGH as long as the 82592 requires DMA transfers. Burst transfers are thus possible. When the 82592 is programmed for Tightly Coupled DMA Interface, the 82592 notifies the DMA controller of the status of transmission or reception, using this pin together with EOP.</td>
</tr>
<tr>
<td>DRQ1</td>
<td>15</td>
<td>28</td>
<td>O</td>
<td><strong>DMA REQUEST (CHANNEL 1)</strong>: This pin is used by the 82592 to request DMA transfer. DRQ1 remains HIGH as long as the 82592 requires DMA transfers. Burst transfers are thus possible. When the 82592 is programmed for Tightly Coupled DMA Interface, the 82592 notifies the DMA controller of the status of transmission or reception, using this pin together with EOP.</td>
</tr>
<tr>
<td>Symbol</td>
<td>Pin No. (DIP)</td>
<td>Pin No. (PLCC)</td>
<td>Type</td>
<td>Name and Function</td>
</tr>
<tr>
<td>-------------------</td>
<td>--------------</td>
<td>---------------</td>
<td>------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>DACK0/</td>
<td>32</td>
<td>2</td>
<td>I</td>
<td><strong>DMA ACKNOWLEDGE (CHANNEL 0):</strong> When LOW, this input signal from the DMA controller notifies the 82592 that the requested DMA cycle is in progress. This signal acts similarly to Chip Select for data and parameter transfers, using DMA channel 0. <strong>DMA ACKNOWLEDGE (CHANNELS 0 AND 1):</strong> When the DACK1/CST/EOP pin is programmed to CST/EOP, this pin provides a DMA acknowledge for both channels 0 and 1. Two DMA acknowledge signals from the DMA controller, DACK0 and DACK1, must be externally ANDed in this mode of operation.</td>
</tr>
<tr>
<td>DACK1/CST/EOP</td>
<td>28</td>
<td>42</td>
<td>I</td>
<td>This is a multifunction, bidirectional pin which can be programmed to DACK1 or CST/EOP during configuration. When it is configured for EOP, it provides an open-drain output. <strong>DMA ACKNOWLEDGE (CHANNEL 1):</strong> When LOW, this input signal from the DMA controller notifies the 82592 that the requested DMA cycle is in progress. This signal acts similarly to Chip Select for data and parameter transfers, using DMA channel 1. <strong>CHIP SELECT (PORT 1):</strong> When LOW, the 82592 is selected by the CPU for command or status transfer through PORT 1. <strong>END OF PROCESS:</strong> A LOW output signal requests the DMA controller to terminate the active DMA service.</td>
</tr>
<tr>
<td>CLK</td>
<td>35</td>
<td>5</td>
<td>I</td>
<td><strong>CLOCK:</strong> In the 40-pin DIP, this is only an input pin. A TTL-compatible clock input to this pin provides the timing for the 82592 parallel subsystem. <strong>CLOCK:</strong> In the 44-pin PLCC, this pin can be a clock input or output, depending on the state of CLKSRC. If CLKSRC is strapped LOW, this pin is a clock input which provides timing for the 82592 parallel subsystem. If CLKSRC is strapped HIGH, the clock for the 82592 parallel subsystem is generated from the internal clock generator. The CLK pin is then a clock output and provides a clock signal whose frequency can be one-half of, or identical to, the frequency of the internally generated parallel subsystem clock, depending on the state of FREQ. Note that the maximum frequency of the clock signal supplied by the CLK pin is 8 MHz.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td><strong>CLKSRC</strong> numeric value <strong>FREQ</strong> clock input or output: <strong>0 (LOW)</strong> Don't Care <strong>1 (HIGH)</strong> Internal Parallel Subsystem Clock Divided by Two <strong>1</strong> Internal Parallel Subsystem Clock</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td><strong>CLK</strong> numeric value <strong>Type</strong> Signal <strong>Clock for the Parallel Subsystem</strong> Clock as provided on the CLK pin. Prescaled clock generated from the internal clock generator. Prescaled clock generated from the internal clock generator</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td><strong>CLKSRC</strong> numeric value <strong>0 (LOW)</strong> Don't Care <strong>1 (HIGH)</strong> Internal Parallel Subsystem Clock Divided by Two <strong>1</strong> Internal Parallel Subsystem Clock</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td><strong>CLOCK SOURCE:</strong> When strapped LOW, a clock signal on the CLK pin provides timing for the parallel subsystem. When strapped HIGH, timing for the parallel subsystem is internally generated from the clock generator provided in the serial subsystem. The internal prescaler is programmed during configuration to determine the frequency of the clock for the parallel subsystem.</td>
</tr>
</tbody>
</table>
FUNCTIONAL DESCRIPTION

Internal Architecture

The 82592 consists of a parallel subsystem, a serial subsystem, and a FIFO subsystem (see Figure 1).

PARALLEL SUBSYSTEM

The parallel subsystem consists of a bus interface unit (BIU), command and status registers, a 24-bit general purpose timer, and three 16-bit event counters.

The BIU provides an 8- and/or 16-bit interface to the external system bus. It handles all data transfers to and from memory (at speeds up to 16 Mbytes/sec.), accepts commands from the CPU, and provides status to the CPU. There are two separate 8-bit I/O ports, Port 0 and Port 1; and two separate 8- or 16-bit DMA channels, Channel 0 and Channel 1. The 8-bit I/O ports are interfaced to the CPU via the data lines D0–D7. The DMA channels can be configured for an 8- or 16-bit data path during initialization, and are typically interfaced to an external DMA controller. When the 82592 is reset by hardware or software, the DMA channels are initialized for an 8-bit data path. The CPU can then configure the 82592 for a 16-bit data path if desired. Once the DMA channels are configured for a 16-bit data path all subsequent DMA transfers are performed on the data lines D0–D15. The two DMA channels are independent and can be used for high-performance operations such as simultaneous transmission and reception.

The 24-bit timer consists of a 24-bit maximum count register, a 24-bit count register, and associated control bits in the command registers. Its clock source can be the transmit clock or the parallel subsystem clock. The timer can be programmed to halt or continue on a terminal count with or without causing an interrupt.

The three 16-bit event counters can be programmed to count valid frames, collided frames, and errored (CRC or Alignment) frames. When these event counters are used in Monitor mode, the 82592 is capable of maintaining the network statistics by itself; i.e., without requesting DMA services or causing interrupts to the CPU.

SERIAL SUBSYSTEM

The serial subsystem consists of a CSMA/CD unit, a data encoder and decoder, collision detect and carrier sense logic, and a clock generator.

The 82592's CSMA/CD unit is highly flexible in implementing the CSMA/CD protocol. It can operate in a variety of IEEE 802.3 and other CSMA/CD LAN environments, including 1BASE5 (StarLAN), 10BASE5 (Ethernet), 10BASE2 (Cheapernet), and the IBM PC Network (Baseband and Broadband). The programmable parameters include:

- Framing (IEEE 802.3 Framing or HDLC Framing)
- Address Field Length
- Station Priority
- Interframe Spacing
- Slot Time
- CRC-32 or CRC-16

The CSMA/CD unit also has a mode of operation which implements deterministic collision resolution (DCR). The DCR algorithm is fully compatible with the MULTIBUS™ II Serial System Bus (SSB) specifications.

The encoder and decoder in the serial subsystem is capable of NRZI, Manchester, and Differential Manchester encoding and decoding at bit rates up to 4 Mb/s in High-Integration Mode, and Manchester encoding at bit rates up to 20 Mb/s in High-Speed Mode. A digital phase-lock loop is used in High-Integration Mode to decode the receive data and to generate the synchronous receive clock.

The collision detect and carrier sense logic generate the internal collision detect and carrier sense signals for the CSMA/CD unit.

The 82592 implements several different internal, logic-based collision detect mechanisms. Two of these, Code Violation and Bit Comparison, are also implemented in the 82588 (8-bit NMOS High Integration LAN Controller), and have been used in a variety of applications. The Code Violation method defines a collision where a transition edge occurs outside the area of normal transitions (as specified by the data encoding method). For example, if there are no mid-bit cell transitions in the Manchester encoded data, this method interprets that condition as a collision. The Bit Comparison method compares the signature of the transmitted frame to the signature of the received frame. If the signatures are different, a collision is assumed to have occurred. Two other internal collision detect methods are Source Address Comparison and StarLAN CPS (Collision Presence Signal) Recognition. The Source Address Comparison compares the source address field of the transmitted frame to the source address field of the received frame. If the source addresses are different, it assumes that a collision has occurred resulting in data corruption in the source address field. The StarLAN CPS Recognition method looks for the specific collision presence signal defined by the
### Table 1. Pin Description (Continued)

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Pin No. (DIP)</th>
<th>Pin No. (PLCC)</th>
<th>Type</th>
<th>Name and Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>FREQ</td>
<td>NA</td>
<td>7</td>
<td>I</td>
<td>FREQUENCY: When strapped LOW, CLK has an output frequency equal to that of the parallel subsystem clock. When strapped HIGH, CLK has an output frequency one-half that of the parallel subsystem clock. The state of this pin is relevant only when CLKSRC is strapped HIGH.</td>
</tr>
<tr>
<td>X1/X2</td>
<td>12/13</td>
<td>25/26</td>
<td>I</td>
<td>HIGH INTEGRATION MODE OSCILLATOR INPUTS: These inputs may be used to connect a quartz crystal which controls the internal clock generator for the serial subsystem. When CLKSRC is strapped HIGH, the clock generator also provides a clock for the parallel subsystem. X1 may also be driven by a MOS-level clock whose frequency is 8, 10, 16, or 18 times the bit rate of Transmit/Receive data. X2 must be left floating if X1 is connected to an external MOS clock.</td>
</tr>
<tr>
<td>TxC</td>
<td>12</td>
<td>25</td>
<td>I</td>
<td>HIGH SPEED MODE TRANSMIT CLOCK: This signal provides the fundamental timing for the serial subsystem. The clock is also used to transmit data synchronously on the TxD pin. For NRZ encoding, data is transferred to the TxD pin on the HIGH to LOW clock transition. For Manchester encoding, the transmitted bit center is aligned with the LOW to HIGH transition.</td>
</tr>
<tr>
<td>RxC</td>
<td>13</td>
<td>26</td>
<td>I</td>
<td>RECEIVE CLOCK: This clock is used to synchronously sample data on the RxD pin. Only NRZ data format is supported for reception. The state of the RxD pin is sampled on the HIGH to LOW transition.</td>
</tr>
<tr>
<td>TCLK/CRS</td>
<td>21</td>
<td>35</td>
<td>I</td>
<td>CARRIER SENSE: In High-Speed Mode this pin is Carrier Sense, CRS, and is used to notify the 82592 that the serial link is active. TRANSMIT CLOCK: In High-Integration Mode this pin is Transmit Clock, TCLK.</td>
</tr>
<tr>
<td>CDT</td>
<td>20</td>
<td>34</td>
<td>I</td>
<td>COLLISION DETECT: This input notifies the 82592 that a collision has occurred. In High-Speed Mode a collision is sensed by this pin only when the 82592 is configured for external Collision Detect (external means are then required for collision detection). In High-Integration Mode collisions are sensed by this pin regardless of the internal or external Collision Detect configuration of the 82592.</td>
</tr>
<tr>
<td>RxD</td>
<td>16</td>
<td>30</td>
<td>I</td>
<td>RECEIVE DATA: This pin receives serial data. It must be HIGH when not receiving.</td>
</tr>
<tr>
<td>TxD</td>
<td>17</td>
<td>31</td>
<td>O</td>
<td>TRANSMIT DATA: This pin transmits data to the serial link. It is HIGH when not transmitting.</td>
</tr>
<tr>
<td>RTS</td>
<td>18</td>
<td>32</td>
<td>O</td>
<td>REQUEST TO SEND: When this signal is LOW the 82592 notifies the channel that it has data to transmit. It is forced HIGH after a reset or when transmission is stopped.</td>
</tr>
<tr>
<td>CTS/LPBK</td>
<td>19</td>
<td>33</td>
<td>I/O</td>
<td>CLEAR TO SEND: An active LOW signal which enables the 82592 to start transmitting data. Asserting this signal HIGH stops the transmission. LOOPBACK: This pin, in conjunction with a pull-down resistor, can be programmed to provide an active HIGH loopback signal to the external interface device.</td>
</tr>
<tr>
<td>VCC</td>
<td>29</td>
<td>1</td>
<td></td>
<td>POWER: +5V ± 10%.</td>
</tr>
<tr>
<td></td>
<td>30</td>
<td>43</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>31</td>
<td>44</td>
<td></td>
<td></td>
</tr>
<tr>
<td>VSS</td>
<td>9</td>
<td>21</td>
<td></td>
<td>GROUND: 0V.</td>
</tr>
<tr>
<td></td>
<td>10</td>
<td>22</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>11</td>
<td>23</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
IEEE 802.3 1BASE5 standard. Other abnormal circumstances, such as no carrier for more than one-half slot time in the receive channel during transmission, are interpreted as collisions by the 82592.

In addition to these internal, logic-based collision detection methods, an external means of collision detection can be used in parallel by using the CDT input pin.

The clock generator in the serial subsystem is available only in High-Integration Mode and provides timing for the serial subsystem. The clock signal can also be routed to the parallel subsystem, if so desired. The oscillator circuit is designed for use with an external, parallel resonant, fundamental mode crystal. The crystal frequency should be selected at 8X, 10X, 16X, or 18X the required serial bit rate.

FIFO SUBSYSTEM

The FIFO subsystem is located between the parallel subsystem and the serial subsystem. It consists of a transmit FIFO, a receive FIFO, and FIFO control logic. The transmit and receive FIFOs are independent of each other and individually provide optimal interfaces between the two subsystems which may have different speeds. There is a total of 64 bytes that can be used for the two separate FIFOs. During configuration these 64 bytes can be divided into one of four possible combinations: 16 and 16 bytes, 16 and 48 bytes, 32 and 32 bytes, or 48 and 16 bytes for the transmit and receive FIFO respectively. The FIFO threshold is also programmed during configuration.

Programming Model—Register Overview

Figure 4 shows the 82592 internal registers that are directly accessible through the 8-bit I/O ports: Port 0 and Port 1.

Figure 5 shows the Port 0 commands, and Figure 6 shows the Port 1 commands. The two separate I/O ports can be accessed at two different addresses selected by CS0 and CS1, or at one address selected by CS0. When the hardware does not support two chip select signals, port switch commands are used to access both ports alternately at one address. If the SWT-TO-PORT-1 command is executed while in Port 0, the port logically becomes Port 1. Software overhead associated with port switching is eliminated if two chip select signals are supplied in hardware.

The 82592 can be configured to have 4 or 6 bytes of status registers in Port 0 (see Figures 4 and 7). When configured to 4 bytes of status registers the first three status registers (STATUS 0 through 2) contain the information about the last command executed or the last frame received. The last status register, STATUS 3, contains the state of the 82592. When the 82592 is configured to 6 bytes of status registers, the two additional bytes are used to report a more complete status of the most recently received frame.

The status of the timer and event counters is available in the Port 1 status registers as shown in Figure 8.

82592 and Host Interaction

The CPU interacts with the 82592 through the system's memory and the 82592's on-chip registers. The CPU creates a data structure in memory, programs the external DMA controller with the start address and byte count of the memory block, and issues a command to the 82592.

The chip select and interrupt lines are used to communicate between the 82592 and the CPU as shown in Figure 9. The interrupt signal is used by the 82592 to attract the CPU's attention. The chip select signal is used by the CPU to attract the 82592's attention. Note that the 82592 does not have any address lines.

There are two kinds of transfers over the bus: command/status and data transfers. The 8-bit command/status transfers are always performed by the CPU. The 8- or 16-bit data transfers are requested by the 82592, and are usually performed by a DMA controller. Table 2 shows the command/status and data-transfer control signals. The CPU writes commands to the 82592 using the CS0 (or CS1) and WR signals, and reads status using the CS0 (or CS1) and RD signals. When data transfers are performed, DACK0 or DACK1 must be asserted by the DMA controller instead of the Chip Select.
Figure 4. Programming Model—Directly Accessible Registers
(Readable Through 8-Bit I/O Port[s])
Figure 5. Port 0 Commands

Figure 6. Port 1 Commands
Figure 7. Port 0 Status Registers

<table>
<thead>
<tr>
<th>Events</th>
<th>Value (Status 0)</th>
</tr>
</thead>
<tbody>
<tr>
<td>CMOS*</td>
<td>0 (CHNL = 1)</td>
</tr>
<tr>
<td>IA-Setup-Done</td>
<td>1</td>
</tr>
<tr>
<td>Configure-Done</td>
<td>2</td>
</tr>
<tr>
<td>MC-Setup-Done</td>
<td>3</td>
</tr>
<tr>
<td>Transmit-Done</td>
<td>4</td>
</tr>
<tr>
<td>TDR-Done</td>
<td>5</td>
</tr>
<tr>
<td>Dump-Done</td>
<td>6</td>
</tr>
<tr>
<td>Diagnose-Passed</td>
<td>7</td>
</tr>
<tr>
<td>End-of-Frame</td>
<td>8</td>
</tr>
<tr>
<td>Request-Next-Buffer</td>
<td>9</td>
</tr>
<tr>
<td>Reception-Aborted</td>
<td>10</td>
</tr>
<tr>
<td>Retransmit-Done</td>
<td>12</td>
</tr>
<tr>
<td>Execution-Aborted</td>
<td>13</td>
</tr>
<tr>
<td>Diagnose-Failed</td>
<td>15</td>
</tr>
</tbody>
</table>

*Available only after Hardware or Software Reset.
The 82592 may have more than one EVENT bit set by the time the CPU reads the Status register.

### Figure 8. Port 1 Status Registers

<table>
<thead>
<tr>
<th>Status Register</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Status 0</td>
<td>Timer Expired Bit 0 = 1</td>
</tr>
<tr>
<td>Status 1</td>
<td>Counter 1 Expired Bit 1 = 1</td>
</tr>
<tr>
<td>Status 2</td>
<td>Counter 2 Expired Bit 2 = 1</td>
</tr>
<tr>
<td>Status 3</td>
<td>Counter 3 Expired Bit 3 = 1</td>
</tr>
</tbody>
</table>

### Table 2. Data Bus Control Signals and Functions

<table>
<thead>
<tr>
<th>Pin Name</th>
<th>RD</th>
<th>WR</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>CS0, CS1</td>
<td></td>
<td></td>
<td>No Transfer to/from Command/Status</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>1</td>
<td>Illegal</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>0</td>
<td>Read from Status Register</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>1</td>
<td>Write to Command Register</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>DACK0, DACK1</th>
<th>RD</th>
<th>WR</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1</td>
<td>X</td>
<td>No DMA Transfer</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>0</td>
<td>Illegal</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>1</td>
<td>Data Read from DMA Channel 0 (or 1)</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>1</td>
<td>Data Write to DMA Channel 0 (or 1)</td>
</tr>
</tbody>
</table>

*Only one of CS0, CS1, DACK0, or DACK1 may be active at any time.*
To initiate an operation such as Transmit or Configure (see Figure 5), the command from the CPU must first be written to the 82592. Any parameters or data associated with the command are transferred from memory to the 82592 using DMA. Upon completion of the operation, the 82592 updates the appropriate status registers and sends an interrupt to the CPU.

Frame Transmission

To transmit a frame, the CPU prepares a Transmit Data Block in memory as shown in Figure 10. Its first two bytes specify the length of the rest of the block. The next few bytes (up to six) contain the destination address of the station the frame is being sent to. The rest of the block is the data field. The CPU programs the DMA controller with the start address of the block, length of the block, and other control information and then issues a Transmit command to the 82592. Upon receiving this command, the 82592 fetches the first two bytes of the block to determine its length. If the link is free and the first data byte was fetched, the 82592 begins transmitting the preamble and concurrently fetches more bytes from the Transmit Data Block and loads them into the transmit FIFO to keep them ready for transmission.

The destination address is transmitted after the preamble. This is followed by the source or the station individual address, which was previously stored in the 82592 by the IA-Setup command. After this, the entire information field is transmitted, followed by a CRC field calculated by the 82592. If a collision is encountered during transmission of the frame, then the transmission is aborted after a jam pattern is sent. If the collision is detected during preamble or SFD (Start Frame Delimiter) transmission, the 82592 transmits the jam pattern after the SFD is transmitted. An interrupt is then generated to inform the CPU of the unsuccessful transmission due to a collision.

The CPU reinitializes the DMA controller and issues a Retransmit command to the 82592. Retransmission is done by the CPU exactly as the Transmit command is done, except the Retransmit command keeps track of the number of collisions encountered. When the 82592 gets the Retransmit command and the backoff timer is expired, it transmits the frame again. Retransmission is repeated until the attempt is successful, or until the preprogrammed retry number expires.

If the 82592 is programmed to generate the EOF signal to the 8237 or 82380 DMA controller, or if it is used with a DMA controller which implements the Tightly Coupled Interface, retransmission is performed without CPU intervention.

Frame Reception

The 82592 can receive frames when its receiver has been enabled. The 82592 checks for an address match for an individual address, a Multicast address, or a Broadcast address. In the Promiscuous mode the 82592 receives all frames. When the address match is successful, the 82592 transfers the frame to memory using the DMA controller. Before enabling the receiver, it is the CPU's responsibility to make a memory buffer area available to the receiver and to properly program the starting address of the DMA controller. The received frame is transferred to the memory buffer in the format shown in Figure 11. This method of reception is called Single Buffer reception; the entire frame is contained in one continuous buffer. Upon completion of reception, the status of the reception is appended at the end of the received frame in the memory buffer, and the total number of bytes transferred to the memory buffer is loaded into the internal status registers 1 and 2. An interrupt is then generated to inform the CPU of the frame reception.

Figure 10. The 82592 Frame Structure and Location of Data Element in System Memory
If the frame size is unknown, memory usage can be optimized by using Multiple Buffer reception. In this mode of operation, the CPU and DMA controller can dynamically allocate memory space as it receives frames. This method requires both DMA channels to receive the frame alternately. As frame reception begins, the 82592 interrupts the CPU and automatically requests assignment of the next available buffer. The CPU does this and loads the second DMA channel with the next buffer's information so the 82592 can immediately switch to the other channel when the current buffer is full. When the 82592 switches from the first to the second buffer it again interrupts the CPU and requests another buffer to be allocated on the previous channel. This process continues until the entire frame is received. The received frame is spread over multiple memory buffers. The link between the buffers is easily maintained by the CPU, using a buffer chain descriptor structure in memory as shown in Figure 12. This dynamic allocation of memory buffers results in efficient use of available storage when handling frames of widely differing sizes.

If the 82592 is programmed to generate the EOP signal to the 8237 or 82380 DMA controller, or if it is used with a DMA controller which implements the Tightly Coupled Interface, buffer reclamation and more advanced data structures for the buffer area can significantly improve system performance.

**EOP Signal to the DMA Controller**

The 82592 can be programmed to assert the EOP signal to the 8237 or 82380 DMA controller when one or more of the following occurs:

- A collision during transmission
- An error (CRC or alignment) during reception
- A good frame reception

If the 8237 or 82380 is programmed for Auto-initialize mode and if the 82592 is programmed to assert the EOP signal on a collision during transmission, the retransmission following a collision is done automatically by the 8237 and the 82592. The 8237 will reinitialize itself automatically and the 82592 will retransmit the same frame from the same memory area without CPU intervention. When the 82592 is programmed for this mode it does not interrupt the CPU upon a collision, and the CPU does not need to issue a Retransmit command to the 82592. The CPU is interrupted only after a successful transmission or retransmission, or after a transmission failure, such as DMA underrun.
If the 82592 is programmed to assert the EOP signal when an error occurs during reception, the 8237 or the 82380 in Auto-initialize mode will be able to reclaim the memory area which would otherwise be wasted for the errored frame reception. If the 82592 is programmed to assert EOP at the end of a frame reception, automatic buffer switching can be accomplished by alternating the DMA channels with the 8237 or the 82380. When the 82380 is used, the buffer switching can be done with only one DMA channel.

The EOP signal must be derived from the DACKT/CST/EOP pin using external logic (see Figure 13).

**82592/82560 Tightly Coupled Interface**

The 82592 has a mode of operation called “Tightly Coupled Interface.” In this mode the 82592 provides a tightly coupled interface to a DMA controller in order to execute some of the time-critical processes of the CSMA/CD protocol without any CPU intervention. By using the 82592’s companion chip, the 82560, or by implementing the Tightly Coupled Interface in a DMA controller, operations such as automatic retransmission, continuous back-to-back frame reception, and transmit and/or receive buffer chaining can be accomplished.

The 82592 provides the status of the current active transmission or reception to the DMA controller by using the DRQ and EOP signals at the end of every DMA cycle. The status is encoded according to Table 3. As long as the 82592 generates DRQ High and EOP Floating at the rising edge of RD or WR, the DMA controller repeats DMA transfers. If the transmission is completed without collisions or if the reception is good (no collision, no CRC, or no Alignment error), then DRQ and EOP both become Low at the end of a DMA transfer which follows the last DMA data transfer. If the transmission encountered a collision or if the reception had an error, DRQ becomes High and EOP becomes Low. The DMA controller must decode these signals appropriately and must reinitialize the DMA channel so it can retransmit the same frame or reclaim the otherwise wasted buffer. It is the DMA controller’s responsibility to reprogram itself for the next appropriate operation.

The 82560 fully implements the Tightly Coupled Interface and provides very-high-performance DMA services for the 82592 with minimal CPU involvement.

**Network Management and Diagnostics**

The 82592 provides a large set of diagnostic and network management functions including: internal and external loopback, monitor mode, optional capture of all frames regardless of destination address (Promiscuous mode), and time domain reflectometry for locating fault points in the network cable. The 82592 Dump command ensures software reliability by dumping the contents of the 82592 internal registers into the system memory.
Table 3. Transmit/Receive Status Encoding on DRQ and EOP

<table>
<thead>
<tr>
<th>DRQ</th>
<th>EOP</th>
<th>Status Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Hi-Z</td>
<td>Idle</td>
</tr>
<tr>
<td>1</td>
<td>Hi-Z</td>
<td>DMA Transfer</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>Transmission or Reception Terminated OK</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>Transmission or Reception Aborted</td>
</tr>
</tbody>
</table>

Other Enhancements

Compared to the previous generation LAN controllers such as the 82588, the 82592 has many functional and performance enhancements. This section lists some of these enhancements which are not covered in other sections.

1. **Multi-IA**—The 82592 implements multiple-individual address (Multi-IA) filtering. It can receive more than one IA frame in this mode.

2. **Power Down Modes**—Two power down modes, Local Power Down and Remote Power Down, are available. When the 82592 is in Remote Power Down mode, it can be powered up remotely by sending a special frame to it.

3. **Automatic Padding and IEEE 802.3 Length Field**—If a frame to be transmitted is shorter than the configured Slot Time, the 82592 automatically appends pad bytes up to the shortest frame greater than the Slot Time. If the data field of a received frame is longer than the byte count indicated in the Length field, the extra bytes are stripped automatically according to the Length field. Erroneous conditions are detected and reported by the 82592. An example of such conditions is reception of a frame which is shorter than the byte count indicated in the Length field.

4. **Automatic Retransmission on Collision During Preamble**—The 82592 can be programmed to retransmit automatically if it detects a collision during transmission of the preamble.

5. **On-Chip Jabber Inhibit Function**—The 82592 can be programmed to provide an on-chip jabber inhibit function.

6. **CRC Transfer to Memory**—The 82592 can be programmed to transfer the CRC field of a received frame into memory.

7. **Loopback Signal to the 82C501**—The 82592 can be programmed to provide an active High loopback signal to the 82C501 (see Figure 14).

8. **StarLAN**—The 82592 can be configured to recognize the IEEE 802.3 1BASE5 Collision Presence Signal (CPS). In this mode it also delays deactivation of the RTS signal at the end of a frame transmission in order to insert an end-of-frame marker as required by the standard.
APPLICATIONS

The 82592 can be used in a variety of applications. When it is used in High-Integration Mode it implements most of the Data Link and Physical Layer functions required by the IEEE 802.3 1BASE5 (StarLAN) and the IBM PC Network (Baseband and Broadband). When it is used in High-Speed Mode it can work with the 82C501 and the 82502 for IEEE 802.3 10BASE5 (Ethernet) and 10BASE2 (Cheapernet) implementations.

If the desired network requires determinism, the 82592's Deterministic Collision Resolution (DCR) method can be used.

Figure 15 shows a block diagram of an 82592/82560 Cheapernet adapter board. The 82560 provides the Tightly Coupled DMA interface for the 82592 and dual-port memory control for the static RAM. The 82592 is interfaced to the 82C501 to provide the Ethernet channel and then to the transceiver to provide the Cheapernet channel. Due to the CMOS process used for these chips, such a board uses much less power than a board based on NMOS or bipolar chips.

![Figure 14. Loopback Output to the 82C501](imageURL)
82588
High Integration LAN Controller

- Integrates ISO Layers 1 and 2
  - CSMA/CD Medium Access Control (MAC)
  - On-Chip Manchester, NRZI Encoding/Decoding
  - On-Chip Logic Based Collision Detect and Carrier Sense
- Supports Mid-Range Industry Standard LANs
  - StarLAN (IEEE 802.3 1BASE5)
  - IBM/PC Network-Baseband and Broadband
- High Level Command Interface Offloads the CPU
- Efficient Memory Use Via Multiple Buffer Reception

The 82588 is a highly integrated CSMA/CD controller designed for cost sensitive, mid-range Local Area Network (LAN) applications, such as personal computer networks.

At data rates of up to 2 Mb/s, the 82588 provides a highly integrated interface and performs: CSMA/CD Data Link Control, Manchester, Differential Manchester or NRZI encoding/decoding, clock recovery; Carrier Sense, and Collision Detection. This mode is called "High Integration Mode." In the 82588 "High Speed Mode", the user can transfer data at a rate of up to 5 Mb/s. In this mode the physical link functions are done external to the 82588.

The 82588 is available in a 28 pin DIP and 44 lead PLCC package and fabricated in Intel's reliable HMOS II 5 volt technology.
### Table 1. Pin Description

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Pin No.</th>
<th>Type</th>
<th>Name and Function</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>DIP</td>
<td>PLCC</td>
<td></td>
</tr>
<tr>
<td>D7</td>
<td>6</td>
<td>10</td>
<td>I/O</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>DATA BUS: The Data Bus lines are bi-directional three state lines connected to the system's Data Bus for the transfer of data, commands, status and parameters.</td>
</tr>
<tr>
<td>D6</td>
<td>7</td>
<td>11</td>
<td></td>
</tr>
<tr>
<td>D5</td>
<td>8</td>
<td>12</td>
<td></td>
</tr>
<tr>
<td>D4</td>
<td>9</td>
<td>13</td>
<td></td>
</tr>
<tr>
<td>D3</td>
<td>10</td>
<td>14</td>
<td></td>
</tr>
<tr>
<td>D2</td>
<td>11</td>
<td>18</td>
<td></td>
</tr>
<tr>
<td>D1</td>
<td>12</td>
<td>19</td>
<td></td>
</tr>
<tr>
<td>D0</td>
<td>13</td>
<td>20</td>
<td></td>
</tr>
<tr>
<td>RD</td>
<td>5</td>
<td>9</td>
<td>I</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>READ: Together with CS, DACK0 or DACK1, Read controls data or status transfers out of the 82588 registers.</td>
</tr>
<tr>
<td>WR</td>
<td>3</td>
<td>4</td>
<td>I</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>WRITE: Together with CS, DACK0 or DACK1, Write controls data or command transfers into the 82588 registers.</td>
</tr>
<tr>
<td>CS</td>
<td>2</td>
<td>3</td>
<td>I</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>CHIP SELECT: When this signal is LOW, the 82588 is selected by the CPU for transfer of command or status. The direction of data flow is determined by the RD or WR inputs.</td>
</tr>
<tr>
<td>CLK</td>
<td>4</td>
<td>5</td>
<td>I</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>CLOCK: System clock. TTL compatible signal.</td>
</tr>
<tr>
<td>RESET</td>
<td>25</td>
<td>40</td>
<td>I</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>RESET: A HIGH signal on this pin will cause the 82588 to terminate current activity. This signal is internally synchronized and must be held HIGH for at least four Clock cycles.</td>
</tr>
<tr>
<td>INT</td>
<td>26</td>
<td>41</td>
<td>O</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>INTERRUPT: Active HIGH signal indicates to the CPU that the 82588 is requesting an interrupt.</td>
</tr>
<tr>
<td>DRQ0</td>
<td>17</td>
<td>26</td>
<td>O</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>DMA REQUEST (CHANNEL 0): This pin is used by the 82588 to request a DMA transfer. DRQ0 remains HIGH as long as 82588 requires data transfers. Burst transfers are done by having the signal active for multiple transfers.</td>
</tr>
<tr>
<td>DRQ1</td>
<td>18</td>
<td>27</td>
<td>O</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>DMA REQUEST (CHANNEL 1): This pin is used by the 82588 to request a DMA transfer. DRQ1 remains HIGH as long as 82588 requires data transfers. Burst transfers are done by having the signal active or multiple transfers.</td>
</tr>
<tr>
<td>DACK0</td>
<td>1</td>
<td>2</td>
<td>I</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>DMA ACKNOWLEDGE (CHANNEL 0): When LOW, this input signal from the DMA Controller notifies the 82588 that the requested DMA cycle is in progress. This signal acts like chip select for data and parameter transfer using DMA channel 0.</td>
</tr>
<tr>
<td>DACK1</td>
<td>27</td>
<td>42</td>
<td>I</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>DMA ACKNOWLEDGE (CHANNEL 1): When LOW, this input signal from the DMA controller notifies the 82588 that the requested DMA cycle is in progress. This signal acts like chip select for data and parameter transfer using DMA channel 1.</td>
</tr>
<tr>
<td>Symbol</td>
<td>Pin No.</td>
<td>Type</td>
<td>Name and Function</td>
</tr>
<tr>
<td>--------</td>
<td>---------</td>
<td>------</td>
<td>-------------------</td>
</tr>
<tr>
<td>X1/X2</td>
<td>15/16</td>
<td>24/25</td>
<td><strong>High Integration Mode</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td><strong>OSCILLATOR INPUTS:</strong> These inputs may be used to connect a quartz crystal that controls the internal clock generator for the serial unit. X1 may also be driven by a MOS level clock whose frequency is 8 or 16 times the bit rate of Transmit/Receive data. X2 must be left floating if X1 has an external MOS clock.</td>
</tr>
<tr>
<td>TxC</td>
<td>15</td>
<td>24</td>
<td><strong>High Speed Mode</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td><strong>TRANSMIT CLOCK:</strong> This signal provides timing information to the internal serial logic, depending upon the mode of data transfer. For NRZ encoding, data is transferred to the TxD pin on the HIGH to LOW clock transition. For Manchester encoding the transmitted bit center is aligned with the TxC LOW to HIGH transition.</td>
</tr>
<tr>
<td>RxC</td>
<td>16</td>
<td>25</td>
<td><strong>RECEIVE CLOCK:</strong> This signal provides timing information to the internal serial logic. NRZ data should be provided for reception (RxD). The state of the RxD pin is sampled on the HIGH to LOW transition of RxC. The operating mode of the 82588 is defined when configuring the chip.</td>
</tr>
<tr>
<td>TCLK/CRS</td>
<td>24</td>
<td>36</td>
<td>In High Speed Mode, this pin is Carrier Sense, input CRS, and is used to notify the 82588 that there is activity on the serial link. In High Integration Mode, this pin is Transmit Clock, TCLK, and is used to output the transmit clock.</td>
</tr>
<tr>
<td>CDT</td>
<td>23</td>
<td>35</td>
<td><strong>COLLISION DETECT:</strong> This input notifies the 82588 that a collision has occurred. It is sensed only if the 82588 is configured for external Collision Detect (external circuitry is then required for detecting the collision).</td>
</tr>
<tr>
<td>RxD</td>
<td>19</td>
<td>31</td>
<td><strong>RECEIVE DATA:</strong> This pin receives serial data.</td>
</tr>
<tr>
<td>TxD</td>
<td>20</td>
<td>32</td>
<td><strong>TRANSMIT DATA:</strong> This pin transmits data to the Serial Link. This signal is HIGH when not transmitting.</td>
</tr>
<tr>
<td>RTS</td>
<td>21</td>
<td>33</td>
<td><strong>REQUEST TO SEND:</strong> When this signal is LOW, the 82588 notifies an external interface that it has data to transmit. It is forced HIGH after a reset and when transmission is stopped.</td>
</tr>
<tr>
<td>CTS</td>
<td>22</td>
<td>34</td>
<td><strong>CLEAR TO SEND:</strong> CTS enables the 82588 to start transmitting data. Raising this signal to HIGH stops the transmission.</td>
</tr>
<tr>
<td>VCC</td>
<td>28</td>
<td>1, 43, 44</td>
<td><strong>POWER:</strong> +5V Supply</td>
</tr>
<tr>
<td>VSS</td>
<td>14</td>
<td>21, 22, 23</td>
<td>Ground</td>
</tr>
</tbody>
</table>
Table 1. Pin Description (Continued)

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Pin No.</th>
<th>Type</th>
<th>Name and Function</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NC</td>
<td>6</td>
<td>DIP</td>
<td></td>
</tr>
<tr>
<td></td>
<td>7</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>8</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>15</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>16</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>17</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>28</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>29</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>30</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>37</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>38</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>39</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>PLCC</td>
<td></td>
</tr>
</tbody>
</table>

NC - NO CONNECT: These pins are reserved for future use.

FUNCTIONAL DESCRIPTION

High Integration Mode

The 82588 LAN Controller is a highly integrated CSMA/CD controller for cost sensitive LAN applications such as personal computer networks. Included on chip is a programmable CSMA/CD controller, an NRZI and Manchester encoder/decoder with clock recovery, and two collision detection mechanisms. With the addition of simple transceiver line drivers or RF Modem, the 82588 performs all the major functions of the ISO Physical and Data Link Layers.

CSMA/CD Controller

The 82588 on-chip CSMA/CD controller is programmable, which allows it to operate in a variety of LAN environments, including industry standards such as StarLAN (IEEE 802.3 1BASE5) and the 2 Mb/s IBM PC Network (both baseband and broadband transmission). Programmable parameters include:
- Framing (End of Carrier of SDLC)
- Address field length
- Station priority
- Interframe spacing
- Slot time
- CRC-32 OR CRC-16

Encoder/Decoder

The on-chip NRZI and Manchester encoder/decoder supports data rates up to 2 Mb/s. Manchester encoding is typically used in baseband applications and NRZI is used in broadband applications.

Collision Detection

One of the 82588's unique features is its on-chip logic based collision detection. To ensure a high probability of collision detection two mechanisms are provided. The Code Violation method defines a collision when a transition edge occurs outside the area of normal transitions as specified by either the Manchester or NRZI encoding methods. Bit Comparison method compares the signature of the transmitted frame to the received frame signature (re-calculated by the 82588 while listening to itself). If the signatures are identical the frame is assumed to have been transmitted without a collision.

System Interface

In addition to providing the functions necessary for interfacing to the LAN, the 82588 has a friendly system interface that eases the design effort. First, the 82588 has a high level command interface; that is the CPU sends the 82588 commands such as Transmit or Configure. This means the designer does not have to write low level software to perform these tasks, and it offloads the CPU in the application. Second, the 82588 supports an efficient memory structure called Multiple Buffer Reception in which buffers are chained together while receiving frames. This is an important feature in applications with limited memory, such as personal computers. Third, the 82588 has two independent sixteen byte FIFO’s, one for reception and one for transmission. The FIFO’s allow the 82588 to tolerate bus latency. Finally the 82588 provides an eight byte data path that supports up to 4 Mbytes/second using external DMA.
Network Management & Diagnostics

The 82588 provides a rich set of diagnostic and network management functions including: internal and external loopback, channel activity indicators, optional capture of all frames regardless of destination address (Promiscuous Mode), capture of collided frames, (if address matches), and time domain reflectometry for locating fault points in the network cable. The 82588 register Dump command ensures reliable software by dumping the content of the 82588 registers into the system memory.

The next section will describe the 82588 system bus interface, the 82588 network interface, and the 82588 internal architecture.

82588/Host CPU Interaction

The CPU communicates with the 82588 through the system's memory and 82588's on-chip registers. The CPU creates a data structure in the memory, programs the external DMA controller with the start address and byte count of the block, and issues the command to the 82588.

The 82588 is optimized for operating with the iAPX 186/188, but due to the small number of hardware signals between the 82588 and the CPU, the 82588 can operate easily with other processors. The data bus is 8 bits wide and there is no address bus.

Chip Select and interrupt lines are used to communicate between the 82588 and the host as shown in the Figure 3. Interrupt is used by the 82588 to draw the CPU's attention. The Chip Select is used by the CPU to draw the 82588's attention.

There are two kinds of transfer over the bus: Command/Status and data transfers. Command/Status transfers are always performed by the CPU. Data transfers are requested by the 82588, and are typically performed by a DMA controller. The table given in Figure 4 shows the Command/Status and data transfer control signals.

The CPU writes to 82588 using CS and WR signals. The CPU reads the 82588 status register using CS and RD signals.
### Pin Name and Function

<table>
<thead>
<tr>
<th>CS*</th>
<th>RD</th>
<th>WR</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>x</td>
<td>x</td>
<td>No transfer to/from Command/Status</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>Illegal</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>Read from status register</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>Write to Command register</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>DACK0[DACK1]*</th>
<th>RD</th>
<th>WR</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>x</td>
<td>x</td>
<td>No DMA transfer</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>Illegal</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>Data Read from DMA channel 0 [or 1]</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>Data Write to DMA channel 0 [or 1]</td>
</tr>
</tbody>
</table>

* Only one of CS, DACK0 and DACK1 may be active at any time.

**Figure 4. Databus Control Signals and Their Functions**

<table>
<thead>
<tr>
<th>COMANDS</th>
<th>VALUE</th>
<th>COMANDS</th>
<th>VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>NOP</td>
<td>0</td>
<td>ABORT</td>
<td>13</td>
</tr>
<tr>
<td>IA-SETUP</td>
<td>1</td>
<td>RECEIVER-ENABLE</td>
<td>8</td>
</tr>
<tr>
<td>CONFIGURE</td>
<td>2</td>
<td>ASSIGN NEXT BUF</td>
<td>9</td>
</tr>
<tr>
<td>MC-SETUP</td>
<td>3</td>
<td>RECEIVE-DISABLE</td>
<td>10</td>
</tr>
<tr>
<td>TRANSMIT</td>
<td>4</td>
<td>STOP-RECEPTION</td>
<td>11</td>
</tr>
<tr>
<td>TDR</td>
<td>5</td>
<td>RESET</td>
<td>14</td>
</tr>
<tr>
<td>DUMP</td>
<td>6</td>
<td>FIX PTR</td>
<td>15 (CHNL = 1)</td>
</tr>
<tr>
<td>DIAGNOSE</td>
<td>7</td>
<td>RLS PTR</td>
<td>15 (CHNL = 0)</td>
</tr>
<tr>
<td>RETRANSMIT</td>
<td>12</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Figure 5. Command Format and Operation Values**
Transmitting a Frame

To transmit a frame, the CPU prepares a Transmit Data Block in memory as shown in Figure 7. Its first two bytes specify the length of the rest of the block. The next few bytes (Up to 6 bytes long) contain the destination address of the node it is being sent to. The rest of the block is the data field. The CPU programs the DMA controller with the start address of the block, length of the block and other control information and then issues the Transmit command to the 82588.

Upon receiving the command, the 82588 fetches the first two bytes of the block to determine the length of the block. If the link is free, and the first data byte was fetched, the 82588 begins transmitting the preamble and concurrently fetches the bytes from the Transmit Data Block and loads them into a 16 byte FIFO to keep them ready for transmitting. The FIFO is a buffer between the serial and parallel part of the 82588. The on-chip FIFOs help the 82588 to tolerate system bus latency as well as provide efficient usage of system bandwidth.

The destination address is sent out after the preamble. This is followed by the source or the station individual address, which is stored earlier on the 82588 using the IA-SETUP command. After that, the entire information field is transmitted followed by a CRC field calculated by the 82588. If during the transmission of the frame, a collision is encountered, then the transmission is aborted and a jam pattern is sent out after completion of the preamble. The 82588 generates an Interrupt indicating the experience of a collision and the frame has to be re-transmitted. Re-transmission is done by the CPU exactly as the Transmit command except the Re-Transmit command keeps track of the number of collisions encountered. When the 82588 gets the Re-Transmit command and the exponential back-off time is expired, the 82588 transmits the frame again. The transmitted frame can be coded to either Manchester, Differential Manchester or NRZI methods.
Collision Detection

The 82588 eliminates the need for external collision detection logic, in most applications, while easing or eliminating the need for complex transceivers. Two algorithms are used for collision detection: Bit Comparison and Code Violation. The Bit Comparison Method is useful in Broadband networks where there are separate transmit and receive channels. Bit Comparison compares the "signature" of the transmitted data and received data at the end of the collision window in any network configuration. This algorithm calculates the CRC after a programmable number of transmitted bits, holds this CRC in a register, and compares it with received data's CRC. A CRC or "signature" difference indicates a collision. The code violation is detected if the encoding of the received data has any bit that does not fit the encoding rules. The code violation method is useful in short bus topology and serial backplane applications where bit attenuation over the bus is negligible.

Figure 7. The 82588 Frame Structure and location of Data element in System Memory

Figure 8. Single Buffer Reception
Receiving a Frame

The 82588 can receive a frame when its receiver has been enabled. The received frame is decoded by either on-chip Manchester, Differential Manchester or NRZI decoders in High Integration Mode and NRZI in High Speed Mode. The 82588 checks for an address match for an individual address, a Multicast address or a Broadcast address. In the Promiscuous mode the 82588 receives all frames. Only when the address match is successful does the 82588 transfer the frame to the memory using the DMA controller. Before enabling the receiver, the CPU makes a memory buffer area available to the Receive Unit and programs the starting address of the DMA controller. The received frame is transferred to the memory buffer in the format shown in Figure 8. This method of reception is called "Single Buffer" reception. The entire frame is contained in one continuous buffer. Upon completion of reception the total number of bytes written into the memory buffer is loaded into status registers 1 and 2 and the status of the reception itself is appended to the received frame. An interrupt to the CPU follows.

If the frame size is unknown, memory usage can be optimized by using "Multiple Buffer" reception.

This way the user does not have to allocate large memory space for short frames. Instead, the 82588 can dynamically allocate memory space as it receives frames. This method requires both DMA channels alternately to receive the frame. As the frame reception starts, the 82588 interrupts the CPU and automatically requests assignment of the next sequential buffer. The CPU does this and loads the second DMA channel with the next buffer information so that the 82588 can immediately switch to the other channel as soon as the current buffer is full. When the 82588 switches from the first to the second buffer it again interrupts the CPU requesting it to allocate another buffer on the other (previous) channel in advance. This process continues until the entire frame is received. The received frame is spread over multiple memory buffers. The link between the buffers is easily maintained by the CPU using a buffer chain descriptor structure in memory (see Figure 9).

This dynamic (pre) allocation of memory buffers results in efficient use of available storage when handling frames of widely differing sizes. Since the buffers are pre-allocated one block in advance, the system is not time critical.

80188 Based System

Figure 10 shows a high performance, high-integration configuration of the 82588 with the 80188 in a typical iAPX188-based microcomputer. The 80188 controls the 82588, as well as providing DMA control services for data transfer, using its on-chip two channel DMA controller.

![Figure 9. Multiple Buffer Reception](image-url)
Link Interface

The Serial Interface Mode configuration parameter selects either a highly integrated Direct Link interface (High Integration Mode) or a highly flexible Transceiver Interface (High Speed Mode).

Application

In the High Integration Mode it is possible to connect the 82588 on a very short "Wired OR" link, on a longer twisted pair cable, or a broadband connection.

Twisted Pair Connection

The link consists of a twisted pair that interconnects the 82588. The transmit data pin is connected via a driver and the receive data pin is connected via a buffer. The twisted pair must be properly terminated to prevent reflections.

In the minimum configuration, TxD and RxD are connected to the twisted pair and CTS is grounded. The 82588 may control the driver with the RTS pin. It is also possible to use external circuitry for performing collision detection, and feeding it to the 82588 through the CDT pin.

Broadband Connection

The 82588 supports data communications over a broadband link in both its modes. Proper MODEM interface should be provided. Collision Detection by Bit Comparison, in High Interface Mode, can be applied to transmission over broadband links.
Figure 10. 80188 Based System
**Absolute Maximum Ratings**

Ambient Temperature Under Bias ....... 0°C to +70°C  
Storage Temperature ............ −65°C to +150°C  
Voltage on Any Pin With  
  Respect to Ground .......... −1.0V to 7V  
Power Dissipation ................. 1.7 Watts

*Notice: Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions above those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

**D.C. Characteristics**

(\(T_A = 0°C \text{ to } +70°C; T_C \text{ (DIP)} = 52°C \text{ to } 108°C, T_C \text{ (PLCC)} = 63°C \text{ to } 116°C; VCC = +5V \pm 10%\))

\(T_XC, R_XC\) have MOS levels (See VMIL, VMIH). All other signals have TTL levels (See VI, VIH, VOL, VOH).

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Parameter</th>
<th>Min</th>
<th>Max</th>
<th>Units</th>
<th>Test Conditions</th>
</tr>
</thead>
<tbody>
<tr>
<td>VIL</td>
<td>Input Low Voltage (TTL)</td>
<td>−0.5</td>
<td>+0.8</td>
<td>V</td>
<td></td>
</tr>
<tr>
<td>VIH</td>
<td>Input High Voltage (TTL)</td>
<td>2.0</td>
<td>VCC + 0.5</td>
<td>V</td>
<td></td>
</tr>
<tr>
<td>VOL</td>
<td>Output Low Voltage (TTL)</td>
<td>0.45</td>
<td>V</td>
<td>IOL = 2.0 mA</td>
<td></td>
</tr>
<tr>
<td>VOH</td>
<td>Output High Voltage (TTL)</td>
<td>2.4</td>
<td>V</td>
<td>IOH = −400 μA</td>
<td></td>
</tr>
<tr>
<td>VMIL</td>
<td>Input Low Voltage (MOS)</td>
<td>−0.5</td>
<td>0.6</td>
<td>V</td>
<td></td>
</tr>
<tr>
<td>VMIH</td>
<td>Input High Voltage (MOS)</td>
<td>3.9</td>
<td>VCC + 0.5</td>
<td>V</td>
<td></td>
</tr>
<tr>
<td>ILI</td>
<td>Input Leakage Current</td>
<td>+10</td>
<td>μA</td>
<td>0 = VIN = VCC</td>
<td></td>
</tr>
<tr>
<td>ILO</td>
<td>Output Leakage Current</td>
<td>±10</td>
<td>μA</td>
<td>0.45 = VOUT = VCC</td>
<td></td>
</tr>
<tr>
<td>ICC</td>
<td>Power Supply Current</td>
<td>400</td>
<td>mA</td>
<td>(T_A = 0°C)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>300</td>
<td>mA</td>
<td>(T_A = +70°C)</td>
<td></td>
</tr>
</tbody>
</table>

**A.C. Characteristics**

(\(T_A = 0°C \text{ to } +70°C; T_C \text{ (DIP)} = 52°C \text{ to } 108°C, T_C \text{ (PLCC)} = 63°C \text{ to } 116°C; VCC = +5V \pm 10%\))

**System Clock Parameters**

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Parameter</th>
<th>Min</th>
<th>Max</th>
<th>Units</th>
<th>Test Conditions</th>
</tr>
</thead>
<tbody>
<tr>
<td>T1</td>
<td>CLK Cycle Period</td>
<td>125</td>
<td></td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>T2</td>
<td>CLK Low Time</td>
<td>53</td>
<td>1000</td>
<td>ns</td>
<td>*5</td>
</tr>
<tr>
<td>T3</td>
<td>CLK High Time</td>
<td>53</td>
<td></td>
<td>ns</td>
<td>*6</td>
</tr>
<tr>
<td>T4</td>
<td>CLK Rise Time</td>
<td>15</td>
<td></td>
<td>ns</td>
<td>*1</td>
</tr>
<tr>
<td>T5</td>
<td>CLK Fall Time</td>
<td>15</td>
<td></td>
<td>ns</td>
<td>*2</td>
</tr>
</tbody>
</table>
## A.C. Characteristics (Continued)

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Parameter</th>
<th>Min</th>
<th>Max</th>
<th>Units</th>
<th>Test Conditions</th>
</tr>
</thead>
<tbody>
<tr>
<td>T6</td>
<td>Reset Active to Clock Low</td>
<td>20</td>
<td></td>
<td>ns</td>
<td>*3</td>
</tr>
<tr>
<td>T8</td>
<td>Reset Pulse Width</td>
<td>4T1</td>
<td></td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>T9</td>
<td>Control Inactive After Reset</td>
<td>T1</td>
<td></td>
<td>ns</td>
<td></td>
</tr>
</tbody>
</table>

### Interrupt Timing Parameters

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Parameter</th>
<th>Min</th>
<th>Max</th>
<th>Units</th>
<th>Test Conditions</th>
</tr>
</thead>
<tbody>
<tr>
<td>T10</td>
<td>CLK High to Interrupt Active</td>
<td>85</td>
<td></td>
<td>ns</td>
<td>*4</td>
</tr>
<tr>
<td>T11</td>
<td>WR Idle to Interrupt Idle</td>
<td>85</td>
<td></td>
<td>ns</td>
<td>*4</td>
</tr>
</tbody>
</table>

### Write Parameters

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Parameter</th>
<th>Min</th>
<th>Max</th>
<th>Units</th>
<th>Test Conditions</th>
</tr>
</thead>
<tbody>
<tr>
<td>T12</td>
<td>CS or DACK0 or DACK1 Setup to WR Low</td>
<td>0</td>
<td></td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>T13</td>
<td>WR Pulse Width</td>
<td>95</td>
<td></td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>T14</td>
<td>CS or DACK0 or DACK1 Hold After WR High</td>
<td>0</td>
<td></td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>T15</td>
<td>Data Setup to WR High</td>
<td>75</td>
<td></td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>T16</td>
<td>Data Hold After WR High</td>
<td>0</td>
<td></td>
<td>ns</td>
<td></td>
</tr>
</tbody>
</table>

### Read Parameters

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Parameter</th>
<th>Min</th>
<th>Max</th>
<th>Units</th>
<th>Test Conditions</th>
</tr>
</thead>
<tbody>
<tr>
<td>T17</td>
<td>CS or DACK0 or DACK1 Setup to RD Low</td>
<td>0</td>
<td></td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>T18</td>
<td>RD Pulse Width</td>
<td>95</td>
<td></td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>T19</td>
<td>CS or DACK0 or DACK1 Address Valid After RD High</td>
<td>0</td>
<td></td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>T20</td>
<td>RD Low to Data Valid</td>
<td>80</td>
<td></td>
<td>ns</td>
<td>*7</td>
</tr>
<tr>
<td>T21</td>
<td>Data Float After RD High</td>
<td>55</td>
<td></td>
<td>ns</td>
<td>*7</td>
</tr>
</tbody>
</table>

### DMA Parameters

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Parameter</th>
<th>Min</th>
<th>Max</th>
<th>Units</th>
<th>Test Conditions</th>
</tr>
</thead>
<tbody>
<tr>
<td>T22</td>
<td>CLK Low to DRQ0 or DRQ1 Active</td>
<td>85</td>
<td></td>
<td>ns</td>
<td>*4</td>
</tr>
<tr>
<td>T23</td>
<td>WR or RD Low to DRQ0 or DRQ1 Inactive</td>
<td>60</td>
<td></td>
<td>ns</td>
<td>*4</td>
</tr>
</tbody>
</table>

**NOTES:**

*1—0.8V—2.0V
*2—2.0V—0.8V
*3—to guarantee recognition at next clock
*4—CL = 50 pF
*5—measured at 1.5V
*6—measured at 1.5V
*7—CL = 20 pF—200 pF
AC TESTING INPUT/OUTPUT WAVEFORM

AC Testing Inputs are Driven at 2.4V for a Logic 1 and 0.45V for a Logic 0. Timing Measurements are Made at 1.5V for Both a Logic 1 and 0.

Rise and Fall Time of Input/Output Signals are Measured Between 0.8V to 2.0V Respectively.

TTL Input/Output Voltage Levels for Timing Measurements

Rise and Fall Time of Input Signals are Measured Between 1.0V to 3.5V Respectively.

Clock Input Voltage Levels for Timing Measurements

Interrupt Timing (Going Active)
Interrupt Timing (Going Inactive)

Reset Timing
Serial Interface A.C. Timing Characteristics
High Integration Mode

TFC is the crystal or serial clock input at the X1 pin. When a serial clock is provided at the X1 pin, the maximum capacitive load allowed on the X2 pin is 15 pF.

**TFC Frequency Range:**

### For Oscillator Frequency = 1 to 16 MHz (High)

<table>
<thead>
<tr>
<th>Parameter</th>
<th>×8 Sampling</th>
<th>×16 Sampling</th>
</tr>
</thead>
<tbody>
<tr>
<td>TCLK Frequency</td>
<td>0.125 – 2 MHz</td>
<td>62.5 kHz – 1 MHz</td>
</tr>
<tr>
<td>T29 = TCLK Cycle Time</td>
<td>8 × T24</td>
<td>16 × T24</td>
</tr>
<tr>
<td>T30 = TCLK High Time</td>
<td>T24 (Typically)</td>
<td>T24 (Typically)</td>
</tr>
<tr>
<td>T31 = TCLK Low Time</td>
<td>7 × T24 (Typically)</td>
<td>15 × T24 (Typically)</td>
</tr>
</tbody>
</table>

### For Oscillator Frequency = 0 to 1 MHz (Low)*

<table>
<thead>
<tr>
<th>Parameter</th>
<th>×8 Sampling</th>
<th>×16 Sampling</th>
</tr>
</thead>
<tbody>
<tr>
<td>TCLK Frequency</td>
<td>0 – 0.125 MHz</td>
<td>0 – 6.25 kHz</td>
</tr>
<tr>
<td>T29 = TCLK Cycle Time</td>
<td>8 × T24</td>
<td>16 × T24</td>
</tr>
<tr>
<td>T30 = TCLK High Time</td>
<td>T25 (Typically)</td>
<td>T25 (Typically)</td>
</tr>
<tr>
<td>T31 = TCLK Low Time</td>
<td>7 × T24 + T26 (Typically)</td>
<td>15 × T24 + T26 (Typically)</td>
</tr>
</tbody>
</table>

*A non-symmetrical clock should be provided so that T25 is less than 1000 ns.

T24 = Serial Clock Period
T25 = Serial Clock High Time
T26 = Serial Clock Low Time

---

**High Speed Mode**
- Applies for TxC, RxC
- f max = 5 MHz ± 100 ppm
- For Manchester, symmetry is required: T63, T64 = \( \frac{1}{2f} \) ± 5%

**High Integration Mode**

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Parameter</th>
<th>Min</th>
<th>Max</th>
<th>Units</th>
<th>Test Conditions</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>External (Fast) Clock Parameters</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>T24</td>
<td>62.5</td>
<td></td>
<td>ns</td>
<td>*1</td>
</tr>
<tr>
<td></td>
<td>T25</td>
<td>18.5</td>
<td>1000</td>
<td>ns</td>
<td>*1, *14</td>
</tr>
<tr>
<td></td>
<td>T26</td>
<td>23.5</td>
<td></td>
<td>ns</td>
<td>*1</td>
</tr>
<tr>
<td></td>
<td>T27</td>
<td>5</td>
<td></td>
<td>ns</td>
<td>*1</td>
</tr>
<tr>
<td></td>
<td>T28</td>
<td>5</td>
<td></td>
<td>ns</td>
<td>*1</td>
</tr>
<tr>
<td></td>
<td>Transmit Clock Parameters</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>T29</td>
<td>500</td>
<td></td>
<td>ns</td>
<td>*3, *12</td>
</tr>
<tr>
<td></td>
<td>T30</td>
<td>*8</td>
<td>1070</td>
<td>ns</td>
<td>*3</td>
</tr>
<tr>
<td></td>
<td>T31</td>
<td>*9</td>
<td></td>
<td>ns</td>
<td>*3</td>
</tr>
<tr>
<td></td>
<td>T32</td>
<td>15</td>
<td></td>
<td>ns</td>
<td>*3</td>
</tr>
<tr>
<td></td>
<td>T33</td>
<td>15</td>
<td></td>
<td>ns</td>
<td>*3</td>
</tr>
</tbody>
</table>
### High Integration Mode (Continued)

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Parameter</th>
<th>Min</th>
<th>Max</th>
<th>Units</th>
<th>Test Conditions</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><strong>Transmit Data Parameters (Manchester, Differential Manchester)</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T34</td>
<td>TxD Transition-Transition</td>
<td>4T24-10</td>
<td>ns</td>
<td>*12</td>
<td></td>
</tr>
<tr>
<td>T35</td>
<td>TCLK Low to TxD Transition Half Bit Cell</td>
<td>*10</td>
<td>*2, *12</td>
<td></td>
<td></td>
</tr>
<tr>
<td>T36</td>
<td>TCLK Low to TxD Transition Full Bit Cell</td>
<td>*11</td>
<td>*2, *12</td>
<td></td>
<td></td>
</tr>
<tr>
<td>T37</td>
<td>TxD Rise Time</td>
<td>15</td>
<td>ns</td>
<td>*2</td>
<td></td>
</tr>
<tr>
<td>T38</td>
<td>TxD Fall Time</td>
<td>15</td>
<td>ns</td>
<td>*2</td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>Transmit Data Parameters (NRZI)</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T39</td>
<td>TxD Transition-Transition</td>
<td>8T24-10</td>
<td>ns</td>
<td>*12</td>
<td></td>
</tr>
<tr>
<td>T40</td>
<td>TCLK Low to TxD Transition</td>
<td>*10</td>
<td>*2, *12</td>
<td></td>
<td></td>
</tr>
<tr>
<td>T41</td>
<td>TxD Rise Time</td>
<td>15</td>
<td>ns</td>
<td>*2</td>
<td></td>
</tr>
<tr>
<td>T42</td>
<td>TxD Fall Time</td>
<td>15</td>
<td>ns</td>
<td>*2</td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>RTS, CTS, Parameters</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T43</td>
<td>TCLK Low To RTS Low</td>
<td>*10</td>
<td>*3, *12</td>
<td></td>
<td></td>
</tr>
<tr>
<td>T44</td>
<td>CTS Low to TCLK Low CTS Setup Time</td>
<td>65</td>
<td>ns</td>
<td></td>
<td></td>
</tr>
<tr>
<td>T45</td>
<td>TCLK low to RTS High</td>
<td>*10</td>
<td>*3, *12</td>
<td></td>
<td></td>
</tr>
<tr>
<td>T46</td>
<td>TCLK Low to CTS Invalid, CTS Hold Time</td>
<td>20</td>
<td>ns</td>
<td>*4, *13</td>
<td></td>
</tr>
<tr>
<td>T47</td>
<td>CTS High to TCLK Low, CTS Setup Time to Stop Transmission</td>
<td>65</td>
<td>ns</td>
<td>*4</td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>IFS Parameters</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T48</td>
<td>Interframe Delay</td>
<td>*5</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>Collision Detect Parameter</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T49</td>
<td>CDT Low to TCLK High, External Collision Detect Setup Time</td>
<td>50</td>
<td>ns</td>
<td>*13</td>
<td></td>
</tr>
<tr>
<td>T50</td>
<td>CDT High to TCLK Low</td>
<td>50</td>
<td>ns</td>
<td>*13</td>
<td></td>
</tr>
<tr>
<td>T51</td>
<td>TCLK High to CDT Inactive, CDT Hold Time</td>
<td>20</td>
<td>ns</td>
<td>*13</td>
<td></td>
</tr>
</tbody>
</table>
### High Integration Mode (Continued)

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Parameter</th>
<th>Min</th>
<th>Max</th>
<th>Units</th>
<th>Test Conditions</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><strong>Collision Detect Parameters (Continued)</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T52</td>
<td>CDT Low to Jamming Start</td>
<td></td>
<td>*6</td>
<td></td>
<td></td>
</tr>
<tr>
<td>T53</td>
<td>Jamming Period</td>
<td></td>
<td>*7</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>Received Data Parameters (Manchester)</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T54</td>
<td>RxD Transition-Transition</td>
<td>4T24</td>
<td>ns</td>
<td>*12</td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>Received Data Parameters (Manchester)</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T55</td>
<td>RxD Rise Time</td>
<td>10</td>
<td>ns</td>
<td>*1</td>
<td></td>
</tr>
<tr>
<td>T56</td>
<td>RxD Fall Time</td>
<td>10</td>
<td>ns</td>
<td>*1</td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>Received Data Parameters (NRZI)</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T57</td>
<td>RxD Transition-Transition</td>
<td>8T24</td>
<td>ns</td>
<td>*12</td>
<td></td>
</tr>
<tr>
<td>T58</td>
<td>RxD Rise Time</td>
<td>10</td>
<td>ns</td>
<td>*1</td>
<td></td>
</tr>
<tr>
<td>T59</td>
<td>RxD Fall Time</td>
<td>10</td>
<td>ns</td>
<td>*1</td>
<td></td>
</tr>
</tbody>
</table>

**NOTES:**

*1—MOS levels.
*2—1 TTL load + 50 pF.
*3—1 TTL load + 100 pF.
*4—Abnormal end to transmission: CTS expires before RTS.
*5—Programmable value: T48 = NIFS × T29 (ns) NIFS—the IFS configuration value.
If NIFS is less than 12, then it is enforced to 12.
*6—Programmable value: T52 = NCDF × T29 + (12 to 15) × T29 (if collision occurs after preamble).
*7—T53 = 32 × T29
*8—Depends on T24 frequency range:
  High Range: T24 = 10
  Low Range: T25 = 10
*9—T31 = T29 - T30 - T32 - T33
*10—2T24 + 40 ns
*11—6T24 + 40 ns
*12—For ×16 sampling clock parameter minimum value should be multiplied by a factor of 2.
*13—To guarantee recognition on the next clock.
*14—62.5 ns minimum in Low Range.

#### Write Timing

![Write Timing Diagram](image-url)

231161-13
Transmit Timings: Clocks RTS and CTS

Transmit Timings—Manchester Data Encoding

Transmit Timings—Lost CTS
### High Speed Mode

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Parameter</th>
<th>Min</th>
<th>Max</th>
<th>Units</th>
<th>Test Conditions</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Transmit/Receive Clock Parameters</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T60</td>
<td>TxC TxC Cycle</td>
<td>200</td>
<td>*13</td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>T61</td>
<td>TxC Rise Time</td>
<td></td>
<td>10</td>
<td>ns</td>
<td>*1</td>
</tr>
<tr>
<td>T62</td>
<td>TxC Fall Time</td>
<td></td>
<td>10</td>
<td>ns</td>
<td>*1</td>
</tr>
<tr>
<td>T63</td>
<td>TxC High</td>
<td>80</td>
<td>1000</td>
<td>ns</td>
<td>*1, *3</td>
</tr>
<tr>
<td>T64</td>
<td>TxC Low</td>
<td>80</td>
<td></td>
<td>ns</td>
<td>*1, *3</td>
</tr>
<tr>
<td><strong>Transmit Data Parameters</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T65</td>
<td>TxD Rise Time</td>
<td>20</td>
<td></td>
<td>ns</td>
<td>*4</td>
</tr>
<tr>
<td>T66</td>
<td>TxD Fall Time</td>
<td>20</td>
<td></td>
<td>ns</td>
<td>*4</td>
</tr>
<tr>
<td>T67</td>
<td>TxC Low to TxD Valid</td>
<td>60</td>
<td></td>
<td>ns</td>
<td>*4, *6</td>
</tr>
<tr>
<td>T68</td>
<td>TxC Low to TxD Transition</td>
<td>60</td>
<td></td>
<td>ns</td>
<td>*2, *4</td>
</tr>
<tr>
<td>T69</td>
<td>TxC High to TxD Transition</td>
<td>60</td>
<td></td>
<td>ns</td>
<td>*2, *4</td>
</tr>
<tr>
<td>T70</td>
<td>TxD Transition—Transition</td>
<td>70</td>
<td></td>
<td></td>
<td>*2, *4</td>
</tr>
<tr>
<td>T71</td>
<td>TxC Low to TxD High</td>
<td>60</td>
<td></td>
<td>ns</td>
<td>*4</td>
</tr>
<tr>
<td><strong>RTS, CTS Parameters</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T72</td>
<td>TxC, Low to RTS Low Time to Activate RTS</td>
<td>60</td>
<td></td>
<td>ns</td>
<td>*5</td>
</tr>
<tr>
<td>T73</td>
<td>CTS Low to TxC Low CTS Setup Time</td>
<td>65</td>
<td></td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>T74</td>
<td>TxC Low to RTS High</td>
<td>60</td>
<td></td>
<td>ns</td>
<td>*5</td>
</tr>
<tr>
<td>T75</td>
<td>TxC Low to CTS Invalid</td>
<td>20</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T75A</td>
<td>CTS High to TxC Low CTS Set-up Time to Stop Transmission</td>
<td>65</td>
<td></td>
<td>ns</td>
<td>*7</td>
</tr>
<tr>
<td><strong>Interframe Spacing Parameters</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T76</td>
<td>Inter Frame Delay</td>
<td></td>
<td></td>
<td>*9</td>
<td></td>
</tr>
<tr>
<td><strong>CRS, CDT, Parameters</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T77</td>
<td>CDT Low to TxC High External Collision Detect Setup Time</td>
<td>45</td>
<td></td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>T78</td>
<td>TxC High to CDT Inactive CDT Hold Time</td>
<td>20</td>
<td></td>
<td></td>
<td>*14</td>
</tr>
<tr>
<td>T79</td>
<td>CDT Low to Jamming Start</td>
<td></td>
<td></td>
<td></td>
<td>*10</td>
</tr>
<tr>
<td>T80</td>
<td>Jamming Period</td>
<td></td>
<td></td>
<td>*11</td>
<td></td>
</tr>
<tr>
<td>T81</td>
<td>CRS Low to TxC High Carrier Sense Setup Time</td>
<td>45</td>
<td></td>
<td>ns</td>
<td>*14</td>
</tr>
<tr>
<td>T82</td>
<td>TxC High to CRS Inactive CRS Hold Time</td>
<td></td>
<td></td>
<td></td>
<td>*14</td>
</tr>
</tbody>
</table>
### High Speed Mode (Continued)

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Parameter</th>
<th>Min</th>
<th>Max</th>
<th>Units</th>
<th>Test Conditions</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>CRS, CDT, Parameters (Continued)</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T83</td>
<td>CRS High to Jaming (Internal Collision Detect)</td>
<td>*12</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T84</td>
<td>CRS High to RxC High. End of Receive Packet</td>
<td>80</td>
<td>ns</td>
<td></td>
<td></td>
</tr>
<tr>
<td>T85</td>
<td>RxC High to CRS High. End of Receive Packet</td>
<td>20</td>
<td>ns</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Receive Clock Parameters</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T86</td>
<td>RxC Rise Time</td>
<td>10</td>
<td>ns</td>
<td>*1</td>
<td></td>
</tr>
<tr>
<td>T87</td>
<td>RxC Fall Time</td>
<td>10</td>
<td>ns</td>
<td>*1</td>
<td></td>
</tr>
<tr>
<td>T88</td>
<td>RxC High Time</td>
<td>80</td>
<td>ns</td>
<td>*1</td>
<td></td>
</tr>
<tr>
<td>T89</td>
<td>RxC Low Time</td>
<td>80</td>
<td>ns</td>
<td>*1</td>
<td></td>
</tr>
<tr>
<td><strong>Received Data Parameters</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T90</td>
<td>RxD Setup Time</td>
<td>45</td>
<td>ns</td>
<td>*1</td>
<td></td>
</tr>
<tr>
<td>T91</td>
<td>RxD Hold Time</td>
<td>45</td>
<td>ns</td>
<td>*1</td>
<td></td>
</tr>
<tr>
<td>T92</td>
<td>RxD Rise Time</td>
<td>20</td>
<td>ns</td>
<td>*1</td>
<td></td>
</tr>
<tr>
<td>T93</td>
<td>RxD Fall Time</td>
<td>20</td>
<td>ns</td>
<td>*1</td>
<td></td>
</tr>
</tbody>
</table>

**NOTES:**

*1 — MOS levels.
*2 — Manchester only.
*3 — Manchester. Needs 50% duty cycle.
*4 — 1 TTL load + 50 pF.
*5 — 1 TTL load + 100 pF.
*6 — NRZ only.
*7 — Abnormal end to transmissions: CTS expires before RTS.
*8 — Normal end to transmission.
*9 — Programmable value.
T76 = NIFS × T60 (ns)  
NIFS - the IFS configuration value.
If NIFS is less than 12, then NIFS is enforced to 12.
*10 — Programmable value:
T79 = NCDF × T60 + (12 to 15) × T60 (ns) (if collision occurs after preamble).
*11 — T80 = 32 × T60  
*12 — Programmable value:
NCSF × TTRC + (12 to 15) × TTRC  
T83 = NCSF × T60 + (12 to 15) × T60  
NCDF - collision detect filter configuration value.
*13 — 2000 ns if configured for Manchester encoding.
*14 — To guarantee recognition on the next clock.
Transmit Data Waveforms
Receive Data Waveforms (NRZ)

Receive Data Waveforms
82560 Host Interface and Memory Controller

- Host Interface to the IBM PC/XT/AT and PS/2™ Buses for 82590, 82592, and 82588 LAN Controllers
- Allows 32-, 16-, and 8-Bit Data Transfers
- Supports Local Static RAM — Up to 32 Kilobytes — Programmable Access Time
- Zero-Wait-State Host Interface Option
- Dual-Channel DMA Controller with Ring Buffer Management Scheme
- Implements Tightly Coupled Interface Mode to 82590/82592 — Automatic Retransmission upon Collision — Transmit Chaining — Back-to-Back Frame Reception — Automatic Buffer Reclamation — Address PROM or Other Peripheral Support — Interfaces Memory-Mapped or I/O-Mapped Adapters
- CHMOS III Technology
- 68-Lead PLCC Package

The 82560 Host Interface and Memory Controller is a companion chip for the Intel 82590 and 82592 Advanced CSMA/CD LAN Controllers as well as the Intel 82588. The 82560 interfaces these controllers to IBM PC/XT/AT and PS/2 systems. It integrates all the interface functions required to implement a nonintelligent, locally buffered LAN solution. The zero wait state and 32-bit data transfers improve the system performance by minimizing the LAN's requirement for Host bandwidth. The 82560's DMA performs data transfers between the LAN controller and the ring-configured local memory. Ring buffer implementation results in highly efficient use of the local memory. The 82560 supports the 82590 and 82592 in their Tightly Coupled Interface (TCI) mode. Without CPU intervention, the 82560 performs transmit chaining, automatic retransmission, back-to-back frame reception, and frame reclamation. The TCI support reduces the software and hardware overhead between frame transfers, and increases the average sustained transfer rate. Combined with the 82590 or 82592, the 82560 provides a high-performance LAN solution for industry standard or custom CSMA/CD networks.
Table 1. 82560 Pin Description

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Pin No.</th>
<th>Type</th>
<th>Name and Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>VCC</td>
<td>5, 23, 57</td>
<td>I</td>
<td>POWER: Connected to +5V power supply.</td>
</tr>
<tr>
<td>VSS</td>
<td>10, 29, 43, 63</td>
<td>I</td>
<td>GROUND: Ground connection.</td>
</tr>
<tr>
<td>CLK</td>
<td>11</td>
<td>I</td>
<td>CLOCK INPUT: This is the system clock input for the 82560. It controls the internal operations of the 82560 and its cycle timing.</td>
</tr>
<tr>
<td>RESET</td>
<td>42</td>
<td>I</td>
<td>RESET: Active high. When active it resets the 82560 to a known passive state.</td>
</tr>
<tr>
<td>D0–D7</td>
<td>40, 41, 44–49</td>
<td>I/O</td>
<td>82560 DATA BUS: Tri-state bus. Used for programmatic access to the 82560 registers. They are also used in the tightly coupled interface (TCI) mode.</td>
</tr>
<tr>
<td>A0–A12</td>
<td>26–39</td>
<td>I</td>
<td>ADDRESS LINES: The 13 address lines select either an 82560 register, or an address in the Local Memory.</td>
</tr>
<tr>
<td>HF0, HF1</td>
<td>25, 24</td>
<td>I</td>
<td>HOST FUNCTION SELECT: These two inputs indicate the type of access requested by the host. These signals are generated by external decode logic and are completely asynchronous to the 82560 system clock. The proper combinations for each access type are shown below:</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>HF1</th>
<th>HF0</th>
<th>Access Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>Idle (No Access Being Requested)</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>Request to Access Shared Portion of Local Memory</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>Request to Access 82560 Registers or the Slave Controller (SCS)</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>Request to Access External PROMs or Latches (GCS)</td>
</tr>
<tr>
<td>RD</td>
<td>17</td>
<td>READ: Active low. This signal is used to indicate the direction of the host transfer. When active, data is being read from the destination (RAM, 560, or GCS port).</td>
</tr>
<tr>
<td>HRDY</td>
<td>20</td>
<td>HOST READY: Active high. This signal from the 82560 is activated when the device on the Local Bus of the LAN adapter is ready to accept data (write cycle) or to output data (read cycle). When no access is being requested by the host (i.e., both HF0 and HF1 are high), this signal is tri-stated in the normal mode, and is driven high in the pipeline mode.</td>
</tr>
<tr>
<td>XCV1</td>
<td>22</td>
<td>TRANSCEIVER ENABLE 1: Enables the transceiver that connects the lower byte of the host and Local data buses. In pipeline mode it enables the transceiver during non-memory host cycles.</td>
</tr>
<tr>
<td>XCV2/PCS</td>
<td>21</td>
<td>TRANSCEIVER ENABLE 2: Enables the transceiver that connects the upper byte of the host and Local data buses. In pipeline mode it enables the latch during memory host cycles.</td>
</tr>
<tr>
<td>INT</td>
<td>50</td>
<td>INTERRUPT OUT: This signal is a logical OR of all enabled interrupt requests. When active it indicates an interrupt request to the CPU. This signal is tristated after reset.</td>
</tr>
<tr>
<td>GCS</td>
<td>59</td>
<td>GENERAL CHIP SELECT: Active low. This signal is asserted by the 82560 when the host requests access to external ROMs or latches.</td>
</tr>
<tr>
<td>Symbol</td>
<td>Pin No.</td>
<td>Type</td>
</tr>
<tr>
<td>--------</td>
<td>--------</td>
<td>------</td>
</tr>
<tr>
<td>BE0</td>
<td>18</td>
<td>I</td>
</tr>
<tr>
<td>BE1</td>
<td>19</td>
<td>I</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Host Bus</th>
<th>Local Bus</th>
<th>BE0</th>
<th>BE1</th>
</tr>
</thead>
<tbody>
<tr>
<td>8-Bit</td>
<td>8-Bit</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>8-Bit</td>
<td>16-Bit</td>
<td>SA0</td>
<td>1</td>
</tr>
<tr>
<td>16-Bit</td>
<td>16-Bit</td>
<td>SA0</td>
<td>SHBE</td>
</tr>
<tr>
<td>16-Bit</td>
<td>32-Bit</td>
<td>SA1</td>
<td>SA1</td>
</tr>
<tr>
<td>32-Bit*</td>
<td>32-Bit</td>
<td>BE0 + BE1</td>
<td>BE2 + BE3</td>
</tr>
</tbody>
</table>

*80386 address pins + stands for logical OR

| DRQ0    | 54     | I    | DMA REQUEST CHANNEL 0: Active high. This is an input from the LAN controller or other peripherals, it requests DMA service. The DMA cycles are run on an on-demand basis, and are prioritized between themselves (two channels) and with the host cycles on an alternating basis. In 82590 Tightly Coupled mode this signal is sampled by the 82560 at the last clock of the Read or Write signal along with DACK1/EOP to determine the state of the transmit or receive process (see Tightly Coupled Interface for more details). |

| DRQ1    | 52     | I    | DMA REQUEST CHANNEL 1: Active high. This is an input from the LAN controller or other peripherals, requesting DMA service. The DMA cycles are run on an on-demand basis, and are prioritized between themselves (two channels) and with the host cycles on an alternating basis. In Tightly Coupled mode this signal is sampled by the 82560 at the last clock of the Read or Write signal (see Tightly Coupled Interface for more details). |

| DACK0/DACK | 55    | O    | Dual Function: This is a dual function pin which serves as DACK0, DMA acknowledge for Channel 0, in all modes except the Tightly Coupled Interface mode. It serves as DACK, DMA acknowledge for both channels, in Tightly Coupled Interface mode. DMA ACKNOWLEDGE0: Active low. Acknowledge DMA requests on channel 0. During special chip select cycles, this signal is controlled by the CPU. DMA ACKNOWLEDGE: Active low. Acknowledge DMA requests on either channel 0, or channel 1. It operates in this mode only when programmed for Tightly Coupled Interface with the 82590 or 82592. This pin can be directly connected to the DACK0/DACK pin of the 82590 or 82592 LAN controllers. |

<p>| DACK1/EOP | 53    | I/O   | Dual Function: This is a dual function, bidirectional pin which serves as DACK1, DMA acknowledge for channel 1, in all modes except 8259X Tightly Coupled Interface mode. It serves as EOP, End of Process indicator, an input, during this Tightly Coupled Interface mode. DMA ACKNOWLEDGE1: Output. Active low. DMA acknowledge for channel 1. During Special Chip Select (SCS) cycles this signal is controlled by the CPU and can be used for accessing the 8259X port 1. The output level is determined by the address of the SCS. |</p>
<table>
<thead>
<tr>
<th>Symbol</th>
<th>Pin No.</th>
<th>Type</th>
<th>Name and Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>DACK1/EOP</td>
<td>53</td>
<td>I/O</td>
<td>END OF PROCESS: Input. In the Tightly Coupled Interface mode, this input, along with the DRQ pin, is sampled by the 82560 at the last clock of the Read or Write signal. The combination of the two pins indicates the status of the Transmit or Receive process. When low, the EOP signal indicates that the active DMA service should be terminated.</td>
</tr>
<tr>
<td>IOWR</td>
<td>56</td>
<td>O</td>
<td>I/O WRITE. Active low. This is the write strobe to the LAN controller or I/O device. It is asserted when data is being written to the LAN controller by either the Host CPU or the 82560 internal DMA. During pipeline read transfers it is the write control signal to the buffer.</td>
</tr>
<tr>
<td>IORD/MWR</td>
<td>58</td>
<td>O</td>
<td>Dual Function: Active low. This signal is used for two different operations. It is a control signal during read cycles from the LAN controller or another I/O device. It is a write strobe during write cycles to the local memory. I/O READ: Active low. It is asserted when data is being read from the LAN controller by either the host CPU or the 82560 internal DMA. During pipeline write transfers it is the read control signal to the buffer. MEMORY WRITE: Active low. It is asserted when data is being written to local memory.</td>
</tr>
<tr>
<td>INTR</td>
<td>51</td>
<td>I</td>
<td>INTERRUPT REQUEST: This signal when active indicates an interrupt request. It is usually connected to the interrupt output of the LAN controller. It may be programmed as active high or low, level or edge triggered, and it can also be masked.</td>
</tr>
<tr>
<td>MA0–12</td>
<td>9–1, 68</td>
<td>O</td>
<td>MEMORY ADDRESS 0–12: These 13 address lines can support two 8-kilobyte or 8-kiloword banks of static memory.</td>
</tr>
<tr>
<td>CSL</td>
<td>62</td>
<td>O</td>
<td>RAM CHIP SELECT (LOW BANK): Active low. This signal is activated during all static-RAM accesses in 8-bit mode, even-byte accesses in 16-bit mode, and even-word accesses in 32-bit mode.</td>
</tr>
<tr>
<td>CSH</td>
<td>61</td>
<td>O</td>
<td>RAM CHIP SELECT (HIGH BANK): Active low. This signal is activated during odd-byte accesses in 16-bit mode or odd-word accesses in 32-bit mode.</td>
</tr>
<tr>
<td>MOE</td>
<td>60</td>
<td>O</td>
<td>MEMORY OUTPUT ENABLE: Active low. This signal is used to enable the memory array’s output buffers during memory read cycles.</td>
</tr>
<tr>
<td>GPI</td>
<td>16</td>
<td>I</td>
<td>GENERAL PURPOSE: Input. This is a general purpose input pin, its state may be read by the CPU.</td>
</tr>
<tr>
<td>CS</td>
<td>12</td>
<td>O</td>
<td>CHIP SELECT: Active low. This pin is normally connected to the Chip Select input of the LAN controller or other peripherals. It is activated during non-DMA accesses to the LAN controller. The CPU activates this signal when it accesses addresses 0, 1, 2, or 3 in the Special Chip Select address space of the 82560.</td>
</tr>
<tr>
<td>RSV1, RSV2</td>
<td>13, 14</td>
<td>I</td>
<td>These pins are reserved and should be tied to VCC.</td>
</tr>
</tbody>
</table>
Figure 2. 82560 PLCC Pinout

Figure 3. Nonintelligent, Buffered Adapter Architecture
FUNCTIONAL OVERVIEW

The 82560 is a dual-port memory controller using interrupt logic and DMA to implement a nonintelligent, buffered LAN adapter for the IBM PC/XT/AT bus. This type of adapter uses on-board memory as a buffer to store frames during transmission and reception. It also uses on-board DMA to transfer data between its local memory and the LAN controller. A block diagram of the buffered, nonintelligent LAN adapter is shown in Figure 3. The architecture is termed nonintelligent because it does not use an on-board CPU to process the transmit or receive frames. The host CPU processes the frames and programs the DMA and LAN controllers. The host interface logic, arbitration logic, and the bus transceivers connect the host bus to the adapter’s local bus. They also control all host accesses to the local bus. The local memory is shared by the host and the LAN controller. It stores information that the host wishes to transfer to the LAN controller, and information received by the LAN controller which should be read by the host. The memory can be shared in two ways—mapping into the host memory space, or mapping into the host I/O space. The DMA controller transfers data between the local buffer memory and the LAN controller. The host CPU may use either string move instructions or a system DMA channel to move data into the buffer memory. The host also accesses the LAN controller registers, the DMA controller registers the boot ROM, and the address ROM through the local bus.

The 82560 integrates the host interface, arbitration logic, memory control logic, interrupt logic, and DMA into one component. It replaces 20–30 MSI and SSI components (see Figure 1). It also provides a Tightly Coupled Interface to the 82588, 82590, and 82592 LAN controllers, and an efficient buffer management scheme, which allows the 82588/82560, 82590/82560, and 82592/82560 combinations to handle time-critical processes such as retransmission, buffer reclamation, and continuous back-to-back frame reception without host CPU intervention. This improves the overall data throughput in the network; and, consequently, system performance. The following discussion describes the 82560 interface to the PC/XT/AT bus, its support of locally buffered memory, and the operation of DMA and the Tightly Coupled Interface (including the buffer management scheme).

HOST INTERFACE

The host interface port connects the 82560 to the PC-bus through external decode logic and bus transceivers. The external decode logic generates the HF0 and HF1 signals indicating the kind of access the host desires. When the request is detected by the 82560 (non-pipeline mode) it deasserts the HRDY signal, thereby suspending the host cycle. HRDY is reactivated when the local device being accessed by the host is ready to accept (Write cycle) or output (Read cycle) data. HRDY reactivation time is programmable as mentioned in the register section; it is described in detail in the 82560 Reference Manual. The request undergoes arbitration, and, if granted, the 82560 activates the XCV1 and XCV2 signals. The XCV signals control the transceiver(s) which interfaces the host data bus to the local data bus. By using one or both transceiver control signals the 82560 can support an 8-, 16-, or 32-bit-wide bus. Once the arbiter grants the host access, the 82560 begins the local bus cycle by generating the appropriate address and control signals.

The host CPU can access the internal registers of the 82560, the local memory controlled by the 82560, or other devices—such as Boot ROM or external Latch—that share the same bus as the 82560. Table 2 lists the various access types that can be requested by the host.

<table>
<thead>
<tr>
<th>Access Type</th>
<th>HF1</th>
<th>HF0</th>
<th>Address</th>
<th>Cycle Status Indications</th>
</tr>
</thead>
<tbody>
<tr>
<td>82560 Registers</td>
<td>0</td>
<td>1</td>
<td>Between 8h and 3Fh</td>
<td>HRDY</td>
</tr>
<tr>
<td>Local Memory Access</td>
<td>1</td>
<td>0</td>
<td>User Defined</td>
<td>HRDY, Memory Control Signals, XCV Signals</td>
</tr>
<tr>
<td>GCS Access (Boot ROM)</td>
<td>0</td>
<td>0</td>
<td>User Defined</td>
<td>HRDY, GCS, IOWR, IORD/MWR</td>
</tr>
<tr>
<td>(General Chip Select)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Special Chip Select</td>
<td>0</td>
<td>1</td>
<td>Less Than 8h</td>
<td>HRDY, DACK Lines, CS, IOWR, IORD/MWR</td>
</tr>
</tbody>
</table>
The 82560 provides eight semaphore ports to resolve contention in a shared resource system. Only the most significant bit of these ports is used. The CPU writes all 0's to the port to clear it. When the port is read, its current value is reported and the most significant bit becomes a 1 at the end of the cycle. The 82560 also supports devices on the local bus other than memory and the LAN controller. These devices can be accessed in two ways: by using the General Chip Select (GCS) signal, or by using the Special Chip Select (SCS) addresses in the 82560 register space. The first method typically supports EPROMs, external latches, and similar devices. The second method is used for accessing the registers of controllers which use the 82560 DMA channels; e.g., the 82590, 82592, or 82588. Each address in the SCS port provides a unique combination of the DACK0-, DACK1-, and CS-pin output states. The CPU activates the chip select of the device being accessed by asserting or deasserting the appropriate signals.

The host CPU can access the 82560 registers and other devices on the local bus at any time. However, local memory can only be accessed by the host after the 82560 memory control registers are initialized. The host accesses local memory in two ways: Page Access or Sequential (I/O mapped or pipeline) Access. After reset, the memory access is I/O-mapped mode but host access to local memory is disabled. The 82560 must be configured for the appropriate memory access mode before local memory can be accessed by the host.

The 82560 memory control logic provides the signals required to interface to static memory. The 82560 can address up to 32 kB of local memory. Each memory address can refer to a byte, a word, or a double word of local memory. Thus the 82560 with its two memory chip selects (low to high bank) and its MOE and MWR outputs, can support 8-, 16-, or 32-bit-wide local buses.

In Page Access mode the local memory is mapped into the host memory space. In this mode the host can directly access local memory through a fixed size window which can be moved around in local memory space. This window is referred to as a "page". Figure 5 shows the paging scheme. The page size can vary from 1 kilobyte to 8 kilowords, and can be located anywhere in local memory. The exact location of the page in the local memory is defined by a page register. By reprogramming the page register the user can relocate the page in local memory.

In I/O-mapped Access mode the memory is mapped into the host I/O address space. Data is transferred between host and local memory using host DMA or string I/O instructions. The 82560 can be programmed to support memory accesses through a single I/O port. The I/O port is defined by an address programmed into an 82560 register. The 82560 maintains the current address, which is updated each time a memory cycle is run. The host does not directly access the local memory. It outputs the I/O address onto the A0−A12 address lines, with the HF lines indicating a memory access. If the I/O address matches the address programmed into the 82560, then the 82560 executes the local memory cycle by outputting the current address onto the memory address lines MA0−12.

The 82560 can be configured to interface with the host in a pipeline mode. In this mode, transparent or edge-triggered latches are needed to isolate the host and local bus during memory cycles. Data is written to the latch (from the host bus) and copied (from the latch) to the local memory. In the host read cycles, data is copied from the latch to the host bus. In anticipation of the next host memory request (sequential), the 82560 then copies the next byte or word from local memory to the latch. Thus the host CPU can operate with 0 wait states by reading from and writing to the latch.

**ARBITER**

All requests for access to devices on the local bus, whether by the host or by the 82560 DMA, undergo arbitration. The host requests are indicated on the HF lines; the DMA requests are indicated on the DRQ lines. Figure 4 shows the basic arbitration cycle of the 82560. Arbitration for the local bus is pipelined. It can take place at any time when the 82560 is idle, or one clock before the end of the current local bus cycle. All requests are sampled on the falling edge of the 82560 clock. Arbitration is completed within one clock cycle. The resultant local bus cycle is started on the falling edge of the next clock. If more than one request is active, arbitration is resolved on an alternating priority basis.

The 82560 deactivates its HRDY line when a host request is detected; the request is synchronized and then arbitrated. If the request is granted, the appropriate local bus cycle begins. After a programmable number of clock cycles HRDY will be reactivated, and the handshake with the host will be complete. DMA requests are synchronized and acknowledged once DMA has been granted access to the local bus. The acknowledge lines are kept inactive until the DMA is granted access to the local bus.
This Diagram Assumes:
The default priority bit to be 1 (bit 7 of the master mode)
I/O or MEM wait states to be 1 clock (Tw = 1)
Non-Pipeline Mode
In the case of host read cycles (in any mode) or host write cycles (in pipeline mode only), “Twh” cycles
will be asserted in addition to “Tw”, between “T1” and “T2” of the host cycles.

**Figure 4. 82560 Arbitration Cycles**

**Figure 5. Page Mechanism**

**DMA MACHINE**

The 82560 provides two DMA channels. Each channel can access 16 kb of memory address, and has request and acknowledge lines and address registers. The DMA normally operates in the Demand mode, and becomes active in response to a DMA request being granted. The requests come in on the DRQ lines and, if granted, are acknowledged by the DACK lines becoming active. Each channel has a control register that includes an enable bit, a direction bit, and output enable bits (CS, DACK0, and DACK1 are active low signals that can be enabled/disabled during DMA cycles). Each channel also has a base, current, stop, lower-limit, and upper-limit register. The current address register (CAR) is incremented after every DMA transfer except when in double host bus mode. The lower-limit register points to the beginning of the ring buffer; the upper-limit register points to the end of the ring buffer. The 82560 performs the wraparound (lower limit to CAR), each time the CAR equals the upper limit. When the contents of the CAR equal those of the stop register, DMA transfer stops and the 82560 generates an interrupt to the CPU. When the double host bus mode is invoked, the DMA machine will alternately activate low and high banks of memory and will increment the address after each high-bank transfer.
LOOSELY COUPLED MODE

The 82560 performs flyby DMA transfers (read from slave and write to memory or vice versa). The operation continues until the current address register equals the stop register or until the DRQ is removed. When the stop register is reached, the 82560 generates an interrupt.

82590 TIGHTLY COUPLED MODE

The Tightly Coupled Interface is a hardware interface between the 82560 and the 82590. This interface allows transmission and reception events to be processed without CPU intervention. It allows the implementation of the time-critical CSMA/CD processes: automatic retransmission, buffer reclamation, and continuous frame reception and transmission. The basic interface is a two-signal DMA handshake between the 82560 and the 82590; this occurs over the DRQ and EOP pins. The 82590 provides the status of the current transmit or receive process, or requests another DMA cycle at the end of each DMA cycle. When configured for the Tightly Coupled Interface, the 82560 and the 82590 use a specific interrupt scheme to minimize CPU overhead and to improve data throughput. The 82590 will not generate interrupts when events occur that can be handled by the 82560 without CPU intervention. Figure 5 illustrates the Tightly Coupled Interface mechanism. Table 3 lists the various combination of the DRQ and EOP signals, and the events they represent.

If both DRQ and EOP are sampled high, the Current Address Register of the channel is incremented and another DMA cycle begins. If a frame is transmitted or received without errors, both DRQ and EOP are low at the end of the DMA cycle and the 82560 will generate an interrupt. If DRQ is high and EOP is low, a collision occurred during transmission, or an error occurred during reception. In this case the Current Address Register will be reloaded with the value in the Base Address Register; and, once again, it will point to the beginning of the frame structure in memory.

The DACK1/CS1.EOP pin of the 82590 is multiplexed and requires external logic to derive the EOP and CS1 signals (see 82590 data sheet). Because the 82560 integrates this logic, its DACK1/EOP pin can be connected (with a pullup resistor) directly to the DACK1/CS1/EOP pin of the 82590, and its DACK0/DACK pin can be connected directly to the DACK pin of the 82590. For more details, see the 8259X Users Manual.

Table 3. DMA Handshake Encoding

<table>
<thead>
<tr>
<th>DRQ</th>
<th>EOP</th>
<th>Event Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>Operation Done</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>Idle</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>Retry Request</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>New DMA Transfer Request</td>
</tr>
</tbody>
</table>

Figure 6. 82560, 82590 DMA Handshake
Transmit

The 82590 can transmit consecutive frames without using the CPU to issue the Transmit command each time. This improves data throughput during transmission and eliminates CPU overhead. The CPU can place multiple transmit frames in memory, with each frame separated from the next by a Transmit Command byte. (For further information see 82590 and 82592 user manuals.) The 82560 supports transmit chaining. It also supports automatic retransmit on collision (provided that the maximum number of collisions is not reached). In this case the current address register is reloaded with the value of the base address register, and the DMA transfer is resumed without CPU involvement. If the maximum number of collisions has been reached, or if transmit failed for any other reason, the 82560 will need CPU intervention. Thus it will generate an interrupt to the CPU. At the end of transmission of each frame, the 82560 updates the status byte (indicating the number of collisions) in the memory.

Figure 7. Example of a 4-kB Transmit Ring Buffer
Receive

Immediately after a channel is enabled for receive, the 82560 will write FFh into the first two bytes of the frame (pointed to by the base register). The current address register is loaded with the contents of the base register and is incremented twice (past the two reserved bytes). If an error occurs during reception, and the save bad frame bit is 0, the CAR is reloaded with the content of the BAR and incremented past the two reserved bytes; however, if the save bad frame bit is set, the CAR is incremented for the next frame and the 82560 generates an interrupt to the CPU. If no error occurs, the last two bytes received (which are always stored in 82560 internal registers) are copied back to the first two bytes of the frame. These are the byte counts. If the 82590 generates an interrupt on each frame reception the 82560 will relay that interrupt to the CPU. At this time the value of the CAR will be copied into BAR, FF will be written into the next two bytes, and CAR will be incremented as before to point to the new frame reception area.

Figure 8. Example of a 4-kB Receive Ring Buffer
## Table 4. Address Map

<table>
<thead>
<tr>
<th>Address</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>3Fh</td>
<td>RESERVED</td>
</tr>
<tr>
<td>2Fh</td>
<td>HOST MODE REGISTER</td>
</tr>
<tr>
<td>2Eh</td>
<td>BASE ADDRESS</td>
</tr>
<tr>
<td>21h</td>
<td>CURRENT ADDRESS</td>
</tr>
<tr>
<td>20h</td>
<td>BASE ADDRESS REGISTER 0</td>
</tr>
<tr>
<td>22h</td>
<td>CURRENT ADDRESS REGISTER 0</td>
</tr>
<tr>
<td>32h</td>
<td>BASE ADDRESS REGISTER 1</td>
</tr>
<tr>
<td>31h</td>
<td>CURRENT ADDRESS REGISTER 1</td>
</tr>
<tr>
<td>30h</td>
<td>BASE ADDRESS REGISTER 1</td>
</tr>
<tr>
<td>3Ah</td>
<td>UPPER LIMIT REGISTER 0</td>
</tr>
<tr>
<td>28h</td>
<td>LOWER LIMIT REGISTER 0</td>
</tr>
<tr>
<td>24h</td>
<td>DMA CONTROL REGISTER 0</td>
</tr>
<tr>
<td>23h</td>
<td>DMA CONTROL REGISTER 1</td>
</tr>
<tr>
<td>1Fh</td>
<td>DMA MODE REGISTER</td>
</tr>
<tr>
<td>1Eh</td>
<td>HOST ADDRESS REGISTER H</td>
</tr>
<tr>
<td>1Ch</td>
<td>SELECT REGISTER</td>
</tr>
<tr>
<td>1Ah</td>
<td>RESERVED</td>
</tr>
<tr>
<td>19h</td>
<td>INT MASK REGISTER</td>
</tr>
<tr>
<td>18h</td>
<td>INT CONTROL/STATUS REGISTER</td>
</tr>
<tr>
<td>16h</td>
<td>82588 STATUS 2 REGISTER</td>
</tr>
<tr>
<td>15h</td>
<td>82588 STATUS 1 REGISTER</td>
</tr>
<tr>
<td>14h</td>
<td>RESERVED</td>
</tr>
<tr>
<td>13h</td>
<td>CONTROL REGISTER</td>
</tr>
<tr>
<td>12h</td>
<td>IDENTIFICATION REGISTER</td>
</tr>
<tr>
<td>11h</td>
<td>MASTER MODE REGISTER</td>
</tr>
<tr>
<td>10h</td>
<td>SEMAPHORES</td>
</tr>
<tr>
<td>0Fh</td>
<td>SCS PORTS§</td>
</tr>
</tbody>
</table>

### SCS PORTS§

<table>
<thead>
<tr>
<th>Bit 1</th>
<th>Bit 0</th>
<th>DACK1</th>
<th>DACK0</th>
<th>Register</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
<td>07h</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td></td>
<td>06h</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td></td>
<td>05h</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td></td>
<td>04h</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td></td>
<td>03h</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td></td>
<td>02h</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td></td>
<td>01h</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
<td>00h</td>
</tr>
</tbody>
</table>

### NOTE:

When writing to 3-byte registers, the most significant byte (higher address) should be written last. The value written into the most significant bytes should be 0. The third bytes are reserved for possible future use.
The 82560 supports a Tightly Coupled Interface (TCI) with the 82588. This interface allows transmit and receive events to be processed without CPU intervention. It allows the combination of the 82588 and 82560 to implement time-critical, CSMA/CD events: automatic retransmission, buffer reclamation, and continuous frame reception (see 82588 Reference Manual). When configured for the 82588 TCI mode, the 82560 uses the 82588 INT pin to determine if an event has occurred. The 82560 then reads the 82588 status register(s) to determine the cause of the interrupt. If the interrupt is due to a collision during transmission, a good frame reception, or errors during frame reception then the 82560 will update its DMA address registers and issue the 82588 the commands necessary for minimizing CPU intervention. The 82560 will regenerate all 82588 interrupts except those generated when a collision occurs during transmission (with the maximum retry count not exceeded). Because transmit and receive interrupts are time-critical processes the 82560 automatically acknowledges such interrupts to reduce dependency on the CPU. It will regenerate the interrupt on its INTOUT pin unless the interrupt is due to a transmit collision.

If the 82588 issues an interrupt due to a collision during transmission, and the maximum retry count has not been exceeded, the 82560 will automatically reload the Current Address Register with the value in the Base Address Register, acknowledge the interrupt, and issue a retransmit command to the 82588. If the interrupt is due to the reception of a good frame, the 82560 will update its Base and Current Address Registers and prepare for a new incoming frame. If the interrupt is due to a receive frame error, the 82560 will reclaim the buffer by resetting the Current Address Register to the beginning of the frame buffer.

If the 82588 is unable to transmit due to having exceeded the maximum retry count or a Lost-CTS condition or a Lost-CRS condition, an interrupt is generated; the 82560 will not update its DMA address registers. It will, however, acknowledge the 82588 interrupt and regenerate the interrupt on its INOUT pin.

The registers can be accessed by the host CPU. The RD signal indicates the direction of data transfer between the 82560 and the CPU. The actual data transfer takes place over the 82560's 8-bit data bus lines (D7–D0). The address of the register being accessed is taken from the address lines A5–A0.

Since the 82560's data bus is 8-bits wide, all access to its registers is on a byte basis. If a register is longer than 1 byte, each byte has to be accessed individually through its unique address in the 82560 register space.

On power-up or reset, the 82560 registers are set to a default configuration. The user must initialize the 82560 for the proper system configuration.

The SCS ports occupy eight addresses in the 82560 register space. The SCS ports should not be thought of as registers. They are merely addresses in the register space which, when addressed, activate a combination of the DACK0, DACK1 or CS pins. The particular combination of these pins signal levels depends on the SCS port address being accessed. The semaphore ports allow resource sharing in a dual processor (intelligent adapter) environment. Each port can be used as a semaphore to implement mutual exclusion.

**CONFIGURATION REGISTERS**

By programming these registers, the 82560 can be tailored to support different PCs, slaves and memories. The memory access mode (I/O or memory mapped) and the type of DMA support (loosely or tightly coupled) can also be programmed.
MASTER MODE REGISTER (10h)

- D7: Host bus interface
- D5: 00 equal data bus width
- D5: 01 double data bus width
- D5: 01 double data bus width with special receive
- D5: 10 reserved
- D4: HRDY delay
- D4: 00 no delay
- D4: 01 0.5 clock delay
- D4: 10 1.5 clock delay
- D4: 11 2.5 clock delay
- Reserved
- D3: Base/Current select (1/0)
- D2: Reserved
- D1: Host/DMA idle priority (1 or 0)
- D0: Reserved

* IN SOME VERSIONS OF 82560, THIS MODE IS NOT TESTED.

CONTROL REGISTER (12h)

- D7: I/O access delay (0 to 3)
- D6: Early/Late write option (1/0)
- D5: Memory access delay (0 to 3)
- D4: Reserved

HOST MODE REGISTER (2Fh)

- D7: Enable/Disable pipeline mode (1/0)
- D6: Pipeline direction read/write (1/0)
- D5: Reserved
- D4: General purpose input
INTERRUPT MASK REGISTER (17h)

- \(D_7\) \(D_6\) \(D_5\) \(D_4\) \(D_3\) \(D_2\) \(D_1\) \(D_0\)
- Low byte of the host-selected address (I/O-mapped memory access)

SELECT REGISTER, HIGH BYTE (1Bh)

- \(D_7\) \(D_6\) \(D_5\) \(D_4\) \(D_3\) \(D_2\) \(D_1\) \(D_0\)
- High bits of the host-selected address
- HRDY delay reference source
- Memory or I/O mapped (1/0)
- Enable/Disable memory access (1/0)

DMA MODE REGISTER (1Fh)

- \(D_7\) \(D_6\) \(D_5\) \(D_4\) \(D_3\) \(D_2\) \(D_1\) \(D_0\)
- Reserved
- Save/Discard bad frame (1/0)
- DMA Mode
  - 00 loosely coupled (regular)
  - 10 8259X Tightly Coupled Interfaced
  - 01 82588 TCI
  - 11 reserved
Interrupt Mask Register (17h)

<table>
<thead>
<tr>
<th>D7</th>
<th>D6</th>
<th>D5</th>
<th>D4</th>
<th>D3</th>
<th>D2</th>
<th>D1</th>
<th>D0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Level/Edge sensitive (1/0)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>High/Low true assert (1/0)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>00 no change*</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>01 enable slave interrupt</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10 disable slave interrupt</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11 reserved</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Reserved</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Disable/Enable Tx chain (1/0)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Enable/Disable interrupt tri-state (1/0)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

*Whenever one of these bits is "1" while writing to this register, other bits are not affected.

Host Address Registers

Contain the initial memory address when the host accesses memory in I/O mapped or pipeline mode.

Identification/Software Reset Register

Writing to this address will reset the chip. Reading from it will provide the user with 82560 stepping information.

Master Mode Register (10h) DMA Control Register* (23h or 33h)

<table>
<thead>
<tr>
<th>D7</th>
<th>D6</th>
<th>D5</th>
<th>D4</th>
<th>D3</th>
<th>D2</th>
<th>D1</th>
<th>D0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Disable/Enable DACK0 during DMA cycles (1/0)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Disable/Enable, DACK1/EOP during DMA cycles (1/0)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Disable CS during DMA cycles (1/0)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Direction bit: memory read/write (1/0)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Enable/Disable DMA channel (1/0)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Receive/Execution channel (1/0)†</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Reserved</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**DMA Channel Function**

<table>
<thead>
<tr>
<th>D5</th>
<th>D3</th>
<th>DMA Channel Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>Transmit</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>Dump (588 or 590/592)</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>Reserved (Do Not Use)</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>Receive</td>
</tr>
</tbody>
</table>

*Each DMA channel has its own control register.
†The following table shows the encoding of bits 3 and 5:
INTERRUPTS

In the non-tightly coupled mode, the 82560 will generate an interrupt when the Current Address register equals the Stop register or when the interrupt input pin is active.

In the tightly coupled modes, the conditions for generating Stop register interrupts are the same however, the 82560 will generate 82590 interrupts only if its source was one of the following.
- Transmission of every frame, or last frame, in the chain is completed (programmable).
- Transmission failed because of a collision, and the maximum number of Transmit retries is reached.
- Transmission failed for a reason other than collision; e.g., lost CRS/CTS.
- Reception failed, and the Save Bad Frame bit is set.
- Reception completed.

The interrupt control register is read by the interrupt routines to determine the exact source of the interrupt.

INTERRUPT STATUS READ REGISTER (16h)

![Diagram of Interrupt Status Read Register]

NOTE:
The interrupt control register is written to acknowledge and reset the interrupt.

INTERRUPT CONTROL WRITE REGISTER (16h)

![Diagram of Interrupt Control Write Register]
SYSTEM INTERACTION

A typical 82560 system interaction is described below.
1. The CPU configures the 82560 by writing to configuration registers.
2. The CPU accesses the local memory (through the 82560) and prepares a block of transmit frames.
3. The CPU writes the proper addresses into the 82560’s DMA address registers, (base, current, lower limit, upper limit and stop).
4. The CPU writes to the 82560’s DMA control registers to configure and enable the channels.
5. The CPU issues a transmit command to the 82590.
6. The 82560 responds to the 82590’s OMA request by transferring data from memory to the 82590.
7. Upon completion of transmission, the 82560 sends an interrupt to the CPU.
8. The CPU reads the 82560 interrupt control register to find the source of the interrupt.
9. The CPU issues a command to the 82590 to clear its interrupt. (If the source of the interrupt was the 82590.)
10. The CPU acknowledges the 82560 interrupt by writing a “1” into the corresponding interrupt control register bit(s).

APPLICATIONS

Figure 9 shows a buffered, nonintelligent StarLAN adapter for the IBM PC bus (using the 82560 and the 82590). Figure 10 shows a buffered, nonintelligent Ethernet adapter for the IBM PC bus (using the 82560, 82592, 82C501 and the 82502).

82560 MACHINE CYCLE

The 82560 machine cycle can be broken down into three basic cycles: Idle (T_{IDLE}), Arbitration (T_A) and Transfer (T_{TSF}). The machine cycle begins when a request (HF or DRQ) becomes active and the 82560 is in the idle state (T_{IDLE}). The requests are synchronized and then undergo arbitration (T_A). Once arbitration is completed, the transfer cycle (T_{TSF}) begins.

Synchronization (T_S) is completed on the falling edge of the clock. If the previous cycle was non-idle, arbitration begins and is completed within one clock period (by the next falling edge of the clock).

The Transfer cycle consists of the following sequential states: the first transfer state (T_1), memory or I/O wait states (T_W), and the second transfer state (T_2). There may be another transfer state, T_{wh} (wait host), during host read or pipeline cycles. When no requests are pending, and the 82560 is not in the transfer or arbitration cycle, it is said to be in the idle state (T_{IDLE}). If the previous cycle was non-idle, the arbitration period (T_A and T_2 of the previous cycle will be done in parallel. (See Figure 4.)

T_W is the programmable portion of the transfer cycle. It can be zero to three clocks long depending on the programmed memory or I/O access delays. If the programmed delay is zero, then there will be no T_W; the first state of the transfer cycle will be T_1. During T_2 the transfer cycle is completed unless the cycle is a host read cycle. In that case the cycle will be extended by inserting T_{wh}. The 82560 will remain in T_{wh} until the HF lines are deasserted. Once HF lines are deasserted, T_2 will begin and one clock period later the bus cycle is complete.
Figure 9. 560 High-Integration Adapter (560 and 560)
ABSOLUTE MAXIMUM RATINGS*

Case Temperature (TC) under Bias ...................... 0°C to +85°C
Storage Temperature .................-65°C to +150°C
Voltage on any Pin with Respect to Ground .............. -0.5V to VCC + 0.5V

*Notice: Stresses above those listed under “Absolute Maximum Ratings” may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions above those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

NOTICE: Specifications contained within the following tables are subject to change.

D.C. CHARACTERISTICS TC = 0°C to +85°C, VCC = +5V ±10%
CLK pin has MOS levels (see VMIL, VMIH) All other signals have TTL levels (see VIL, VIH, VOL, VOH).

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Parameter</th>
<th>Min</th>
<th>Max</th>
<th>Units</th>
<th>Test Conditions</th>
</tr>
</thead>
<tbody>
<tr>
<td>VIL</td>
<td>Input Low Voltage (TTL)</td>
<td>-0.5</td>
<td>+0.8</td>
<td>V</td>
<td></td>
</tr>
<tr>
<td>VIH</td>
<td>Input High Voltage (TTL)</td>
<td>2.0</td>
<td>VCC + 0.5</td>
<td>V</td>
<td></td>
</tr>
<tr>
<td>VOL</td>
<td>Output Low Voltage (TTL)</td>
<td>0.45</td>
<td>V</td>
<td></td>
<td></td>
</tr>
<tr>
<td>VOH</td>
<td>Output High Voltage (TTL)</td>
<td>2.4</td>
<td>VCC</td>
<td>V</td>
<td>IOH = -400 μA</td>
</tr>
<tr>
<td>VMIL</td>
<td>Input Low Voltage (MOS)</td>
<td>-0.5</td>
<td>0.6</td>
<td>V</td>
<td></td>
</tr>
<tr>
<td>VMIH</td>
<td>Input High Voltage (MOS)</td>
<td>VCC - 0.6</td>
<td>VCC + 0.5</td>
<td>V</td>
<td></td>
</tr>
<tr>
<td>IIL</td>
<td>Input Leakage Current</td>
<td>±10</td>
<td>μA</td>
<td>0 = VIN = VCC - 0.45</td>
<td></td>
</tr>
<tr>
<td>IIO</td>
<td>I/O Leakage Current</td>
<td>±10</td>
<td>μA</td>
<td>0.45 = VOUT = VCC - 0.45</td>
<td></td>
</tr>
<tr>
<td>CIN</td>
<td>Capacitance of Input Buffer</td>
<td>10</td>
<td>pF</td>
<td>FC = 1 MHz</td>
<td></td>
</tr>
<tr>
<td>COUT</td>
<td>Capacitance of Input/Output Buffer</td>
<td>20</td>
<td>pF</td>
<td>FC = 1 MHz</td>
<td></td>
</tr>
<tr>
<td>ICC</td>
<td>Power Supply Current</td>
<td>50</td>
<td>mA</td>
<td>10 MHz</td>
<td></td>
</tr>
</tbody>
</table>

A.C. CHARACTERISTICS C_L on all outputs is 50 pF. The user should add 0.2 ns/pF up to 100 pF

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Parameter</th>
<th>Min</th>
<th>Max</th>
<th>Test Conditions</th>
</tr>
</thead>
<tbody>
<tr>
<td>SYSTEM CLOCK INPUT PARAMETERS</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T1</td>
<td>CLK Cycle Period</td>
<td>100</td>
<td></td>
<td>(Note 1)</td>
</tr>
<tr>
<td>T2</td>
<td>CLK Low Time</td>
<td>45</td>
<td></td>
<td>(Note 1)</td>
</tr>
<tr>
<td>T3</td>
<td>CLK High Time</td>
<td>45</td>
<td></td>
<td>(Note 1)</td>
</tr>
<tr>
<td>T4</td>
<td>CLK Rise Time</td>
<td>5</td>
<td></td>
<td>(Note 2)</td>
</tr>
<tr>
<td>T5</td>
<td>CLK Fall Time</td>
<td>5</td>
<td></td>
<td>(Note 3)</td>
</tr>
</tbody>
</table>

HOST ACCESS CYCLE—NON PIPELINE MODE PARAMETERS

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Parameter</th>
<th>Min</th>
<th>Max</th>
<th>Test Conditions</th>
</tr>
</thead>
<tbody>
<tr>
<td>T6</td>
<td>HF or DREQ Setup Time</td>
<td>10</td>
<td></td>
<td></td>
</tr>
<tr>
<td>T7</td>
<td>HF Active Time (Low)</td>
<td>2*T1 + 10</td>
<td></td>
<td>(Note 5)</td>
</tr>
<tr>
<td>T8</td>
<td>HF Inactive Time (High)</td>
<td>T1 + 10</td>
<td></td>
<td></td>
</tr>
<tr>
<td>T9</td>
<td>HF to HRDY Low</td>
<td>50</td>
<td></td>
<td></td>
</tr>
<tr>
<td>T10</td>
<td>HF Active to HDRY High</td>
<td>2*T1 + 50</td>
<td></td>
<td>(Note 4) (Note 9)</td>
</tr>
<tr>
<td>T11</td>
<td>HRDY High to HF Inactive</td>
<td>0</td>
<td></td>
<td>(Note 5)</td>
</tr>
</tbody>
</table>
A.C. CHARACTERISTICS

C\textsubscript{i} on all outputs is 50 pF. The user should add 0.2 ns/pF up to 100 pF (Continued)

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Parameter</th>
<th>Min</th>
<th>Max</th>
<th>Test Conditions</th>
</tr>
</thead>
<tbody>
<tr>
<td>T12</td>
<td>HF Inactive to HRDY Float</td>
<td>75</td>
<td></td>
<td></td>
</tr>
<tr>
<td>T13</td>
<td>HF Active to XCVR Lines Low</td>
<td>T1 + T2</td>
<td>2*T1 + T2 + 75</td>
<td>(Note 6)</td>
</tr>
<tr>
<td>T14</td>
<td>HF Inactive to XCVR Lines High</td>
<td>(Note 7)</td>
<td>75</td>
<td></td>
</tr>
<tr>
<td>T15</td>
<td>HF Active to RD Low</td>
<td>T1 + T2 + 10</td>
<td></td>
<td></td>
</tr>
<tr>
<td>T16</td>
<td>RD Hold after HF Inactive</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>T17</td>
<td>HF Active to Input Add. Valid</td>
<td>-20</td>
<td></td>
<td></td>
</tr>
<tr>
<td>T18</td>
<td>Address Hold after HF Inactive</td>
<td>0</td>
<td></td>
<td>(Note 8)</td>
</tr>
<tr>
<td>T19</td>
<td>HF Active to 82560 Data Valid</td>
<td>3*T1 + 80</td>
<td></td>
<td>(Note 9)</td>
</tr>
<tr>
<td>T20</td>
<td>Data Hold after HF Inactive</td>
<td>T1 + T2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>T21</td>
<td>HF Active to 82560 Add Valid (MAN)</td>
<td>2*T1 + T2 + 75</td>
<td>(Note 9)</td>
<td></td>
</tr>
<tr>
<td>T22</td>
<td>Add Valid or Chip Select Active Time</td>
<td>2*T1</td>
<td>(Note 10)</td>
<td></td>
</tr>
<tr>
<td>T23</td>
<td>HF Active to CS Active</td>
<td>2*T1 + T2 + 50</td>
<td>(Note 9)*</td>
<td></td>
</tr>
<tr>
<td>T24</td>
<td>CS Enveloping Controls</td>
<td>20</td>
<td></td>
<td></td>
</tr>
<tr>
<td>T25</td>
<td>Control Active Time</td>
<td>(Note 11)</td>
<td>(Note 11)</td>
<td></td>
</tr>
<tr>
<td>T26</td>
<td>HF to Data Valid</td>
<td>3*T1-30</td>
<td></td>
<td></td>
</tr>
<tr>
<td>T27</td>
<td>Data Hold after HRDY High</td>
<td>(Note 12)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

HOST ACCESS CYCLE—PIPELINE MODE PARAMETERS

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Parameter</th>
<th>Min</th>
<th>Max</th>
<th>Test Conditions</th>
</tr>
</thead>
<tbody>
<tr>
<td>T28</td>
<td>HF Active Time</td>
<td>T1 + 10</td>
<td>(Note 13)</td>
<td>(Note 9)</td>
</tr>
<tr>
<td>T29</td>
<td>HF Active to Port CS Active</td>
<td>2*T1 + 75</td>
<td></td>
<td>(Note 6)</td>
</tr>
<tr>
<td>T30</td>
<td>HF Inactive to HRDY Low</td>
<td>75</td>
<td></td>
<td></td>
</tr>
<tr>
<td>T31</td>
<td>HRDY Low to HRDY High</td>
<td>(Note 14)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>T32</td>
<td>Port CS Active Time</td>
<td>2*T1</td>
<td>(Note 14)</td>
<td></td>
</tr>
<tr>
<td>T33</td>
<td>HF Inactive to Buffer Write</td>
<td>10</td>
<td></td>
<td></td>
</tr>
<tr>
<td>T34</td>
<td>Write Active Time</td>
<td>T1-10</td>
<td>T1 + 10</td>
<td></td>
</tr>
</tbody>
</table>

DMA PARAMETERS

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Parameter</th>
<th>Min</th>
<th>Max</th>
<th>Test Conditions</th>
</tr>
</thead>
<tbody>
<tr>
<td>T35</td>
<td>DRQn High or INTR to Clock Low Setup Time</td>
<td>50</td>
<td></td>
<td>(Note 15)</td>
</tr>
<tr>
<td>T36</td>
<td>DRQn Low to Clock Low, Hold Time</td>
<td>10</td>
<td></td>
<td></td>
</tr>
<tr>
<td>T37</td>
<td>EOP Pulse Width</td>
<td>T1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>T38</td>
<td>Address Delay Time</td>
<td>T2 + 75</td>
<td></td>
<td></td>
</tr>
<tr>
<td>T39</td>
<td>CS, CSn, DAKn Delay Time</td>
<td>T2 + 50</td>
<td></td>
<td></td>
</tr>
<tr>
<td>T40</td>
<td>CSn Delay Time (Slave to SRAM Flyby)</td>
<td>50</td>
<td></td>
<td></td>
</tr>
<tr>
<td>T41</td>
<td>IORD_MWR, IOWR Delay Time</td>
<td>45</td>
<td></td>
<td></td>
</tr>
<tr>
<td>T42</td>
<td>IORD_MWR, IOWR Active Time</td>
<td>(Note 16)</td>
<td>(Note 16)</td>
<td></td>
</tr>
</tbody>
</table>
**A.C. CHARACTERISTICS**

C\textsubscript{r} on all outputs is 50 pF. The user should add 0.2 ns/pF up to 100 pF (Continued)

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Parameter</th>
<th>Min</th>
<th>Max</th>
<th>Test Conditions</th>
</tr>
</thead>
<tbody>
<tr>
<td>T43</td>
<td>Interrupt Delay Time</td>
<td>75</td>
<td></td>
<td></td>
</tr>
<tr>
<td>T44</td>
<td>Interrupt Gap</td>
<td>3*T1-10</td>
<td></td>
<td></td>
</tr>
<tr>
<td>T45</td>
<td>Reset Setup Time</td>
<td>50</td>
<td></td>
<td></td>
</tr>
<tr>
<td>T46</td>
<td>Reset Active Time (High)</td>
<td>4*T1</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

*For pin HRDY 4 mA.*

**NOTES:**

1. Measured at V\textsubscript{CC}/2.
2. 3.2V to 1.8V.
3. 1.8V to 3.2V.
4. The following configuration affect the HRDY output going active (high).
   Legend:
   - TID—The configuration of HRDY delay (master mode register, TID = 0,5,1,5,2,5 ).
   - TIO—The configuration of I/O access delay (Control register, TIO = 0,1,2,3 ).
   - TMEM—The configuration of MEM access delay (Control register, TMEM = 0,1,2,3 ).
   - Tsystem = delay from HRDY to HF inactive.
   "If bit 5 of Register at Address 1BH then (TID+2)*T1 + 75
    else [TID + TIO(or TMEM) + 2]*T1 + 75"
5. The user should not that the XCVR lines goes inactive immediately after HF inactivation.
6. Provided that the HOST wins arbitration.
7. In the case of HOST write cycle the XCVR lines will go high at the end of the 82560 cycle even if HF lines are still active.
   In the case of HOST read cycles, the 82560 will terminate the local cycle after HF lines are inactivated.
8. Address lines are latched at the end of T1 of 82560 HOST bus cycles.
9. The maximum time specified assumes that the HOST wins the arbitration. If the HOST loses the arbitration to a DMA request two possible scenarios are:
   a) Arbitration lost to a single DMA cycle. In this case [(greater of TIO and TMEM) + 2]*T1 should be added to the max. time.
   b) Arbitration lost to a DMA cycle which is followed by four locked DMA cycles. In this case [(greater of TIO and TMEM) + 5 + 10]*T1 should be added to the max. time. This might happen in the rare case when the HOST request coincides with the last receive or transmit transfer, in the TCI mode.
10. [TIO(or TMEM) + 2]*T1 + 10.
    In the case of long (HF) HOST memory read requests, it would be extended until the request is removed.
11. Min = [TIO(or TMEM) + 1]*T1-10, Max = [TIO(or TMEM) + 1]*T1 + 10.
12. This parameter depends on T10. In terms of machine states, data remains valid until the end of the cycle (end of state T2).
13. (TMEM + 2)*T1 + 75 + Tsystem.
    Tsystem = delay from HRDY to HF inactive.
    This maximum time refers to a second memory request immediately following a first one, assuming that the first one was not delayed by a DMA cycle.
14. [TIO(or TMEM) + 2]*T1 + 75.
15. This is an asynchronous signal (DRQn only in its leading edge). It is internally synchronized. Meeting this parameter, assures recognition on the next clock.
16. Min = [(greater of TIO and TMEM) + 1]*T1 + T2 + 10
    Max = [(greater of TIO and TMEM) + 1]*T1 + T2 + 10
A.C. Testing Inputs are Driven at 2.4V for a Logic "1" and 0.45V for a Logic "0". Timing Measurements are made at 1.5V for both a Logic "1" and "0".

WAVEFORMS

HOST READ CYCLE—NON PIPELINE MODE
WAVEFORMS (Continued)

HOST WRITE CYCLE—NON PIPELINE MODE

![Waveform Diagram]

**Host Write Cycle—Non Pipeline Mode**

- **CLK**: Clock signal.
- **HF**: High Frequency signal.
- **HRDY**: Ready signal.
- **XCvn**: Crossbar signal.
- **RD**: Read signal.
- **AO-A12, BMn**: Address and Bus Mastership signals.
- **Data Valid**: Data valid signal.
- **MAO-12**: Master Address signal.
- **CSn, GCS, CS, DACKn**: Chip Select and Acknowledge signals.
- **IORD/MWR, IOWR**: Input/Output Read/Write signals.

**Timing Phases**:
- **T0, T1, T2, T3, T6, T7, T8, T9, T10, T11, T12, T13, T14, T15, T16, T17, T18, T20, T21, T22, T23, T24, T25, T26, T27**: Timing intervals for various signals.

**Host Write to 82560**
- T25

**Host Write to I/O or SRAM**
- T21, T22, T23, T24, T25
HOST READ CYCLE—PIPELINE MODE

WAVEFORMS (Continued)
WAVEFORMS (Continued)

HOST WRITE CYCLE—PIPELINE MODE

DMA FLYBY CYCLE—SLAVE TO SRAM
DMA FLYBY CYCLE—SRAM TO SLAVE

WAVEFORMS (Continued)
WAVEFORMS (Continued)

INTERRUPT

![Interrupt Waveform Diagram]

RESET

![Reset Waveform Diagram]
An 82586 Data Link Driver

CHARLES YAGER
INTRODUCTION

This application note describes a design example of an IEEE 802.2/802.3 compatible Data Link Driver using the 82586 LAN Coprocessor. The design example is based on the “Design Model” illustrated in “Programming the 82586”. It is recommended that before reading this application note, the reader clearly understands the 82586 data structures and the Design Model given in “Programming the 82586”.

“Programming the 82586” discusses two basic issues in the design of the 82586 data link driver. The first is how the 82586 handler fits into the operating system. One approach is that the 82586 handler is treated as a “special kind of interface” rather than a standard I/O interface. The special interface means a special driver that has the advantage of utilizing the 82586 features to enhance performance. However the performance enhancement is at the expense of device dependent upper layer software which precludes the use of a standard I/O interface.

The second issue “Programming the 82586” discusses which algorithms to choose for the CPU to control the 82586. The algorithms used in this data link design are taken directly from “Programming the 82586”. Command processing uses a linear static list, while receive processing uses a linear dynamic list.

The application example is written in C and uses the Intel C compiler. The target hardware for the Data Link Driver is the iSBC 186/51 COMMputer, however a version of the software is also available to run on the LANHIB Demo board.

1.0 FITTING THE SOFTWARE INTO THE OSI MODEL

The application example consists of four software modules:

- Data Link Driver (DLD): drives the 82586, also known as the 82586 Handler.
- Logical Link Control (LLC): implements the IEEE 802.2 standard.
- User Application (UAP): exercises the other software modules and runs a specific application.
- C hardware support: written in assembly language, supports the Intel C compiler for I/O, interrupts, and run time initialization for target hardware.

Figure 1 illustrates how these software modules combined with the 82586, 82501 and 82502 complete the first two layers of the OSI model. The 82502 implements an IEEE 802.3 compatible transceiver, while the 82501 completes the Physical layer by performing the serial interface encode/decode function.

The Data Link Layer, as defined in the IEEE 802 standard documents, is divided into two sublayers: the Logical Link Control (LLC) and the Medium Access Control (MAC) sublayers. The Medium Access Control sublayer is further divided into the 82586 Coprocessor plus the 82586 Handler. On top of the MAC is the LLC software module which provides IEEE 802.2 compatibility. The LLC software module implements the Station Component responses, dynamic addition and deletion of Service Access Points (SAPs), and a class 1 level of service. (For more information on the LLC sublayer, refer to IEEE 802.2 Logical Link Control Draft Standard.) The class 1 level of service provides a connectionless datagram interface as opposed to the class 2 level of service which provides a connection oriented level of service similar to HDLC Asynchronous Balanced Mode.

On top of the Data Link Layer is the Upper Layer Communications Software (ULCS). This contains the Network, Transport, Session, and Presentation Layers. These layers are not included in the design example, therefore the application layer of this ap note interfaces directly to the Data Link layer.
The application layer is implemented in the User Application (UAP) software module. The UAP module operates in one of three modes: Terminal Mode, Monitor Mode, and High Speed Transmit Mode. The software initially enters a menu driven interface which allows the program to modify several network parameters or enter one of the three modes.

The Terminal Mode implements a virtual terminal with datagram capability (connectionless "class 1" service). This mode can also be thought of as an async to IEEE 802.3/802.2 protocol converter.

The Monitor Mode provides a dynamic update on the terminal of 6 station related parameters. While in the monitor mode, any size frame can be repeatedly transmitted to the cable in a software loop.

High Speed Transmit Mode transmits frames to the cable as fast as the software possibly can. This mode demonstrates the throughput performance of the Data Link Driver.

The UAP gathers network statistics in all three modes as well as when it is in the menu. In addition, the UAP module provides the capability to alter MAC and LLC addresses and re-initialize the data link. (Figure 2 shows a combined software and hardware block diagram.)

2.0 LARGE MODEL COMPILATION

All the modules in this design example are compiled under the Large Model option. This has the advantages of using the entire 1 Mbyte address space, and allowing the string constants to be stored in ROM. In the Large Model it is important to consider that the 82586's data structures, SCB, CB, TBD, FD, and RBD, must reside within the same data segment. This data segment is determined at locate time.

The C_Assy_Support module has a run time start function which loads the DLD data segment into a global variable SEGMT_. This data segment is used by the 82586 Handler for address translation purposes. The 82586 uses a flat address while the 80186 uses a segmented address. Any time a conversion between 82586 and 80186 addresses are needed the SEGMT_ variable is used.

Points for the 80186 in the large model are 32 bits, segment and offset. All the 82586 link pointers are 16 bit offsets. Therefore when trading pointers between the 82586 and the 80186, two functions are called: Offset (ptr), and Build_Ptr (offset). Offset (ptr) takes a 32 bit 80186 pointer and returns just the offset portion for the 82586 link pointer. While Build_Ptr (offset) takes an 82586 link pointer and returns a 32 bit 80186 pointer, with the segment part being the SEGMT_ variable. Offset () and Build_Ptr() are simple functions written in assembly language included in the C_Assy_Support module.

In the small model, Offset () and Build_Ptr() are not needed, but the variable SEGMT_ is still needed for determining the SCB pointer in the ISCP, and in the Transmit and Receive Buffer Descriptors.

3.0 THE 82586 HANDLER

3.1 The Buffer Model

The buffer model chosen for the 82586 Handler is the "Design Model" as described in "Programming the 82586". This is based on the 82586 driver as a special driver rather than as a standard driver. Using this approach the ULCS directly accesses the 82586's Transmit and Receive Buffers, Buffer Descriptors and Frame Descriptors. This eliminates buffer copying. Transmit and receiver buffer passing is done entirely through pointers.
The only hardware dependencies between the Data Link and ULCS interface are the buffer structures. The ULCS does not handle the 82586's CBs, SCB or initialization structures. To isolate the data link interface from any hardware dependencies while still using the design model, another level of buffer copying must be introduced. For example, when the ULCS transmits a frame it would have to pass its own buffers to the data link. The data link then copies the data from ULCS buffers into 82586 buffers. When a frame is received, the data link copies the data from the 82586's buffers into the ULCS buffers. The more copying that is done the slower the throughput. However, this may be the only way to fit the data link into the operating system. The 82586 Handler can be made hardware independent by adding a receive and transmit function to perform the buffer copying.

The 82586 Handler allocates buffers from two pools of memory: the Transmit pool, and the Receive pool as illustrated in Figure 3. The Transmit pool contains Transmit Buffer Descriptors (TBDs) and Transmit Buffers (TBs). The Receive pool contains Frame Descriptors (FDs), Receive Buffer Descriptors (RBDs), and Receive Buffers (RBs).

When the ULCS wants to transmit, it requests a TBD from the handler. The handler returns a pointer to a free TBD. Each TBD has a TB attached to it. The ULCS fills the buffer, sets the appropriate fields in the TBD, and passes the TBD pointer back to the handler for transmission. After the frame is transmitted, the handler places the TBD back into the free TBD pool. If the ULCS needs more than one buffer per frame, it simply requests another TBD from the handler and performs the necessary linkage to the previous TBD.

On the receive side, the RFA pool is managed by the 82586 itself. When a frame is received, the 82586 interrupts the handler. The handler passes a FD pointer to the ULCS. Linked to the FD is one or more RBDs and RBs. The ULCS extracts what it needs from the FD, RBDs and RBs, and returns the FD pointer back to the handler. The handler places the FD and RBDs back into the free RFA pool.

3.2 The Handler Interface

The handler interface provides the following basic functions:

- initialization
- sending and receiving frames
- adding and deleting multicast addresses
- getting transmit buffers
- returning receive buffers

Figure 4 lists the Handler Interface functions.

On power up, the initialization function is called. This function initializes the 82586, and performs diagnostics. After initialization, the handler is ready to transmit and receive frames, and add and delete multicast addresses.

To send a frame, the ULCS gets one or more transmit buffers from the handler, fills them with data, and calls the send function. When a frame is received, the handler calls a receive function in the ULCS. The ULCS receive function removes the information it needs and returns the receive buffers to the handler. The addition and deletion of multicast addresses can be done "on the fly" any time after initialization. The receiver doesn't have to be disabled when this is done.

The command interface to the handler is totally asynchronous—the ULCS can issue transmit commands or multicast address commands whenever it wants. The commands are queued by the handler for the 82586 to execute. If the command queue is full, the send frame procedure returns a false status rather than true. The size of the command queue can be set at compile time by setting the CB—CNT constant. Typically the command queue never has more than a few commands on it because the 82586 can execute commands faster than the ULCS can issue them. This is not the case in a heavily loaded network when deferrals, collisions, and retries occur.

The command interface to the 82586 handler is hardware independent; the only hardware dependence is the buffering. A hardware independent command interface doesn't have any performance penalty, but some 82586 programmability is lost. This shouldn't be of concern since most data links do not change configuration parameters during operation. One can simply modify a few constants and recompile to change frame and network parameters to support other data links.
### Handler Interface Functions

<table>
<thead>
<tr>
<th>Function Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Init_586()</td>
<td>Initialize the Handler</td>
</tr>
<tr>
<td>Send_Frame (ptbd, padd)</td>
<td>Sends a frame to the cable.</td>
</tr>
<tr>
<td></td>
<td>ptbd—Transmit Buffer Descriptor pointer</td>
</tr>
<tr>
<td></td>
<td>padd—Destination Address pointer</td>
</tr>
<tr>
<td>Recv_Frame (pfd)</td>
<td>Handler calls this function which resides in the ULCS.</td>
</tr>
<tr>
<td></td>
<td>pfd—Frame Descriptor pointer</td>
</tr>
<tr>
<td>Add_Multicast_Address (pma)</td>
<td>Adds one multicast address</td>
</tr>
<tr>
<td></td>
<td>pma—Multicast Address pointer</td>
</tr>
<tr>
<td>Delete_Multicast_Address (pma)</td>
<td>Deletes one multicast address</td>
</tr>
<tr>
<td>Get_Tbd( )</td>
<td>Get a Transmit Buffer Descriptor pointer</td>
</tr>
<tr>
<td>Put_Free_Rfa (pfd)</td>
<td>Returns a Frame Descriptor and Receive Buffer Descriptors to the 82586.</td>
</tr>
</tbody>
</table>

**Figure 4. List of Handler Interface Functions**

![Free CB Pool Diagram](image)

**Figure 5. Free CB Pool**

![Free Transmit Buffer Descriptor Pool Diagram](image)

**Figure 6. Free Transmit Buffer Descriptor Pool**
3.3 Initialization

The function which initializes the 82586 handler, Init_586(), is called by the ULCS on power up or reinitialization. Before this function is called, an 82586 hardware or software reset should occur. The Initialization occurs in three phases. The first phase is to initialize the memory. This includes flags, vectors, counters, and data structures. The second phase is to initialize the 82586. The third phase is to perform self test diagnostics. Init_586() returns a status byte indicating the results of the diagnostics.

Init_586() begins by toggling the 82501 loopback pin. If the 82501 is powered up in loopback, the CRS and CDT pin may be active. To reset this condition, the loopback pin is toggled. The 82501 should remain in loopback for the first part of the initialization function.

Phase 1 executes initialization of all the handlers flags, interrupt vectors, counters, and 82586 data structures. There are two separate functions which initialize the CB and RFA pools: Build_CB() and Build_Rfa().

3.3.1 BUILDING THE CB AND RFA POOLS

Build_CB() builds a stack of free linked Command Blocks, and another stack of free linked Transmit Buffer Descriptors. (See Figures 5 and 6.) Each stack has a Top of Stack pointer, which points to the next free structure. The last structure on the list has a NULL link pointer.

The CBs within the list are initialized with 0 status, EL bit set, and a link to the next CB. The TBD structures are initialized with the buffer size, which is set at compile time with the TBUF_SIZE constant, a link to the next TBD, and an 82586 pointer to the transmit buffer. This pointer is a 24 bit flat/physical address. The address is built by taking the transmit buffer's data segment address, shifting it to the left by 4 and adding it to the transmit buffer offset. An 80186 pointer to the transmit buffer is added to the TBD structure so that the 80186 does not have to translate the address each time it accesses the transmit buffer.

Build_Rfa() builds a linear linked Frame Descriptor list and a Receive Buffer Descriptor list as shown in Figure 7. The status and EL bits for all the free FDs are 0. The last FD's EL bit is 1 and link pointer is NULL. The first FD on the FD list points to the first RBD on the RBD list. The RBDs are initialized with both 82586 and 80186 buffer pointers. The 80186 buffer pointer is added to the end of the RBD structure. Begin and end pointers are used to mark the boundaries of the free lists.

3.3.2 82586 Initialization

The 82586 initialization data structure SCP is already set since it resides in ROM, however, the ISCP must be loaded with information. Within the SCP ROM is the pointer to the ISCP; the ISCP is the only absolute address needed in the software. Once the ISCP address is determined, the ISCP can be loaded. The SCB base is obtained from the C_Assy_Support module. The global variable SEGMENT_contains the address of the

![Figure 7. Free RFA](https://example.com/f7.png)
data segment of the handler. The 80186 shifts this value to the left by 4 and loads it into the SCB base. The SCB offset is now determined by taking the 32 bit SCB pointer and passing it to the Offset() function.

The 82586 interrupt is disabled during initialization because the interrupt function is not designed to handle 82586 reset interrupts. To determine when the 82586 is finished with its reset/initialization, the SCB status is polled for both the CX and CNA bits to be set. After the 82586 is initialized, both the CX and CNA interrupts are acknowledged.

The 82586 is now ready to execute commands. The Configuration is executed first to place the 82586 in internal loopback mode, followed by the IA command. The address for the IA command is read off of a prom on the PC board.

### 3.3.3 SELF TEST DIAGNOSTICS

The final phase of the handler initialization is to run the self test diagnostics. Four tests are executed: Diagnose command, Internal loopback, External loopback through the 82501, and External loopback through the transceiver. If these four tests pass, the data link is ready to go on line.

The function that executes these diagnostics is called Test__Link(). If any of the tests fail, Test__Link() returns immediately with the Self__Test global variable set to the type of failure. This Self__Test global variable is then returned to the function which originally called Init_586(). Therefore Init_586() can return one of five results: FAILED__DIAGNOSE, FAILED__LPBK__INTERNAL, FAILED__LPBK__EXTERNAL, FAILED__LPBK__TRANSCEIVER or PASSED.

---

**Figure 8. Initialization Diagnostics: Test__Link()**

1-182
The Diagnose() function, called by Test__Link(), does not return until the diagnose command is completed. If the interrupt service routine detects that a Diagnose command was completed then it sets a flag to allow the Diagnose() function to return, and it also sets the Self__Test variable to FAIL if the Diagnose command failed. If the Diagnose command completed successfully, the loopback tests are performed.

Before any loopback tests are executed, the Receive Unit is enabled by calling Ru__Start(). Loopback tests begin by calling Send__Lpbk__Frame(), which sends 8 frames with known loopback data and its own destination address. More than one loopback frame is sent in case one or more of them are lost. Also several of the frames will have been received by the time flags.lpbk__test is checked.

Two flag bits are used for the loopback tests: flags.lpbk__mode, and flags.lpbk__test. flags.lpbk__mode is used to indicate to the receive section that the frames received are potentially loopback frames. The receive section will pass receive frames to the Loopback Check() function if the flags.lpbk__mode bit is set. The Loopback__Check() function first compares the source address of the frame with its station address. If this matches then the data is checked with the known loopback data. If the data matches, then the flags.lpbk__test bit is set, indicating a successful loopback. The flow of the Test__Link() function is displayed in Figure 8.

3.4 Command Processing

Command blocks are queued up on a static list for the 82586 to execute. The flow of a command block is given in Figure 9. When the handler executes a command it first has to get a free command block. It does this by calling Get__Cb() which returns a pointer to a free command block. The CB structure is a generic one in which all commands except the MC-Setup can fit in. The handler then loads into the CB structure the type of command and associated parameters. To issue the command to the 82586 the Issue__CU__Cmd() function is called with the pointer to the CB passed to this function. Issue__CU__Cmd() places the command on the 82586’s static command block list. After the 82586 executes the command, it generates an interrupt. The interrupt routine, Isr__586(), processes the command and returns the Command Block to the free command block list by calling Put__Cb().

3.4.1 ACCESSING COMMAND BLOCKS-GET__CB() and PUT__CB()

Get__Cb() returns a pointer to a free command block. The free command blocks are in a linear linked list structure which is treated as a stack. The pointer cb__tos points to the next available CB. Each time a CB is requested, Get__Cb() pops a CB off the stack. It does this by returning the pointer of cb__tos. cb__tos is then updated with the CB’s link pointer. When the CB list is empty, Get__Cb() returns NULL.

There are two types of nulls, the 82586 ‘NULL’ is a 16 bit offset, OFFFFH, in the 82586 data structures. The 80186 null pointer, ’pNULL’, is a 32 bit pointer; with OFFFFH offset and the 82586 handler’s data segment, SEGMT__, as the base.

Put__Cb() pushes a free command block back on the list. It does this by placing the cb__tos variable in the returned CB’s link pointer field, then updates cb__tos with the pointer to the returned CB.

3.4.2 ISSUING CU COMMANDS-ISSUE__CU__CMD()

This function queues up a command for the 82586 to execute. Since static lists are used, each command has its EL bit set. There is a begin__cbl pointer and an end__cbl pointer to delineate the 82586’s static list. If there are no CBs on the list, then begin__cbl is set to pNULL. (Figure 10 illustrates the static list.) Each time a command is issued, a deadman timer is set. When the 82586 interrupts the CPU with a command completed, the deadman timer is reset.

Issue__Cu__Cmd() begins by disabling the 82586’s interrupt. It then determines whether the list is empty or not. If the list is empty, begin and end pointers are loaded with the CB’s address. The CU must then be started. Before a CU_START can be issued, the SCB’s cbl_offset field must be loaded with the address of the command, the Wait__Scb() function must be called to insure that the SCB is ready to accept a command, and the deadman timer must be initialized. If the list is not empty, then the command block is queued at the end of the list, and the interrupt service routine Isr__586(), will continue generating CAs for each command linked on the CB list until the list is empty.
3.4.3 INTERRUPT SERVICE ROUTINE—ISR_586()

ISR_586() starts off by saving the interrupts that were generated by the 82586 and acknowledging them. Acknowledgment must be done immediately because if a second interrupt were generated before the acknowledgment, the second interrupt would be missed. The interrupt status is then checked for a receive interrupt and if one occurred the Recv___Int__Processing() function is called. After receive processing is check the CPU checks whether a command interrupt occurred. If one did, then the deadman timer is reset and the results of the command are checked. There are only two particular commands which the interrupt results are checked for: Transmit and Diagnose. The Diagnose command needs to be tested to see if it passed, plus the diagnose status flag needs to be set so that the initialization process can continue.

The transmit command status provides network management and station diagnostic information which is useful for the “Network Management” function of the ISO model. The following statistics are gathered in the interrupt routine: good_transmit_cnt, sqe_err_cnt, defer_cnt, no_crss_cnt, underrun_cnt, max_col_cnt. To speed up transmit interrupt processing a flag is tested to determine whether these statistics are desired, if not this section of code is skipped.

The sqe error requires special considerations when used for statistic gathering or diagnostics. The sqe status bit will be set if the transceiver’s self test passed. However if the sqe status bit is not set, the transceiver may still have passed its self test. Several events can prevent the sqe bit from being set. For example, the first transmit command status after power up will not have the sqe bit set because the sqe is always from the previous command. Also if any collisions occur, the sqe bit might not be set. This has to do with the timing of when the sqe signal comes from the transceiver. It is possible that a JAM signal from a remote station can overlap the sqe signal in which case the 82586 will not set the sqe status bit. Therefore the sqe error count should only be recorded when no collisions occur.

One other situation can occur which will prevent the SQE status bit from being set. If transmit command reaches the maximum retry count, the next transmit command’s SQE bit will not be set.

The final phase of interrupt command processing determines if another command is linked, and returns the CB to the free command block list. Another command being linked is indicated by the CB link field not being NULL. In this case the deadman timer and the 82586’s CU are re-started. If the CB link is NULL, there are no further commands to execute, and begin_cbl is set to pNULL.

3.4.4 SENDING FRAMES—SEND____FRAME (PTBD, PADD)

Send_Frame() receives two parameters, a pointer to the first Transmit Buffer Descriptor, and a pointer to the destination address. There may be one or more TBDs attached. The last TBD is indicated by its link
field being NULL and the EOF bit set. It is the responsibility of the ULCS to make sure this is done before calling Send_Frame().

Send_Frame() begins by trying to obtain a command block. If the free command block list is empty, the send frame function returns with a false result. It is up to the ULCS to either continue attempting transmission or attempt at a later time. The send frame function calculates the length field by summing up the TBDs actual count field. After the length field is determined, send frame checks to see if padding is required. If padding is necessary, Send Frame will change the act count field in the TBD to meet the minimum frame requirements. This technique transmits what ever was in the buffer as padding data. If security is an issue, the padding data in the buffer should be changed.

3.4.5 ACCESSING TRANSMIT BUFFERS—GET_TBD() AND PUT_TBD()

Get_Tbd() returns a pointer to a free Transmit Buffer Descriptor, and Put_Tbd() returns one or more linked Transmit Buffer Descriptors to the free list. The TBD which Get_Tbd() allocates has its link pointer set to NULL, and its EOF bit cleared. If another buffer is needed, the link field in the old TBD must be set to point to the new TBD. The last TBD used should have its link pointer set to NULL and its EOF bit set. Figure 11 shows the flow chart of getting buffers and sending a frame.

Put_Tbd(ptbd) is called by the Isr_586() function when the 82586 is done transmitting the buffers. A pointer to the first TBD is passed to Put_Tbd(). Put_Tbd() finds the end of the list of TBDs and returns them to the free buffer list.

3.4.6 MULTICAST ADDRESSES

The 82586 handler maintains a table of multicast addresses. Initially this table is empty. To enable a multicast address the Add_Multicast_Address(pma) function is called; to disable a multicast address, Delete_Multicast_Address(pma) function is called. Both functions accept a parameter which points to the multicast address. Add and Delete functions perform linear searches through the Multicast Address Table (MAT).

Add scans the entire MAT once to check if the address being added is a duplicate of one already loaded. Add will not enter a duplicate multicast address. If there are no duplicates Add goes to the beginning of the MAT and looks for a free location. If it finds one, it loads the new address into the free location and sets the location status to INUSE. If no free locations are available, Add returns a false result.

Delete looks for a used location in the MAT. When it finds one, it compares the address in the table with the address passed to it. If they match, the location status is set to FREE and a TRUE result is returned. If no match occurs, the result returned is FALSE.

If Add or Delete change the MAT, they update the 82586 by calling Set_Multicast_Address(). This function executes an 82586 MC Setup command. Set_Multicast_Address() uses the addresses in the MAT to build the MC Setup command. The MC Setup command is too big to be built from the free CBs.
command blocks are 18 bytes long, while the MC Setup command can be up to 16,392 bytes. Therefore a separate Multicast Command Block (ma_cb) must be allocated and used. The size of the ma_cb and MAT are determined at compile time based on the MULTI_ADDR_CNT constant. The design example allows up to 16 multicast addresses.

Since there is only one ma_cb, and it is not compatible with the other CBs, it must be treated differently. Only one ma_cb can be on the 82586 command list. The ma_cb command word is used as a semaphore. If it is zero, the command is available. If not, Set_Multicast_Address() must wait until the ma_cb is free. Also the interrupt routine can't return the ma_cb to the free CB list. It just clears the cmd field, to indicate that ma_cb is available.

The 82586's receiver does not have to be disabled to execute the MC Setup command. If the 82586 is receiving while this command is accessed, the 82586 will finish reception before executing the MC Setup command. If the MC Setup command is executing, the 82586 automatically ignores incoming frames until the MC Setup is completed. Therefore multicast addresses can be added and deleted on the fly.

3.4.7 RESETTING THE 82586—RESET_586()

The 82586 rarely if ever locks up in a well behaved network; (i.e. one that obeys IEEE 802.3 specifications). The lock-ups identified were artificially created and would normally not occur. This data link driver has been tested in an 8 station network under various loading conditions. No lock-ups occurred under any of the data link drivers test conditions. However the reset software has been tested by simulating a lockup. This can be done by having the 82586 transmit, and disabling the CTS pin for a time longer than the deadman timer.

An 82586 deadlock is not a fatal error. The handler is designed to recover from this problem. As mentioned before, each time the 82586 is given a CA to begin executing a command, a deadman timer is set. The deadman timer is reset when a CNR interrupt is generated. If the CNR interrupt is not generated before the deadman timer expires, the 82586 must be reset.

Resetting of the 82586 should not be done while the handler software is executing. This could create a software deadlock by interrupting a critical section of code in the handler. To insure that the Reset_586() function is not executed while the handler is executing, all of the entry points to the handler (i.e. interface functions) set a semaphore flag bit called flags.reset_sema. This flag is cleared when the interface functions are exited.

If the Deadman timer interrupt occurs while flags.reset_sema is set, another flag is set (flag.reset_pend) indicating that the Reset_586() function should be called when the interface functions are exited. However if the deadman timer interrupt occurs when flags.reset_sema is clear, Reset_586() is called immediately. Figure 12 shows the logic for entering and exiting interface functions.

Reset_586() begins by disabling the 82586 interrupt, placing the ESI in loopback, and resetting the 82586. The reset can be a software or a hardware reset. However, there are certain lockups in the 82586 where only a hardware reset will suffice. (The 82586 errata sheet explicitly indicates which deadlocks require a hardware reset.) After the reset, Reset_586() executes a Configure, IA-Setup, and a MC-Setup command; the MC Setup command is built from the multicast address table (MAT). The 82586 Command Queues and Receive Frame Queues are left untouched so that the 82586 can continue executing where it left off before the deadlock. This way no frames or commands are lost. This requires that a separate reset CB and reset Multicast CB is used, because other CBs already in use cannot be disturbed.
3.5 Receive Frame Processing

The following functions are used for Receive Frame Processing:

- **Recv__Int__Processing()** Called by Isr__586() to remove FDs and RBDs from the 82586's RFA.
- **Recv__Frame (pfd)** Called by Recv__Int__Processing(). This function resides in the ULCS.
- **Check__Multicast (pfd)** Used for perfect Multicast filtering.
- **Put__Free__Rfa (pfd)** Returns FDs and RBDs to the 82586's RFA.
- **Ru__Start( )** Restarts the RU when in the IDLE or No Resources state.

### 3.5.1 RECEIVE INTERRUPT PROCESSING - RECV__INT__PROCESSING()

The Recv__Int__Processing() function is called by Isr__586() when the FR bit in the SCB is set. The Recv__Int__Processing() function checks whether any FDs and RBDs on the free list have been used by the 82586. If they have, Recv__Int__Processing() removes the used FDs and RBDs from the free list, and passes them to the ULCS.

The Recv__Int__Processing() function is a loop where each pass removes a frame from the 82586's RFA. When there are no more used FDs and RBDs on the RFA, the function calls Ru__Start(), then returns to Isr__586(). The first part of the loop checks to see if the C bit in the first FD of the free FD list is set. If the C bit is set, the function determines if one or more RBDs are attached. If there are RBDs attached, the end of the RBD list is found. The last RBD's link field is used to update begin__rbd pointer, and then it's set to NULL.

After the receive frame has been delineated from the RFA, some information about the frame is needed to determine which function to pass it to. Since the save bad frame configure bit is not set, the only bad frame on the list could be an out of resource frame. An out of resource frame is returned to the RFA by calling Put__Free__RFA (pfd). If the flags.lpbb_mode bit is set, the frame is given to the loopback check function. If the destination address of the frame indicates a multicast, the check multicast function is called. If the frame has passed all of the above tests and still has not been returned, it is passed to the Recv__Frame( ) function which resides in the ULCS.

Check__Multicast (pfd) determines whether the multicast address received is in the multicast address table. This is necessary because the 82586 does not have perfect multicast address filtering. Check__Multicast does a byte by byte comparison of the destination address with the addresses in the multicast address table. If no match occurs, it returns false, and Recv__Int__Processing calls Put__Free__RFA (pfd) to return the frame to the RFA. If there is a match, Check__Multicast() returns TRUE and Recv__Int__Processing() calls Recv__Frame( ), passing the pointer to the FD of the frame received.

### 3.5.2 RETURNING FDs AND RBDs - PUT__FREE__RFA (pfd)

Put__Free__RFA combines Supply__FD and Supply__RBD algorithms described in “Programming the 82586” into one function. The begin and end pointers delineate what the CPU believes is the beginning and end of the free list. The decision of whether to restart the RU is made when examining both the free FD list and the free RBD list. This is why two ru_start_flags are used, one for the FD list and one for the RBD list. Both flags are initialized to FALSE.

The function starts off by initializing the FD so that the EL bit is set, the status is 0, and the FD link field is NULL. The rbd pointer is saved before the rbd pointer field in the FD is set to NULL. The free FD list is examined and if it's empty, begin_fd and end_fd are loaded with the address of the FD being returned. In this case the RU should not be restarted, because there is only one FD on the free list. If the free FD list is not empty, the FD being returned is placed on the end of the list, the end pointer is updated, and the RU start flag is set TRUE.

To begin the RBD list processing the end of the returned RBD list is determined, and this last RBD's EL bit is set. If the free RBD list is empty, the returned RBD list becomes the free RBD list. If there is more than one RBD on the returned list, the ru start flag is set TRUE. If the free RBD list is not empty, the returned RBD list is appended on the end of the free list, the end_rbd pointer is updated, and the ru start flag is set TRUE.

The last part of Put__Free__RFA( ) is to determine whether to call Ru__Start( ). Both ru start flags are ANDed together, and if the result is TRUE, the Ru__Start( ) function is called.

### 3.5.3 RESTARTING THE RECEIVE UNIT - RU__START( )

The Ru__Start( ) function checks two things before it decides to restart the RU. The first thing it checks is whether the RU is already READY. If it is, there is no reason to restart it. If the RU is IDLE or in NO__RESOURCES, then the second thing to check is whether the first free FD on the free FD list has its C bit set. If it does, then the RU should not be restarted. The reason is that the free FD list should only contain free FDs.
when the RU is started. If the C bit is set in the FD, then not all the used FD have been removed yet. If the RU is started when used FDs are still in the RFA, the 82586 will write over the used FDs and frames will be lost. Therefore Ru__Start() is exited if the first FD in the RFA has its C bit set. If the RU is not READY, and begin_fd doesn’t point to a used FD, then the RU is restarted.

Note that in “Programming the 82586” there are two more conditions to be met before the RU is started: two or more FD on the RFA, and two or more RBD on the RFA. These conditions are checked in Put_Free_RFA(), and Ru__Start() isn’t called unless they are met.

4.0 LOGICAL LINK CONTROL

The IEEE 802.2 LLC function completes the Data Link Layer of the OSI model. The LLC module in this design example implements a class 1 level of service which provides a connectionless datagram interface. Several data link users or processes can run on top of the data link layer. Each user is identified by a link service access point (LSAP). Communication between data link users is via LSAPs. An LSAP is an address that identifies a specific user process or another layer (see Figure 13). The LSAP addresses are defined as follows:

<table>
<thead>
<tr>
<th>Layer</th>
<th>LSAP Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data Link Layer (Station Component)</td>
<td>00H</td>
</tr>
<tr>
<td>Transport Layer</td>
<td>FEH</td>
</tr>
<tr>
<td>Network Management Layer</td>
<td>08H</td>
</tr>
<tr>
<td>User Processes</td>
<td>multiples of 4 in the range 0CH &lt; LSAP &lt; FCH</td>
</tr>
</tbody>
</table>

Each receiving process is identified by a destination LSAP (DSAP) and each sending process is identified by a source LSAP (SSAP). Before a destination process can receive a packet, its DSAP must be included in a list of active DSAPs for the data link.

Figure 14 illustrates the relationship between the Station Component and the SAP components. (The SAP components are user processes.) The Station Component receives all of the good frames from the Handler and checks the DSAP address. If the DSAP address is 0, then the frame is addressed to the Station Component and a Station Component Response is generated. If the DSAP address is on the active DSAP list, then the Station Component passes the frame to the addressed SAP. If the DSAP address is unknown, the frame is returned to the handler.

![Figure 13. Data Link Interface](image-url)
There are 3 commands and 2 responses which the class 1 LLC layer must implement. Figure 15 shows IEEE 802.2 Class 1 commands and responses and Figure 16 shows the IEEE 802.2 Class 1 frame format.

<table>
<thead>
<tr>
<th>Commands</th>
<th>Responses</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>UI</td>
<td>XID</td>
<td>Unnumbered Information</td>
</tr>
<tr>
<td></td>
<td>TEST</td>
<td>Exchange ID</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Remote Loopback</td>
</tr>
</tbody>
</table>

Any frames addressed to active SAPs are passed directly to them. The Station Component will not respond to SAP addressed frames. Therefore it is the responsibility of the SAPs to recognize and respond to frames addressed to them. When a SAP transmits a frame, it builds the IEEE 802.2 frame itself and calls the Handler's Send_Frame() function directly. The LLC module is not used for SAP frame transmission. The only functions which the LLC module implement are the dynamic addition and deletion of DSAPs, multiplexing the frames to user SAPs, and the Station Component command recognition and responses. This is one implementation of the IEEE 802.2 standard. Other implementations may have the LLC module do more functions, such as SAP command recognitions and responses. A list of the functions included in the LLC module is as follows:

<table>
<thead>
<tr>
<th>LLC Functions</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Init__Llc()</td>
<td>Initializes the DSAP address table and calls Init__586()</td>
</tr>
<tr>
<td>Add__Dsap__Address</td>
<td>Add a DSAP address to the active list</td>
</tr>
<tr>
<td>Delete__Dsap__Address</td>
<td>Delete a DSAP address</td>
</tr>
<tr>
<td>Recv__Frame</td>
<td>Receives a frame from the 82586 Handler</td>
</tr>
<tr>
<td>Station__Component__Response</td>
<td>Generates a response to a frame addressed to the Station Component</td>
</tr>
</tbody>
</table>

From Figure 15 it can be seen that there are no LLC class 1 UI responses because information frames are not acknowledged at the data link level. The only command frames that may require responses are XID and TEST. If a command frame is addressed to the Station Component, it checks the control field to see what type of frame it is. If it’s an XID frame, the Station Component responds with a class 1 XID response frame. If it’s a TEST frame, the Station Component responds with a TEST frame, echoing back the data it received. In both cases, the response frame is addressed to the source of the command frame.
4.1 Adding and Deleting LSAPs

When a user process wants to add a LSAP to the active list, the process calls Add_Dsap_Address(dsap, pfunc). The dsap parameter is the actual DSAP address, and the pfunc parameter is the address of the function to be called when a frame with the associated DSAP address is received.

The LLC module maintains a table of active dsaps which consists of an array of structures. Each structure contains two members: stat - indicates whether the address is free or inuse, and (*p_sap_func)() contains the address of the function to call. The index into the array of structures is the DSAP address. This speeds up processing by eliminating a linear search. Delete_Dsap_Address (dsap) simply uses the DSAP index to mark the stat field FREE.

5.0 APPLICATION LAYER

For most networks the application layer resides on top of several other layers referred to here as ULCS. These other layers in the OSI model run from the network layer through the presentation layer. The implementation of the ULCS layers is beyond the scope of this application note, however Intel provides these layers as well as the data link layer with the OpenNET product line. For the purpose of this application note the application layer resides on top of the data link layer and its use is to demonstrate, exercise and test the data link layer design example.

There can be several processes sitting on top of the data link layer. Each process appears as a SAP to the data link. The UAP module, which implements the application layer, is the only SAP residing on top of the data link layer in this application example. Other SAPs could certainly be added such as additional "connectionless" terminals, a networking gateway, or a transport layer, however in the interest of time this was not done.

5.1 Application Layer Human Interface

The UAP provides a menu driven human interface via an async terminal connected to port B on the iSBC 186/51 board. The menu of the commands is listed in Figure 17 along with a description that follows:

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>T</td>
<td>Terminal Mode - implements a virtual terminal with datagram capability (connectionless &quot;class 1&quot; service). This mode can also be thought of as an async to IEEE 802.2/802.3 protocol converter.</td>
</tr>
<tr>
<td>M</td>
<td>Monitor Mode</td>
</tr>
<tr>
<td>X</td>
<td>High Speed Transmit Mode</td>
</tr>
<tr>
<td>V</td>
<td>Change Transmit Statistics</td>
</tr>
<tr>
<td>P</td>
<td>Print All Counters</td>
</tr>
<tr>
<td>C</td>
<td>Clear All Counters</td>
</tr>
<tr>
<td>A</td>
<td>Add a Multicast Address</td>
</tr>
<tr>
<td>Z</td>
<td>Delete a Multicast Address</td>
</tr>
<tr>
<td>S</td>
<td>Change the SSAP Address</td>
</tr>
<tr>
<td>D</td>
<td>Change the DSAP Address</td>
</tr>
<tr>
<td>N</td>
<td>Change Destination Node Address</td>
</tr>
<tr>
<td>L</td>
<td>Print All Addresses</td>
</tr>
<tr>
<td>R</td>
<td>Re-Initialize the Data Link</td>
</tr>
<tr>
<td>B</td>
<td>Change the Number Base</td>
</tr>
</tbody>
</table>

Each time a frame has been successfully transmitted the Good frames transmitted count is incremented. The same holds true for reception. CRC, Alignment, Out of Resources, and Overrun Errors are all obtained from the SCB. Underrun, lost CRS, SQE error, Max retry, and Frames that deferred are all transmit statistics that are obtained from the Transmit command status word. 82586 Reset is a count which is incremented each time the 82586 locks up. This count has never normally been incremented.

Figure 17. Menu of Data Link Driver Commands
Clear All Counters - Resets all of the counters.

Add/Delete Multicast Address - Adds and Deletes Multicast Addresses.

Change SSAP Address - Deletes the previous SSAP and adds a new one to the active list. The SSAP in this case is this station's LSAP. When a frame is received, the DSAP address in the frame received is compared with any active LSAPs on the list. The SSAP is also used in the SSAP field of all transmitted frames.

Change DSAP Address - Delete the old DSAP and add a new one. The DSAP is the address of the LSAP which all transmit frames are sent to.

Change Destination Node Address - Address a new node.

Print All Addresses - Display on the terminal the station address, destination address, SSAP, DSAP, and all multicast addresses.

Re-initialize Data Link - This causes the Data Link to completely reinitialize itself. The 82586 is reset and reinitialized, and the selftest diagnostic and loopback tests are executed. The results of the diagnostics are printed on the terminal. The possible output messages from the 82586 selftest diagnostics are:

- Passed Diagnostic Self Tests
- Failed: Self Test Diagnose Command
- Failed: Internal Loopback Self Test
- Failed: External Loopback Self Test
- Failed: External Loopback Through Transceiver Self Test

Change Base - Allows all numbers to be displayed in Hex or Decimal.

5.2 A Sample Session

The following text was taken directly from running the Data Link software on a 186/51 board. It begins with the iSDM monitor signing on and continues into executing the Data Link Driver software.

---

Passed Diagnostic Self Tests

Enter the Address of the Destination Node in Hex -> 00AA000179E

Enter this Station's LSAP in Hex -> 20

Enter the Destination Node's LSAP in Hex -> 20

Do you want to Load any Multicast Addresses? (Y or N) -> Y

Enter the Multicast Address in Hex -> 00AA001llll

Would you like to add another Multicast Address? (Y or N) -> N

This Station's Host Address is: 00AA0001868

The Address of the Destination Node is: 00AA000179E

This Station's LSAP Address is: 20

The Address of the Destination LSAP is: 20

The following Multicast Addresses are enabled: 00AA001llll
Commands are:

T - Terminal Mode  
M - Monitor Mode  
X - High Speed Transmit Mode  
V - Change Transmit Statistics  
P - Print All Counters  
C - Clear All Counters  
A - Add a Multicast Address  
Z - Delete a Multicast Address  
S - Change the SSAP Address  
D - Change the DSAP Address  
N - Change Destination Node Address  
L - Print All Addresses  
B - Change the number Base  

Enter a command, type H for Help -- > P

Good frames transmitted: 24  
Good frames received: 1  
CRC errors received: 0  
Alignment errors received: 0  
Out of Resource frames: 0  
Receiver overrun frames: 0  
82586 Reset: 0  
Transmit underrun frames: 0  
Lost CRS: 0  
SQE errors: 9  
Maximum retry: 4  
Frames that deferred: 4  

Enter a command, type H for Help -- > T  
Would you like the local echo on? (Y or N) -- > Y  
This program will now enter the terminal mode.  
Press 'C then CR to return back to the menu.

Hello this is a test.

/* "C CR "*/

Enter a command, type H for Help -- > M  
Do you want this station to transmit? (Y or N) -- > Y  
Enter the number of data bytes in the frame -- > 1500  
Hit any key to exit Monitor Mode.

<table>
<thead>
<tr>
<th># of Good Frames Transmitted</th>
<th># of Good Frames Received</th>
<th>CRC Errors</th>
<th>Alignment Errors</th>
<th>No Resource Errors</th>
<th>Receive Overrun Errors</th>
</tr>
</thead>
<tbody>
<tr>
<td>32</td>
<td>0</td>
<td>000000</td>
<td>000000</td>
<td>000000</td>
<td>000000</td>
</tr>
</tbody>
</table>

/* "CR "*/

Enter a command, type H for Help -- > X  
Hit any key to exit High Speed Transmit Mode.

/* "CR "*/

Enter a command, type H for Help -- > R  
Passed Diagnostic Self Tests
5.3 Terminal Mode

The Terminal mode buffers characters received from the terminal and sends them in a frame to the cable. When a frame is received from the cable, data is extracted and sent to the terminal. One of three events initiate the UAP to send a frame providing there is data to send: buffering more than 1500 bytes, receiving a Carriage Return from the terminal, or receiving an interrupt from the virtual terminal timer.

The virtual terminal timer employs timer 1 in the 80130 to cause an interrupt every .125 seconds. Each time the interrupt occurs the software checks to see if it received one or more characters from the terminal. If it did, then it sends the characters in a frame.

The interface to the async terminal is a 256 byte software FIFO. Since the terminal communication is full duplex, there are two half duplex FIFOs: a Transmit FIFO and a Receive FIFO. Each FIFO uses two functions for I/O: Fifo_In() and Fifo_Out(). A block diagram is displayed in Figure 18.

The serial I/O for the async terminal interface is always polled except in the Terminal mode where it is interrupt driven. The Terminal mode begins by enabling the 8274 receive interrupt but leaves the 8274 transmit interrupt disabled. This way any characters received from the terminal will cause an interrupt. These characters are then placed in the Transmit FIFO. The only time the 8274 transmit interrupt is enabled is when the Receive FIFO has data in it. The receive FIFO is filled from frames being received from the cable. Each time a transmit interrupt occurs a byte is removed from the Receive FIFO and written to the 8274. When the Receive FIFO empties, the 8274 transmit interrupt is disabled.

The flow control implemented for the terminal interface is via RTS and CTS. When the Transmit FIFO is full, RTS goes inactive preventing further reception of characters (see Table 1). If the Receive FIFO is full, receive frames are lost because there is no way for the data link using class 1 service to communicate to the remote station that the buffers are full. Lost receive frames are accounted for by the Out of Resources Frame counter.

The Async Terminal bit rate sets the throughput capability of the station in the terminal mode because the bottleneck for this network is the RS232 interface. Using this fact a simple test was conducted to verify the data link driver's capability of switching between the receiver's No Resource state and the Ready State. For example if station B is sending frames in the High Speed Transmit mode to station A which is in the Terminal mode, frames will be lost in station A. Under these circumstances station A's receiver will be switching from Ready state to Out of Resources state. The sum of Good frames received plus Out of Resource frames from station A should equal Good frames transmitted from station B; unless there were any underruns or overruns.

### Table 1. FIFO State Table

<table>
<thead>
<tr>
<th>Function</th>
<th>Present State</th>
<th>Next State</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>FIFO_T__IN()</td>
<td>EMPTY</td>
<td>IN USE</td>
<td>Start Filling Transmit Buffer</td>
</tr>
<tr>
<td></td>
<td>IN USE</td>
<td>FULL</td>
<td>Shut Off RTS</td>
</tr>
<tr>
<td>FIFO_T__OUT()</td>
<td>FULL</td>
<td>IN USE</td>
<td>Enable RTS</td>
</tr>
<tr>
<td></td>
<td>IN USE</td>
<td>EMPTY</td>
<td>Stop Filling Transmit Buffer</td>
</tr>
<tr>
<td>FIFO_R__IN()</td>
<td>EMPTY</td>
<td>IN USE</td>
<td>Turn on Txlnt</td>
</tr>
<tr>
<td></td>
<td>IN USE</td>
<td>FULL</td>
<td>Stop Filling FIFO from Receive Buffer</td>
</tr>
<tr>
<td>FIFO_R__OUT()</td>
<td>FULL</td>
<td>IN USE</td>
<td>Start Filling FIFO from Receive Buffer</td>
</tr>
<tr>
<td></td>
<td>IN USE</td>
<td>EMPTY</td>
<td>Turn Off Txlnt</td>
</tr>
</tbody>
</table>

The Async Terminal bit rate sets the throughput capability of the station in the terminal mode because the bottle neck for this network is the RS232 interface. Using this fact a simple test was conducted to verify the data link driver's capability of switching between the receiver's No Resource state and the Ready State. For example if station B is sending frames in the High Speed Transmit mode to station A which is in the Terminal mode, frames will be lost in station A. Under these circumstances station A's receiver will be switching from Ready state to Out of Resources state. The sum of Good frames received plus Out of Resource frames from station A should equal Good frames transmitted from station B; unless there were any underruns or overruns.
5.3.1 Sending Frames

The Terminal Mode is entered when the Terminal_Mode() function is called from the Menu interface. The Terminal_Mode() function is one big loop, where each pass sends a frame. Receiving frames in the Terminal Mode is handled on an interrupt driven basis which will be discussed next.

The loop begins by getting a TBD from the 82586 handler. The first three bytes of the first buffer are loaded with the IEEE 802.2 header information. The loop then waits for the Transmit FIFO to become not EMPTY, at which point a byte is removed from the Transmit FIFO and placed in the TBD. After each byte is removed from the Transmit FIFO several conditions are tested to determine whether the frame needs to be transmitted, or whether a new buffer must be obtained. A frame needs to be transmitted if: a Carriage Return is received, the maximum frame length is reached, or the send_frame flag is set by the virtual terminal timer. A new buffer must be obtained if none of the above is true and the max buffer size is reached.

If a frame needs to be sent the last TBD’s EOP bit is set and its buffer count is updated. The 82586 Handler’s Send_Frame() function is called to transmit the frame, and continues to be called until the function returns TRUE.

The loop is repeated until a 'c' followed by a Carriage Return is received.

5.3.2 Receiving Frames

Upon initialization the UAP module calls the Add_Dsap_Address(dsap, pfunc) function in the LLC module. This function adds the UAP’s LSAP to the active list. The pfunc parameter is the address of the function to call when a frame has been received with the UAP’s LSAP address. This function is Recv_Data__1(). Recv_Data__1() looks at the control field of the frame received and determines the action required.

The commands and responses handled by Recv_Data__1() are the same as the Station Component’s commands and responses given in Figure 15. One difference is that Recv_Data__1() will process a UI command while the Station Component will ignore a UI command addressed to it.

Recv_Data__1() will discard any UI frames received unless it is in the Terminal Mode. When in the Terminal Mode, Recv_Data__1() skips over the IEEE 802.2 header information and uses the length field to determine the number of bytes to place in the Receive FIFO. Before a byte is placed in the FIFO, the FIFO status is checked to make sure it is not full. Recv_Data__1() will move all of the data from the frame into the Receive FIFO before returning.

When a frame is received by the 82586 handler an interrupt is generated. While in the 82586 interrupt routine the receive frame is passed to the LLC layer and then to the UAP layer where the data is placed in the Receive FIFO by Recv_Octal_Data__1(). Since Recv_Data__1() will not return until all of the data from the frame has been moved into the Receive FIFO, the 8274 transmit interrupt must be nested at a higher priority than the 82586 interrupt to prevent a software lock. For example if a frame is received which has more than 256 bytes of data, the Receive FIFO will fill up. The only way it can empty is if the 8274 interrupt can nest the 82586 interrupt service routine. If the 8274 could not interrupt the 82586 ISR then the software would be stuck in Recv_Data__1() waiting for the FIFO to empty.

5.4 Monitor Mode

The Monitor Mode dynamically updates 6 station related parameters on the terminal as shown below.

The Monitor_Mode() function consists of one loop. During each pass through the loop the counters are updated, and a frame is sent. Any size frame can be transmitted up to a size of the maximum number of transmit buffers available. Frame sizes less than the minimum frame length are automatically padded by the 82586 Handler.

The data in the frames transmitted in the Monitor Mode are loaded with all the printable ASCII characters. This way when one station is in the Monitor Mode transmitting to another station in the Terminal Mode, the Terminal Mode station will display a marching pattern of ASCII characters.

<table>
<thead>
<tr>
<th># of Good Frames Transmitted</th>
<th># of Good Frames Received</th>
<th>CRC Errors</th>
<th>Alignment Errors</th>
<th>No Resource Errors</th>
<th>Receive Overrun Errors</th>
</tr>
</thead>
<tbody>
<tr>
<td>32</td>
<td>0</td>
<td>00000</td>
<td>00000</td>
<td>00000</td>
<td>00000</td>
</tr>
</tbody>
</table>
5.5 High Speed Transmit Mode

The High Speed Transmit Mode demonstrates the throughput performance of the 82586 Handler. The Hs_Xmit_Mode() function operates in a tight loop which gets a TBD, sets the EOF bit, and calls Send_Frame(). The flow chart for this loop is shown in Figure 19.

The loop is exited when a character is received from the terminal. Rather than polling the 8274 for a receive buffer full status, the 8274's receive interrupt is used. When the Hs_Xmit_Mode() function is entered, the hs_stat flag is set true. If the 8274 receive interrupt occurs, the hs_stat flag is set false. This way the loop only has to test the hs_stat flag rather than calling inb() function each pass through the loop to determine whether a character has been received.

The performance measured on an 8 MHz 186/51 board is 593 frames per second. The bottle neck in the throughput is the software and not the 82586. The size of the buffer is not relevant to the transmit frame rate. Whether the buffer size is 128 bytes or 1500 bytes, linked or not, the frame rate is still the same. Therefore assuming a 1500 byte buffer at 593 frames per second, the effective data rate is 889,500 bytes per second.

This can easily be demonstrated by using two 186/51 boards running the Data Link software. The receiving stations counters should be cleared then placed in the Monitor mode. When placing it in the monitor mode, transmission should not be enabled. When the other station is placed in the High Speed Transmit Mode a timer should be started. One can use a stop watch to determine the time interval for transmission. The frame rate is determined by dividing the number of frames received in the Monitor station by the time interval of transmission.
APPENDIX A
COMPILING, LINKING, LOCATING, AND RUNNING THE SOFTWARE ON THE 186/51 BOARD

********* Instructions for using the 186/51 board *********

Use 27128A for no wait state operation. 27128s can be used but wait states will have to be added.

Copy HI.BYT and LO.BYT files into EPROMs
PROMs go into U34 - HI.BYT and U39 - LO.BYT on the 186/51 board

JUMPERS REQUIRED
Jumper the 186/51 board for 16K byte PROMs in U34 and U39 Table 2-5 in 186/51 HARDWARE REFERENCE MANUAL (Rev-001)

186/51(ES) 186/51 (S)/186/51
E151-E152 OUT E199-E203 OUT
E152-E150 IN E203-E191 IN
E94-E95 IN E120-E119 IN
E100-E106 IN E116-E112 IN
E107-E113 IN E111-E107 IN
E133-E134 IN E94-E93 IN

also change interrupt priority jumpers - switch 8274 and 82586 interrupt priorities

WIRE WRAP
E36-E47 IN
E39-E44 IN
E79-E45 IN

USE SDM MONITOR
The SDM Monitor should have the 82586's SCP burned into ROM. The ISCP is located at OFFFOH. Therefore for the SCP the value in the SDM ROM should be:

ADDRESS DATA
FFFF6H XXOOH
FFFF8H XXXXH
FFFFAH XXXXH
FFFFCH FFFOH
FFFFEH XXOOH

To run the program begin execution at 0D000:6H
I.E. G D000:6
GOOD LUCK!

************* submit file for compiling one module: *************
run
c86.86 :F6:%0 LARGE ROM DEBUG DEFINE(DEBUG) include(:F6:)
ext

************* submit file for linking and locating: *************
run
:F6:uap.obj, lclib.lib to :F6:dld.lnk segsize(stack(4000h)) notype
loc86 :F6:dld.lnk to :F6:dld.loc&
initcode (OD0000H) start(begin) order(classes(data, stack, code)) &
addresses(classes(data(3000H), stack(OCBOOH), code(OD0020H)))
ch86 :F6:dld.loc to :F6:dld.rom
exit

************* submit file for burning EPROMs using IPPS: *************
ipps
i 86
f :F6:dld.rom (OD0000h)
3
c
0 to :F6:lo.byty
1
1 to :F6:hi.byty

t 27128
9
c :F6:lo.byty t p
n
C :F6:hi.byty t p
n
exit
/PCD/USR/CHUCK/CSRC/DLD.H

.fullName

../../../S29B6 Structures and Constants

/* general purpose constants */
define INUSE 0
define EMPTY 1
define FULL 2
define FREE 1
define TRUE 1
define FALSE 0
define NULL 0xFFFF

/* Define Data Structures */
define RBUF_SIZE 128 /* receive buffer size */
define TBUF_SIZE 128 /* transmit buffer size */
define ADD_LEN 6
#define MULTI_ADDR_CNT 16
typedef unsigned short int u_short;

/* results from Test_Link(); loaded into Self_Test char */
define PASSED 0
define FAILED_DIAGNOSE 1
define FAILED_LP8K_INTERNAI 2
define FAILED_LP8K_EXTERNAL 3
define FAILED_LP8K_TRANSCEIVER 4

/* Frame Commands */
define UI 0x03 /* Unnumbered Information Frame */
define XID 0xAF /* Exchange Identification */
define TEST 0x03 /* Remote Loopback Test */
define P_F_BIT 0x10 /* Poll/Final Bit Position */
define C_R_BIT 0x01 /* Command/Response bit in SSAP */
define DSAP_CNT 8 /* Number of allowable DSAPs; must be a multiple of 2**(N), and DSAP addresses assigned must be divisible by 2**(8-N). (i.e. the N LSBs must be 0) */
define DSAP_SHIFT 5 /* DSAP_Shifts must equal B-N */
define XID_LENGTH 6 /* Number of Info bytes for XID Response frame */

/* System Configuration Pointer SCP */
struct SCP {
  u_short sysbus; /* 82596 bus width. 0 - 16 bits
                   1 - 8 bits */
}

231421-17
```c
/* Intermediate System Configuration Pointer ISCPtr */
struct ISCPtr {
    u_short busy; /* set to 1 by cpu before its first CA,
                    cleared by 52506 after reading */
    u_short offset; /* offset of system control block */
    u_short base1; /* base of system control block */
    u_short base2; /* */
};

/* System Control Block BCB */
struct BCB {
    u_short stat; /* Status word */
    u_short cmd; /* Command word */
    u_short cb1_offset; /* Offset of first command block in CRL */
    u_short RFA_offset; /* Offset of first frame descriptor in RFA */
    u_short crc_errs; /* CRC errors accumulated */
    u_short aln_errs; /* Alignment errors */
    u_short rsc_errs; /* Frames lost because of no Resources */
    u_short ovr_errs; /* Overrun errors */
};

/* Command Block */
struct CB {
    u_short stat; /* Status of Command */
    u_short cmd; /* Command */
    u_short link; /* Link field */
    u_short param1; /* Parameters */
    u_short param2; /* */
    u_short param3; /* */
    u_short param4; /* */
    u_short param5; /* */
    u_short param6; /* */
};

/* Multicast Address Command Block MA_CB */
struct MA_CB {
    u_short stat; /* Status of Command */
    u_short cmd; /* Command */
    u_short link; /* Link field */
    u_short mc_cnt; /* Number of MC addresses */
    char mc_addr[ADD_LEN+MULTI_ADDR_CNT]; /* MC address area */
};

/* Transmit Buffer Descriptor TBD */
struct TBD {
    /* */
};
```
u_short act_cnt;  /* Number of bytes in buffer */
  u_short link;   /* offset to next TBD */
  u_short buff_l; /* lower 16 bits of buffer address */
  u_short buff_h; /* upper 8 bits of buffer address */
      struct TBD *buff_ptr; /* not used by the 586; used by the
                        software to save address translation
                        routine. */
    
/* Transmit Buffers */
  struct TBD {
    char data [TBUF_SIZE];
  };

/* Frame Descriptor FD */
  struct FD {
    u_short stat;    /* Status Word of FD */
    u_short el_e;   /* EL and 8 bits */
    u_short link;   /* link to next FD */
    u_short rbd_offset; /* Receive buffer descriptor offset */
    char dest_addr[ADD_LEN]; /* Destination address */
    char src_addr[ADD_LEN]; /* Source address */
    u_short length; /* Length field */
  };

/* Receive Buffer Descriptor RBD */
  struct RBD {
    u_short act_cnt; /* Actual number of bytes received */
    u_short link;   /* Offset to next RBD */
    u_short buff_l; /* Lower 16 bits of buffer address */
    u_short buff_h; /* upper 8 bits of buffer address */
    u_short size;   /* size of buffer */
    struct RB *buff_ptr; /* not used by the 586; used by the
                        software to save address translation
                        routine */
  };

/* Receive Buffers */
  struct RB {
    char data[RBUF_SIZE];
  };

struct FRAME_STRUCT {
    unsigned char dsap; /* Destination Service Access Point */
    unsigned char ssap; /* Source Service Access Point */
    unsigned char cmd; /* ISD Data Link Command */
};

/* LBAP Address Table */
  struct LAT { char stat; /* INUSE or FREE */

AP-235

/PCG/USB/CHUCK/CSRC/DLD.H

int (*_sap_func)(); /* Pointer to LSAP function, associated with dsap address */

struct MAT { /* Multicast Address Table */
    char stat; /* INUSE or FREE */
    char addr[ADD_LEN]; /* actual ac address */
};

/* general purpose flags */

struct FLAGS {
    unsigned diag_done: 1; /* diagnose command complete */
    unsigned stat_on: 1; /* network diagnostic statistics on/off */
    unsigned reset_sens: 1; /* don't reset when this bit is set */
    unsigned reset_send: 1; /* reset when this bit is set */
    unsigned lpkl_test: 1; /* loopback test flag */
    unsigned lpkl_mode: 1; /* loopback mode on/off */
};

/* General purpose bits */

#define ELBIT 0x0800
#define EOFBIT 0x0800
#define SBIT 0x0400
#define IBIT 0x0200
#define CBIT 0x0800
#define BBIT 0x0400
#define OBKIT 0x2000

/* SCB patterns */

#define CX 0x0800
#define FR 0x0400
#define NMA 0x0200
#define RDD 0x0100
#define RESET 0x0080
#define CU_START 0x0100
#define RU_START 0x0010
#define RU_ABORT 0x0040
#define CU_MASK 0x0070
#define RU_MASK 0x0070
#define RU_READY 0x0040

/* 82586 Commands */

#define NOP 0x0000
#define IA 0x0001
#define CONFIGURE 0x0002
#define NEXT_SETUP 0x0003
#define TRANSMIT 0x0004
#define TDR 0x0005
#define DUMP 0x0006
#define DIAGNOSE 0x0007

231421-20

1-201
/**********************************************************
 * /PCD/USB/CHUCK/CSRC/DLD.h
 * / 82356 Command and Status Masks */
 * define CMD_MASK 0x0007
 * define MDCERRBIT 0x2000
 * define COLLMASK 0x000F
 * define DEFERMASK 0x0080
 * define MDCRSMASK 0x0400
 * define UNDERRUNMASK 0x0100
 * define SDENASK 0x0040
 * define MAXCOLMASK 0x0020
 * define OUT_OF_RESOURCES 0x0200
 */ Configure Parameters */
 * define FIFO_LIM 0x0800 /* use FIFO lim of 8 */
 * define BYTE_CNT 0x0008
 * define BRDY 0x0040
 * define SKY_SF 0x0080
 * define ADDR_LEN 0x0000 /* address length of 6 bytes */
 * define AC_LDC 0x0000 /* preamble length of 9 bytes */
 * define PREAM_LEN 0x2000 /* preamble length of 9 bytes */
 * define INT_LPBC 0x0000
 * define EXT_LPBC 0x0000
 * define LIN_PRIO 0x0000 /* no priority */
 * define ACR 0x0000
 * define BGF_MET 0x0000 /* IFB time 9 & usec */
 * define IFS 0x0000 /* slot time 12 usec */
 * define SLOT_TIME 0x0020 /* slot time 51.2 usec */
 * define RETRY_NUM 0x0000 /* retry number 15 */
 * define PHM 0x0000
 * define BC_DIS 0x0000
 * define MANCHESTER 0x0000
 * define TDMA_CRS 0x0000
 * define NCRC_REGS 0x0010
 * define CRC_16 0x0020
 * define BT_STUFF 0x0040
 * define PAD 0x0080
 * define CRSF 0x0000 /* no carrier sense filter */
 * define CRS_SRC 0x0000
 * define CDIF 0x0000 /* no collision detect filter */
 * define CDT_SRC 0x0000
 * define MIN_DATA_LEN_MIN_FRM_LEN 0x0040 /* 64 bytes */
 * define MIN_DATA_LEN_MIN_FRM_LEN 0x0040 /* 64 bytes */
 * define MIN_DATA_LEN_MIN_FRM_LEN 0x0040 /* 64 bytes */
 * define MIN_DATA_LEN_MIN_FRM_LEN 0x0040 /* 64 bytes */
 * define MAX_FRAME_SIZE 1500 - 3 /* assumes Ethernet/IEEE 802.3 frames with 6 bytes of address */
* define MAX_FRAME_SIZE 1500 - 3 /* assumes Ethernet/IEEE 802.3 frames with 6 bytes of address */
*/ Configure Parameters */
 */ define FIFO_LIM 0x0800 /* use FIFO lim of 8 */
 */ define BYTE_CNT 0x0008
 */ define BRDY 0x0040
 */ define SKY_SF 0x0080
 */ define ADDR_LEN 0x0000 /* address length of 6 bytes */
 */ define AC_LDC 0x0000 /* preamble length of 9 bytes */
 */ define PREAM_LEN 0x2000 /* preamble length of 9 bytes */
 */ define INT_LPBC 0x0000
 */ define EXT_LPBC 0x0000
 */ define LIN_PRIO 0x0000 /* no priority */
 */ define ACR 0x0000
 */ define BGF_MET 0x0000 /* IFB time 9 & usec */
 */ define IFS 0x0000 /* slot time 12 usec */
 */ define SLOT_TIME 0x0020 /* slot time 51.2 usec */
 */ define RETRY_NUM 0x0000 /* retry number 15 */
 */ define PHM 0x0000
 */ define BC_DIS 0x0000
 */ define MANCHESTER 0x0000
 */ define TDMA_CRS 0x0000
 */ define NCRC_REGS 0x0010
 */ define CRC_16 0x0020
 */ define BT_STUFF 0x0040
 */ define PAD 0x0080
 */ define CRSF 0x0000 /* no carrier sense filter */
 */ define CRS_SRC 0x0000
 */ define CDIF 0x0000 /* no collision detect filter */
 */ define CDT_SRC 0x0000
 */ define MIN_DATA_LEN_MIN_FRM_LEN 0x0040 /* 64 bytes */
 */ define MIN_DATA_LEN_MIN_FRM_LEN 0x0040 /* 64 bytes */
 */ define MIN_DATA_LEN_MIN_FRM_LEN 0x0040 /* 64 bytes */
 */ define MIN_DATA_LEN_MIN_FRM_LEN 0x0040 /* 64 bytes */
 */ define MAX_FRAME_SIZE 1500 - 3 /* assumes Ethernet/IEEE 802.3 frames with 6 bytes of address */
*/ define MAX_FRAME_SIZE 1500 - 3 /* assumes Ethernet/IEEE 802.3 frames with 6 bytes of address */
*/ Configure Parameters */
* define FIFO_LIM 0x0800 /* use FIFO lim of 8 */
* define BYTE_CNT 0x0008
* define BRDY 0x0040
* define SKY_SF 0x0080
* define ADDR_LEN 0x0000 /* address length of 6 bytes */
* define AC_LDC 0x0000 /* preamble length of 9 bytes */
* define PREAM_LEN 0x2000 /* preamble length of 9 bytes */
* define INT_LPBC 0x0000
* define EXT_LPBC 0x0000
* define LIN_PRIO 0x0000 /* no priority */
* define ACR 0x0000
* define BGF_MET 0x0000 /* IFB time 9 & usec */
* define IFS 0x0000 /* slot time 12 usec */
* define SLOT_TIME 0x0020 /* slot time 51.2 usec */
* define RETRY_NUM 0x0000 /* retry number 15 */
* define PHM 0x0000
* define BC_DIS 0x0000
* define MANCHESTER 0x0000
* define TDMA_CRS 0x0000
* define NCRC_REGS 0x0010
* define CRC_16 0x0020
* define BT_STUFF 0x0040
* define PAD 0x0080
* define CRSF 0x0000 /* no carrier sense filter */
* define CRS_SRC 0x0000
* define CDIF 0x0000 /* no collision detect filter */
* define CDT_SRC 0x0000
* define MIN_DATA_LEN_MIN_FRM_LEN 0x0040 /* 64 bytes */
* define MIN_DATA_LEN_MIN_FRM_LEN 0x0040 /* 64 bytes */
* define MIN_DATA_LEN_MIN_FRM_LEN 0x0040 /* 64 bytes */
* define MIN_DATA_LEN_MIN_FRM_LEN 0x0040 /* 64 bytes */
* define MAX_FRAME_SIZE 1500 - 3 /* assumes Ethernet/IEEE 802.3 frames with 6 bytes of address */
* define MAX_FRAME_SIZE 1500 - 3 /* assumes Ethernet/IEEE 802.3 frames with 6 bytes of address */
# Define constants for storage area

#define CB_CNT 8 /* Number of available Command Blocks */
#define FD_CNT 16 /* Number of available Frame Descriptors */
#define RBD_CNT 64 /* Number of available Receive Buffer descriptors */
#define TSB_CNT 16 /* Number of available Transmit Buffer descriptors */

/ * loopback parameters passed to Configure() */
#define INTERNAL_LOOPBACK 0x4000
#define EXTERNAL_LOOPBACK 0x8000
#define NO_LOOPBACK 0x0000

#include "dli.h" /* 586 Data Structures */

/ * IB timer Addresses */
#define TIMER1_CTL 0xFF5E
#define TIMER1_CNTL 0xFF5B
#define TIMER2_CTL 0xFF66
#define TIMER2_CNTL 0xFF60

/ * external functions */

/ * I/O */
int inw(); /* input word : inw(address) */
void outw(); /* output word : outw(address, value) */
void init_intv(); /* initialise the interrupt vector table */
void enable(); /* enable B0186 interrupts */
void disable(); /* disable B0186 interrupts */

extern char *Build_Ptr();
char *SEGHT; /* Data segment value */
char *pNULL; /* NULL pointer */

/ * Macro 'type' of definitions */
#define CA outw(0xC0B.0) /* the command to issue a Channel Attention */
#define ESI_LOOPBACK outw(0xCB.0) /* put the ESI in Loopback */
#define NO_ESI_LOOPBACK outw(0xCB.0) /* take the ESI out of Loopback */
#define ESI_0130 outb(0xE0, 0x63) /* End Of Interrupt */
#define TIMER1_EDI_B0186 outw(0xFF22, 0x04) /* EDI for Timer 1 on the 186 */
#define TIMER1_EDI_0130 outb(0xE0, 0x64) /* EDI for 1B6's Timer1 on the 130 */
int Self_Test;  /* used for diagnostic purposes */
unsigned long good_smnt_cnt;
unsigned long no_crs_cnt;
unsigned long sqe_err_cnt;
unsigned long max_col_cnt;
unsigned long recv_frame_cnt;
unsigned long reset_cnt;

#define LPBK_FRAME_SIZE 4  /* loopback frame storage */
char *lpbk_frame(LPBK_FRAME_SIZE) = (0x55, 0xAA, 0x55, 0xAA);

#define whoami_io_addr 0x0000  /* I/O address of Host Address Prom */
char *whoami[ADD_LEN];  /* Ram array where host address is stored */

/* transmission statistics variables */
unsigned long good_smnt_cnt;
unsigned long no_crs_cnt;
unsigned long sqe_err_cnt;
unsigned long max_col_cnt;
unsigned long recv_frame_cnt;
unsigned long reset_cnt;

/* Allocate storage for structures and buffers */
struct FLAGS flags;

/* 8086 structures */

/* System Configuration Pointer: Rom Initialization */
/* struct SCP scp = (0x0000, 0x0000, 0x0000, 0x0000, 0x0000); */

/* struct ISC isp: Intermediate System Configuration Pointer */
/* struct SBC sbc: */
/* System Control Block */
/* struct CB cb(CB_CNTJ, */
/* Command Blocks */
/* *cbtos, *begin_cbl, *end_cbl); */
/* pointer to the beginning of the free command block list (cb_tos) and the beginning and end of the 82586 cbl */

/* struct TBD tbld(TBD_CNTJ, */
/* Transmit Buffer Descriptor */
/* *tbd_tos); */
/* pointer to the free Transmit buffer descriptors */

/* struct TB tbuf(TBD_CNTJ); */
/* Transmit Buffers */

/* struct FD fd(FD_CNTJ, */
/* Frame Descriptors */
/* *begin_fd, *end_fd); */
/* pointers to the beginning and end of the free FD list */

/* struct RBD rbd(RBD_CNTJ); */
/* Receive Buffer Descriptors */
/PCD/UBR/CHUCK/CBRC/DLD.C

#define _IO (io) 

struct RBUFB (RBUF_CNT) { /* Receive Buffers */
    struct MAT mat [MULTI_ADDR_CNT]; /* Multicast Address Table */
    struct MA_CB ma_cb; /* Multicast Address Command Block */
}

/* The following structures are used only in Reset_8086() function */
struct CB res_CB; /* Temporary CB for reinitializing the 86 */
struct MA_CB res_MA_CB; /* Temporary MA_CB for reloading Multicast */

/* Hardware Support Functions */
Enable_8086_Int() { 
    int c;
    c = inb (0XFE); /* read the 80130 interrupt mask register */
    outb (0XEE, 0X00F7 & c) /* write to the 80130 interrupt mask register */
}
Disable_8086_Int() { 
    int c;
    c = inb (0XEE);
    outb (0XEE, 0X0008 & c);
}

Set_Timeout() { 
    outb (0XFE, 0X0000); /* Set Enable bit in Timer Mode/Control register */
}
Reset_Timeout() { 
    outb (0XFF3E, 0X6009); /* Reset Enable bit in Timer Mode/Control register */
}

Init_Timer() /* 86's Timer 2 is a prescaler for Timer 1. It clocks Timer 1 
    every 32.7 msec. The deadman timeout is set for 1.29 sec */
    { 
        outb (0XFF3B, 0X000C); /* Set Timer Interrupt Control register */
        outb (0XFF3C, 0XFFFE); /* set max count register for timer2 to 0FFFH */
        outb (0XFF5A, 3B); /* set max count register A for timer 1 */
        outb (0XFF66, 0XDD01); /* Set Timer2 Mode/Control register */
        outb (0XFF3E, 0X0009); /* Set Timer Mode/Control register */
        outb (0XFF2B, (inb (0XFF2B) & 0XFFE7)); /* Enable 86 Timer interrupt */
        outb (0XEE, (inb (0XEE) & 0X000F)); /* enable 80130 interrupt from 80186 */
    }
/* end hardware support functions */
Clear_CNT()
/CD/UBR/CHUCK/C5RC/DLC. C

<
    sbc.crc_errno = 0;    /* clear 586 error statistic counters */
    sbc.um_errno = 0;
    sbc.tsc_errno = 0;
    sbc.err_errno = 0;
    good_xmit_cnt = 0;    /* init data link statistics */
    underrun_cnt = 0;
    no_crs_cnt = 0;
    deferr_cnt = 0;
    sqe_err_cnt = 0;
    mas_col_cnt = 0;
    recv_frame_cnt = 0;
    reset_cnt = 0;
>
Init_586()
{
    struct ISCP *iscp;
    u_short i;
    struct MAT *mat;

    NO_EBI_LOOPBACK;    /* Done for 82501. Inactivates CRB if powered up
                       in loopback */
    EBI_LOOPBACK;
    init_intv();    /* Initialization DLDs interrupt vectors */
    Init_Timer();
    flags.reset_zmme = 0;    /* Initialize Reset Flags */
    flags.reset_pend = 0;
    flags.stat_on = 1;
    Disable_586_INT();

    iscp = 0x00000FF0;    /* Initialize the ISCP pointer */
    iscp->busy = 1;
    iscp->offset = Offset(bscb);
    iscp->base1 = SEVMT << 4;
    iscp->base2 = (SEVMT >> 12) & 0x000F;
    pNULL = Build_Ptr(NULL);    /* build a NULL pointer - 8086 type: 32 bits */
    Build_RFa();    /* init Receive Frame Area */
    Build_Ck();    /* init Command Block list */
    mac_ch.end = 0;    /* multicast address semaphore init */
    Clear_Cnt();
    sbc.stat = 0;

    CA;    /* wait for the 586 to complete initialization */

    for ( i = 0; i <= 0x0FF00; i++)
}
if (scb_stat == (CX : CNA))
    break;

if (i > OFFFF)
    Fatal("DLD: init - Did not get an interrupt after Reset/CA\n");

/* Ack the reset interrupt */
scb_cmd = (CX : CNA);
Wait_Scb();
Enable_586_Int();

if (i < 200)
    Fatal("DLD: in10" - Did not get an int.

/* Initialize the Multicast Address Table */
for (i = 0; i < ADD_LEN; i++)
    whoami(ADD_LEN - 1) - i = inb(whoami_io_addr + i*2);

Configure(INTERNAL_LOOPBACK); /* Put 586 in internal loopback */
SetAddress(); /* Set up the station address */
/* run diagnostics */
Test_Link();

if (Self_Test != PASSED)
    return(Self_Test);

Configure(NO_LOOPBACK); /* Configure the 82586 */
return(Self_Test);

Build_Fd()
{
    struct FD *pf;
    struct RBD *prbd;
    unsigned long badd;

    /* Build a linear linked frame descriptor list */
    for (pf = &fd[0]; pf <= &fd[FD_CNT - 1]; pf++) {
        pf->stat = pf->el_e = 0;
        pf->link = Offset(pf+1);
        pf->rd_a_offset = NULL;
    }
}
/PCD/USB/CHUCK/CBR/C/DLD.C

/* point to &fd(FD_CNT - 1) */
end_fd = --pfd;
pfd->link = NULL; /* last fd link is NULL */
pfd->el = ELBIT; /* last fd has EL bit set */
begin_fd = pfd = &fd[0]; /* point to first fd */
pfd->fd_offset = Offset(&rbd[0]); /* link first fd to first rbd */

/* Build a linear linked receive buffer descriptor list */
for (prrd = &rbd[0], pbuf = &buf[0], prbd = &rbd[RBD_CNT - 1], prbd++, pbuf++) {
    badd = SEGMT << 4;
badd += Offset(pbuf);
    prbd->offset = badd;
    prbd->buf = badd >> 16;
    prbd->buf_size = pbuf;
    prbd->act_cnt = 0;
    prbd->link = Offset(prbd + 1);
    prbd->size = RBUF_SIZE;
}

/* Build a stack of free command blocks */

struct CB *pcb;
struct TB *ptbd;
struct TB *pbuf;
unsigned long badd;

for (pcb = &cb[0], pcb <= &cb[CB_CNT - 1], pcb++) {
   pcb->stat = 0;
    pcb->cmd = ELBIT;
    pcb->link = Offset(pcb + 1);
}

/* Build a stack of transmit buffer descriptors */
for (ptbd = &tbd[0], pbuf = &tbuf[0], ptbd = &tbd[TBD_CNT - 1], ptbd++, pbuf++) {
    ptbd->act_cnt = TBUF_SIZE;
    ptbd->link = Offset(ptbd + 1);
    badd = SEGMT << 4;
}
```c
badd = Offset(pbuf);
ptbd->buff_l = badd;
ptbd->buff_h = badd >> 16;
ptbd->buff_ptr = pbuf;

/* ptbd->link = NULL; /* last tbd link is NULL */
tbd_tos = &tbd[0]; /* Set the Top Of the Stack */

/* Get a Command Block from the free list */
struct CB *Get_Cb() /* return a pointer to a free command block */
{
    struct CB *pcb;
    if (Offset(pcb = cb_tos) == NULL)
        return(pNULL);
    cb_tos = (struct CB *) Build_Ptr(pcb->link);
    pcb->link = NULL;
    return(pcb);
}

/* Put a Command Block back onto the free list */
Put_Cb(pcb)
{
    struct CB *pcb;
    pcb->stat = 0;
    pcb->link = Offset(cb_tos);
    cb_tos = pcb;
}

struct TBD *Get_Tbd() /* return a pointer to a free transmit buffer descriptor */
{
    struct TBD *ptbd;
    flags.reset_sema = 1;
    Disable_SBD_Int();
    if ((ptbd = tbd_tos) != pNULL) {
        tbd_tos = (struct TBD *) Build_Ptr(ptbd->link);
        ptbd->link = NULL;
    }
    Enable_SBD_Int();
    flags.reset_sema = 0;
    if (flags.reset_send == 1)
        Reset_SBD();
    return(ptbd);
}

Put_Tbd(ptbd)
```
/PCO/UBR/CHUCK/CRBC/DLD.C

struct TBD *ptbd;
{
  struct TBD *p;
  // find the end of the tbd list returned. ptbd is the beginning */
  for (p = ptbd; p->link != NULL; p = (struct TBD *) Build_Ptr(p->link)) ;
  p->act_cnt = TBUF_SIZE;     /* clear EOFBIT and update size on last tbd */
  p->link = Offset(tbd_topt);
  tbd_topt = ptbd;
}

SetAddress()
{
  struct CB *pcb;
  #ifdef DEBUG
  if (!(pcb = Get_CB()) == pNULL)
    Fatal("dlc: GetAddress - couldn't get a CB
  
  else
    pcb = Get_CB();
  #endif /* DEBUG */
  bcopy((char *)&pcb->parm1, &whoami[0], ADD_LEN);  /* move the prom
    address to IA cmd */
  pcb->cmd = IA : ELBIT;
  Issue_CU_Cmd(pcb);
}

Wait_Scb()  /* wait for the scb command word to be clear */
{ u_short i, stat;
  for (stat = FALSE, stat = FALSE, ) {  
    for (i = 0; i < 0xFFFOO, i++)
      if (scb.cmd == 0)
        break;
    if (i > 0xFFFOO) {
      Bug("DLD: Scb command not clear\n"");
      CA;
    } else
      stat = TRUE;
}
Issue_CU_Cmd pcb) /* Queue up a command and issue a start CU command if no other commands are queued */
{
    struct CB *pcb;
    Disable_586_Int();
    if (begin_cbl == pNULL) { /* if the list is inactive start CU */
        begin_cbl = end_cbl = pcb;
        scb.cbl_offset = Offset(pcb);
        WaitForScb();
        scb.cmd = CU_START;
        Set_Timeout(); /* set deadman timer for CU */
        CA;
    } else {
        end_cbl->link = Offset(pcb);
        end_cbl = pcb;
    }
    Enable_586_Int();
}

Isr7() {
    outb(OxEO, Ox67); /* EDI B0130 */
}

Isr6() {
    Write("\nInterrupt \n\n");
    outb(OxEO, Ox66); /* EDI B0130 */
}

Isr5() {
    Write("\nInterrupt 5\n");
    outb(OxEO, Ox65); /* EDI B0130 */
}

/* Deadman Timer Interrupt Service Routine */
Isr_Timeout() /* Interrupt 4 */
{
    Reset_Timeout();
    if (flags.reset_sema == 1)
        flags.reset_pend = 1;
    else
        Reset_586();
    TIMER1_EOI_B0186;
    TIMER1_EOI_B0130;
}

/* Interrupt 0 is Uart in UAP Module */
/* Interrupt 2 is Timer in UAP Module */
ISR1()
{
  Write("\nInterrupt 1\n");
  outb(0x80, 0x61);  /* EDL 80130 */
}

/* 586 Interrupt service routine; Interrupt 3 */
ISR_586()
{
  u_short  stat_scb;
  struct CB *pcb;
  enable();  /* nesting only the uart interrupt */
  Wait.Cmd();
  scb.Cmd = (stat_scb = scb.stat) & (CX | CNA | FR | RNR);
  CA;
  if (stat_scb & (FR | RNR))
    Rcv_IntProcessing();
  if (stat_scb & CNA)  /* end of cb processing */
    Reset_Timeout();  /* clear deadman timer */
    pcb = Build_Ptr(scb cb1_offset);
#include DEBUG
  if (begin_cbl == pNULL)
    Bug("DIL: begin_cbl == NULL in interrupt routine\n");
    return;
  }
  if ((pcb->stat & OxCOOG) != Ox8000)
    Fatal("DIL: C bit not set or B bit set in interrupt routine\n");
#include DEBUG #
  switch (pcb->cmd & CMD_MASK) {
    case TRANSMIT:
      if (flags.stat_on == 1)  /* if Transmit Statistics are collected do */
        /* if sqe bit = 0 and there were no collisions -> sqe error
            this condition will occur on the first transmission if
            there were no collisions, or if the previous transmit
            command reached the max collision count, and the current
            transmission had no collisions */
      if ((pcb->stat & (BEMASK | MAXCOLMASK | COLLMASK)) == 0)
        sqe_err_cnt;
      if (pcb->stat & DEFERMASK)
        defer_cnt;
}
if (pcb->stat & NOERRBIT)
  ++good_start_cnt;
else {
  if (pcb->stat & NOCRBMASK)
    ++nocnt;
  if (pcb->stat & UNDERRUNMASK)
    ++underrun_cnt;
  if (pcb->stat & MAXCOLMASK)
    ++max_col_cnt;
}
if (pcb->parm != NULL)
  Put_Tbl(Build_Ptr(pcb->parm));
  break;

  case DIAGNOSE:
    flags.diag_done = 1;
    if ((pcb->stat & NOERRBIT) == 0)
      Self_Test = FAILED_DIAGNOSE;
      break;

    default:
      /* check to see if another command is queued */
    if (pcb->link == NULL)
      begin_cbl = pcb;
    else /* restart the CU and execute the next command on the cbl */
      begin_cbl = Build_Ptr(pcb->link);
    sb_cbl_offset = pcb->link;
    Wait_Scb();
    sb_cbl = CU_START;
    CA;
    Wait_Scb();
    Set_Timeout(); /* Start deadman timer */
}
if ((pcb->cmd & CMD_MASK) == MC_SETUP)
  pcb->cmd = 0; /* clear MC_SETUP cmd word; this will implement a
  lock semaphore so that it won't be reused until
  it is completed */
else
  Put_Cb(pcb); /* Don't return MC_SETUP cmd block. It's not a
  general purpose command block from free CB list */
disable(); /* disable cpu int so that the 586 isr will not nest */
ED1_B0130;
typedef struct FD TFD;
#define FD_LINK_T FLD

struct FD

struct FD* TFD;

/* points to the Frame Descriptor */

struct RBD* TRBD;

/* points to the first rbd for the frame */

for (FD = begin_fd; FD != pNULL; FD = begin_fd)

if (FD->stat & CBIT) {

begin_fd = (struct FD*) Build_Ptr(FD->link);

pbd = (struct RBD*) Build_Ptr(FD->pbd_offset);

if (pbd != pNULL) { /* check to see if a buffer is attached */

if (pbd != begin_rbd)

endif /* DEBUG */

for (q = pbd; (q->act_cnt & EOFBIT) != EOFBIT;

q = (struct RBD*) Build_Ptr(q->link));

begin_rbd = (struct RBD*) Build_Ptr(q->link);

q->link = NULL;

if (FD->stat & OUT_OF_RESOURCES)

Put_Free_RFA(FD);

else {

/* if the DLD is in a loopback test, check the frame recv */

if (FD->loopback_mode == 1)

Loopback_Check(FD);

else

/* if it's a multicast address check to see if it's */

/* in the multicast address table, if not discard the frame */

if (((FD->dest_addr[0] & 01) == 01) && (!Check_Multicast(FD)))

Put_Free_RFA(FD);

else

Recv_Frame(FD),

++recv_frame_cnt;

}

else {

Ru_Start(); /* If RU has gone into no resources, restart it */

break;

}

Loopback_Check(FD) /* Called by Recv_Int_Processing; checks address */

and data of potential loopback frame */

struct FD* TF;
if ( bcmp((char *)&spfd->src_addr[0], &whoami[0].ADD_LEN) != 0 ) {  
  Put_Free_RFA(pfd);
  return;
}

pbuf = (struct RBD *) Build_Ptr((pfd->rbd_offset); /* point to receive buffer descriptor */  

if ( bcmp((char *)&pbuf, &lpbk_frame[0].LPBK_FRAME_SIZE) != 0 ) {  
  Put_Free_RFA(pfd);
  return;
}

flags.lpbk_test = 1; /* passed loopback test */  

Check_Multicast(pfd) /* returns true if multicast address is in MAT */ {  
  struct FD *pfd;
  struct MAT *pmat;

  for (pmat = &mat[0]; pmat <= &mat[MULTI_ADDR_CNT - 1]; pmat++)
    if ( (pmat->stat == INUSE && (bcmp((char *)&spfd->dest_addr[0], &pmat->addr[0].ADD_LEN) == 0))
      break;

  if (pmat > &mat[MULTI_ADDR_CNT - 1])
    return(FALSE);
  return(TRUE);
}

/* Test the Link Function: executes Diagnose and Loopback tests */
Test_Link() {  
  Self_Test = PASSED;
  Diagnose();
  if (Self_Test == FAILED_DIAGNOSE)
    return;
  Ru_Start(); /* start up the Ru for loopback tests */

  flags.lpbk_mode = 1; /* go into loopback mode */
  flags.lpbk_test = 0; /* set loopback test to false */

  if (flags.lpbk_test == 0) {  
    if (flags.lpbk_mode == 0) {  
      Self_Test = FAILED_LPBK_INTERNAL;
      flags.lpbk_mode = 0;
      return;
    }
  }

  if (flags.lpbk_test == 0) {  
    Configure(INTERNAL_LOOPBACK); /* external loopback test w/ ESI in lpbk */
    Send_Lpbk_Frame();
    if (flags.lpbk_test == 0) {  
      Self_Test = FAILED_LPBK_INTERNAL;
    }
  }  

flags.ipbk_mode = 0;
return;
}
if (flags.ipbk_test == 0) {
    Self_Test = FAILED_LP8K_TRANSCIEVER;
}
flags.ipbk_mode = 0; /* leave loopback mode */

Send_Lp8k_Frame()
{
    struct TBD *ptbd;
    int i;
    for (i = 0; i < B; i++) { /* send lp8k frame B times, since it's
                               best effort delivery */
        ifdef DEBUG
            if ((ptbd = Get_Tbd()) == pNULL)
                Fatal("did - Send_Lp8k_Frame - couldn't get a TBD
        
        else
            ptbd = Get_Tbd();
        endif /*DEBUG*/
        ptbd->act_cnt = EOFBIT | LP8K_FRAME_SIZE;
        bcopy((char *)ptbd->buff_ptr, &lpk_frame[0], LP8K_FRAME_SIZE);
        while (!Send_Frame(ptbd, &whoami[0]));
        
        Diagnose()
        {
            struct CB *pcb;
        ifdef DEBUG
            if ((pcb = Get_CB()) == pNULL)
                Fatal("diag - Diagnose - couldn't get a CB
        
            else
                pcb = Get_CB();
        endif /*DEBUG*/
            flags.diag_done = 0;
            Self_Test = FALSE;
            pcb->cmd = DIAGNOSE | ELBIT;
            Issue_CU_Cmd(pcb);
            while (flags.diag_done == 0); /* wait for Diag cmd to finish */
Configure(loopflag)

u_short loopflag;

if (DEBUG)
  if (pcb = Get_CB()) == pNULL)
    Fatal("conflu - Configure - couldn't get a CB\n");
else
  pcb = Get_CB();
endif // DEBUG /

#elif DEBUG /

  /* Ethernet default parameters */
  pcb->param1 = 0x0000;
  pcb->param2 = 0x2500 ; loopflag;
  pcb->param3 = 0x6000;
  pcb->param4 = 0x2000;
  pcb->param5 = 0x0000;
  if (loopflag == NO_LOOPBACK)
    pcb->param6 = 0x0040;
  else
    pcb->param6 = 0x0000; /* loopback frame is less bytes than
    the minimum frame length */
  pcb->cmd = CONFIGURE ! EBit;

  Issue_CU_Cmd(pcb);

  /* Send a frame to the cable, pass a pointer to the destination address
  and a pointer to the first transmit buffer descriptor. */
  Send_Frame(std, padd); /* return false if it can't get a Command block */

  struct TBD *ptbd;
  char *padd;

  struct CB *pcb;

  u_short length;

  flags.reset same = 1;
  if (pcb = Get_CB()) == pNULL)
    if (flags.reset same = 0;
      if (flags.reset_send = 1)
        Reset_BBB();
      return(FALSE);

  pcb->param1 = Offset(ptbd);
/PCO/USB/CHUCK/CSRC/DLD.C

/* move destination address to command block */
bcopy((char *)bpbd->parm2, (char *)padd, ADD_LEN);

/* calculate the length field by summing up all the buffers */
for (length = 0; ptbd->link != NULL; ptbd = Build_Ptr(ptbd->link))
    length += ptbd->act_cnt;

/* add the last buffer */
/* check to see if padding is required, do not do padding on loopback */
/* this will not work if MIN_DATA_LEN > TBUFSIZE */
if (length < MIN_DATA_LEN) & /* assumes a 4 byte CRC */
    (bcmp(&whoami[0], (char *)padd, ADD_LEN) != 0))
    ptbd->act_cnt = MIN_DATA_LEN;  EDI BIT;
pcb->parm5 = length;  /* length field */

pcb->cmd = TRANSMIT | ELBIT;

Issue_CU_Cmd(pcb);
flags.reset_sema = 0;
if (flags.reset_pend == 1)
    Reset_RB64();
return(TRUE);

Add_Multicast_Address(pma) /* pma - pointer to multicast address */
char *pma; /* returning false means the Multicast address table is full */
{
    struct MAT *pmat;
    flags.reset_sema = 1;
    /* if the multicast address is a duplicate of one already in the MAT, 
    then return */
    for (pmat = mat; pmat <= &mat[MULTI_ADDR_CNT - 1]; pmat++)
        if (pmat->stat == INUSE &
            (bcmp(&pmat->addr[0], (char *)pma, ADD_LEN) == 0))
            return(TRUE);
    
    for (pmat = mat; pmat <= &mat[MULTI_ADDR_CNT - 1]; pmat++)
        if (pmat->stat == FREE) {
            pmat->stat = INUSE;
            bcopy(&pmat->addr[0], (char *)pma, ADD_LEN);
            break;
        }
/PCG/USR/CHUCK/CSR/MDL.C

if (pmat > &mat[MULTI_ADDR_CNT - 1]) {
  flags.reset_same = 0;
  if (flags.reset_pend == 1)
    Reset_586();
  return(FALSE);
}

Set_Multicast_Address();
flags.reset_same = 0;
if (flags.reset_pend == 1)
  Reset_586();
return(TRUE);

Delete_Multicast_Address(pma) /° returning false means the multicast address
was not found #/

char *pma;
{
  struct MAT *pmat;
  flags.reset_same = 1;
  if (pmat = mat; pmat <= &mat[MULTI_ADDR_CNT - 1]; pmat++)
    if (pmat->stat == INUSE &&
        (bcmp(pmat->addr0, (char *) pma, ADD_LEN) == 0)) {
      pmat->stat = FREE;
      break;
    }

if (pmat > &mat[MULTI_ADDR_CNT - 1]) {
  flags.reset_same = 0;
  if (flags.reset_pend == 1)
    Reset_586();
  return(FALSE);
}

Set_Multicast_Address();
flags.reset_same = 0;
if (flags.reset_pend == 1)
  Reset_586();
return(TRUE);

Set_Multicast_Address()
{
  struct MAT *pmat;
  struct MA_CB *pma_cb;
  u_short i;
  if (pmat = mat;
      while (pma_cb->cmd != 0) ; /* if the MA_CB is inuse, wait until it's free */
      pma_cb->xlink = NULL;

231421-38
for (pamat = mat; pamat < &mat[MULTI_ADDR_CNT - 1]; pamat++)
    if (pamat->stat == INUSE) {
        bcopy(&pama_cb->mc_addr[1], &pamat->addr[0], ADD_LEN);
        i += ADD_LEN;
    }
pma_cb->mc_cnt = i;
pma_cb->cmd = MC_SETUP | ELBIT;
    Issue_CU_Cmd(pma_cb);

Put_Free_RFA(pfd) // Return Frame Descriptor and Receive Buffer
    Descriptors to the Free Receive Frame Area */
{
    struct FD *pfd;
    struct RBD *prbd; // points to beginning of returned RBD list */
    char *ru_start_flag_fd, /* indicates whether to restart RU */
         *ru_start_flag_rbd;
    flags.reset_sema = 1;
    ru_start_flag_fd = ru_start_flag_rbd = FALSE;
    pfd->el_s = ELBIT;
    pfd->stat = 0;
    prbd = (struct RBD *) Build_Ptr(pfd->rbd_offset); /* pick up the link to the rbd */
    pfd->link = pfd->rbd_offset = NULL;
    /* Disable_586_Int(); this command is only necessary in a multitasking
program. However in this single task environment this routine is originally
called from isr_586(). therefore interrupts are already disabled */

    if (begin_fd == pNULL)
        begin_fd = end_fd = pfd;
    else {
        end_fd->link = Offset(pfd);
        end_fd->el_s = 0;
        end_fd = pfd;
        ru_start_flag_fd = TRUE;
    }
    if (prbd == pNULL) {
        /* if there is a rbd attached to the fd then
find the beginning and end of the rbd list */

        for (q = prbd; q->link != pNULL; q = Build_Ptr(q->link))
            q->act_cnt = 0;
        /* now prbd points to the beginning of the rbd list and
q points to the end of the list */
        q->size = RBUF_SIZE | ELBIT;
        q->act_cnt = 0;
    }
if (begin_rbd == pNull) { /* if there is nothing on the list create a new list */
    begin_rbd = prbd;
    end_rbd = q;
    if (prbd == q) ru_start_flag_rbd = TRUE; /* if there is more than one rbd returned start the RU */
} else { /* if the rbd list already exists add on to the new returned rbd */
    end_rbd->link = Offset(prbd);
    end_rbd->size = RBUF_SIZE;
    end_rbd = q;
    ru_start_flag_rbd = TRUE;
}
if (ru_start_flag_fd && ru_start_flag_rbd)
    Ru_Start();
/* Enable_56b_Int() if Disable_56b_Int() is used above */
flags.reset_sema = 0;
if (flags.reset_end == 1)
    Reset_56b();
}
Ru_Start()
{
    if ((sch.stat & RU_MASK) == RU_READY) /* if the RU is already 'ready' */
        return;
    if ((begin_fd->stat & CBIT) == CBIT)
        return;
    begin_fd->rbd_offset = Offset(begin_rbd); /* link the beginning of the rbd list to the first fd */
    sch.rfa_offset = Offset(begin_fd);
    Wait_Scb();
    sch.cmd = RU_START;
    CA;
}
Software_Reset()
{
    sch.cmd = RESET;
    CA;
    Wait_Scb();
}
Issue_Reset_Cmds()
{
    Wait_Scb();
    sch.cmd = CU_START;
    CA;
/PC0/USR/CHUCK/CBRC/DLD.C

Wait_Scb();

outb(0xFFFE, 0); /* shut off timer 1 interrupt */
outb(TIMER1_CNT, 0);
outb(0x0, 0); /* use timer 1 without interrupt as a deadman */
while (!(inb(0xFFFE) & 0x0020) == 0) /* if Max Cnt bit is set before CNA
is set, 586 Cnd deadlocked */
if ((scb.stat & CNA) == CNA)
  break;
if (scb.stat & CNA == CNA)
  Fatal("DLD: Issue_Reset_Cmds — Command deadlock during reset procedure\n");
Reset_Timeout();
scb.cmd = CNA; /* Acknowledge CNA interrupt */
CA;
Wait_Scb();

/* Execute a reset. Configure, SetAddress, and MC_Setup, then restart the
Receive Unit and the Command Unit */
Reset_586()
{
  struct MAT *pmat;
  u_short j;
  +reset_cnt;
  Disable_586_Int();
  ESI_LOOPBACK;
  Software_Reset();
  scb.stat = 0;
  CA; /* wait for the 586 to complete initialization */
  for (i = 0; i <= 0xFFFO; i++)
    if (scb.stat == (CX & CNA))
      break;

  if (i > 0xFFFO)
    Fatal("DLD: init — Did not get an interrupt after Software Reset\n");
  /* Ack the reset interrupt */
  Wait_Scb();
  scb.cmd = (CX & CNA);
  CA;
  Wait_Scb();

  ifdef DEBUG
      if (begin_cbl == pNULL)
        Fatal("DLD: begin_cbl = NULL in Reset_586");
  endif /* DEBUG */

231421-41
/* Configure the SGB */
/* Ethernet default parameters: Configure is not necessary when using
default parameters */
res_cb.link = NULL;
res_cb.param1 = 0x0080;
res_cb.param2 = 0x2600;
res_cb.param3 = 0x6000;
res_cb.param4 = 0xFFF00;
res_cb.param5 = 0x0000;
res_cb.param6 = 0x0040;
res_cb.cmd = CONFIGURE | ELBIT;

scl_cbl.offset = Offset(bres_cb.stat);

Issue_Reset.Cmds();

/* Set the Individual Address */
bcopy((char *)&bres_cb.param1, &whoami[0], ADD_LEN); /* move the prom
address to IA cmd */
res_cb.cmd = IA | ELBIT;

Issue_Reset.Cmds();

/* reload the multicast addresess */
i = res_ma.cb.stat = 0;
res_ma.cb.link = NULL;
for (pmat = &mat[0]; pmnt <= &mat[MULTI_ADDR_CNT - 1]; pmnt++)
    if (pmnt->stat == INUSE ) {
        bcopy(bres_ma.cb.mc_addr[i], pmnt->addr[0], ADD_LEN);
        i += ADD_LEN;
    }

res_ma.cb.mc_cnt = i;
res_ma.cb.cmd = NO_SETUP | ELBIT;
scl_cbl.offset = Offset(bres_ma.cb.stat);

Issue_Reset.Cmds();

/* Restart the Command Unit and the Receive Unit */
flags.reset_nema = 0;
flags.reset_nend = 0;
NO_ESI_LOOPBACK;

Recv_Unit_Processing();
scl_cbl.offset = begin_cbl;
Wait_Csb();
SCB_Cmd = CU_START;
Set_Timeout(); // Set Deadman Timer
CA:
Enable_SCB_Int();

/* bcopy — byte copy routine */
bcopy(dst, src, nbytes)
char *dst, *src;
int nbytes;
{
while (nbytes--) dst++ = src++;}

/* bcmp — byte compare */
bcmp(s1, s2, nbytes)
char *s1, *s2;
int nbytes;
{
while (nbytes-- && *s1++ == *s2++)
return(*--s1 - *--s2);}
/PCG/UBR/CHUCK/CSRC/LLC.C

/#include "dl.h"
extern char *pNULL;
extern struct TBD *Get_Tbd();
extern char *Build_Ptr();
readonly char xid_frame[XID_LENGTH] = { 0, 0, XID, 0s8i, 0s0i, 0};
/* DSAP, SSAP, XID, xid class 1 response */
struct LAT lat[DSAP_CNT];
Init_Llc()
 { struct LAT *plat;
   for (plat = &lat[0]; plat <= &lat[DSAP_CNT - 1]; plat++)
     plat->stat = FREE;
   return(Init_DB6());
 }
/* Function for adding a new DSAP */
Add_Dsap_Address(dsap, pfunc) /* DSAP must be divisible by 2**(8-N), where 
    2**N = DSAP_CNT. (i.e. N LSBs must be 0). 
    The function will return FALSE if does not 
    meet the above requirements, or the DSAP 
    Address Table is full, or the address has 
    already been used. NULL DSAP address is 
    reserved for the Station Component */
in dsap, (*pfunc) ();
 { struct LAT *plat;
   if ((dsap << (8-DSAP_SHIFT) & 0x0OFF) != 0 || dsap == 0)
     return(FALSE);
   /* Check for duplicate dsaps. */
   if ((plat = &lat[dsap >> DSAP_SHIFT]->stat == FREE) {
     plat->stat = INUSE;
     plat->p_add_func = pfunc;
     return(TRUE);
   } else
     return(FALSE);
 }
/* Function for deleting DSAPs */
Delete_Dsap_Address(dsap) /* If the specified connection exists, it is severed. 
     If the connection does not exist, the command is ignored. */

23123-44
/PCD/USB/CHUCK/CBRC/LLC.C

int dsap;
{
latt�ap >> DSAP_SHIFT).stat = FREE;
}

Recv_Frame(pfd)
{
struct FD *pfd;
struct RBD *prbd;
struct FRAME_STRUCT *pfs;
struct LAT *plat;

prbd = (struct RBD *) Build_Ptr(pfd->rbd_offset);
pfs = (struct FRAME_STRUCT *) prbd->buff_ptr;

if (pfd->rbd_offset != NULL) { /* There has to be a rbd attached
to the fd, or else the frame is
tooshort */

if (pfs->dsap == 0) { /* if the frame is addressed to the Station
Component, then a response may be required */

if ((pfs->dsap & C_R_BIT)) { /* if the frame received is a response,
instead of a command, then reject it.
Because this software does not implement
DUPLICATE_ADDRESS_CHECK, -> no response
frames should be recv'd */

Station_Component_Response(pfd);

} /* not addressed to Station Component */

else if ((pfs->dsap < (B-DSAP_SHIFT) & OR OFF) == 0 &
(plat = &lat((pfs->dsap) >> DSAP_SHIFT))->stat == INUSE) { /*
(plat->p_sap_func)(pfd); /* call the function
associated
with the dsap received */

} /* check to see if the dsap addressed is active */

}

}

Put_Free_RFAPfd); /* return the pfd if not given to the user saps */

Station_Component_Response(pfd)
{
struct FD *pfd;

struct FRAME_STRUCT *pfrs, *pfrs;
struct TBD *ptbd, *begin_ptbd, *eq;
struct RBD *prbd;

prbd = (struct RBD *) Build_Ptr(pfd->rbd_offset);
pfrs = (struct FRAME_STRUCT *) prbd->buff_ptr;

switch (pfrs->cmd & "P_F_BIT") {
case XID:
while((ptbd = Get_Tbd()) == pNULL);
ptbd->act_cnt = EOBBIT | XID_LENGTH;
scopy((char *)ptbd->buff_ptr, &sid_frame[0], XID_LENGTH);
ptfs = (struct FRAME_STRUCT *)ptbd->buff_ptr.
ptfs->cmd = ptfs->cmd;

while( ptbd != pNULL ) {
  while(!Send_Frame(ptbd, Build_Ptr(pfd->src_addr))); break;

  case TEST:
    for (prbd = (struct RBD *)Build_Ptr(prbd->fbd_offset),
         q = begin_ptbd = pNULL; prbd = pNULL;
         prbd = Build_Ptr(prbd->fbd_offset),
         while ((ptbd = Get_Tbd()) == pNULL);
    if (q == pNULL)
      q->link = Offset(ptbd),
    else
      begin_ptbd = ptbd;
    ptbd->act_cnt = prbd->act_cnt;
    bcopy((char *)ptbd->buff_ptr, (char *)prbd->buff_ptr,
         ptbd->act_cnt & 0x3FFF);
    ptfs = (struct FRAME_STRUCT *)begin_ptbd->buff_ptr.
    ptfs->cmd = ptfs->cmd;
    ptfs->ssap = prfs->ssap & C_R_BIT; /* return the frame to the sender */
    while(!Send_Frame(begin_ptbd, Build_Ptr(pfd->src_addr))); break;
  }
}
#include "dId.h"

/* ASCII Characters */
#define ESC 0x1B
#define LF 0x0A
#define CR 0x0D
#define BS 0x08
#define BEL 0x07
#define SP 0x20
#define DEL 0x7F
#define CTL_C 0x03

/* Hardware */
#define CH_B_CTL 0x00DE
#define CH_A_CTL 0x00DC
#define CH_B_DAT 0x00DA
#define CH_A_DAT 0x00DB
#define UART_STAT_MK 0x70

/* Interrupt cases for 8274 */
#define UART_TX_B 0
#define UART_RECV_B 0x08
#define UART_RECV_ERR_B 0x0C
#define EXT_STAT_INT_B 0x04
#define EXT_STAT_INT_A 0x14

char fifo_t[256];
char fifo_r[256];
char wrq[5], wrb[5];
unsigned char _in_fifo_t, out_fifo_t, in_fifo_r, out_fifo_r, actual;
unsigned char t_buf_stat, r_buf_stat;
char cbuf[80]; /* Command line buffer */
char line[81]; /* Monitor Mode display line */
unsigned char dsep, ssep, send_flag, local_echo;
char Dest_Addr[ADD_LEN];
char Multi_Addr[ADD_LEN];

int tmstat, /* terminal mode status, for leaving terminal mode */
    int dtes, monitor_flag, hs_stat, /* flags */
    extern struct TBD *Get_Tbd();
extern char *Build_Ptr();
extern struct FLAGS *flags;
extern char sid_frame();
extern char whoami();
extern struct MAT mat[3];
extern struct LAT lat[3];
extern char *pNULL;

extern unsigned long good_smnt_cnt;
extern u_short underrun_cnt;
extern u_short no_cra_cnt;
extern unsigned long defer_cnt;
extern u_short sqe_err_cnt;
extern u_short mac_cel_cnt;
extern unsigned long recv_frame_cnt;
extern u_short reset_cnt;

extern struct BCB sb;

/*Macro 'type' of definitions*/
#define RTS_ONB outb(CH_B_CTL,0x05); outb(CH_B_CTL,wrb[5]=wrb[5]!0x02)
#define RTS_OFFB outb(CH_B_CTL,0x00); outb(CH_B_CTL,wrb[5]=wrb[5]!0x0F)
#define RTS_ONA outb(CH_A_CTL,0x05); outb(CH_A_CTL,wrb[5]=wrb[5]!0x02)
#define RTS_OFFA outb(CH_A_CTL,0x00); outb(CH_A_CTL,wrb[5]=wrb[5]!0x0F)
#define UART_TX_DI_B outb(CH_B_CTL,0x01); outb(CH_B_CTL,wrb[1]=wrb[5]!0x02)
#define UART_TX_EI_B outb(CH_B_CTL,0x01); outb(CH_B_CTL,wrb[1]=wrb[5]!0x10)
#define UART_RX_DI_B outb(CH_B_CTL,0x01); outb(CH_B_CTL,wrb[1]=wrb[5]!0x02)
#define UART_RX_EI_B outb(CH_B_CTL,0x01); outb(CH_B_CTL,wrb[1]=wrb[5]!0x10)
#define RESET_TX_INT outb(CH_B_CTL,0x20)
#define EDI_0874 outb(CH_A_CTL,0x20) /* 8274 int is IR3 on B0130 */
#define EDI_08130_08274 outb(0xEO,0x60)
#define EDI_08130_TIMER outb (0xEO,0x62)

Enable_Uart_Int()
{
    int c;
    c = inb(0xE2); /* read the 80130 interrupt mask register */
    outb(0xE2, 0x00FE & c); /* write to the 80130 interrupt mask register */
}

Disable_Uart_Int()
{
    int c;
    c = inb(0xE2);
    outb(0xE2, 0x0001 | c);
}

Enable_Timer_Int()
{
    int c;
    outb(0xEA, 0x25);
    outb(0xEA, 0x00); /* Timer1 interrupts every .125 sec */
    send_flag = FALSE;
    c = inb(0xE2); /* read the 80130 interrupt mask register */
    outb(0xE2, 0x00FB & c); /* write to the 80130 interrupt mask register */
}
```c
/PCD/USR/CHUCK/CSRC/UAP.C

Disable_Timer_Int()
{
    int c;
    c = inb(0x0E2);
    outb(0x0E2, 0x0004 | c);
}

Co(c)
char c;
{
    while ((inb(CH_B_CTL) & 4) == 0);
    outb(CH_B_DAT, c);
}

Ci()
{
    while ((inb(CH_B_CTL) & 1) == 0);
    return(inb(CH_B_DAT) & 0x7F));
}

Read(msg, cnt, pact)
char *msg;
unsigned char cnt, *pact;
{
    unsigned char i;
    char c, buf[2001];
    for (i = c = 0; (c = Ci) & (c != LF) & (i < 19B);)
    {
        c = Ci() & 0x7F;
        if (c == BS || c == DEL) {
            if (i > 0) {
                Co(BS); Co(BP); Co(BB);
            } else {
                if (c == BS) Co(c);
                buf[i++] = c;
            } else {
                if ((c == CR) || (c == LF)) {
                    buf[i++] = CR;
                    buf[i++] = LF;
                } else Co(BEL);
            }
        } else {
            if (i > cnt) *pact = cnt;
            else *pact = i;
            for (i = 0, i < *pact; i++)
                *msg++ = buf[i];
```
/PCG/USR/CHUCK/CBRC/UAP.c

> Read_Char()
> {
>   unsigned char ch;
>   Read(cbdbuf[O], &ch, &actual);
>   i = Skip(cbdbuf[O]);
>   return(cbdbuf[i]);
> }

> Write(msg)
> {
>   char *msg;
>   while (*msg != '\0') {
>     if (*msg == '\n')
>       Co(CR);
>     Co(*msg++);
>   }
>
> Fatal(msg) /* write a message to the screen then stop */
> char *msg;
> {
>   Write("Fatal: ");
>   Write(msg);
>   for(i);
> }

Bug(msg) /* write a message to the screen then continue */
char *msg;
{
   Write("Bug: ");
   Write(msg);
}

Ascii_To_Char(c) /* convert ASCII-Hex to Char */
char c;
{
   if ('0' <= c && (c <= '9'))
     return(c - '0');
   if ('A' <= c && (c <= 'F'))
     return(c - 0037);
   if ('a' <= c && (c <= 'f'))
     return(c - 0037);
   return(0xFF);
}

Lower_Case(c)
char c;
{
   if ('A' <= c && (c <= 'Z'))
     return(c);
   if ('a' <= c && (c <= 'z'))
     return(c + 0020);
   return(c);
}
# /PCO/USR/CHUCK/CSRC/UAP.C

```c
Char_To_Ascii(c, ch) /* convert char to ASCII-Hex */
unsigned char  c, ch[3];
{
  unsigned char  i;
  i = (c & 0xFF) >> 4;
  if (i < 10)
    ch[0] = i + 0x30;
  else
    ch[0] = i + 0x37;
  i = (c & 0xFF);  
  if (i < 10)
    ch[1] = i + 0x30;
  else
    ch[1] = i + 0x37;
  ch[2] = '\0';
}

Skip(pmsg) /* skip blanks */
char *pmsg;
{
  int  i;
  for (i = 0; *pmsg == ' '; i++, pmsg++)
    return(i);
}

Read_Int() /* Read a 16 bit integer */
{
  u_short  ud, wh, wd1, wh1, j;
  char    i, done, has, dover, hover;
  for (done = FALSE; done == FALSE; ) {
    Read(&cbuf[0]);
    i = Skip(&cbuf[0]);
    for (hex = dover = hover = FALSE, ud = wh = wd1 = wh1 = 0;
         (j = Ascii_To_Char(cbuf[i])) <= 15, j++) {
      if (j > 9)
        has = TRUE;
      ud = ud*10 + j;
      wh = wh*16 + j;
      if (ud < ud1)
        dover = TRUE;
      if (wh < wh1)
        hover = TRUE;
      wd1 = ud, wh1 = wh;
    }
    if (cbuf[i] == 'H' || cbuf[i] == 'h' || cbuf[i] == CR || cbuf[i] == LF || cbuf[i] == '/') {
      if (cbuf[i] == 'H' || cbuf[i] == 'h')
        hex = TRUE;
      if (hex == TRUE && hover == FALSE)
        done = TRUE;
      if (hex == FALSE && dover == FALSE)
        done = TRUE;
```
# /PCD/USR/CHUCK/CBRC/UAP.C

```c
if (!done) {
    Write("\nThis number is too big.\nIt has to be less than 65536.\n\n", Enter number --> "");
} else
    Write("Illegal Character\nEnter a number -->" );
```

```c
if (hex)
    return(wh);
return(ud);
```

```c
Int_To_Ascii (value, base, ld, ch, width) /* convert an integer to an ASCII string */
unsigned long value;
short char ch[1], ld;
short i, j;
for (i = 0; i < width; i++) {
    j = value % base;
    if (j < 10) ch[i] = j + 0x30;
    else ch[i] = j + 0x37;
    value = value / base;
} for (i = width - 1; ch[i] == '0' & i > 0; i--)
    ch[i] = ld;
ch[width] = '\'0';
```

```c
Write_Long_Int (dw, i)
unsigned long dw;
short i;
short j;
char ch[13];
short j;
char ch[11];
if (hex) Int_To_Ascii (dw, 16, \"\", &ch[0], 8);
else Int_To_Ascii (dw, 10, \"\", &ch[0], 10);
for (j = 0; ch[j] != '\0'; i++, j++)
    line[i] = ch[j];
```

```c
Write_Short_Int (w, l)
short w, i;
short j;
char ch[6];
unsigned long dw;
dw = w;
if (hex)
    Int_To_Ascii (dw, 16, \"\", &ch[0], 4);
else ...
```

---

1-233
/PC0/USR/CHUCK/CBRC/UAP.C

Int_To_Ascii(dw, 10, 'O', &ch[0]; 5);
for (j = 0; ch[j] != 'O'; i++, j++)
line[i] = ch[j];

else 0.

Read_Ascii(pmsg, add, cnt)  /* pmsg - pointer to the output message */
# add - pointer to the address */
# cnt - number of bytes in the address */
char *pmsg, *add[], cnt;

cnt b;  
for (i = 0; 4 i < 2*cnt; i++, j++) <
if (b = Read_Char();  
if ((b == 'Y') || (b == 'y'))
return(TRUE);  
if ((b == 'N') || (b == 'n'))
return(FALSE);  
Write("Enter a Y or N --> ");

Read.Addr(pmsg, add, cnt)  /* pmsg - pointer to the output message */
/* add - pointer to the address */
/* cnt - number of bytes in the address */
char *pmsg, *add[], cnt;

cnt b;  
for (i = 0; 4 i < 2*cnt; i++, j++) <
if (b = Read_Char();  
if ((b == 'Y') || (b == 'y'))
return(TRUE);  
if ((b == 'N') || (b == 'n'))
return(FALSE);  
Write("Enter a Y or N --> ");

Write.Addr(padd, cnt)
char *padd[], cnt;

{  
unsigned char i, c[3],
for ( ; cnt > 0; cnt-- )

231421-53
```c
/PCD/UBR/CHUCK/CBRC/IVAP.C

i = paddr(cnt-1);
Char_To_Acct(i, &c[0]);
Write(&c[0]);
>
c[0] = 'n';
c[1] = 'O';
Write(&c[0]);
>
Recv_Data_i(pfd) /* Receives the frame from the 802.2 module */
{
struct FD
  *pfd;
struct FRAME_STRUCT *prfs, *ptfs;
struct TBD *ptbd;
struct RBD *prbd;
char *prbuf;
int cnt;

prbd = (struct RBD *) Build_Ptr(pfd->rbd_offset);
prfs = (struct FRAME_STRUCT *) Build_Ptr(prbd->buff_ptr);

switch (prfs->cmd & "P_F_BIT") {
  case UI:
    if (monitor_flag) {
      break; /* Don't put data in fifo unless terminal mode */
      prbuf = (char *) prfs;
      prbuf += 3; /* skip over the header info and point to the data */
      cnt = 3;
      for (; prbd != NULL; cnt = 0, prbuf = (char *) prbd->buff_ptr) {
        for (; cnt < (prbd->act_cnt & 0x003FFFF) & prbd->length > 0;
          cnt++, prbuf++, prbd->length--) {
          while(r_buf_stat == FULL);
          FIFO_R_In(prbuf);
        }
        prbd = Build_Ptr(prbd->link);
      }
      ifdef DEBUG
      if (prbd->length == 0 & prbd != NULL)
        Fatal(" Unexpected frame ");
      #endif /* DEBUG */
      break;
    }
  case XID
    while ((ptbd = Get_Tbd()) == NULL);
    ptbd->act_cnt = EOFBIT | XID_LENGTH;
    bcopy((char *) ptbd->buff_ptr, &xid_frame[0], XID_LENGTH);
    ptfs = (struct FRAME_STRUCT *) ptbd->buff_ptr;
    ptfs->cmd = prfs->cmd;
    ptfs->sapp = prfs->sapp | C_R_BIT; /* return the frame to the sender */
    ptfs->sapp = spapp;
    while(!Send_Frame(ptbd, Build_Ptr(pfd->src_addr)));
  }
}
```
break;
  case TEST:
    for (prbd = (struct RBD *)&Build_Ptr(pfd->rbd_offset),
         q = begin_pbd = pNULL; prbd ?= pNULL;
         prbd = Build_Ptr(prbd->rbd)) {
      if (q != pNULL)
        q->link = Offset(ptbd);
      else
        begin_pbd = ptbd,
        prbd->act_cnt = ptbd->act_cnt;
        memcpy((char *)ptbd->buff_ptr, (char *)prbd->buff_ptr,
               ptbd->act_cnt & Ox3FF),
        q = ptbd;
      }
    }
    pfs = (struct FRAME_STRUCT *)&begin_pbd->buff_ptr;
    pfs->cmd = prfs->cmd;
    pfs->ssap = prfs->ssap & C_R_BIT; /* return the frame to
         the sender */
    while((Send_Frame(begin_pbd, Build_Ptr(pfd->src_addr))))
      break;
    }
  Put_Free_RFA(pfd); /* return the frame */
  }
  }
Fifo_T_Out() /* called by main program */
  { char c;
    c = fifo_t[outfifo_t++];
    Disable_Uart_Int();
    if (outfifo_t == in_fifo_t) /* if the fifo is empty */
      t_buf_stat = EMPTY; /* stop filling Transmit Buffer Descriptors */
    else /* if the fifo was full and is now draining */
      if (t_buf_stat == FULL & out_fifo_t - 80 == in_fifo_t) /* turn on */
        RTS_DNB;
        t_buf_stat = INUSE;
    }
    Enable_Uart_Int();
    return(c);
  }
Fifo_T_In() /* called by Uart receive interrupt */
  { char c;
    fifo_t[in_fifo_t++] = c;
    if (t_buf_stat == EMPTY)
if (in_fifo) {
    // If FIFO is empty, disable transmitter.
    //}
}

if (out_fifo) {
    // if FIFO is empty, service
    //}
}

if (UART_TX_FIFO) {
    // Prime the interrupt
    //}
}

if (UART_TX_FIFO) {
    // Prime the interrupt
    //}
}

if (UART_TX_FIFO) {
    // Prime the interrupt
    //}
}
```c
/*PCD/UBS/CHUCK/CBR/C/UART.C

case UART_RECV_ERR_B:
    outb(CH_B_CTL, 1); /* point to RRI in 8274 */
    stat = inb(CH_B_CTL);
    outb(CH_B_CTL, 0x30);
    if (stat & 0x0010)
        Write("nParity Error Detected\n");
    if (stat & 0x0020)
        Write("nOverrun Error Detected\n");
    if (stat & 0x0040)
        Write("nFraming Error Detected\n");
    break;

case UART_RECV_B:
    c = inb(CH_B_DAT);
    if (hs_stat == TRUE) { /* Flag to terminate High Speed Transmit mode */
        break;
    }
    if (local_echo)
        Co(c);
    if (c == CTL_C)
        tmstat = FALSE;
    else
        Fifo_T_In(c);
    break;

case EXT_STAT_INT_B:
    outb(CH_B_CTL, 0x10); /* reset external status interrupts */
    break;

case EXT_STAT_INT_A:
    outb(CH_A_CTL, 0x10);
    break;

default:
}

ISR2()
{
    send_flag = TRUE;
    outb(0x6A, 123);
    outb(0x8E, 0x00); /* Timer 1 interrupts every 125 sec */
    outb(0x8E, 0x62); /* EDI 00130 */
}

231421-57
```
Load_Loop()
{
    int    Recv_Data_1;
    for(;;)
    {
        Read_Adr("\nEnter this Station's LSAP in Hex --> ", &asap, 1);
        if (!Add_Deap_Address(&asap, Recv_Data_1))
            Write("\n\nError: LSAP Address must be one of the following:\n\n2OH, 4OH, 6OH, 8OH, AOH, COH, EOH \n");
        else break;
    }
}

Load_Multicast()
{
    for (;;)
    {
        Read_Adr("\nEnter the Multicast Address in Hex --> ", &Multi_Addr[0], ADD_LEN);
        if ((Multi_Addr[0] & 0xO1) == 0)
            Write("\n\nSorry, the LSB of the Multicast Address must be 1\n");
        else if (!Add_Multicast_Address(&Multi_Addr[0])
            Write("\n\n\n\nSorry, Multicast Address Table is full\n");
            break;
        else
            Write("\n\nWould you like to add another Multicast Address?\n");
            Write("\n(Y or N) --> ");
            if (!Yes()) break;
    }
}

Remove_Multicast()
{
    for (;;)
    {
        Read_Adr("\nEnter the Multicast Address that you want to delete in Hex --> ", &Multi_Addr[0], ADD_LEN);
        if ((Multi_Addr[0] & 0xO1) == 0)
            Write("\n\n\n\n\nSorry, the LSB of the Multicast Address must be 1\n");
        else if (!Delete_Multicast_Address(&Multi_Addr[0])
            Write("\n\n\n\n\n\n\nSorry, that Multicast Address doesn't exist\n");
            break;
        else
            Write("\n\nWould you like to delete another Multicast Address?\n");
            Write("\n(Y or N) --> ");
            if (!Yes()) break;
    }
}
Print_Addresses()
{
    struct MAT *pmat;
    int stat;

    Write("\n This Station's Host Address is: ");
    Write_Addr(&mat[IO]. ADD_LEN);
    Write("\n The Address of the Destination Node is: ");
    Write_Addr(&mat[1O]. ADD_LEN);
    Write("\n This Station's LBAP Address is ");
    Write_Addr(&msgap. 1);
    Write("\n The Address of the Destination LBAP is. ");
    Write_Addr(&dspan. 1);

    if (stat = FALSE:
        for (pmat = mat[IO]; pmat <= &mat[MULTI_ADDR_CNT - 1]; pmat++)
            if (pmat->stat == INUSE) {
                stat = TRUE;
                break;
            }

            if (stat){
                Write("\n The Following Multicast Addresses are enabled: ");
                for (pmat = mat[IO]; pmat <= &mat[MULTI_ADDR_CNT - 1]; pmat++)
                    if (pmat->stat == INUSE) {
                        Write_Addr(&pmat->addr[0]. ADD_LEN);
                        Write(" ");
                    }
            }
        else
            Write("\n There are no Multicast Addresses enabled.\n");

    Init_DataLink()
{
    int stat;

    if ((stat = Init_Lic()) == PASSED)
        Write("\n\nPassed: Diagnostic Self Tests\n\n");
    else if(stat == FAILED_DIAGNOSE)
        Write("\n\nFailed: Diagnostic Self Tests\n\n");
    else if(stat == FAILED_LPBK_INTERNAL)
        Write("\n\nFailed: Internal Loopback Self Tests\n");
    else if(stat == FAILED_LPBK_EXTERNAL)
        Write("\n\nFailed: External Loopback Self Tests\n");
    else if(stat == FAILED_LPBK_TRANSCIVER)
        Write("\n\nFailed: Loopback Through Transceiver Self Tests\n");

    Init_Vap()
{
    outb(0xED, 0x31); /* initialize 80130 pic - ICW1 */
    outb(0x22, 0x20); /* ICW2 */

    231421-59
/PC0/USB/CHUCK/CSRC/UAP.C

outb(0x2E, 0x10);  /* ICW */
outb(0x2E, 0x0D);  /* ICW */
outb(0x2E, 0x10);  /* ICW */
outb(0x2E, 0xFF);  /* mask all interrupts */
outb(0xFF, 0x00);  /* set 80186 vector base */

/* Initialize the 80130 timer for Terminal Mode */
outb(0x8E, 0x34);
outb(0x8E, 0x88);
outb(0x8E, 0x08);  /* SYSTICK set for 1 msec */
outb(0x8E, 0x7D);
outb(0x8E, 0x25);
outb(0x8E, 0x00);  /* Timer 1 interrupts every .125 sec */

/* Initialize the 8274 */
outb(CH_B_CTL, 0x10); outb(CH_B_CTL, 0x20); outb(CH_B_CTL, 0x30);
outb(CH_A_CTL, 0x10); outb(CH_B_CTL, 0x30);
outb(CH_B_CTL, 0); outb(CH_B_CTL, wrb[2] = 0x14);
outb(CH_B_CTL, 1); outb(CH_B_CTL, wrb[1] = 0x15);
outb(CH_B_CTL, 9); outb(CH_B_CTL, wrb[9] = 0xEA);

/* IEEE 802.2/802.3 Compatible Data Link Driver */

Init_DataLink();

dhex = FALSE;
monitor_flag = TRUE;

Read_Addr("\n\nEnter the Address of the Destination Node in Hex --> ",
&Dest_Addr[0]. ADD_LEN);

Load_Lsap();

Read_Addr("\nEnter the Destination Node's LSAP in Hex --> ", &lsa.pr. 1);

if (Yes())
Load_Multicast();

Print_Addresses();

Terminal_Mode()
{
int frame_cnt, buf_cnt;
struct TBD *tbd, *eq, *begin_tbd;
char *buf, c;

if(Yes())
Write("\nWould you like the local echo on? (Y or N)-->");

}
/PC/UBR/CHUCK/CSRC/UAP.C

local_echo = TRUE;
else
local_echo = FALSE;

Write("\nThis program will now enter the terminal mode.\n\n");
Write("\nPress 'C' then CR to return back to the menu\n\n");

// Initialize Fifo variables /
out_fifo_t = in_fifo_t = out_fifo_r = in_fifo_r = 0;
t_buf_stat = EMPTY; r_buf_stat = EMPTY;

EDI_0130_R274;
Enable_Uart_Int();
Enable_Timer_Int();
monitor_flag = FALSE;
tmsstat = TRUE;
while (tmsstat) {
  for (frame_cnt = 0; frame_cnt < MAX_FRAME_SIZE; q = ptbd) {
    while ((ptbd = Get_Tbd()) == NULL); /* get a unit buffer from the
data link */
    pbuf = (char *) ptbd->buf_ptr; /* point to the buffer */
    buf_cnt = 0;
    if (frame_cnt == 0) { /* if this is the first buffer, add on IEEE 802.2
    header information */
      begin_ptbd = ptbd;
      *pbuf++ = ldfp;
      *pbuf++ = ssap;
      *pbuf++ = ui;
      buf_cnt = 0;
    } else q->link = Offset(ptbd); /* if this isn't the first buffer
    link the previous buffer with the new one */
    /* fill up a data link unit buffer from async transmit fifo */
    for (; buf_cnt < TBUF_SIZE & frame_cnt < MAX_FRAME_SIZE,
         buf_cnt++, pbuf++, frame_cnt++) {
      if (frame_cnt == 0 && send_flag) {
        while (t_buf_stat == EMPTY); /* wait until fifo has data */
        if ((c = *pbuf = Fifo_T_Out()) == CR) {
          ++buf_cnt; ++pbuf; ++frame_cnt;
          break;
        }
      } else if (c == CR || buf_cnt < TBUF_SIZE || send_flag) { /* last buffer in list */
        ptbd->act_cnt = buf_cnt || EOFBIT;
        send_flag = FALSE;
        break;
      }
    }
  }
  while (Send_Frame(begin_ptbd, &Dest_Addr[0])); /* keep trying until
  successful */
}
struct TBD *Build_Frame(cnt)
{
    u_short cnt;
    u_short buf_cnt, frame_cnt, i;
    struct TBD *ptbd, eq, *begin_ptbd;
    char *pbuf;

    1 = 0x20; frame_cnt = 0;
    for (; ; q = ptbd) {
        while ((ptbd = Get_Tbd()) == NULL) /* get a xmit buffer from the */
            data link */
            pbuf = (char *) ptbd->buff_ptr; /* point to the buffer */
            buf_cnt = 0;
        if (frame_cnt == 0) { /* if this is the first buffer, add on IEEE 802 */
            begin_ptbd = ptbd;
            *pbuf++ = dsep;
            *pbuf++ = sep;
            *pbuf++ = UI;
            buf_cnt = 3;
        } else q->link = Offset(ptbd); /* if this isn't the first buffer */
            /* fill up a data link xmit buffer with ASCII characters */
            for (; buf_cnt < TBUF_SIZE & cnt > 0;
                i++, buf_cnt++, pbuf++, cnt--, frame_cnt++) {
                *pbuf = i;
                if (i > 0x7E)
                    i = 0x1F;
            }
        if (cnt == 0) { /* last buffer in list */
            ptbd->act_cnt = buf_cnt | EOFBIT;
            break;
        }
    }
    return(begin_ptbd);
}

Monitor_Mode()
{
    u_short xmit_cnt, i;
    struct TBD *Build_Frame(), *ptbd;

    Write("Do you want this station to transmit? (Y or N) -- ");
    if (Yes())

231421-62
for (xmit = FALSE; xmit == FALSE; ) {
  Write("n Enter the number of data bytes in the frame --> ");
  cnt = Read_Int();
  if (cnt > 2048)
    Write("n Sorry, the number has to be less than 2048\n");
  else
    xmit = TRUE;
}

else xmit = FALSE;

Write("n Hit any key to exit Monitor Mode.\n");
Write("# of Good # of Good CRC Alignment No Receive\n");
Write(" Transmitted Received Errors Errors Resource Overrun\n");
```
/* 01234567890123456789012345678901234567890123456789012345678901234567
    xxxxxxxxxx xxxxxxxxxx xxx xxx
    xxxxxxxxxx xxxxxxxxxx
    11 25 33 44 57 71 */
```

for (i = 0; i < 79; i++)
  line[i] = Ox20;
line[79] = CR;
line[80] = '\n';

while (((inb(CH_B_CTL) & 1) == 0) {
  for (i = 0; i < 72; i++)
    line[i] = Ox20;
  Write_Long_Int(good_xmit_cnt, 11);
  Write_Long_Int(recv_frame_cnt, 25);
  Write_Short_Int(scb_crc_errs, 33);
  Write_Short_Int(scb_align_errs, 44);
  Write_Short_Int(scb_rsc_errs, 57);
  Write_Short_Int(scb_ovr_errs, 71);
  Write(&line[33]);
  if (xmit) {
    ptd = Build_Frame(cnt);
    while(!Send_Frame(ptbd, &Dest_Addr[0]));
  }
  i = Ci();
}

Hx_Xmit_Mode()
{
  struct TBD *ptbd;
  Write("n Hit any key to exit High Speed Transmit Mode.\n");
  hs_stat = TRUE;
  EOI_80130_8274;
  Enable_Vart_INT();
  
  /* Execute this loop until a recv char interrupt happens at Uart */

  231421-63
while (hs_stat) {
    while ((ptbd = Get_Tbd()) == NULL) /* get a smit buffer from
the data link */
    ptbd->act_cnt = EDFBIT; /* set the End Of Frame bit */
    while (Send_Frame(ptbd, &Dest_Addr(0))) /* Send Frame */
}
Disable_UART_Int();
}
Print_Cnt() {
char ch[i], base, width, i;
unsigned long temp;
{
dhex {
    width = 0;
    width = 4;
    base = 1A;
} else {
    base = 10;
    width = 10;
    width = 9;
}
Write("\n\nGood frames transmitted: ");
for (i = 1; i <= 11 - width; i++)
    Co(SP);
Int_To_Ascii(good_smit_cnt, base, ', ', &ch[0], width);
for (i = width - 1; i >= 0; i--)
    Co(ch[i]);
Write(" Good Frames received: ");
for (i = 1; i <= 15 - width; i++)
    Co(SP);
Int_To_Ascii(frame_cnt, base, ', ', &ch[0], width);
for (i = width - 1; i >= 0; i--)
    Co(ch[i]);
Write("\n\nBad_frames received: ");
for (i = 1; i <= 15 - width; i++)
    Co(SP);
temp = sch_crc_errs;
Int_To_Ascii(temp, base, ', ', &ch[0], width);
for (i = width - 1; i >= 0; i--)
    Co(ch[i]);
Write(" Alignment errors received: ");
for (i = 1; i <= 10 - width; i++)
    Co(SP);
temp = sch_align_errs;
Int_To_Ascii(temp, base, ', ', &ch[0], width);
for (i = width - 1; i >= 0; i--)
    Co(ch[i]);
Write("\n\nOut of Resource frames. ");
for (i = 1; i <= 12 - width; i++)
    Co(SP);
temp = sch_out_err;
Int_To_Ascii(temp, base, ', ', &ch[0], width);
for (i = width - 1; i >= 0; i--) 
  CoC(h[i]);
Write(" Receiver overrun frames: ");
for (i = 1; i <= 12 - width; i++) 
  Co(SP);
  temp = LLC_uvt_errs;
Int_To_Ascii(temp, base, ",", &ch[0], width);
for (i = width - 1; i >= 0; i--) 
  CoC(h[i]);
Write(" Transmit underrun frames: ");
for (i = 1; i <= 11 - width; i++) 
  Co(SP);
  temp = underrun_cnt;
Int_To_Ascii(temp, base, ",", &ch[0], width);
for (i = width - 1; i >= 0; i--) 
  CoC(h[i]);
Write(" in Last CRN: ");
for (i = 1; i <= 25 - width, i++) 
  Co(SP);
  temp = no_crs_cnt;
Int_To_Ascii(temp, base, ",", &ch[0], width);
for (i = width - 1, i >= 0; i--) 
  CoC(h[i]);
Write(" SUE errors: ");
for (i = 1, i <= 21 - width, i++) 
  Co(SP);
  temp = max_col_cnt;
Int_To_Ascii(temp, base, ",", &ch[0], width);
for (i = width - 1, i >= 0, i--) 
  CoC(h[i]);
Write(" Frames that deferred: ");
for (i = 1, i <= 15 - dwidth, i++) 
  Co(SP);
Int_To_Ascii(defer_cnt, base, ",", &ch[0], dwidth);
for (i = dwidth - 1, i >= 0, i--) 
  CoC(h[i]);
}

Print_Help()
{
  Write(" in Commands are. "\n  Write(" T - Terminal Mode M - Monitor Mode\n"),
231421-65
1-246
Main()
{
    int c;
    Init_Uap(),
    Print_Help();
    for (;;) {
        Write ("Enter a command, type H for Help --> ");
        c = Read_Char();
        switch (Lower_case(c)) {
        case 'h':
            Print_Help();
            break;
        case 'm':
            Monitor_Mode();
            break;
        case 't':
            Terminal_Mode();
            break;
        case 'c':
            Clear_Cnt();
            break;
        case 'v':
            Write("Transmit Statistics are now ");
            if (flags.stat_on == 1)
                Write("on. 
Would you like to change it? (Y or N) -->");
            else
                Write("off. 
Would you like to change it? (Y or N) -->");
            if (Yes()) {
                if (flags.stat_on == 1)
                    flags.stat_on = 0;
                else flags.stat_on = 1;
            }
            break;
        case 'p':
            Print_Cnt();
            break;
        case 'c':
            Clear_Cnt();
            break;
        case 'a':
            Load_Multicast();
            break;
        case 'r':
            Remove_Multicast();
            break;
        case 's':
            break;
        default:
            break;
        }
    }
}
/PCO/USR/CHUCK/CBRC/UAP.C

Delete_Deep_Address(ssap);
Load_Lsap();
break;
case 'd':
    Read_Addr("\n\nEnter the Destination Node's LSAP in Hex ---> ", &dsap, 1);
break;
case 'n':
    Read_Addr("\n\nEnter the Address of the Destination Node in Hex ---> ", &Dest_Addr[0], ADD_LEN);
break;
case 'l':
    Print_Addresses();
break;
case 'v':
    Software_Reset();
    Init_DataLink();
    Add_Deep_Address(ssap,Recv_Data_1);
break;
case 'b':
    Write("\n\nThe current base is ");
    if (dhex == TRUE)
        Write("Hex. \n Would you like to change it? (Y or N) ---> ");
    else
        Write("Decimal. \n Would you like to change it? (Y or N) ---> ");
    if (Yes()) {
        if (dhex == TRUE)
            dhex = FALSE;
        else dhex = TRUE;
    }
break;
default:
    Write("\n Unknown command\n");
    break;
}
/PC0/UBR/CHUCK/CBRC/ASSY.ASM

.name c_assy_support
.stack segment stack 'stack'
.stack label word
.stack ends

DLD_DATA segment public 'DATA'
extrn SEGMT_.word ; data segment address
DLD_DATA ends

UAP_DATA segment public 'DATA'
UAP_DATA ends

DLD_CODE segment public 'CODE'
extrn Isr_Timeout_far, Isr_686_far, Isr7_far
extrn Isr6_far, Isr5_far, Isr1_far
DLD_CODE ends

UAP_CODE segment public 'CODE'
extrn Isr_Vert_far, Isr2_far, Main_far
UAP_CODE ends

DQ_CODE segment public 'CODE'
.public inw, outw, init_intv, enable, disable, Build_Ptr,
.public Offset, begin, inw, outw
arg1 equ [BP + 6]
arg2 equ [BP + 8]

.assume CS:DQ_CODE
.assume DS:DLD_DATA

; initialization program for the 8259a data link driver
;
begin

stj
mov ax, DLD_DATA;set base of driver and
mov SEGMT_.ax ; pass the segment value to the c program
mov ds, ax

.call Main_;  ; go to the c program

hit

inb proc far
push BP
mov BP, SP
push DX
mov DX, arg1
in AL, DX
pop DX
mov SP, BP

231421-68
/PCD/USR/CHUCK/C SRC/ASSY. ASM

popBP
ret

inb_ endp

outb_ proc far
pushBP
movBP,BP
pushDX
pushAX
movDX, arg1
movAX, arg2
outDX, AL
popAX
popDX
movSP, BP
popBP
ret

outb_ endp

inw_ proc far
pushBP
movBP,BP
pushDX
movDX, arg1
inAX, DX
popDX
movSP, BP
popBP
ret

inw_ endp

outw_ proc far
pushBP
movBP,BP
pushDX
pushAX
movDX, arg1
movAX, arg2
outDX, AX
popAX
popDX
movSP, BP
popBP
ret

outw_ endp

Build_Ptr_ proc far
pushBP
movBP,BP
movDX, DLD_DATA
movAX, arg1
movSP, BP
popBP
ret

Build_Ptr_ endp

Offset_ proc far
/*PCO/USR/CHUCK/CSRC/ABBY.ASM*/

push BP
mov BP, BP
mov AX, arg1
mov BP, BP
ret

服_ever_end

serve_int_isr proc far
push AX
push BX
push CX
push DX
push SI
push DI
push DB
push EB
mov AX, DLD_DATA
mov DS, AX
mov EB, AX
call Isr_386_
pop EB
pop DS
pop DI
pop SI
pop DX
pop CX
pop BX
pop AX
iret

serve_int_isr endp

serve_int_5274 proc far
push AX
push BX
push CX
push DX
push SI
push DI
push DB
push EB
mov AX, UART_DATA
mov DS, AX
mov EB, AX
call Isr_UART_
pop EB
pop DS
pop DI
pop SI
pop DX

231421-70.
/PCD/USR/CHUCK/CSRC/ASSY.ASM

pop CX
pop BX
pop AX
iret

serve_int_8274 endp

serve_int_timeout proc far
push AX
push BX
push CX
push DX
push SI
push DI
push DB
push EB

mov AX, DLD_DATA
mov DS, AX
mov EB, AX

call _Isr_Timeout_

pop EB
pop DS
pop DI
pop SI
pop DX
pop CX
pop BX
pop AX

test

serve_int_timeout endp

serve_int7_isr proc far
push AX
push BX
push CX
push DX
push SI
push DI
push DB
push EB

mov AX, DLD_DATA
mov DS, AX
mov EB, AX

call _Isr7_

pop EB
pop DS
pop DI
pop SI
pop DX
pop CX
pop BX
pop AX

231421-71
iret
serve_int7_isr endp

serve_int7_isr proc far
push AX
push BX
push CX
push DX
push SI
push DI
push DS
push ES
mov AX, DLD_DATA
mov DS, AX
mov ES, AX
call 1sr6_
pop ES
pop DS
pop DI
pop SI
pop DX
pop CX
pop BX
pop AX
iret
serve_int7_isr endp

serve_int5_isr proc far
push AX
push BX
push CX
push DX
push SI
push DI
push DS
push ES
mov AX, DLD_DATA
mov DS, AX
mov ES, AX
call 1sr5_
pop ES
pop DS
pop DI
pop SI
pop DX
pop CX
pop BX
pop AX
iret
serve_int5_isr endp
; serve_int2_isr proc far
push AX
push BX
push CX
push DX
push SI
push DI
push DS
push ES
mov AX, UAP_DATA
mov DS, AX
mov ES, AX
call Isr2_
pop ES
pop DS
pop DI
pop SI
pop DX
pop CX
pop BX
pop AX
iret
serve_int2_isr endp

; serve_int1_isr proc far
push AX
push BX
push CX
push DX
push SI
push DI
push DS
push ES
mov AX, DLD_DATA
mov DS, AX
mov ES, AX
call Isr1_
pop ES
pop DS
pop DI
pop SI
pop DX
pop CX
pop BX
pop AX
iret
serve_int1_isr endp

enable_proc far ret

231421-73
```assembly
ret
enable_ endp

disablecli
ret
disable_ endp

init_intv_ proc far
push DS
push AX
xor AX, AX
mov DS, AX

; Interrupt types for the 186/51 computer
mov DS:word ptr 80h, offset serve_int_9274 ; int 0
mov DS:word ptr 82h, DS:CODE
mov DS:word ptr 84h, offset serve_int_isr ; int 1
mov DS:word ptr 86h, DS:CODE
mov DS:word ptr 88h, offset serve_int2_isr ; int 2
mov DS:word ptr 8Ah, DS:CODE
mov DS:word ptr 8Ch, offset serve_int_isr ; int 3
mov DS:word ptr 8Eh, DS:CODE
mov DS:word ptr 92h, offset serve_int_timeout ; int 4
mov DS:word ptr 94h, DS:CODE
mov DS:word ptr 96h, offset serve_int_isr ; int 5
mov DS:word ptr 98h, DS:CODE
mov DS:word ptr 9Ah, offset serve_int_isr ; int 6
mov DS:word ptr 9Ch, offset serve_int_isr ; int 7
mov DS:word ptr 9Eh, DS:CODE

pop AX
pop DS
ret

init_intv_ endp

DS:CODE ends end begin:'ds:did_data, ss:stack:stktop

231421-74
```
Implementing Ethernet/Cheapernet with the Intel 82586

KIYOSHI NISHIDE
APPLICATIONS ENGINEER
PREFACE

Intel's three VLSI chip set, the 82586, 82501, and 82502, is a complete solution for IEEE 802.3 10M bps LAN standards—10BASE5 (Ethernet) and 10BASE2 (Cheapernet). The 82586 is an intelligent peripheral which completely manages the processes of transmitting and receiving frames over a network under the CSMS/CD protocol. The 82586 with its on-chip four DMA channels offloads the host CPU of the tasks related to managing communication activities. The chip, for example, does not depend on the host CPU for time critical functions, such as transmissions/retransmissions and receptions of frames. The 82501 is a 10 MHz serial interface chip specially designed for the 82586. The primary function of the 82501 is to perform Manchester encoding/decoding, provide 10 MHz transmit and receive clocks to the 82586, and drive the transceiver (AUI) cable in Ethernet applications. In addition, the 82501 provides a loopback function and on-chip watchdog timer. The 82502 is a CMOS transceiver chip. The 82502 is the chip which actually drives the coaxial cable used for Ethernet or Cheapernet.

This Ap Note presents a design example of a simple but general Ethernet/Cheapernet board based on the three chip set. The board is called LANHIB (LAN High Integration Board) and uses an 80186 microprocessor as the host CPU. The LANHIB is an independent single board computer and requires only a power supply and ASCII terminal. Demo software, called TSMS (Traffic Simulator and Monitor Station) is also included in this Ap Note. The TSMS program is a network debugger and exercise tool used to exercise the 82586. In addition, flowcharts for troubleshooting are provided in order to minimize debugging time of the LANHIB board.

1.0 INTRODUCTION

A brief overview of the CSMA/CD protocol is described in Section 2. Ethernet and Cheapernet are also compared in this section.

Section 3 discusses hardware of the LANHIB in detail. This section should be helpful not only to understand the LANHIB, but also to learn in general how a system based on the three chip set can be put together. Since the 82502 involves analog circuitry, an explanation on proper layout is provided.

Demo software is presented in Section 4.0. It covers EPROM programming procedures and three sample sessions. Step by step operations at a terminal are illustrated in the figures.

Section 5 describes LANHIB troubleshooting procedures. Flowcharts are used to guide troubleshooting.

Complete LANHIB schematics and parts list are found in Appendix A. If a LANHIB is to be built, the schematics and Section 5 can be submitted to an available wire wrap facility. In parallel to board construction, Sections 3 and 4 can be studied. A factory wire wrap board for the LANHIB is offered at a discount price by Augat Corporation. Please return the enclosed card for more information.

Listing of the TSMS program and LANHIB Initialization Routine are in Appendix B. The source codes and related files are available on a diskette by returning the card enclosed in this design kit or through Insite (Intel's Software Index and Technology Exchange Library).

2.0 ETHERNET/CHEAPERNET OVERVIEW

2.1 CSMA/CD

Carrier Sense Multiple Access with Collision Detection (CSMA/CD) is a simple and efficient means of determining how a station transmits information over common medium that is shared with other stations. CSMA/CD is the access method defined by the IEEE 802.3 standard.

Carrier Sense (CS) means that any station wishing to transmit "listens" first. When the channel is busy (i.e., some other station is transmitting) the station waits (defers) until the channel is clear before transmitting ("listen before talk").

Multiple Access (MA) means that any stations wishing to transmit can do so. No central controller is needed to decide who is able to transmit and in what order.

Collision Detection (CD) means that when the channel is idle (no other station is transmitting) a station can start transmitting. It is, however, possible for two or more stations to start transmitting simultaneously causing a "collision". In the event of a collision, the transmitting stations will continue transmitting for a fixed time to ensure that all transmitting stations detect the collision. This is known as jamming. After the jam, the stations stop transmitting and wait a random period of time before retrying. The range of random wait times increases with the number of successive collisions such that collisions can be resolved even if a large number of stations are colliding.

There are three significant advantages to the CSMA/CD protocol. The first and foremost is that CSMA/CD is a proven technology. One CSMA/CD network, Ethernet, has been used by Xerox since 1975. Ethernet is so well understood and accepted that IEEE adopted
it (with minor changes) as the IEEE 802.3 10Base5 (10 Mbps, Baseband, 500 meters per segment) standard. Reliability is the second advantage to the 802.3 protocol. This media access method enables the network to operate without central control or switching. Thus, if a single station malfunctions, the rest of the network can continue operation. Finally, since CSMA/CD networks are passive and distributed in nature, they allow for easy expansion. New nodes can be added at any time without reinitializing the entire network.

2.2 Ethernet and Cheapernet

The IEEE 802.3 Type 10BASE5 standard (Ethernet) has gained wide acceptance by both large and small corporations as a high speed (10 Mbps) Local Area Network. The Ethernet channel is a low noise, shielded 50Ω coaxial cable over which information is transmitted at 10 million bits per second. Each segment of cable can be up to 500 meters in length and can be connected to longer network lengths using repeaters. Repeaters regenerate the signal from one cable segment onto another. At each end of a cable segment a terminator is attached. This passive device provides the proper electrical termination to eliminate reflections. The transceiver transmits and receives signals on the coaxial cable. In addition, it isolates the node from the channel so that a failure within the node will not affect the rest of the network. The transceiver is also responsible for detecting collisions—simultaneous transmissions by two or more stations. Ethernet transceivers are connected to the network coaxial cable using a simple tap, and to the station it serves via the transceiver cable which can be up to 50 meters in length. The transceiver cable is made of four individually shielded twisted pairs of wires. An Ethernet interface at a computer (DTE), which includes a serial interface and data link controller, provides the connection to the user or server station. It also performs frame manipulation, addressing, detecting transmission errors, network link management, and encoding and decoding of the data to and from the transceiver.

The IEEE 802.3 Type 10BASE2 (Cheapernet) has the same functional and electrical specifications as Type 10BASE5 (Ethernet) with only two exceptions in physical (or rather mechanical) characteristics. Cheapernet is as shown in Figure 1 just a different implementation of the IEEE standard. Ethernet and Cheapernet are both 10 million bits/second CSMA/CD LANs and use the identical network parameters, such as slot time = 51.2 µs. Ethernet and Cheapernet can, therefore, be built by the same VLSI components with the same software (Figure 2).

The two physical differences attribute to the cost reduction purpose of Cheapernet—cheaper implementation of Ethernet. First, the cable used in Cheapernet may be a lower cost 50Ω coaxial cable than the one for Ethernet. The most common coaxial cable for Cheapernet is RG58 which cost about $0.15/ft. A typical Ethernet cable costs about $0.83/ft.

Second, the transceiver is integrated into the DTE in Cheapernet. The coaxial cable physically comes to the DTE, connects to the transceiver within the DTE, and goes to the next DTE (see Figure 3). The kind of connector used at the DTE is an off-the-shelf BNC "T" connector. Topology is, therefore, a simple daisy chaining. This cabling scheme contributes to further cost reduction due to omission of the Transceiver (AUI) Cable, cheaper connectors, and easier installation. The Ethernet transceiver cable costs about $1.49/ft. More flexible thin coaxial cables and familiar BNC "T" connectors are making Cheapernet a user installable Ethernet compatible network.
Figure 2. 82586/82501/82502 in Ethernet and Cheapernet

Figure 3. Ethernet Cabling vs Cheapernet Cabling
Table 1. Differences between Ethernet and Cheapernet

<table>
<thead>
<tr>
<th></th>
<th>Ethernet (10BASE5)</th>
<th>Cheapernet (10BASE2)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data Rate</td>
<td>10 M bits/sec.</td>
<td>10 M bits/sec.</td>
</tr>
<tr>
<td>Baseband or Broadband</td>
<td>Baseband (Manchester)</td>
<td>Baseband (Manchester)</td>
</tr>
<tr>
<td>Cable Length per Segment</td>
<td>500m</td>
<td>185m</td>
</tr>
<tr>
<td>Nodes per Segment</td>
<td>100</td>
<td>30</td>
</tr>
<tr>
<td>Node Spacing</td>
<td>2.5m</td>
<td>0.5m</td>
</tr>
<tr>
<td>Cable Type</td>
<td>0.4 in diameter 50Ω Double Shielded example: Ethernet Coax.</td>
<td>0.2 in diameter 50Ω Single or Double Shielded example: RG 58 A/U or RG 58 C/U</td>
</tr>
<tr>
<td>Transceiver Cable</td>
<td>Yes, up to 50m</td>
<td>No, not needed</td>
</tr>
<tr>
<td>Capacitance per node</td>
<td>4 pF</td>
<td>8 pF</td>
</tr>
<tr>
<td>Typical Connector</td>
<td>Clamp-on Tap Connector or Type N Plug Connector</td>
<td>BNC Female Connector</td>
</tr>
</tbody>
</table>

Because of the lower quality cables and connectors used in Cheapernet, there are some drawbacks. The maximum distance for one Cheapernet cable segment is only 185m (600 feet), as compared to 500m (1640 feet) for Ethernet. The maximum number of nodes allowed for one Cheapernet cable segment is 30. Ethernet on the other hand allows a maximum of 100 nodes per segment. A BNC “T” connector used in Cheapernet introduces more electrical discontinuity on the transmission line than the clamp-on tap connector widely used for Ethernet. The maximum capacitance load allowed at a Cheapernet connection is 8 pF, while it is 4 pF for Ethernet. These differences are summarized in Table 1.0.

Since Ethernet and Cheapernet share the same functional and electrical characteristics, both may be mixed in a network as shown in Figure 4. In this hybrid Ethernet/Cheapernet network, it is important to keep the network propagation delay within 46.4 μs. The network may be expanded as required within this round trip propagation delay limit. Ethernet, for example, may serve as a backbone for Cheapernet in a hybrid Ethernet/Cheapernet network.

![Figure 4. Ethernet/Cheapernet Hybrid Network](image_url)
3.0 ETHERNET/CHEAPERNET NODE DESIGN

Details on LAN High Integration Board (LANHIB) design are presented in this section. The LANHIB is an 82586/80186 shared bus board and can be configured to Ethernet or Cheapernet. The 82586 is used in minimum mode to reduce chip count.

The reader is advised to refer to the 80186, 82586, 82501, and 82502 data sheets. Basic understanding of the 80186 microprocessor is assumed. Figure 5 shows the block diagram of the LANHIB. Schematics are in Appendix A.

3.1 82586 (Min Mode) Interface to the 80186

The 82586 can be placed in minimum mode by strapping the MN/MX pin to VCC. In the minimum mode, the chip directly provides all bus control signals—ALE, RD, WR, DT/R, and DEN, saving the 8288 Bus Controller. The 80186, which is the only other bus master on the shared bus, also generates these bus control signals directly. The HOLDs and HLDA of these two chips are connected together so that only one of the two bus masters can exclusively drive the bus at a time under the HOLD/HLDA protocol. Except for the ALE, all bus signals including address and data lines float when the chip does not have control of the bus. In this design example, RDs, WRs, DT/R and DEN from the two chips are connected together respectively. ALEs from the two chips are connected to an OR-gate to generate a system ALE. Multiplexed address data lines AD0–AD15 and address lines A15–A19 of the two chips are also connected line by line correspondingly.

3.2 82586 Address Latch Interface

Figure 6 shows the timing of the address signals with respect to the ALE signal. The ALE of the 82586 is OR-ed with the ALE of the 80186 and the result is connected to the latch enable inputs of Octal Transceiver Latches. The latches transfer the input data to the output as long as the latch enable is high, and captures the input data into the latch when the latch enable goes low. In this timing diagram, the setup and hold times of the input data (82586 address) required by the address latch can be verified. Estimating 7 ns of propagation delay in the 74S32, the setup time is T38 + 7, which is 32 ns at 8 MHz. The hold time for A19 is shorter than the other address lines because it is valid only during T1. The hold time for A19 is T4 – T36 – 7, which is 3 ns. The hold time for the other address lines is T39 – 7, which is 38 ns. In this design, a 74F373 was chosen to latch address lines A16–A19 and two 74LS373s were used to latch address lines AD0–AD15. Required setup and hold times of the 74F and 74LS 373s are summerized in Table 2.

Note that address lines A16–A18 and BHE of the 82586 are not really needed to be latched. These lines stay valid for an entire memory cycle.

Figure 6. 82586 Address Timing

1-262
3.3 80186 Address Latch Interface

The address latch used by the 82586 is shared by the 80186. Figure 7 shows the 80186 address line timing with respect to the ALE. Again estimating 7 ns delay in the 74S32, the setup time for the latch is \( T_{A_{VAL}} + 7 \) and the hold time is \( T_{L_{AL}} - 7 \). These are 37 ns and 23 ns respectively at 8 MHz. Comparing to the required values shown in Table 2, it is quite obvious that the setup and hold times of the latch are met by wide margins. Note that the 80186’s address lines A16–A18 and BHE are not valid for an entire memory cycle; therefore, they have to be latched.

3.4 82586 Memory Interface

The 74LS373 has a delay of 18 ns for input data to reach the output assuming the latch enable is high. A demultiplexed valid address (output of the address latch), therefore, becomes available after \( T_{29} + 18 \) measuring from the beginning of \( T_1 \) (Figure 8). The demultiplexed address remains valid until the ALE of the next memory access becomes active. Upper address lines, A14 through A20, are connected to a 16L8 PAL, which provides address decode logic for all memory devices. The PAL truth table is in Appendix A. The PAL has a maximum of 35 ns propagation delay, so chip selects will become active after \( 55 + 18 + 35 \) ns (max.) from the beginning of \( T_1 \) as indicated in Figure 8. Since address decode logic is implemented by a PAL, any memory expansion would only require a reprogramming of this PAL.

Two 74LS245 bus transceiver chips are controlled by the DT/R and DEN. Output enable and disable times of the 74LS245 are 40 and 25 ns respectively. The maximum propagation delay when the output enable is active is 12 ns.

---

Table 2. 74F and 74LS Data Setup and Hold Time Specifications at 25°C

<table>
<thead>
<tr>
<th></th>
<th>74F373</th>
<th>74LS373</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Min</td>
<td>Nom</td>
</tr>
<tr>
<td>Data Setup Time</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>Data Hold Time</td>
<td>3</td>
<td></td>
</tr>
</tbody>
</table>

---

Figure 7. 80186 Address Timing
Figure 8. 82586 Memory Interface Timing
Figure 9. 80186 Memory Interface Timing
Address access time is \(3 \times T_1 - T_29 - 18 - T_8 - 12 + n \times T_1\), where \(n\) is the number of wait states. For 0 wait states operation at 8 MHz, it is 270 ns minimum. Chip select access time is \(3 \times T_1 - T_29 - 18 - T_8 - 12 + n \times T_1 - 35\), which is 235 ns for 0 wait state operation. Command access time for a read cycle is \(2 \times T_1 - T_40 - T_8 - 12 + n \times T_1\), which is 123 ns.

Address setup time for a write cycle is \(T_1 - T_29 - 18 + T_23\), which is 52 ns minimum.

To meet these timing requirements, 2764-20s must be used for ROM. Static RAM chips, HM6264P-15, offer very wide timing margins and were selected for this design.

### 3.5 80186 Memory Interface

Figure 9 shows the timing of the 80186 memory interface. By comparing this figure to Figure 7, it is easy to notice that the 80186 offers a little faster bus interface. For example, TCLRL which is equivalent to T40 (0 to 95 ns) of the 82586 is specified as 10 to 70 ns. Since the memory choice satisfies the 82586 memory timing parameters, it also satisfies the 80186 memory timing parameters.

### 3.6 Memory Map

With 2764-20 EPROMs and 6264P-15 SRAMs, this board has 32 K bytes of ROM space and 16 K bytes of RAM space. Memory map is given in Figure 10. If 27128-20 EPROMs are used, the ROM space becomes 64 K bytes.

### 3.7 80186 I/O Interface

#### 3.7.1 82586 CHANNEL ATTENTION GENERATION

The active low Peripheral Chip Select 0 (PCS0) was used to generate a channel attention (CA) signal to the 82586. This way of CA generation satisfies the requirement that the width of a CA which must be wider than a clock period of the system clock.

#### 3.7.2 82586 HARDWARE RESET PORT

PCS0 of the 80186 will reset the 82586 if any I/O command is executed using this I/O chip select.

#### 3.7.3 82530 INTERFACE

82530 interface to the 80186 was derived from the design example presented in the 82530 SCC-80186 Interface Ap Brief. This document is attached to this Ap Note as Appendix C.

### 3.8 82586 Ready Signal Generation

82586 asynchronous ready (ARDY) signal is generated from a shift register. The shift register provides the 82586 a “normally ready” signal. When a wait state is needed, the ready signal is dropped to the low state. As shown in Table 3, the 82586 can be programmed to have 0 to 8 wait states by setting the DIP switch properly. Even though the on-board memory devices are
fast enough for 0 wait states operation, this programmable wait state capability was added so that the effect of wait states on the 82586 performance could be evaluated.

3.9 82501 Circuits

Since the 82501 is designed to work with the 82586, no interfacing circuits are required.

The transceiver cable side of the 82501 requires some passive components. The receive and collision differential inputs must be terminated by 78Ω ±5% resistors. Common mode voltages on these differential inputs are established internally. 240Ω ±5% pull down resistors must be connected on the TRMT and TRMT output pins.

A 0.022 μF ±10% capacitor connected between pin 1 and 2 of the 82501 is for the analog phase-looped loop.

Connected between the X1 and X2 pins is a 20 MHz parallel resonant quartz crystal (antiresonant with 20 pF load fundamental mode). An internal divide-by-two counter generates the 10 MHz clock. Since both Ethernet and Cheapernet tolerate an error of only ±0.01% in bit rate, a high quality crystal is recommended. The accuracy of a crystal should be equal to or better than ±0.002% @ 25°C and ±0.005% for 0–70°C. A 30–35 pF capacitor is connected from each crystal pin (X1 and X2) to ground in order to adjust effective capacitance load for the crystal, which should be about 20 pF including stray capacitance.

3.10 82502 Circuits

3.10.1 ISOLATION AND POWER REQUIREMENTS

The IEEE 802.3 standard requires an electrical isolation within the transceiver (MAU). Cheapernet (10BASE2) requires the isolation means to withstand 500V ac, rms for one minute. Ethernet (10BASE5) requires 250 Vrms. This electrical isolation is normally accomplished by transformer coupling of each signal pair. The kind of transformers recommended for the 82502 are the pulse transformers which have a 1:1 turn ratio and at least 50 microhenry inductance. PE64102 and PE64107 manufactured by Pulse Engineering are found to be good selections for this purpose. The PE 64102 offers 500 Vrms isolation. The PE64107 offers 2000 Vrms isolation. Both products provide three transformers in one package. Even though the current Type 10BASE5 specification requires only 250 Vrms, it is very common to have a higher isolation, at least 500 Vrms, in transceivers.

The standard specifies the voltage input level and maximum current allowed on the power pair of the transceiver cable. The voltage level may be between +11.28V dc and +15.75V dc. The maximum current is limited to 500 mA. Since the 82502 requires +10V ±10% and +5V ±10% as power, there has to be a DC/DC converter. In addition the DC/DC converter must be isolated due to the requirement described above. The DC/DC converter should be able to supply about 100 mA on the +10V line and 60 mA on the 5V line. The efficiency required in the converter is, therefore, (11V × 100 mA + 5.5V × 60 mA) / (11.28V - 0.5A × 4Ω × 500 mA) × 100 = 31% worst case. 4Ω is the maximum round trip resistance the power pair may have. 82502's CMOS process is the major contributor to this low DC/DC efficiency requirement.

Since the DC/DC converter has an isolation transformer inside, the output voltages are all floating voltages. The 0V output of the converter, for example, has no voltage relationship with the DTE's ground. The VSS and AVss pins of the 82502 should be connected to the 0V output of the DC/DC converter which is the 82502's ground (reference voltage).

Both Pulse Engineering and Reliability Incorporated produce DC/DC converters that meet the 82502's requirements. The Pulse Engineering's part number is PE64369 (enclosed in this design kit). The device measures about 1.5" x 1.5" x 0.5" and provides 2000 Vrms breakdown. The Reliability's part number is 2E12R10-5. Preliminary data sheets are available from Reliability.

3.10.2 OTHER PASSIVE AND ACTIVE DEVICES FOR THE 82502

A 78Ω ±5% resistor is required to terminate the transmit pair of the Transceiver cable. The chip has an internal circuit that establishes a common mode voltage, thus no voltage divider is required. The receive and collision pair drivers need pull up resistors. A 43.2 ±1% resistor must be connected from each output pin to +5V.
A 243Ω ±0.5% precision resistor is required on the REXT pin to the ground. The accuracy of this resistor is very important since this resistor is a part of current and voltage reference circuits in the analog sections of the 82502.

Grounding the HBD (Heartbeat Disable) pin will allow the chip to perform Signal Quality Error check (Heartbeat) as required by the IEEE 802.3. The chip will transmit the collision presence signal after each transmission during Interframe Spacing (IFS) time. In a repeater application, this feature is disabled (HBD = +5V).

Diodes connected on the CXTD pin are to reduce the capacitive loading onto the coaxial cable. One diode is sufficient, but two will provide a protection in case one burns out (Short Circuit). The diode should have about 2 pF shunt capacitance at Vd = 0V and be able to handle at least 100 mA when biased in forward direction. A few candidates are IN5282, IN3600, and IN4150.

A 100Ω fusible resistor connected on the CXRD pin is purely for protection. It is there as a fuse, not as a resistor. The 82502 works without this resistor. The IEEE 802.3, however, states that “component failures within the MAU (Media Attachment Unit or Transceiver) electronics should not prevent communication among other MAUs on the coaxial cable.” It is recommending a transceiver design that minimizes the probability of total network failure. The fusible resistor will provide an open circuit in an event of excess current. A short circuit from the CXRD pin to ground will not bring down the network due to the blown fuse.

A 1 MΩ resistor connected between the coaxial cable shield and the Transceiver cable shield will provide a static discharge path. The Ethernet coaxial cable should also have an effective earth ground at one point in a network as required by the standard. A 0.01 μF in parallel to the 1 MΩ resistor provides ground for RF signals.

### 3.10.3 LAYOUT CONSIDERATION FOR THE 82502 CIRCUITS

It is strongly recommended that the board have a special ground plane for the 82502 (see Figure 11). The 0V (reference) output of the isolated DC/DC converter should be connected to the ground plane. The VSS and AVSS pins of the 82502 should be connected to the ground plane with minimum lead wires.

There should be a 0.22 μF capacitor connected between the coaxial cable shield and ground. The signal path from the coax. shield through the 0.22 μF capacitor to

![Figure 11. Ground Plane for the 82502](image-url)
the ground should be kept as short as possible—leads of the 0.22 μF capacitor should be as short as possible.

The path length from the CXTD pin through two diodes to the center conductor of the coax should also be minimized.

These are recommendations which will produce a more reliable circuit if followed carefully. Remember that the 82502 has analog circuits in it.

4.0 DEMONSTRATION SOFTWARE

The demonstration software included in this Ap Note is called “Traffic Simulator and Monitor Station” (TSMS) program. The TSMS program is written in PL/M and has the following features:

1. Programmable network load generation
2. Network statistical monitoring capabilities
3. Interactive command execution of all 82586 commands
4. Interactive buffer monitoring

The environment created with the TSMS program was found to be very useful for network debugging and other individual station's hardware and software debugging. The TSMS software listing is found in Appendix B.

NOTE:
The 82586 Date Link Driver presented in AP Note 235 also runs on the LANHIB. Please refer to the Ap Note for detailed operations of the software.

4.1 Programming PROMs to Run the TSMS Program

By returning the card enclosed in this kit or by contacting Insite, the TSMS program and related batch files can be obtained on a diskette. TSMS related files that are on the diskette are:

READ.ME
TSMS.PLM
IO.PLM
INI186.PLM
LANHIB.BAT
SBC.BAT
IUPHIB.BAT
IUPSBC.BAT
HI.BYT
LO.BYT
ROM.BAT

The READ.ME file contains instructions for programming PROMs. HI.BYT and LO.BYT are the files which can be downloaded to PROMs directly. These files are already configured for the LANHIB. The batch file ROM.BAT invokes the Intel PROM Programming Software (iPPS) under the DOS operation system and programs two 2764 EPROMs. The Intel Universal Programmer must be placed in ON-LINE mode.

Other files contained in the diskette are for compiling and locating the original TSMS program. Using these files, the original TSMS program can be changed or can be compiled for an iSBC 186/51. ‘TSMS.PLM’ is the original TSMS source program. ‘IO.PLM’ contains the IO driver needed when the TSMS program is run on the iSBC 186/51. INI186.PLM is the LANHIB initialization routine. LANHIB.BAT is the batch file that compiles, links, and locates the TSMS program and the LANHIB initialization routine. SBC.BAT compiles, links, and locates the TSMS program and the IO driver for the iSBC 186/51. IUPHIB.BAT programs two 2764s for the LANHIB. IUPSBC.BAT programs two 2764s for the iSBC 186/51.

Therefore, if the TSMS program is to be run on the LANHIB (Demo board), steps required are:

1. C: > LANHIB
2. C: > IUPHIB

If the TSMS program is to be run on the iSBC 186/51, steps required are:

1. C: > SBC
2. C: > IUPSBC

4.2 Capabilities and Limits of the TSMS Program

The TSMS program initializes the LANHIB Ethernet/Cheapesternet station by executing 82586’s Diagnose, Configure, IA-Setup, and MC-Setup commands. The program asks a series of questions in order to set up a linked list of these 82586 commands. After initialization is completed, the program automatically starts the 82586’s Receive Unit (monitoring capability). Transmissions are optional (traffic simulation capability).

The TSMS program has two modes of operation: Continuous mode and Interactive Command Execution mode. The program automatically gets into the Continuous mode after initialization. The Interactive Command Execution mode can be entered from the Continuous mode. Once entered in the Continuous mode, the software uses the format shown in Figure 12 to display information. Detailed description of each of these fields is as follows:

Host Address: host (station) address used in the most recently prepared IA-Setup command. The software simply writes the address stored in the IA-Setup command block with its least significant bit being in the most right position. Note that if the IA-Setup com-
Host Address: 00 AA 00 00 18 6D
Multicast Address(es): No Multicast Addresses Defined
Destination Address: FF FF FF FF FF FF
Frame Length: 118 bytes
Time Interval between Transmit Frames: 159.4 microseconds
Network Percent Load generated by this station: 35.7%
Transmit Frame Terminal Count: Not Defined
82586 Configuration Block: 08 00 26 00 60 00 F2 00 00 40

Figure 12. Continuous Mode Display

Multicast Address(es): multicast addresses used in the most recently prepared MC-Setup command. As in the case of host address, the software simply writes the addresses stored in the MC-Setup command block. Note that if the MC-Setup command was just set up and not executed, the addresses displayed in this field may not be the addresses stored in the 82586.

Destination Address: destination address stored in the transmit command block if AL-LOC=0. If AL-LOC=1, destination address is picked up from the transmit buffer. The least significant bit is in the most right position.

Frame Length: transmit frame byte count including destination address, source address, length, data, and CRC field.

Time Interval Between Transmit Frames: approximate time interval obtainable between transmit frames (Figure 13). The number is correct if there are no other stations transmitting on the network.

Network Percent Load Generated by This Station: approximate network percent load that is generated by this station (Figure 13). The number is correct if there are no other stations transmitting on the network.

Transmit Frame Terminal Count: number of frames this station will transmit before it stops network traffic load generation. If this station is transmitting indefinitely, this field will be 'Not Defined'.

82586 Configuration Block: configuration parameters used in the most recently prepared Configure command. As in the case of IA-Setup command, the software simply writes the parameters from the Configure command block. The least significant byte (FIFO Limit) of the configuration parameters is printed in the most left position.

# of Good Frames Transmitted: number of good frames transmitted. This is a snap shot of the 32-bit transmit frame counter. It is incremented only when both C and OK bits of the transmit command status are set after an execution. The counter is 32-bit wide.

# of Good Frames Received: number of good frames received. This is a snap shot of the 32-bit receive frame counter. It is incremented only when both C and OK bits of a receive frame descriptor status are set after a reception. The counter is 32-bit wide.

CRC Errors: number of frames that had a CRC error. This is a snap shot of the 16-bit CRC counter maintained by the 82586 in the SCB.

Alignment Errors: number of frames that had an alignment error. This is a snap shot of the 16-bit alignment counter maintained by the 82586 in the SCB.

No Resource Errors: number of frames that had a no resource error. This is a snap shot of the 16-bit no resource counter maintained by the 82586 in the SCB.

Receive Overrun Errors: number of frames that had a receive overrun error. This is a snap shot of the 16-bit receive overrun error counter maintained by the 82586 in the SCB.

If the station is actively transmitting, # of good frames transmitted should be incrementing. If the station is actively receiving frames, # of good frames received should be incrementing. In this continuous mode, a user can see the activities of the network.
Hitting any key on the keyboard while the program is running in the Continuous mode will exit the mode. The program will respond with a message ‘Enter Command (H for Help) → ’. In this Interactive Command Execution mode, a user can set up any one of the 82586 action commands and/or execute any one of the 82586 SCB control commands. Setting up a Dump command and executing a SCB Command Unit Start command will, for example, execute the Dump command. Display commands are also available to see the contents of the 82586's data structure blocks. A display command will enable a user to see the contents of the 82586's dump (see Section 6.3).

Typing ‘E’ after ‘Enter command (H for help) → ’, executing a SCB Command Unit Start command with a transmit command, or executing a SCB Receive Unit Start command will exit the Interactive Command Execution mode. The program will be back in the Continuous mode. Using this Interactive Command Execution mode, one can, for example, reconfigure the station and come back to the Continuous mode. Section 6 lists actual example executions of the TSMS program.

The TSMS program should be run in an 8 MHz system. The software running at 8 MHz with a maximum of 2 wait states has been tested and verified to be able to receive back-to-back frames separated by 9.6 microseconds and still keep track of the correct number of frames received. This capability, for example, can be used to find out exactly how many frames a new station in the network had transmitted.

The software does not perform extensive loopback tests and hardware diagnostics during the initialization. A loopback operation can be performed interactively in the Interactive Command Execution mode.

The software allows a user to set up only 8 multicast addresses maximum. It is not possible with this program to set up more than 8 multicast addresses.

The command chaining feature of the 82586 is not used in the Interactive Command Execution mode. Each command setup performed by a ‘S’ command after ‘Enter command (H for help) → ’ sets up a command with its EL bit set, I bit reset, and S bit reset. Diagnose, Configure, IA-Setup, and MC-Setup commands are chained together during the initialization routine and executed at once with only one CA.

The software sets up 5 Receive Frame Descriptors linked in a circular list. Therefore, a user can see only the last 5 frames the station has received. It also sets up 5 receive buffers, each being 1514 bytes long, linked in circle. Therefore, the 82586 never goes into the NO RESOURCES state.

4.3 Example Executions of the TSMS Program

This section presents three example executions of the TSMS program. When the TSMS program needs a command to be typed, it asks a question with ‘ → ’. Anything after ‘ → ’ is what a user needs to type in on the keyboard. To switch from the continuous mode to the interactive command execution mode, type any key on the keyboard.

4.3.1 EXAMPLE 1: EXTERNAL LOOPBACK EXECUTION

In this example, 500 external loopback transmissions and receptions are executed (Figure 14). In order for the software to process each loopback properly, a large delay was given between transmissions.

4.3.2 EXAMPLE 2: FRAME RECEPTION IN PROMISCUOUS MODE

The 82586 is configured to receive any frame that exists in the network (Figure 15). In this example, the station received 100 frames.

4.3.3 EXAMPLE 3: 35.7% NETWORK TRAFFIC LOAD GENERATION

The station is programmed to transmit 118 byte long frames with a time interval of 159.4 microseconds in between (Figure 16). The network load is about 35.7 percent if no other stations are transmitting in the network.

A key was hit to enter the Interactive Command Execution mode. In that mode, a Dump command was executed and the result was displayed. After the Dump execution, a transmit command was set up again and the station was put in the Continuous mode.
Initialization begun

Configure command is set up for default values.
Do you want to change any bytes? (Y or N) ==> Y
Enter byte number (1 - 11) ==> 4
Enter byte 4 (4H) ==> A6H
Any more bytes? (Y or N) ==> Y
Enter byte number (1 - 11) ==> 11
Enter byte 11 (BH) ==> 6
Any more bytes? (Y or N) ==> N
Configure the 586 with the prewired board address ==> N
Enter this station's address in Hex ==> 000000002200
You can enter up to 8 Multicast Addresses.
Would you like to enter a Multicast Address? (Y or N) ==> N
You entered 0 Multicast Address(es).

Would you like to transmit?
Enter a Y or N ==> Y
Enter a destination address in Hex ==> 000000002200
Enter TYPE ==> 0
How many bytes of transmit data?
Enter a number ==> 2
Transmit Data is continuous numbers (0, 1, 2, 3, ...)
Change any data bytes? (Y or N) ==> N
Enter a delay count ==> 10000000000
The number is too big.
It has to be less than or equal to 65535 (FFFFH).
Enter a number ==> 60000
Setup a transmit terminal count? (Y or N) ==> Y
Enter a transmit terminal count ==> 500
Destination Address: 00 00 00 00 22 00
Frame Length: 20 bytes
Time Interval between Transmit Frames: 30.18 milliseconds
Network Percent Load generated by this station: .0 %
Transmit Frame Terminal Count: 500

Good enough? (Y or N) ==> Y
Receive Unit is active.

Figure 14. External Loopback Execution
---Transmit Command Block---
0000 at 033E
8004
FFFF
034E
2200
0000
0000
0000

Hit <CR> to continue

transmission started!

************************** Station Configuration **************************

Host Address: 00 00 00 00 22 00
Multicast Address(es): No Multicast Addresses Defined
Destination Address: 00 00 00 00 22 00
Frame Length: 20 bytes
Time Interval between Transmit Frames: 30.18 milliseconds
Network Percent Load generated by this station: .0 %
Transmit Frame Terminal Count: 500
82586 Configuration Block: 08 00 A6 00 60 00 F2 00 00 06

************************** Station Activities **************************

<table>
<thead>
<tr>
<th># of Good Frames</th>
<th># of Good Received</th>
<th>CRC Errors</th>
<th>Alignment Errors</th>
<th>No Resource Errors</th>
<th>Receive Overrun Errors</th>
</tr>
</thead>
<tbody>
<tr>
<td>500</td>
<td>500</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Figure 14. External Loopback Execution (Continued)
Traffic Simulator and Monitor Station Program

Initialization begun

Configure command is set up for default values.
Do you want to change any bytes? (Y or N) ==> Y
Enter byte number (1 - 11) ==> 9
Enter byte 9 (9H) ==> 1
Any more bytes? (Y or N) ==> N
Configure the 586 with the prewired board address ==> Y
You can enter up to 8 Multicast Addresses.
Would you like to enter a Multicast Address? (Y or N) ==> N
You entered 0 Multicast Address(es).

Would you like to transmit?
Enter a Y or N ==> N

Receive Unit is active.

**************************** Station Configuration *************************
Host Address: 00 AA 00 00 18 6D
Multicast Address(es): No Multicast Addresses Defined
82586 Configuration Block: 08 00 26 00 60 00 F2 01 00 40

**************************** Station Activities **************************
<table>
<thead>
<tr>
<th># of Good Frames Transmitted</th>
<th># of Good CRC Errors</th>
<th>Alignment No</th>
<th>Receive Resource Errors</th>
<th>Overrun Errors</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>100</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Enter command (H for help) ==> D

Command Block or Receive Area? (R or C) ==> R

Frame Descriptors:
- 4000 at 036C A000 at 0382 A000 at 0398 A000 at 03AE A000 at 03C4
- 0000 0000 0000 0000 0000
- 0382 0382 0382 0382 0382
- 03DA 03E4 03EE 03EE 03EE
- 2200 2200 2200 2200 2200
- 2200 2200 2200 2200 2200
- 0000 0000 0000 0000 0000

Figure 15. Frame Reception in Promiscuous Mode
Figure 15. Frame Reception in Promiscuous Mode (Continued)
Receive Buffer 4:
002C:18F4 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
002C:1904 10 11 12 13 14 15 16 17 18 19 1A 1B 1C 1D 1E 1F
002C:1914 20 21 22 23 24 25 26 27 28 29 2A 2B 2C 2D 2E 2F
002C:1924 30 31 32 33 34 35 36 37 38 39 3A 3B 3C 3D 3E 3F
002C:1934 40 41 42 43 44 45 46 47 48 49 4A 4B 4C 4D 4E 4F
002C:1944 50 51 52 53 54 55 56 57 58 59 5A 5B 5C 5D 5E 5F
002C:1954 60 61 62 63

Hit <CR> to continue

Enter command (H for help) ==> E

***************************** Station Configuration *****************************
Host Address: 00 AA 00 00 18 6D
Multicast Address(es): No Multicast Addresses Defined
82586 Configuration Block: 08 00 26 00 60 00 F2 01 00 40

***************************** Station Activities *****************************

<table>
<thead>
<tr>
<th># of Good Frames</th>
<th># of Good Received</th>
<th>CRC Errors</th>
<th>Alignment Errors</th>
<th>No Resource Errors</th>
<th>Receive Overrun Errors</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>100</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Figure 15. Frame Reception in Promiscuous Mode (Continued)
Initialization begun

Configure command is set up for default values.
Do you want to change any bytes? (Y or N) ==> N
Configure the 586 with the prewired board address ==> Y
You can enter up to 8 Multicast Addresses.
Would you like to enter a Multicast Address? (Y or N) ==> N
You entered 0 Multicast Address(es).

Would you like to transmit?
Enter a Y or N ==> Y
Enter a destination address in Hex ==> FFFFFFFF

Enter TYPE ==> 0
How many bytes of transmit data?
Enter a number ==> 100
Transmit Data is continuous numbers (0, 1, 2, 3, ...)
Change any data bytes? (Y or N) ==> N

Enter a delay count ==> 0
Setup a transmit terminal count? (Y or N) ==> N

Destination Address: FF FF FF FF FF FF
Frame Length: 118 bytes
Time Interval between Transmit Frames: 159.4 microseconds
Network Percent Load generated by this station: 35.7 %
Transmit Frame Terminal Count: Not Defined

Good enough? (Y or N) ==> Y

Receive Unit is active.

---Transmit Command Block---
0000 at 033E
8004
FFFF
034E
FFFF
FFFF
FFFF
FFFF
0000

Hit <CR> to continue

Figure 16. 35.7% Network Load Generation
transmission started!

********** Station Configuration **********

Host Address: 00 AA 00 00 18 6D
Multicast Address(es): No Multicast Addresses Defined
Destination Address: FF FF FF FF FF FF
Frame Length: 118 bytes
Time Interval between Transmit Frames: 159.4 microseconds
Network Percent Load generated by this station: 35.7 %
Transmit Frame Terminal Count: Not Defined
82586 Configuration Block: 08 00 26 00 60 00 F2 00 00 40

********** Station Activities **********

| # of Good | # of Good | CRC | Alignment | No | Receive |
| Frames    | Errors    |     | Errors    |    | Errors  |
| Transmitted| Received |     |           |    |         |
| 10459     | 0         | 0   | 0         | 0  | 0       |

Enter command (H for help) ==> H

Commands are:
S - Setup CB           D - Display RFD/CB
P - Print SCB          C - SCB Control CMD
L - ESI Loopback On   N - ESI Loopback Off
A - Toggle Number Base Z - Clear Tx Frame Counter
Y - Clear Rx Frame Counter E - Exit to Continuous Mode

Enter command (H for help) ==> S

Enter command block type (H for help) ==> H

Command block type:
N - Nop     I - IA Setup
C - Configure M - MA Setup
T - Transmit R - TDR
D - Diagnose S - Dump Status
H - Print this message

Enter command block type (H for help) ==> S

Enter command (H for help) ==> C

Do you want to enter any SCB commands? (Y or N) ==> Y
Enter CUC ==> 1
Enter RES bit ==> 0
Enter RUC ==> 0
Issued Channel Attention

Enter command (H for help) ==> D

Figure 16. 35.7% Network Load Generation (Continued)
Command Block or Receive Area? (R or C) ==> C
---Dump Status Command Block---
A000 at 0364
8006
FFFF
27D6

Dump Status Results
at 27D6
00 E8 3F 26 08 60 00 FA 00 00 40 FF 6D 18 00 00
AA 00 40 20 00 00 00 00 FF FF FF FF B5 9E EE CF
62 63 3F 80 00 00 00 00 00 00 00 00 00 00 00 70 03 06 00
DC 05 00 00 0C 04 DC 05 E4 03 DA 03 DA 03 78 05
82 03 6C 03 F8 03 64 80 D6 27 E8 21 FF FF 4E 03
06 80 FF FF 64 03 00 00 D2 02 00 00 00 00 00 00
00 00 D6 27 00 01 00 28 00 00 00 00 30 26 00 00
20 00 40 06 30 01 00 00 90 00 10 01 00 00 6C 03
00 00 6A 03 0E 00 6C 28 00 00 74 03 00 00 00 00
00 00 00 00 00 00 00 00

Enter command (H for help) ==> S
Enter command block type (H for help) ==> T
Enter a destination address in Hex ==> FFFFFFFFFFFFFF

Enter TYPE ==> 0
How many bytes of transmit data?
Enter a number ==> 100
Transmit Data is continuous numbers (0, 1, 2, 3, ...)
Change any data bytes? (Y or N) ==> N

Enter a delay count ==> 0

Setup a transmit terminal count? (Y or N) ==> N

Destination Address: FF FF FF FF FF FF
Frame Length: 118 bytes
Time Interval between Transmit Frames: 159.4 microseconds
Network Percent Load generated by this station: 35.7%
Transmit Frame Terminal Count: Not Defined

Good enough? (Y or N) ==> Y

Enter command (H for help) ==> C
Do you want to enter any SCB commands? (Y or N) ==> Y
Enter CUC ==> 1
Enter RES bit ==> 0
Enter RUC ==> 0
Issued Channel Attention

Figure 16. 35.7% Network Load Generation (Continued)
5.0 IN CASE OF DIFFICULTY

This section presents methods of troubleshooting ("de-bugging") a LANHIB board. When a LANHIB board is powered up with the TSMS program stored in EPROMs, it should display "TRAFFIC SIMULATOR AND MONITOR STATION PROGRAM" message on a terminal screen. If the message is not displayed, the board has to be debugged. Section 5.1 describes basic 80186/82586 system troubleshooting procedures. Section 5.2 is for troubleshooting 82501 and 82502 circuits. After the 80186/82586 system is debugged, the 82501/82502 circuits have to be tested.

5.1 Troubleshooting 80186/82586 System

Shown in Figure 17 is a flow chart for troubleshooting 80186/82586 system. The procedure requires an oscilloscope. A logic analyzer is needed if problems appear to be serious. The procedures will debug the board to the point where the 82530 is initialized properly. If the 82530 can be initialized properly, ROM and RAM interfaces must be functioning. Board initialization routines (INI186.PLM) linked to the TSMS program requires ROM and RAM accesses. Since the 82586 shares most of the system with the 80186, no special debugging is required for the 82586. Wiring of all 82586 parallel signal pins should, however, be checked.

The flow chart branches to two major paths after the first decision box. One path debugs the RS-232 channel and the other debugs the 80186/82586 system. The waveform of the TRXCB output of the 82530 determines which path to be taken. If the 82530 is getting programmed properly, there should be 153.6 KHz (1/f = 6.51 µs) clock on this output pin. If there is a clock, the problem is probably in the RS-232 interface. If there is no clock, then the system has to be debugged using a logic analyzer.

5.2 Troubleshooting 82501/82502 Circuits

If the TSMS program runs on the LANHIB but the 82586 is not able to transmit or receive, there must be a problem in 82501/82502 circuits. The flow chart in Figure 19 will guide troubleshooting in these circuits. An oscilloscope is required.

The board should be configured to Cheapernet and disconnected from the network. Two terminators will be required to terminate a "T" BNC connector providing an effective load resistance of 25Ω to the 82502. The 82586 must have the system and transmit clocks running upon reset. Since the transmit clock is generated by the 82501, the 82501 transmit clock output pin (pin 16) should be checked. The TSMS program executes 82586's Diagnose, Configure, IA-Setup, and MC-Setup commands during initialization. If the 82586 has active CR (Carrier Sense) signal, it cannot complete execution of these commands. The 82501 should, therefore, be checked if it is generating inactive CR signal to the 82586 after power up. The LANHIB powers up the 82501 in non-loopback mode.
After making sure that the 82501 is generating proper signals to the 82586, the TSMS program is restarted with an initialization shown in Figure 20. The 82586 is configured to EXT-LPBK = 1, TONO-CRS = 1, and MIN-FRM-LEN = 6. The chip is also loaded with a destination address identical to the source address. If there are no problems in the 82501/82502 circuits, the station will be receiving its own transmitted frames. If problems exist, the station will only be transmitting. Since the 82586 is configured to TONO-CRS (Transmission On NO Carrier Sense), the chip will keep transmitting regardless of the state of carrier sense. The 82501/82502 circuits can then be probed with an oscilloscope at the locations indicated in Figure 21. Probing will catch problems like wiring mistakes, missing load resistors, etc.

Once the station is debugged, it can be connected to the network. If there is a problem in the network, the 82586’s TDR command can be used to find the location and nature of the problem.

Figure 17. Flowchart for 80186/82586 System Troubleshooting
CONNECT A LOGIC ANALYZER ON THE MULTIPLEXED BUS.
1. CONNECT AD15-ADO, ALE, RD, WR, ROMHI, ROMLO, RAMHI, RAMLO, AND CS PIN(PIN 33) OF 82530.
2. USE CLKOUT OF 80186 TO CLOCK THE LOGIC ANALYZER. SAMPLE DATA ON RISING EDGES.
3. TRIGGER THE LOGIC ANALYZER ON ALE BECOMING HIGH.

SHOWN IN FIGURE 18 IS AN EXAMPLE OF A LOGIC ANALYZER TRACE. COMPARE WHAT'S OBTAINED TO THE ONE IN FIGURE 18.

IF DIFFERENT, POSSIBLE PROBLEMS ARE:
1. HIGH BYTE EPROM AND LOW BYTE EPROM ARE SWAPPED.
2. ADDRESS/DATA LINES ARE NOT CONNECTED PROPERLY.
3. ADDRESS DECODE PAL IS NOT PROGRAMMED PROPERLY.
   etc.

CHECK IF 82530 IS GETTING INITIALIZED PROPERLY ON THE LOGIC ANALYZER. TRY OTHER LOGIC ANALYZER TRIGGERING EVENT, e.g. CS PIN(PIN 33) OF 82530 BECOMING LOW.

MAKE SURE THERE IS 153.6 KHZ(1/f=6.51 µsec.) SQUARE WAVE ON TRXCB(PIN 26) OF 82530.

CHECK RS-232 DRIVER & RECEIVER CHIPS. ARE THEY CONNECTED PROPERLY? NOTE THAT THE 1488(75188) REQUIRES +12V & -12V AND THAT THE 1489(75189) REQUIRES ONLY +5V.

CHECK RS-232 DCE & DTE CONNECTIONS. THE LANHIB IS A DCE AND AN ASCII TERMINAL IS A DTE. ONLY PIN2(TXD), 3(RXD), AND 7(GROUND) ARE USED.

CHECK CONFIGURATION OF THE ASCII TERMINAL. BAUD RATE SHOULD BE SET TO 9600. ALSO 8 BITS/CHAR, NO PARITY, AND 2 STOP BITS/CHAR.

START DEMO
Figure 18. Example of Logic Analyzer Trace

Logics are:

- AD15–AD0
- ALE
- RD#
- WR#
- ROMHI#
- ROMLO#
- RAMHI#
- RAMLO#
- CS# Pin (Pin 33) of 82530

---

0097 00 41 01001111
0098 00 41 01001111
0099 00 41 01001111
TRIG 00 41 11101111 ...... Logic Analyzer is triggered on ALE = HI.
0101 FF F0 01001111 ...... 80186 jumps to FFFOH after reset.
0102 06 EA 00101111 ...... JMP instruction (Direct Intersegment)
0103 06 EA 00101111 SEGMENT OFFSET = 0006H
0104 06 EA 00101111 SEGMENT SELECTOR = FFCOH
0105 06 EA 00101111 (80186 inserts 3 WAIT states before
0106 06 EA 00101111 UMCS register is programmed.)
0107 06 EA 11101111
0108 FF F2 01101111
0109 CO 40 00101111
0110 CO 00 00101111
0111 CO 00 00101111
0112 CO 00 00101111
0113 CO 00 00101111
0114 CO 00 11101111
0115 FF F4 01101111
0116 FF FF 00101111
0117 FF FF 00101111
0118 FF FF 00101111
0119 FF FF 00101111
0120 FF FF 00101111
0121 FF FF 11101111
0122 FF F6 01101111
0123 00 40 00101111
0124 00 00 00101111
0125 00 00 00101111
0126 00 00 00101111
0127 00 00 00101111
0128 00 00 11101111
0129 FC 06 01101111 ...... Jumped to FC06H
0130 2E FA 00101111
0131 2E FA 00101111
0132 2E FA 00101111
0133 2E FA 00101111
0134 2E FA 00101111
0135 2E FA 11101111
0136 FC 06 01101111
0137 16 BE 00101111
0138 16 BE 00101111

Figure 18. Example of Logic Analyzer Trace
DISCONNECT COAX. PUT TERMINATORS ON BOTH ENDS OF "T" CONNECTOR. MAKE SURE THE BOARD IS CONFIGURED TO CHEAPERNET.

UPON POWER UP, DOES 82501 GENERATE:
1. 10 MHz TxC AND RxC TO 82586?
2. INACTIVE CRS TO 82586?

RUN TSMS PROGRAM.

WHEN A TRANSMISSION IS ATTEMPTED, DOES THE TSMS PROGRAM DISPLAY "NO CARRIER SENSE" MESSAGE?

POWER DOWN AND RE-START TSMS PROGRAM WITH 82586 CONFIGURED TO:
1. EXT-LPBK = 1
2. TONG-CRS = 1
3. MIN-FRM-LEN = 6
EXECUTE LOOPBACKS BY USING DESTINATION ADDR SAME AS SOURCE ADDR. TRANSMIT ONLY A FEW DATA BYTES.

AN EXAMPLE EXECUTION IS SHOWN IN FIGURE 20.

IF THE STATION IS NOT RECEIVING WHILE IT'S TRANSMITTING, THERE IS A PROBLEM. PROBE SIGNALS AT LOCATIONS SHOWN IN FIGURE 21. IT'S PROBABLY A WIRING PROBLEM.

MAKE SURE THE 82501 IS POWERED UP IN NON-LOOPBACK MODE.

BOARD SHOULD BE FUNCTIONAL.
**Traffic Simulator and Monitor Station Program**

Initialization begun

Configure command is set up for default values.
Do you want to change any bytes? (Y or N) ==> Y
Enter byte number (1 - 11) ==> 4
Enter byte 4 (4H) ==> A6H
Any more bytes? (Y or N) ==> Y
Enter byte number (1 - 11) ==> 9
Enter byte 9 (9H) ==> 08H
Any more bytes? (Y or N) ==> Y
Enter byte number (1 - 11) ==> 11
Enter byte 11 (BH) ==> 6
Any more bytes? (Y or N) ==> N
Configure the 586 with the prewired board address ==> N
Enter this station's address in Hex ==> 000000002200
You can enter up to 8 Multicast Addresses.
Would you like to enter a Multicast Address? (Y or N) ==> N
You entered 0 Multicast Address(es).

Would you like to transmit?
Enter a Y or N ==> Y
Enter a destination address in Hex ==> 000000002200

Enter TYPE ==> 0
How many bytes of transmit data?
Enter a number ==> 2
Transmit Data is continuous numbers (0, 1, 2, 3, ...)
Change any data bytes? (Y or N) ==> N
Enter a delay count ==> 0
Setup a transmit terminal count? (Y or N) ==> N

Destination Address: 00 00 00 00 22 00
Frame Length: 20 bytes
Time Interval between Transmit Frames: 159.4 seconds
Network Percent Load generated by this station: 11.0 %
Transmit Frame Terminal Count: Not Defined

Good enough? (Y or N) ==> Y

Figure 20. TSMS Initialization for 82501/82502 Circuits Troubleshooting
Figure 21. Probing 82501/82502 Circuits

NOTE:
Numbers are probing sequence.
APPENDIX A
LANHIB SCHEMATICS
PARTS LIST
PAL EQUATIONS
DIP SWITCH SETTINGS
WIRE WRAP SERVICES
## PARTS LIST

<table>
<thead>
<tr>
<th>REFERENCE</th>
<th>DESCRIPTION</th>
<th>PART NO.</th>
<th>CODE</th>
<th>QTY</th>
</tr>
</thead>
<tbody>
<tr>
<td>U1</td>
<td>IC</td>
<td>74F932</td>
<td>OBD</td>
<td>1</td>
</tr>
<tr>
<td>U2</td>
<td>IC</td>
<td>74F936</td>
<td>OBD</td>
<td>1</td>
</tr>
<tr>
<td>U3, U4</td>
<td>IC</td>
<td>74L5245</td>
<td>OBB</td>
<td>2</td>
</tr>
<tr>
<td>U5</td>
<td>IC</td>
<td>74F932</td>
<td>OBD</td>
<td>1</td>
</tr>
<tr>
<td>U6</td>
<td>IC/CAP</td>
<td>74L5737</td>
<td>OBD</td>
<td>2</td>
</tr>
<tr>
<td>U7</td>
<td>IC</td>
<td>81168</td>
<td>INT</td>
<td>1</td>
</tr>
<tr>
<td>U8</td>
<td>IC</td>
<td>82596</td>
<td>INT</td>
<td>1</td>
</tr>
<tr>
<td>U9</td>
<td>IC</td>
<td>1618</td>
<td>OBD</td>
<td>1</td>
</tr>
<tr>
<td>U10</td>
<td>IC</td>
<td>74L572</td>
<td>OBD</td>
<td>1</td>
</tr>
<tr>
<td>U11</td>
<td>IC</td>
<td>74L574</td>
<td>OBD</td>
<td>2</td>
</tr>
<tr>
<td>U12, U27</td>
<td>IC</td>
<td>74M574</td>
<td>OBD</td>
<td>1</td>
</tr>
<tr>
<td>U13</td>
<td>IC</td>
<td>74L5165</td>
<td>OBD</td>
<td>1</td>
</tr>
<tr>
<td>U14</td>
<td>IC</td>
<td>82591</td>
<td>INT</td>
<td>1</td>
</tr>
<tr>
<td>U15</td>
<td>Pulse Transformer Pack</td>
<td>PE64482</td>
<td>PE</td>
<td>1</td>
</tr>
<tr>
<td>U16</td>
<td>IC</td>
<td>82582</td>
<td>INT</td>
<td>1</td>
</tr>
<tr>
<td>U17</td>
<td>DC/DC Converter</td>
<td>PE64369</td>
<td>PE</td>
<td>1</td>
</tr>
<tr>
<td>U18, U20</td>
<td>IC</td>
<td>64K-bit EPROM</td>
<td>2764-28</td>
<td>INT</td>
</tr>
<tr>
<td>U21</td>
<td>IC</td>
<td>74SC88</td>
<td>OBD</td>
<td>1</td>
</tr>
<tr>
<td>U22, U25</td>
<td>IC</td>
<td>51626-15</td>
<td>HIT</td>
<td>2</td>
</tr>
<tr>
<td>U24</td>
<td>IC</td>
<td>51626-15</td>
<td>HIT</td>
<td>2</td>
</tr>
<tr>
<td>U26</td>
<td>IC</td>
<td>74L544</td>
<td>OBD</td>
<td>1</td>
</tr>
<tr>
<td>U27</td>
<td>IC</td>
<td>82530</td>
<td>INT</td>
<td>1</td>
</tr>
<tr>
<td>U28</td>
<td>IC</td>
<td>1489</td>
<td>OBD</td>
<td>1</td>
</tr>
<tr>
<td>U31</td>
<td>IC</td>
<td>1489</td>
<td>OBD</td>
<td>1</td>
</tr>
<tr>
<td>U32</td>
<td>IC</td>
<td>74L528</td>
<td>OBD</td>
<td>2</td>
</tr>
<tr>
<td>R1-R3, R6</td>
<td>Resistor</td>
<td>10K ohm</td>
<td>1/4W%</td>
<td>5%</td>
</tr>
<tr>
<td>R9, R18</td>
<td>Resistor</td>
<td>100 ohm</td>
<td>1/4W%</td>
<td>5%</td>
</tr>
<tr>
<td>R11</td>
<td>Resistor</td>
<td>1K ohm</td>
<td>1/4W</td>
<td>5%</td>
</tr>
<tr>
<td>R13-R16</td>
<td>Resistor</td>
<td>45 ohm</td>
<td>1/8W, 1%</td>
<td>COL</td>
</tr>
<tr>
<td>R17</td>
<td>Resistor</td>
<td>100 ohm</td>
<td>1/8W, 5%</td>
<td>COL</td>
</tr>
<tr>
<td>R18, R22</td>
<td>Resistor</td>
<td>249 ohm</td>
<td>1/8W, 5%</td>
<td>COL</td>
</tr>
<tr>
<td>R20, R26</td>
<td>Resistor</td>
<td>1K ohm</td>
<td>1/4W, 5%</td>
<td>COL</td>
</tr>
<tr>
<td>C1, C2</td>
<td>Capacitor</td>
<td>100uF, 10V</td>
<td>COL</td>
<td>OBD</td>
</tr>
<tr>
<td>C3-C5</td>
<td>Capacitor</td>
<td>100uF, 5%</td>
<td>COL</td>
<td>OBD</td>
</tr>
<tr>
<td>C6</td>
<td>Capacitor</td>
<td>220uF, 5%</td>
<td>COL</td>
<td>OBD</td>
</tr>
<tr>
<td>C7</td>
<td>Capacitor</td>
<td>1uF, 50V</td>
<td>COL</td>
<td>OBD</td>
</tr>
<tr>
<td>C11, C12</td>
<td>Capacitor</td>
<td>0.01uF, 50V</td>
<td>COL</td>
<td>OBD</td>
</tr>
<tr>
<td>C18, C9</td>
<td>Capacitor</td>
<td>0.022uF, 50V</td>
<td>COL</td>
<td>OBD</td>
</tr>
<tr>
<td>CR1</td>
<td>Diode</td>
<td>IN5282</td>
<td>OBD</td>
<td>1</td>
</tr>
<tr>
<td>CR2, CR3</td>
<td>Diode</td>
<td>IN5282</td>
<td>OBD</td>
<td>2</td>
</tr>
<tr>
<td>V1</td>
<td>Parallel Resonant Crystal</td>
<td>16N Hz</td>
<td>COL</td>
<td>OBD</td>
</tr>
<tr>
<td>V2</td>
<td>Parallel Resonant Crystal</td>
<td>20M Hz</td>
<td>COL</td>
<td>OBD</td>
</tr>
</tbody>
</table>

### POWER SUPPLY CONNECTIONS

- **+12V**
- **+5V**
- **-2uF, 22uF, 22uF**
- **+12V**
- **-2uF, 22uF**
- **-20V**
- **-12V**

### NOTES:
1. The board requires +5V, +12V, and -12V multibus power pins for these voltages and ground are shown above.
2. Each IC should have a 0.1 uF capacitor between power pin and ground pin. Parts list does not include decoupling capacitors.
3. Parts list does not include decoupling capacitors.

### MFR CODE

<table>
<thead>
<tr>
<th>MFR N CODE</th>
<th>MANUFACTURE</th>
<th>LOCATION</th>
</tr>
</thead>
<tbody>
<tr>
<td>INT</td>
<td>INTEL CORPORATION</td>
<td>SANTA CLARA, CA</td>
</tr>
<tr>
<td>HIT</td>
<td>HITACH AMERICA LTD</td>
<td>SAN JOSE, CA</td>
</tr>
<tr>
<td>OBD</td>
<td>ORDER BY DESCRIPTION</td>
<td>ANY COMMERCIAL (COL) SOURCE</td>
</tr>
<tr>
<td>PE</td>
<td>PULSE ENGINEERING</td>
<td>SAN DIEGO, CA</td>
</tr>
<tr>
<td>RCB</td>
<td>RCB COMPONENTS INC</td>
<td>MANCHESTER, NH</td>
</tr>
<tr>
<td>TI</td>
<td>TEXAS INSTRUMENTS</td>
<td>DALLAS, TX</td>
</tr>
</tbody>
</table>

292010-78
Module Addr_dec
Title 'LANHIB Address Decode Logic
Kiyoshi Nishide Intel Corp. March, 1986'

*Declarations

PAL1 device '
A0, A14, A15 pin 1, 2, 3;
A16, A17, A18 pin 4, 5, 6;
A19, BHE pin 7, 8;
HLDA, S2 pin 9, 11;
RAMLO, RAMHI pin 18, 17;
ROMLO, ROMHI pin 19, 12;
ROM pin 13;
R104 pin 16;

Equations

!ROMHI = A15 & A16 & A17 & A18 & A19 & (HLDA # S2) & R104;
!ROMLO = !A15 & A16 & A17 & A18 & A19 & (HLDA # S2) & R104;
!ROM = A17 & A18 & A19 & (HLDA # S2) & !R104;
!RAMHI = !A14 & !A15 & !A16 & !A17 & !A18 & !A19 & !BHE & (HLDA # S2);
!RAMLO = !A0 & !A14 & !A15 & !A16 & !A17 & !A18 & !A19 & (HLDA # S2);

End Addr_dec

PAL Equations
DIP SWITCH SETTINGS FOR VARIOUS OPERATIONS

“1” indicates ON (Switch is closed).
“0” indicates OFF (Switch is open).
“X” indicates Don’t Care.

1. To configure the board to Ethernet or Cheapernet:

<table>
<thead>
<tr>
<th>SW3 87654321</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ethernet</td>
<td>XX000000</td>
</tr>
<tr>
<td>Cheapernet</td>
<td>XX111111 Transceiver Cable should not be connected.</td>
</tr>
</tbody>
</table>

2. To run the TSMS program or the Data Link Driver program:

<table>
<thead>
<tr>
<th>SW4 87654321</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>TSMS Program</td>
<td>XXXX0001 TSMS program uses the 82530 in Asynchronous Polling mode. Data Link Driver program uses the 825830 in Asynchronous Polling and Vectored Interrupt modes.</td>
</tr>
<tr>
<td>Data Link Driver Program</td>
<td>XXXX0001</td>
</tr>
</tbody>
</table>

3. To select the 2764-20 EPROMs or 27210 EPROM:

<table>
<thead>
<tr>
<th></th>
<th>SW3 87654321</th>
</tr>
</thead>
<tbody>
<tr>
<td>2764-20 EPROMs</td>
<td>0XXXXXXX</td>
</tr>
<tr>
<td>27210 EPROM</td>
<td>1XXXXXXX</td>
</tr>
</tbody>
</table>

4. Dip Switch Setting Examples:

<table>
<thead>
<tr>
<th>SW3 87654321</th>
<th>SW4 87654321</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0X1111111</td>
<td>TSMS Program from the 2764-20 EPROMs in Cheapernet Configuration</td>
</tr>
<tr>
<td></td>
<td>0X000000</td>
<td>TSMS Program from the 2764-20 EPROMs in Ethernet Configuration</td>
</tr>
<tr>
<td></td>
<td>1X1111111</td>
<td>TSMS Program or the Data Link Driver program from the 27210 EPROM in Cheapernet Configuration</td>
</tr>
<tr>
<td></td>
<td>1X0000000</td>
<td>TSMS Program or the Data Link Driver program from the 27210 EPROM in Ethernet Configuration</td>
</tr>
</tbody>
</table>

5. Dip Switch SW2 programs the number of wait states for the 82586 (see Table 3).
COMPANIES OFFERING WIRE WRAP SERVICES

AUGAT
Interconnection Systems Division
40 Perry Avenue
P.O. Box 1037
Attleboro, MA 02703
(617) 222-2202

100935 South Wilcrest Drive
Houston, TX 77099
(713) 495-3100

Automation Delectronics Corporation
1650 Locust Avenue
Bohemia, NY 11716
(516) 567-7007

dataCon, Inc.
Eastern Division
60 Blanchard Road
Burlington, MA 01803
(617) 273-5800

Mid-Western Division
502 Morse Avenue
Schaumburg, IL 60193
(312) 529-7690

Western Division
20150 Sunburst Street
Chatsworth, CA 91311-6280
(818) 700-0600

South-Western Division
1829 Monetary Lane
Carrollton, TX 75006
(214) 245-6161

European Division
In der Klinge 5
D-7100 Heilbronn, West Germany
(01731) 217 12

DATAWRAP
37 Water Street
Wakefield, MA 01880
(617) 938-8911

Elma/EMS
A Division of Sandberg Industries
Berkshire Industrial Park
Bethel, CT 06801
(203) 797-9711

1851 Reynolds Avenue
Irvine, CA 92714
(714) 261-9473

3042 Scott Boulevard
Santa Clara, CA 95054
(408) 970-8874

WRAPEX Corporation
96 Mill Street
Woonsocket, RI 02895
(401) 769-3805
APPENDIX B
SOFTWARE LISTINGS—TSMS PROGRAM AND
LANHIB INITIALIZATION ROUTINE
Traffic Simulator/Monitor Station Program

This software can be conditionally compiled to work on the iSBC 1B6/51 or on the LANHIB if "set(SBC18651)" is added to the compiler call statement. This source program will be compiled for the iSBC18651. */

tsms:
do;
  declare main label public.
  
  /* literals */
  $IF SBC18651$
  declare lit literally 'literally',
  true    lit '1',
  false   lit '0',
  forever lit 'while 1',
  ISCP$LOC$LO lit 'OFFFOH',
  ISCP$LOC$HI lit '0',
  SCB$BASE$LO lit '0',
  SCB$BASE$HI lit '0',
  CAS$PORT lit 'OCBH',
  BOARD$ADDRESS$BASE lit 'OF0H',
  INT$TYPE$586 lit '20H',
  INT$TYPE$TIMERO lit '30H',
  INT$CTL$TIMERO lit 'OFF32H',
  INT$7  lit '27H',
  PIC$MASK$130 lit '0E2H',
  PIC$MASK$1B6 lit 'OFF2BH',
  ENABLE$586 lit '0FEH',
  ENABLE$5B6$1B6 lit '0EEH',
  PIC$EOI$130 lit '0EOH',
  EDI$CMD$130 lit '64H',
  EDI$CMD$4$130 lit '64H',
  PIC$EOI$1B6 lit 'OFF22H',
  EDI$CMD$0$1B6 lit '0',
  PIC$VTR$1B6 lit 'OFF20H',

292010–31
declare scp structure
  (sysbus byte,
   unused (5) byte,
   iscp$addr$lo word,
   iscp$addr$hi word)
at (OFF56H) data (0, 0, 0, 0, 0, 0, ISCP$LOC$LO, ISCP$LOC$HI),

/* Intermediate System Configuration Pointer */
5 1 declare iscp*ptr pointer,

iscp based iscp*ptr structure
{
  busy byte, /* set to 1 by CPU before its first CA to 586,
  cleared by 586 after reading info from it */
  unused byte, /* unused */
  scb$b word, /* offset of system control block */
  scb$b (2) word /* base of system control block */
};

/*@ System Control Block */

6 1 declare scb structure
{
  status word, /* cause(s) of interrupt, CU state, RU state */
  cmd word, /* int acks, CU cmd, RESET bit, RU cmd */
  cbl$offset word, /* offset of first command block in CBL */
  rpa$offset word, /* offset of first packet descriptor in RPA */
  crc$errs word, /* crc error encountered so far */
  aln$errs word, /* alignment errors */
  rsc$errs word, /* no resources */
  ovrn$errs word /* overrun errors */
};

/*@ 82586 Action Commands */

/*@ NOP */

7 1 declare nop structure
{
  status word,
  cmd word,
  link$offset word
};

/*@ Individual Address Setup */

8 1 declare ia$setup structure
{
  status word,
  cmd word,
  link$offset word,
  ia$address (6) byte
};

/*@ Configure */

9 1 declare configure structure
{
  status word,
  cmd word,
  link$offset word,
  byte$cnt byte,
  info (11) byte
};
multicast address setup

declare mcsetup structure

status word,
cmd word,
linkoffset word,
mcbytecount word,
mcaddress (4B) byte /* only 8 MC addresses are allowed */

transmit structure

status word,
cmd word,
linkoffset word,
bdoffset word,
destadr (6) byte,
type word

transmit buffer descr1ptor

actcount word,
linkoffset word,
ad0 word,
ad1 word

transmit buffer

1518 byte

diagnose structure

status word,
cmd word,
linkoffset word,
result word

traffic simulator/monitor station program (continued)
/* Dump Status */
16 1 declare dump structure
   (status word,
    cmd word,
    link@offset word,
    buff@ptr word);

/* Dump Area */
17 1 declare dump@area (170) byte;

/* Frame Descriptor */
/* Receive frame area is made of 5 RFDs, 5 RBDs, and 5 1514 bytes long buffers. */
18 1 declare rfd (5) structure
   (status word,
    elts word,
    link@offset word,
    bd@offset word,
    dest@adr (3) word,
    src@adr (3) word,
    type word);

/* Receive Buffer Descriptor */
19 1 declare rbd (5) structure
   (act@count word,
    next@bd@link word,
    ad0 word,
    ad1 word,
    size word);

/* Receive Buffer */
20 1 declare rbuf (5) structure
   (buffer (1514) byte);

/* global variables */
21 1 declare status word./* UART status */
actual word, ch byte at (&cbuf), cbuf (80) byte, ch byte, count byte.
define count word, receive count word, preamble word, address length byte, address byte, cbuf, gobyte, goback byte, reset byte, delay word, current offset word, current frame byte, no transmission byte, stop count word, stop byte, mcount byte, x byte, y byte.

declare (a, c) word, (b, d, e) pointer;
end read;

declare (a, c) word, (b, d) pointer;
end write;

csts: procedure byte external;
end csts;

declare (str, ptr, addr) pointer, base586 dword, w based ptr, addr (2) word,
ptr = @ptr;

offset: procedure (ptr) word;

declare (ptr, ptr2loc) pointer, base586 dword, w based ptr2loc (2) word.

/* actual number of chars UART transferred */
/* buffer for a line of chars */
/* number base switch */
/* counter for received frames */
/* counter for transmitted frames */
/* preamble length in word */
/* address in byte */
/* address location control of 82586 */
/* crc length */
/* if set, go back to Continuous Mode */
/* reset flag */
/* delay counter for transmission delay */
/* offset of current command block */
/* offset of frame descriptor just used */
/* transmit terminal frame count */

/* external procedures */
read: procedure (a, b, c, d, e) external;
write: procedure (a, b, c, d) external;

csts:

/* utility procedures */
offset:

declare (ptr, ptr2loc) pointer, base586 dword, w based ptr2loc (2) word,
ptr2loc = @ptr;

/* 82586 SCB Base Address (20-bit wide in this 186 based system) */

Traffic Simulator/Monitor Station Program (Continued)
base586 = (shl(double (iscp scb$bc(1)), 16) and 00000000H) + iscp scb$b(0),  
return low((shl(double (w(1)), 4) + w(0)) - base586);
end offset,

written procedure (a, b, c, d),
/* This procedure writes a line and put a CR/LF at the end */
declare (a, c) word,
(b, d) pointer,
call write(a, b, c, d);
end writeln,

end offset,

write procedure.
/* This procedure writes a CR/LF. */
call write(0, @(ODH, OAH), 2, @status);
end write;

cr$lf procedure.
/* This procedure writes a CR/LF. */
call write(0, @(ODH, OAH), 2, @status);
end cr$lf;

pause: procedure;
/* This procedure breaks a program flow, and waits for a char to be typed */
call write(0, @(ODH, OAH, ‘Hit <CR> to continue’), 23, @status),
call read(i, @c$buf, 80, @actual, @status),
call cr$lf;
end pause,

skip: procedure byte,
/* This procedure skips all leading blank characters and returns the first non-blank character */
declare i byte:
i = 0;
do while (@c$buf(i) == ‘ ‘),
i = i + 1;
end;
return i;
end skip;

read$char: procedure byte;

Traffic Simulator/Monitor Station Program (Continued)
// This procedure reads a line and returns the first non-blank character. */
50 2 declare i word;
59 2 call read(1, @cbuf, 80, @actual, @status),
60 2 i = skip;
61 2 return(c@buf(i));
62 2 end read@char;

63 1 read@bit: procedure byte;
// This procedure reads a bit and returns the value. */
64 2 declare b byte;
65 2 do forever;
66 3 b = read@char;
67 3 if b = '1' then return 1;
68 3 else if b = '0' then return 0;
69 3 else
70 3 call write(0, @( 'Enter a 0 or 1 == > '), 20, @status);
71 3 end;
72 3 end read@bit;

74 1 yes: procedure byte;
// This procedure reads a character and determines if it is a Y(y) or N(n) */
75 2 declare b byte;
76 2 do forever;
77 3 b = read@char;
78 3 if (b = 'Y') or (b = 'y') then return true;
79 3 else if (b = 'N') or (b = 'n') then return false;
80 3 else
81 3 call write(0, @(0DH, OAH, ' Enter a Y or N == > '), 22, @status);
82 3 end;
83 3 end yes;

85 1 char$to$int: procedure (c) byte;
// This procedure converts a byte of ASCII integer to an integer */
86 2 declare c byte;
87 2 if ('0' <= c) and (c <= '9') then return (c - 30H);
88 2 else if ('A' <= c) and (c <= 'F') then return (c - 37H),
89 2 else
90 2 2
if ('a' <= c) and (c <= 'f') then return (c - 57H).
else return OFFH.

end char$to$int.

int$to$ascii procedure (value, base, ld, bufadr, width);

/* This procedure converts an integer < OFFFFFFFFH to an array of ASCII codes.
Input variables are value = integer to be converted,
base = number base to be used for conversion,
ld = leading character to be filled in,
bufadr = buffer address of the array,
width = size of array. */

declare value dword,
bufadr pointer,
(ld, j, base, ld, width) byte,
chars based bufadr (1) byte;

do i = 1 to width;
   j = value mod base;
   if j < 10 then chars (width - i) = j + 30H;
   else chars (width - i) = j + 37H;
   value = value / base,
end;

return value;

end int$to$ascii;

out$word: procedure (w$ptr, distance);

/* An integer at (selector of w$ptr) (offset of w$ptr + distance) is printed
as a 4 digit hexadecimal number. */

declare chars(4) byte,
w$ptr pointer,
distance byte,
w based w$ptr (1) word;

call int$to$ascii(w(distance), 16, '0', @chars(0), 4);
call write(0, @chars(0), 4, @status);
end out$word;

write$int: procedure(dw, t);

/* An integer (dw) is printed in hexadecimal (t = 1) or in decimal (t = 0). */
declare dw dword,
   chars (10) byte,
   t byte.

118 2  if t then
119 2   do:
120 3    call int*to*ascii(dw, 16, 0, @chars(0), 8);
121 3    call write(0, @chars(0-char*count), char*count, @status),
122 3    end;
123 2  else
124 3    call int*to*ascii(dw, 16, 0, @chars(0), 10),
125 3    call write(0, @chars(10-char*count), char*count, @status),
126 3    end.
127 2  end write*int.

128 1  out*dec*hex, procedure(dw):
/* This procedure prints an integer in decimal and hexadecimal */
129 2  declare dw dword;
130 2   call write*int(dw, 0),
131 2   call write(0, @('/ '), 2, @status),
132 2   call write(0, @('H'), 2, @status),
133 2   call write(0, @('H'), 2, @status);
134 2  end out*dec*hex.

135 1  write*offset procedure(w$ptr),
/* This procedure takes a pointer variable, converts it to a 82586 type offset,
   and prints it in hexadecimal */
136 2  declare w$ptr pointer,
    w word;
137 2   call write(0, @('/ at '), 4, @status),
138 2   w = offset(w$ptr),
139 2   call out*word(w, 0),
140 2   call write(0, @('/ '), 2, @status),
141 2  end write*offset.

142 1  write*address procedure (ptr),
/* This procedure takes a pointer variable and prints it in the
   'selector offset' format */
143 2  declare (ptr, ptr$loc) pointer,
    w based ptr$loc (2) word.
144 2  ptr$loc = @ptr.
call outword(@w(1), 0),
call write(0, @(''), 1, @status),
call outword(@w(0), 0),
call write(0, @(' '), 1, @status),

end write$address,

print$wds: procedure(w$ptr, no$words),

/* This procedure prints no$words number of words starting at w$ptr. */

declare w$ptr pointer,

(i, no$words) byte,

if no$words <> 0 then

do;

call cr$lf,

do i = 0 to no$words - 1;

call outword(w$ptr, 1);

if i = 0 then

call write$offset(w$ptr);

call cr$lf;

end;

end print$wds;

print$str: procedure (str$ptr, len),

/* This procedure prints len number of bytes starting at str$ptr. */

declare (len, i) byte,

chars (2) byte,

str$ptr pointer,

str based str$ptr (1) byte;

if len <> 0 then

do i = 0 to (len - 1);

call int$to$asci(str(i), 16, 'O', @chars(0), 2);

call write(0, @chars(0), 2, @status);

call write(0, @(' '), 2, @status);

call cr$lf;

end;

end print$str;

printf$buff: procedure (ptr, cnt),

/* This procedure prints cnt number of buffer contents starting at ptr. */

declare ptr pointer,

bt based ptr (1) byte,

(i, j) byte,

cnt word;
if cnt > 16 then
  do
    i = shr(cnt, 4) - 1;
    j = 0 to i;
    call write@address(@bt(16*i));
    call print@str(@bt(16*i), 16);
    if (j = 20) or (j = 40) or (j = 60) or (j = 80) then
      call pause,
    end;
    i = i + 1,
    if cnt-16*i <> 0 then call write@address(@bt(16*i)),
    call print@str(@bt(16*i), cnt-16*i);
  end;
  else
    do
      call write@address(@bt(0));
      call print@str(@bt(0), cnt);
    end;
end print@buff;

read@int: procedure (limit) dword;

! This procedure reads integer characters and forms an integer. If the
! integer is bigger than 'limit' or an overflow error is encountered, then
! an error message is printed. */

declare (wd, wh, limit) dword;
  (i, j, k, done, hex, dover, hover) byte.

do forever,
  call read(i, @c.buf, 80, @actual, @status);
  i, k, done, dover, hover = false;
  wd, wh = 0;
  if @c.buf(i)<> 'h' and (c.buf(i) <> 'h') and (c.buf(i) <> ODH) and
     (c.buf(i) <> OAH) and (c.buf(i) <> 'i') or (i = k) then
    call write@int(0, @ODH, OAH, 'Illegal character'), 20, @status;
  else
    do
      if (c.buf(i) = 'h') or (c.buf(i) = 'h') then hex = true,
      if hex then

Traffic Simulator/Monitor Station Program (Continued)
224 4 do, if not hover and (wh <= limit) then return wh.
225 5 end.
226 4 else
227 5 if not hover and (ud <= limit) then return ud.
228 4 call writeln(O, @(ODH, OAH, 'The number is too big.'), 25, @status).
229 4 call write(O, @( 'It has to be less than or equal to '), 36, @status).
230 4 call writeln(O, @( ' '), 1, @status).
231 4 end;
232 4 call writeln(O, @(hex(limit)), 24, @status).
233 4 call writeln(O, @( ' '), 1, @status).
234 4 end;
235 3 call write(O, @(Enter a number == ' ), 20, @status).
236 3 end;
237 2 end read$int;

Traffic Simulator/Monitor Station Program (Continued)
269 1 percent: procedure;

    /* This procedure calculates and prints a network percent load generated by this station. The equation used in this procedure was obtained from actual measurements. */

270 2 declare i word,
    (j, k) dword,
    pcent (3) byte;

271 2 j = (tbd.act.count and 3FFFH)*8;
272 2 if not addrloc then k = (2*address.length + 2 + crc + preamble)*8;
274 2 else k = (crc + preamble)*8;
275 2 if delay <> 0 then

276 2 *IF NOT SBC18651
277 2 i = low((1000*(j + k))/(1805 + k + 5*double(delay) + j));
278 2 *ELSE
279 2 i = low((1000*(j + k))/(2021 + k + 5*double(delay) + j));
280 2 *ENDIF

281 2 *IF NOT SBC18651
282 2 i = low((1000*(j + k))/(1810 + k + j));
283 2 *ELSE
284 2 i = low((1000*(j + k))/(2026 + k + j));
285 2 *ENDIF

286 2 call int#to#ascii(i, 10, 0, @pcent(0), 3);
287 2 call write(0, @pcent(0), 2, @status);
288 2 call write(0, @(' '), 1, @status);
289 2 call write(0, @pcent(2), 1, @status);
290 2 call writeln(0, @(' '), 2, @status);
291 2 end percent;

292 1 print#network#addr: procedure (ptr);

    /* This station's address is printed with its least significant bit in the most right position. */

293 2 declare ptr pointer,
    addr based ptr (1) byte,
    char (6) byte,
    i byte;

Traffic Simulator/Monitor Station Program (Continued)
do i = 1 to address\$length.
   char(i-1) = addr(address\$length-1);
end.
call print\$str(char(0), address\$length).
end print\$network\$addr.

print\$parameters procedure.
/* This procedure prints transmission parameters. */
declare w dword.
   stgs (6) byte;
call write(0, @(' Destination Address. '), 22, @status),
if not addrloc then
   call print\$network\$addr(@transmit. dest\$adr(0));
else
call print\$network\$addr(@ts\$buffer(0));
   w = (tbd.act\$count and 3FFFH) + address\$length * 2 + 2 + crc;
   w = (tbd. act\$count and 3FFFH) + crc;
call write(0, @(' Frame Length: '), 15, @status);
call write\$int(w, 0);
call write\$int(w. 0);
call write\$int(w, 0);
call write\$int(0, @(' Time Interval between Transmit Frames: '), 40, @status);
if delay <> 0 then
   do;
   IF NOT SBC18651
   w = 1810 + (double(delay) - 1) * 5;
   ELSE
   w = 2020 + (double(delay) - 1) * 5;
   ENDIF
call int\$to\$asci(w, 10, 0, @stgs, 6);
if w >= 10000 then
   do:
call write(0, @stgs(0), 2, @status),
call write(0, @(' '), 1, @status),
call write(0, @stgs(2), 2, @status),
call write\$int(0, @('@ milliseconds'), 12, @status);
end;
else
do:
call write(0, @stgs(0), 5, @status),
call write(0, @(' '), 1, @status),
call write(0, @stgs(5), 1, @status),
call write\$int(0, @('@ microseconds'), 13, @status);
end;
end;
else
Traffic Simulator/Monitor Station Program (Continued)
199 NOT SBC18651
   call writeln(O, O(' 159.4 microseconds'), 19, @status);
$ELSE
   call writeln(O, O(' 172.8 microseconds'), 19, @status).
$ENDIF
   call write(0, O(' Network Percent Load generated by this station'), 49, @status);
   call percent;
   call write(0, O(' Transmit Frame Terminal Count'), 32, @status);
   if stop then call write@int(stop@count, dhex);
   else call write(0, O('Not Defined'), 11, @status);
   call cr@lf;
   end print@parameters;
   print@scb: procedure;
      /* prints the SCB */
   call writeln(0, @0DH, OAH, '### System Control Block ###'), 30, @status);
   call print@wds(scb@status, B);
   end print@scb;
   wait@scb: procedure;
      /* This procedure provides a wait loop for the SCB command word to become cleared. */
   declare i word;
   i = 0;
   do while (scb.cmd <> O) and (i < 8000H);
      i = i + 1;
   end;
   if scb.cmd <> O then
      do,
      call write(0, @0DH, OAH, ' Wait Time = '), 15, @status);
      call write@int(i, O),
      call cr@lf,
   end;
   end wait@scb;
   start@timer0: procedure;
      /* 80186 timer0 is started. */

Traffic Simulator/Monitor Station Program (Continued)
output(TIMER0$CTL) = 0E000H.
end start$timer0:

ISR: procedure interrupt INT$TYPE$S86 reentrant:
  /* interrupt service routine for 82586 interrupt */
declare 1 byte:
  /* Enable 82586 Interrupt */
$IF SBC18651
  output (PIC$E_DI$130) = EDI$CMD0$130.
  enable;
$ELSE
output (PIC$E_DI$186) = EDI$CMD0$186.
  enable;
$ENDIF
  /* Frame Received Interrupt has the highest priority */
if (scb.status and 4000H) = 4000H then
disable:
sotb.cmd = 4000H;
output (CA$PORT) = CA;
call wait$scb;
if rfd(current$frame).status = OA000H then
do:
  receive$count = receive$count + 1;
  current$frame = current$frame + 1;
  if current$frame = 5 then current$frame = 0;
  end,
  return;
end;
endif
if (scb.status and 2000H) = 2000H then
do:
disable:
sotb.cmd = 2000H;
output (CA$PORT) = CA;
call wait$scb;
enable:
if (transmit.status and OA000H) = OA000H then
do:
count = count + 1i
else
do:
Traffic Simulator/Monitor Station Program (Continued)
transmit status = 0;
if delay = 0 then
do;
disable;
scb.cmd = 0100H;
output(CA&PORT) = CA;
call wait*scb;
return;
else

do;
call start*timer0.
return;
end;
end;
end;
end;
end;
end;
end;
end;
end;
end;
end;
end;

Traffic Simulator/Monitor Station Program (Continued)
do;
  disable;
  scb.cmd = 8000H;
  output (CA*PORT) = CA;
  call wait*scb;
end;
if (scb.status and 1000H) = 1000H then
  do;
    disable;
    scb.cmd = 1000H;
    output (CA*PORT) = CA;
    call wait*scb;
    call write (O. @ODH, 'Receive Unit became not ready.', @status));
end;
if reset then
  do;
    if iscp.busy then
do;
      call writeln(0, @(ODH, OAH, 'Reset failed.'), 16, @status));
    disable;
    scb.cmd = 0000H;
    output (CA*PORT) = CA;
    call wait*scb;
    output (CA*PORT) = CA;
    call writeln(O, @('Software Reset Executed!'), 25, @status));
  end;
else reset = false;
end;

procedure interrupt INT*TYPE*TIMER;
/* interrupt service routine for 80186 timer interrupt*/
srb.cmd = O100H;
output(CA*PORT) = CA;
call wait*scb;

IF SBC18651
  output(PIC*EDI*130) = EDI*CMD4*130;
  enable;
  output(PIC*EDI*186) = EDI*CMD0*186;
ELSE
  output(PIC*EDI*186) = EDI*CMD4*186;
ENDIF

end ts*isr:

Traffic Simulator/Monitor Station Program (Continued)
*IF SBC18651

isr#7 procedure interrupt INT#7.
/* The 80150 generates an interrupt 7 if the original interrupt is not
active any more when the first interrupt acknowledge is received. */
call write(O, @(ODH, 'Interrupt 7', ODH), 13, @status);
end isr#7;
*ENDIF

read#byte procedure (k) byte:
474 1 declare k word;
475 2 call write(O, @(ODH, OAH, ' Enter byte '), 14, @status);
476 2 call out#dec#hex(k);
477 2 call write(O, @( ' == '), 5, @status);
478 2 return read#int(0FFH);
479 2 end read#byte.

480 2

init#1B6#timerO: procedure;
/* This procedure initializes the 80186 timer 0. */
481 1 declare k byte;
482 2 if SBC18651

output(INT$CTL$TIMERO) = B;
call write(O, @(ODH, OAH, ' Enter a delay count == > '), 27, @status);
delay = read#int(0FFFFH);
if (delay < 100) and (delay <> 0) then
  do:
call cr$1f;
call cr$1f;
call loop$char(35, ' *');
call write(O, @( ' WARNING '), 9, @status);
call loop$char(35, ' *');
call writelnO, @(ODH, OAH, 'A delay count between 0 and 100 may be very ','dangerous when this station starts'), 80, @status),
call writelnO, @( 'to receive many frames separated only by the ','IFS period (9.6 microseconds)'), 75, @status),
call writelnO, @( 'If this station never receives a frame, then ','ignore this warning '), 65, @status),
call loop$char(79, ' *');
  end:
output(MAX$COUNT$A) = delay;
call cr$1f;
output(PIC$MASK$186) = 3EH.

Traffic Simulator/Monitor Station Program (Continued)

292010-50
$\text{ELSE}$

483 2 output(INT\$\text{CTL}\$\text{TIMER0}) = OCH;

484 2 call write(O, @ODH. OAH, ' Enter a delay count -> '), 27, @status);

485 2 delay = read\$int(OFFFFH);

486 2 output(MAX\$COUNT\$A) = delay;

487 2 call cr\$lf;

488 2 output(PIC\$MASK\$I86) = ENABLE\$I86$I86;

$\text{ENDIF}$

489 2 end init$I86$I86$timer0;

490 1 setup\$ia\$parameters. procedure;

491 2 declare i byte;

492 2 call write(O, @ODH. OAH, ' Configure the I86 with the prewired '

493 2 'board address -> '), 57, @status);

494 2 if yes then

495 3 do i = 0 to address\$length - 1;

496 3 ia\$setup.ia\$address(i) = input(BOARD\$ADDRESS\$BASE + 10 - 2 * i);

497 3 end;

498 3 else

499 3 do;

500 3 call write(O, @ODH. OAH, ' Enter this station's address',

501 3 ' in Hex -> '), 43, @status);

502 3 call put\$address(ia\$setup.ia\$address(0));

503 3 end;

504 1 setup\$mc\$parameters: procedure;

505 2 declare (j, k, done) byte;

506 4 j = 0;

507 2 call writeln(O, @ODH. OAH, ' You can enter up to 8 Multicast Addresses. ');

508 2 done = false;

509 2 call write(O, @(' Would you like to enter a Multicast Address?'',

510 2 ' Y or N -> '), 35, @status);

511 2 do while not done;

512 4 if yes then

513 5 k = j * address\$length;

514 5 j = j + 1;

515 5 call cr\$lf;

516 5 if j = 9 then

517 5 do;

518 5 call write(O, @(' You already entered 8 Multicast addresses. ');

519 5 done = true;

520 5 else

521 5 do;

Traffic Simulator/Monitor Station Program (Continued)
CALL putaddress(@mc@setup.mc@address(k));
call write(0, @ODH, OAH, ' More Multicast Addresses?', '(Y or N) => '), 42, @status);
end;
end;
else done = true;
end;
end;
end;
end;
if ja then ja - 1;
mc@count = address@length + j;
mc@setup.mc@byte@count = mc@count;
call write(0, @ODH, OAH, ' You entered '), 15, @status),
call write(int(j), 0);
call writeinfo(0, @( ' Multicast Address(es)'), 23, @status);
end setup@mc@parameters;

setup@configure@parameters: procedure;
declare (k, j) byte;
configure.byte@cnt = 11;
configure.info(0) = 8;
configure.info(1) = 0;
configure.info(2) = 26H;
configure.info(3) = 0;
configure.info(4) = 96;
configure.info(5) = 0;
configure.info(6) = 1F2H;
configure.info(7) = 0;
configure.info(8) = 0;
configure.info(9) = 64;
j = 0;
call write(0, @ODH, OAH, ' Configure command is set up for default',
' values', @ODH, OAH, ' Do you want to change any bytes?',
'(Y or N) => '), 99, @status);
do while yes;
do while j = 0;
call write(0, @ODH, OAH, ' Enter byte number (1 - 11) => '), 34,
@status);
j = readbyte(11);
if j = 0 then
call write(0, @ODH, OAH, ' Illegal byte number'), 22, @status);
end;
if j = 1 then configure.byte@cnt = readbyte(j);
else configure.info(j - 2) = readbyte(j);
j = 0;
call write(0, @ODH, OAH, ' Any more bytes? (Y or N) => '), 32,
@status);
end;
preamble = shl(1, shr((configure.info(2) and 3OH), 4)+1),
address@length = configure.info(2) and 07H;
if address@length = 7 then address@length = 0;
add@loc = shr((configure.info(2) and O8H), 3);
if shr((configure.info(7) and 20H), 5) then crc = 2; else crc = 4;
if shr((configure.info(7) and 10H), 4) then crc = 0;
end setup@configure@parameters;
Traffic Simulator/Monitor Station Program (Continued)
setup@x$parameters procedure.
declare (size, i) word.

do forever,
nottransmission = false,
transmit bd$offset = offset (@tbd act$count),
if not ad$loc then
do;
call write(0, @(ODH, OAH, ' Enter a destination address in Hex ==0 '), 42, @status),
call put$address(@transmit dest$adr(O));
end;
else call writeln(0, @( ' B2586 is configured to pick up DA, IA, ' and TYPE from TX buffer.'), 64, @status),
call cr$@;

if not ad$loc then
do;
call write(0, @(ODH, OAH, ' Enter TYPE == '), 18, @status),
transmit.type = read$int(0FFFFH);
end;
call writeln(0, @(ODH, OAH, ' How many bytes of transmit data?'), 35, @status),
call write(0, @( ' Enter a number == '), 20, @status),
size = read$int(1518),
tbd act$count = size or 8000H;
if size <> 0 then
do;
tbd lin$offset = 0FFFFH;
tbd ad0 = offset (@tbd$buffer(0)),
tbd ad1 = 0;
do i = 0 to 1517;
tx$buffer(i) = i;
end;
call writeln(0, @(ODH, OAH, ' Transmit Data is continuous numbers (0, 1, 2, 3, ' and ... '), 57, @status),
call write(0, @( ' Change any data bytes? (Y or N) == '), 27, @status),
do while yes,
call write(0, @(ODH, OAH, ' Enter a byte number == '), 27, @status),
i = read$int(size),
call write(0, @(ODH, OAH, ' Byte '), 8, @status),
call out$dec$hex(i),
call write(0, @( ' currently contains '), 20, @status),
call out$dec$hex(tx$buffer(i)),
call write(0, @( '.'), 1, @status),
tx$buffer(i) = read$byte(i);
call write(0, @(ODH, OAH, ' Any more bytes? (Y or N) == '), 32, @status),
do while yes,
call write(0, @(ODH, OAH, ' Enter a byte number == '), 27, @status),
i = read$int(size),
call write(0, @(ODH, OAH, ' Byte '), 8, @status),
call out$dec$hex(i),
call write(0, @( ' currently contains '), 20, @status),
call out$dec$hex(tx$buffer(i)),
call write(0, @( '.'), 1, @status),
tx$buffer(i) = read$byte(i);
call write(0, @(ODH, OAH, ' Any more bytes? (Y or N) == '), 32, @status),
doi;
end;
end;
else transmit bd$offset = 0FFFFH;
call cr$@;
call init1B6#timer0;
call write(O, @ODH, OAH, 'Setup a transmit terminal count?',
   '(Y or N) => '), 49, @status);
if yes then
   do:
      stop = true;
      call write(O, @ODH, OAH, 'Enter a transmit',
         ' terminal count => '), 39, @status);
      stop = count = read$int(0FFFFFFFH);
   end;
else stop = false;
call cr$lf;
call print#parameters;
call write(O, @ODH, OAH, 'Good enough? (Y or N) => '), 29, @status);
if yes then return;
end;

loop$char: procedure (i, j);
declare (i, j, k) byte;
do k = 1 to i;
call write(O, @j, i, @status);
end;
end loop$char;

init: procedure;
declare i byte;
call cr$lf,
call loop$char(13, OAH),
call loop$char(13, '\f');
call writeln(O, @(TRAFFIC SIMULATOR AND MONITOR',
   ' STATION PROGRAM '), 46, @status),
call loop$char(7, OAH);
call writeln(O, @(TRAFFIC SIMULATOR AND MONITOR',
   ' STATION PROGRAM '), 46, @status),
call cr$lf,
call set$interrupt(INT'TYPE'SB6, isr);
call set$interrupt(INT$7, isr7),
output (PIC$MASK$130) = ENABLE$5B6$186,
output (PIC$EDI$130) = EDI$CMD$5B6$186,
output (PIC$EDI$130) = EDI$CMD$5B6$186,
output (PIC$EDI$1B6) = EDI$CMD$1B6,
output (PIC$VTR$1B6) = 30H;

*ELSE

658 2  output (PIC$EDI$1B6) = EDI$CMD$1B6,
659 2  output (PIC$EDI$1B6) = EDI$CMD$1B6,
660 2  output (PIC$MASK$1B6) = ENABLE$5B6.

*ENDIF

/* locate iscp */
661 2  iscp$ptr = ISCP$LOC$LO;
/* set up fields in ISCP */
662 2  iscp. busy = 1,
663 2  iscp. scb$b(0) = SCB$BASE$LO;
664 2  iscp. scb$b(1) = SCB$BASE$HI;
665 2  iscp. scb%o = offset (@scb.status);
/* set up SCD */
666 2  scb. status = 0;
667 2  scb. cbl$offset = offset (@diagnose. status),
668 2  scb. rpa$offset = offset (@fd(0). status);
669 2  scb. crc$errs = 0;
670 2  scb. a1m$errs = 0;
671 2  scb. rsc$errs = 0;
672 2  scb. ovrn$errs = 0;
/* set up Diagnose command */
673 2  diagnose. status = 0;
674 2  diagnose cmd = 7;
675 2  diagnose link$offset = offset (@configure status),
/* set up CONFIGURE command */
676 2  configure. status = 0.
677 2  configure cmd = 2;
678 2  configure. link$offset = offset (@ia$setup. status),
680 2  call setup$configure$parameters;
/* set up IA command */
681 2  ia$setup. status = 0.
682 2  ia$setup. cmd = 1;
683 2  ia$setup. link$offset = offset (@mc$setup. status),
684 2  call setup$ia$parameters;

Traffic Simulator/Monitor Station Program (Continued)
/* set up MC command */
683 2 mc$setup status = 0,
686 2 mc$setup.cmd = 8003H,
687 2 mc$setup.link$offset = OFFFFH,
688 2 call setup$mc$parameters.

/* set up one transmit cb linked to itself */
689 2 transmit status = 0,
690 2 call write0. @('ODH. OAH. ' Would you like to transmit?'), 30. @status),
691 2 call write(0. @(' Enter a Y or N ==> '), 20. @status),
692 2 if yes then
693 2 do,
694 3 transmit cmd = 8004H,
695 3 transmit link$offset = OFFFFH,
696 3 transmit bd$offset = offset (@bd act$count),
697 3 call setup$tx$parameters.
698 3 end.
699 2 else no$transmission = true.

/* initialize receive packet area */
700 2 do i = 0 to 3
701 3 rfd(i). status = 0,
702 3 rfd(i). else$ = 0,
703 3 rfd(i). link$offset = offset (@rfd(i+1). status),
704 3 rfd(i). bd$offset = OFFFFH,
705 3 rbd(i). act$count = 0,
706 3 rbd(i). next$bd$link = offset (@rbd(i+1). act$count),
707 3 rbd(i). ad0 = offset (@rbuf(i). buffer(0)),
708 3 rbd(i). ad1 = 0,
709 3 rbd(i). size = 1500.
710 3 end;
711 3 rfd(0). bd$offset = offset (@rbd(0). act$count),
712 2 rfd(4). status = 0;
713 2 rfd(4). else$ = 0,
714 2 rfd(4). link$offset = offset (@rfd(0). status),
715 2 rfd(4). bd$offset = OFFFFH,
716 2 rbd(4). act$count = 0,
717 2 rbd(4). next$bd$link = offset (@rbd(0). act$count),
718 2 rbd(4). ad0 = offset (@rbuf(4). buffer(0)),
719 2 rbd(4). ad1 = 0,
720 2 rbd(4). size = 1500.

/* initialize counters */
721 2 count = 0,
722 2 receive$count = 0,
723 2 current$frame = 0.

/* issue the first CA */
output(CA&PORT) = CA.

end init;

print$help procedure:

call writeln(0, @('ODH, OA, ' Commands are ')', 16, status);
call writeln(0, @('ODH, OA, ' S - Setup CB
output(CA.PORT)

D - Display H/D/CB'), 45, status);
call writeln(0, @('P - Print SCB
C - SCB Control CMD'), 44, status);
call writeln(0, @('L - ESI Loopback On
N - ESI Loopback Off'), 45, status);
call writeln(0, @('A - Toggle Number Base'), 23, status);
call writeln(0, @('Z - Clear Tx Frame Counter'), 27, status);
call writeln(0, @('Y - Clear Rx Frame Counter'), 27, status);
call writeln(0, @('E - Exit to Continuous Mode'), 28, status);

end print$help;

enter@scb$cmd. procedure,

declare i byte;

/* enter a command into the SCB */
call cr$if;
if scb$cmd <> 0 then do,
call writeln(0, @('SCB command word is not cleared'), 32, status),
call write(O, @('Try a Channel Attention? (Y or N) ==> '), 39, status);
if yes then
    output(CA&PORT) = CA;
call writeln(0, @('Issued channel attention'), 25, status),
call cr$if;
call cr$if;
return;
end;
call write(O, @('Do you want to enter any SCB commands? (Y or N) ==> '), 33, status);
if not yes then return;
call write(O, @('Enter CUC ==> '), 17, status),
i = read@int(4);
scb$cmd = scb$cmd or shl(double(i), 0);
if i = 1 then scb$ch$offset = cur$ch$offset;
call write(O, @('Enter RES bit ==> '), 21, status),
i = read@bit;
scb$cmd = scb$cmd or shl(i, 7);
call write(O, @('Enter RUC ==> '), 17, status),
i = read@int(4);
scb$cmd = scb$cmd or shl(i, 4);

Traffic Simulator/Monitor Station Program (Continued)
if ((scb_cmd & offset = offset (@transmit status))
    and (false. cmd and 0100H) = 0100H) or ((scb cmd and 0100H) = 0010H)
    and not ((scb cmd and 0080H) = 0080H)
    then goback = 1.

    call writeln(0. @ODH. OAH. ' Issued Channel Attention' ). 27. @status).
    call cr@l; 1
    output(CA@PORT) = CA; 2
end enter@scb@cmd.

print@type@help. procedure:

    call writeln(0. @ODH. OAH. OAHI. 'Command block type '). 22. @status).
    call writeln(0. @(' N - Nop I - IA Setup'). 35. @status).
    call writeln(0. @(' C - Configure M - MA Setup'). 35. @status).
    call writeln(0. @(' T - Transmit R - TDM'). 30. @status).
    call writeln(0. @(' D - Diagnose S - Dump Status'). 3B. @status).
    call writeln(0. @(' H - Print this message'). 23. @status).
end print@type@help;

setup@cb@procedure:

    declare (t. valid) byte;
    valid = false.
do while not valid:
        call writeln(0. @ODH. OAH. ' Enter command block type (H for'.
            ' help) =? '). 45. @status);
        t = read@char.
        if (t = 'M') or (t = 'm') and (t <> 'T') and (t <> 't') and
            (t <> 'N') and (t <> 'n') and (t <> 'R') and (t <> 'r') and
            (t <> 'D') and (t <> 'd') and (t <> 'C') and (t <> 'c') and
            (t <> 'I') and (t <> 'i') and (t <> 'M') and (t <> 'm') and
            (t <> 'S') and (t <> 's') then
        
        call writeln(0. @ODH. OAH. ' Illegal command block type'). 29.
            @status).
        else
            if (t = 'H') or (t = 'h') then call print@type@help.
            if (t = 'N') or (t = 'n') then call print@type@help.
            end valid = true.
        if (t = 'I') or (t = 'i') then
        do.
            cur@cb@offset = offset (@nop@status).
            nop@status = 0.
            nop@cmd = B000H.
            nop@link+offset = OFFFH.
            end.
        if (t = 'I') or (t = 'i') then
        do.
            cur@cb@offset = offset (@ia@student@status).
            ia@student@status = 0.
            ia@student@cmd = B001H.
            ia@student@link+offset = OFFFH.
            call setup@ia@parameters.
        end.
Traffic Simulator/Monitor Station Program (Continued)
```
806  2 if (t = 'C') or (t = 'c') then
807  2   do
808  3   cur$cb$offset = offset (@configure.status);
809  3   configure.status = 0;
810  3   configure.cmd = B002H;
811  3   configure.link$offset = OFFFFH;
812  3   call setup$configure$parameters.
813  3 end;
814  2 if (t = 'M') or (t = 'm') then
815  2   do
816  3   cur$cb$offset = offset (@mc$setup.status);
817  3   mc$setup.status = 0;
818  3   mc$setup.cmd = B003H;
819  3   mc$setup.link$offset = OFFFFH;
820  3   call setup$mc$parameters.
821  3 end;
822  2 if (t = 'T') or (t = 't') then
823  2   do
824  3   cur$cb$offset = offset (@transmit.status);
825  3   transmit.status = 0;
826  3   transmit.cmd = B004H;
827  3   transmit.link$offset = OFFFFH;
828  3   call setup$transmit$parameters.
829  3 end;
830  2 if (t = 'R') or (t = 'r') then
831  2   do
832  3   cur$cb$offset = offset (@tdr.status);
833  3   tdr.status = 0;
834  3   tdr.cmd = B005H;
835  3   tdr.link$offset = OFFFFH;
836  3   tdr.result = 0;
837  3 end;
838  2 if (t = 'S') or (t = 's') then
839  2   do
840  3   cur$cb$offset = offset (@dump.status);
841  3   dump.status = 0;
842  3   dump.cmd = B006H;
843  3   dump.link$offset = OFFFFH;
844  3   dump.buf$ptr = offset (@dump$area(0));
845  3 end;
846  2 if (t = 'D') or (t = 'd') then
847  2   do
848  3   cur$cb$offset = offset (@diagnose.status);
849  3   diagnose.status = 0;
850  3   diagnose.cmd = B007H;
851  3   diagnose.link$offset = OFFFFH;
852  3 end;
853  2 end setup$cb;

54  1 display$command$block: procedure;
55  2 declare (i, j) byte,
56        wh pointer,
57        sel selector,
58        w word;

Traffic Simulator/Monitor Station Program (Continued)
```
call crlf;
if curcb&offset = OFFFH then
  call write(O, @('No Command Block to display'), 28, @status),
  if curcb&offset = offset (@nop.status) then
    do:
      call write(O, @('---NOP Command Block---'), 23, @status),
      call print@uds(@nop.status, 3);
    end;
  if curcb&offset = offset (@tdr.status) then
    do:
      call write(O, @('---TDR Command Block---'), 23, @status),
      call print@uds(@tdr.status, 4);
    end;
  if curcb&offset = offset (@diagnose.status) then
    do:
      call write(O, @('---Diagnose Command Block---'), 28, @status),
      call print@uds(@diagnose.status, 3);
    end;
  if curcb&offset = offset (@transmit.status) then
    do:
      call write(O, @('---Transmit Command Block---'), 28, @status),
      if not addb&length then i = address@length;
      else i = address@length + 1;
      if addb&length then call print@uds(@transmit.status, 4);
      else call print@uds(@transmit.status, 1/2+1);
      call crlf;
      call crlf;
      if transmit b&offset <> OFFFH then
        do:
          call write(O, @('---Transmit Buffer Descriptor---'), 33, @status),
          call print@uds(@tbd.act.count, 4);
          call write(O, @(ODH, OAH, OAH,
            display the transmit buffer? (Y or N) =
            '), 46, @status);
          if yes then
            do:
              call crlf;
              call writeln(O, @('Transmit Buffer: '), 17, @status),
              w = tbd.act.count and 3FFFH;
              call print@buff(@tx@buffer(0), w); end;
        end;
      end:
      if curcb&offset = offset (@ia@setup.status) then
        do:
          call write(O, @('---IA Setup Command Block---'), 28, @status),
          call print@uds(@ia@setup.status, 6);
        end;
      if curcb&offset = offset (@configure.status) then
        do:
          call write(O, @('---Configure Command Block---'), 29, @status),
          call print@uds(@configure.status, 9);
        end;
      if curcb&offset = offset (@mc@setup.status) then
        do:
          call write(O, @('---MC Setup Command Block---'), 28, @status),
          i = 4 + mc@count/2;
        end;
Traffic Simulator/Monitor Station Program (Continued)
Traffic Simulator/Monitor Station Program (Continued)
965 3    call cr1ef.
966 3    end.
967 2    call write(O, @ODH, OAH, OAH, 'Display the receive buffer',
968         46, @status);
969 2    if not yes then return.
970 2    call writeint(O, @ODH, OAH, 'Receive Buffers', 19, @status).
971 2    do i = 1 to 4,
972 3    call write(O, @ODH, OAH, 'Receive Buffer', 18, @status),
973 3    call writeint(i, 0),
974 3    k = rbd(i) act#count and 3FFFH,
975 3    call printbuff(@rh() buffer(O), k);
976 3    call pause;
978 3    end;
979 2  end displayreceivearea.
980 1    display@cb@pa: procedure;
981 2    declare i byte;
982 2    call write(O, @ODH, OAH, Command Block or Receive Area',
983         47, @status),
984 2    do while (i <> 'R') and (i <> 'r') and (i <> 'C') and (i <> 'c'),
985 3    call writeint(O, @ODH, OAH, 'Illegal command', 18, @status),
986 3    call write(O, @('Enter R or C == > '), 18, @status),
987 3    i = read#char;
988 3    end;
989 2    if (i = 'R') or (i = 'r') then call display@receivearea,
990 2    else call display@command@section;
992 2  end display@cb@pa;
993 1    process@cmd: procedure;
994 2    declare u, i byte;
995 2    goback = 0;
996 2    b = read#char;
997 2    call cr1ef;
998 2    if (b <> 'h') and (b <> 'H') and (b <> 's') and (b <> 'S') and
999 2    (b <> 'd') and (b <> 'd') and (b <> 'p') and (b <> 'p') and
1000 2    (b <> 'c') and (b <> 'c') and (b <> 'e') and (b <> 'e') and
1001 2    (b <> 'l') and (b <> 'l') and (b <> 'n') and (b <> 'n') and
1002 2    (b <> 'y') and (b <> 'y') and (b <> 'q') and
1003 2    l (b <> 'A') and (b <> 'a') then
1004 2    call write(O, @('Illegal command'), 16, @status);)
1005 2    if (b = 'H') or (b = 'h') then call print@help;
1006 2    if (b = 'A') or (b = 'a') then
1007 2    if dhex then
1008 2    do,
1009 2    dhex = false;
1010 3    call write(O, @('Counters are displayed in decimal '), 35,
1011 3    @status),
1012 3    end.
1013 2    else

Traffic Simulator/Monitor Station Program (Continued)
do, dhex = true,
call write(O, @(' Counters are displayed in hexadecimal '), 29, @status); end.

if (b = 'L') or (b = 'l') then do,
output(ESI%PORT) = LOOPBACK,
call write(O, @(' ESI is in Loopback Mode '), 25, @status); end.

if (b = 'N') or (b = 'n') then do,
output(ESI%PORT) = NO%LOOPBACK,
call write(O, @(' ESI is NOT in Loopback Mode '), 29, @status); end.

if (b = 'Z') or (b = 'z') then do,
count = 0,
call write(O, @(' Transmit Frame Counter is cleared '), 35, @status); end.

if (b = 'Y') or (b = 'y') then do,
receive%count = 0,
scb crc%err, scb aln%err, scb rsc%err, scb ovrn%err = 0,
call write(O, @(' Receive Frame Counter is cleared '), 34, @status); end.

if (b = 'C') or (b = 'c') then call enter%scb%cmd,
if (b = 'S') or (b = 's') then call setup%cb,
if (b = 'P') or (b = 'p') then call print%scb,
if (b = 'D') or (b = 'd') then call display%scb%rpa,
if (b = 'E') or (b = 'e') then goback = 1;
call cr%if,
call wr%if.
end process%cmd.
do forever;

goback = 0;
call write(O, @(' Enter command (H for help) ==> '), 24, @status);
do forever;
if csts then do,
disable;
call process%cmd,
enable;
if goback then return;
call write(O, @(' Enter command (H for help) ==> '), 34, @status);
end;
end.
edgetout;
end.
update procedure:

```
call cr$l;
call loop$char(10, OA$);
call loop$char(28, '*');
call write(0, @(' Station Configuration '), 23, @status);
call loop$char(27, ' ');
call cr$l;
call cr$l;
call write(0, @(' Host Address '), 15, @status);
call print$network$addr(@ia$setup ia$address(0));
i = 0;
call write(0, @(' Multicast Address(es)'), 24, @status);
if mc$setup.mc$byte$count = 0
then call writeln(0, @('No Multicast Addresses Defined'), 30, @status),
else
do while i < mc$setup mc$byte$count:
call print$network$addr(@mc$setup mc$address(i)),
call loop$char(24, ' ');
i = i + 6;
end;
call write(0, @ODH), 1, @status),
call write(0, @(' 82586 Configuration Block '), 28, @status);
call print$str(@configure, 1nfo(0), 10);
call cr$l;
call loop$char(29, '*');
call writeln(0, @(' Station Activities '), 20, @status);
call loop$char(29, '*');
call writeln(0, @(' Multicast Addresses Defined'), 30, @status),
call writeln(0, @(' Station Configuration '), 23, @status),
call writeln(0, @(' Host Address '), 15, @status);
call print$network$addr(@ia$setup ia$address(0));
call writeln(0, @(' 82586 Configuration Block '), 28, @status);
call print$str(@configure, 1nfo(0), 10);
call cr$l;
call loop$char(29, '*');
call writeln(0, @(' Station Activities '), 20, @status);
call loop$char(29, '*');
call writeln(0, @(' Multicast Addresses Defined'), 30, @status),
call writeln(0, @(' Station Configuration '), 23, @status),
call writeln(0, @(' Host Address '), 15, @status);
call write(0, @ODH), 1, @status),
call write(0, @(' 82586 Configuration Block '), 28, @status);
call print$str(@configure, 1nfo(0), 10);
call cr$l;
call loop$char(29, '*');
call writeln(0, @(' Station Activities '), 20, @status);
call loop$char(29, '*');
call writeln(0, @(' Multicast Addresses Defined'), 30, @status),
call writeln(0, @(' Station Configuration '), 23, @status),
call writeln(0, @(' Host Address '), 15, @status);
call write(0, @ODH), 1, @status),
call write(0, @(' 82586 Configuration Block '), 28, @status);
call print$str(@configure, 1nfo(0), 10);
call cr$l;
call loop$char(29, '*');
call writeln(0, @(' Station Activities '), 20, @status);
call loop$char(29, '*');
call writeln(0, @(' Multicast Addresses Defined'), 30, @status),
call writeln(0, @(' Station Configuration '), 23, @status),
call writeln(0, @(' Host Address '), 15, @status);
call write(0, @ODH), 1, @status),
call write(0, @(' 82586 Configuration Block '), 28, @status);
call print$str(@configure, 1nfo(0), 10);
call cr$l;
call loop$char(29, '*');
call writeln(0, @(' Station Activities '), 20, @status);
call loop$char(29, '*');
call writeln(0, @(' Multicast Addresses Defined'), 30, @status),
call writeln(0, @(' Station Configuration '), 23, @status),
call writeln(0, @(' Host Address '), 15, @status);
call write(0, @ODH), 1, @status),
call write(0, @(' 82586 Configuration Block '), 28, @status);
call print$str(@configure, 1nfo(0), 10);
call cr$l;
call loop$char(29, '*');
call writeln(0, @(' Station Activities '), 20, @status);
call loop$char(29, '*');
call writeln(0, @(' Multicast Addresses Defined'), 30, @status),
call writeln(0, @(' Station Configuration '), 23, @status),
call writeln(0, @(' Host Address '), 15, @status);
call write(0, @ODH), 1, @status),
call write(0, @(' 82586 Configuration Block '), 28, @status);
call print$str(@configure, 1nfo(0), 10);
call cr$l;
call loop$char(29, '*');
call writeln(0, @(' Station Activities '), 20, @status);
call loop$char(29, '*');
call writeln(0, @(' Multicast Addresses Defined'), 30, @status),
call writeln(0, @(' Station Configuration '), 23, @status),
call writeln(0, @(' Host Address '), 15, @status);
call write(0, @ODH), 1, @status),
call write(0, @(' 82586 Configuration Block '), 28, @status);
call print$str(@configure, 1nfo(0), 10);
call cr$l;
call loop$char(29, '*');
call writeln(0, @(' Station Activities '), 20, @status);
call loop$char(29, '*');
call writeln(0, @(' Multicast Addresses Defined'), 30, @status),
call writeln(0, @(' Station Configuration '), 23, @status),
call writeln(0, @(' Host Address '), 15, @status);
call write(0, @ODH), 1, @status),
call write(0, @(' 82586 Configuration Block '), 28, @status);
call print$str(@configure, 1nfo(0), 10);
call cr$l;
call loop$char(29, '*');
call writeln(0, @(' Station Activities '), 20, @status);
call loop$char(29, '*');
call writeln(0, @(' Multicast Addresses Defined'), 30, @status),
call writeln(0, @(' Station Configuration '), 23, @status),
call writeln(0, @(' Host Address '), 15, @status);
call write(0, @ODH), 1, @status),
call write(0, @(' 82586 Configuration Block '), 28, @status);
call print$str(@configure, 1nfo(0), 10);
call cr$l;
call loop$char(29, '*');
call writeln(0, @(' Station Activities '), 20, @status);
call loop$char(29, '*');
call writeln(0, @(' Multicast Addresses Defined'), 30, @status),
call writeln(0, @(' Station Configuration '), 23, @status),
call writeln(0, @(' Host Address '), 15, @status);
call write(0, @ODH), 1, @status),
call write(0, @(' 82586 Configuration Block '), 28, @status);
call print$str(@configure, 1nfo(0), 10);
call cr$l;
call loop$char(29, '*');
call writeln(0, @(' Station Activities '), 20, @status);
call loop$char(29, '*');
call writeln(0, @(' Multicast Addresses Defined'), 30, @status),
end update;

main

```

Traffic Simulator/Monitor Station Program (Continued)
do while (diagnose status and B00OH) <> 8000H.

end.

if diagnose status <> OAOO0H then call writeln(O, @('Diagnose failed'), 17, @status).

if configure status <> OAO00H then call writeln(O, @('Configure failed'), 18, @status).

if ia setups status <> OAOO0H then call writeln(O, @('IA Setup failed'), 17, @status).

if mc setups status <> OAOO0H then call writeln(O, @('MC Setup failed'), 17, @status).

scb chunk offset = offset (@transmit, status);

call writeln(O, @('Receive Unit is active'), 26, @status), disable.

scb cmd = O0100H.

output(CA@PORT) = CA.

call waitscb.

if not no transmission then do.

call write(O, @('---Transmit Command Block---'), 2B, @status);

call printnus (@transmit, status, 8);

call crsflf.

cur$cb$offset = offset (@transmit, status);

call pause;

do i = 1 to 60;

call time(250);

end.

call writeln(O, @('ODH, OAH, 'transmission started'), 23, @status);

call crsflf.

disable;

call writeln(O, @('ODH, OAH, 'transmission started'), 23, @status);

call crsflf.

disable;

call writeln(O, @('ODH, OAH, 'Receive Unit is active'), 26, @status).

end.

call update;

do forever.

call writeln(O, @('ODH, OAH, '), 2, @status);

do y = 0 to 5.

do case y:

call writeln(int(count, dhex));

call writeln(int(receive$count, dhex));

call writeln(scb crc Errs, dhex));

call writeln(scb aln$Errs, dhex));

call writeln(scb ovnerrs, dhex));

end.

call loop$char(char$count, ' '),

call loop$char(char$count, ' '),

end.

call getout.

call update.

char$count = 13 - char$count.

call loop$char(char$count, ' '),

call loop$char(char$count, ' '),

end.

end trans.

MODULE INFORMATION:

| CODE AREA SIZE | 23C3H | 9155D |
| CONSTANT AREA SIZE | OF88H | 3973D |
| VARIABLE AREA SIZE | 25B6H | 9822D |
| MAXIMUM STACK SIZE | 0092H | 146D |
| 1994 LINES READ | 9155D |
| 0 PROGRAM WARNINGS |
| 0 PROGRAM ERRORS |

DICTIONARY SUMMARY:

159KB MEMORY AVAILABLE
23KB MEMORY USED (14%)

END OF PL/M-86 COMPILATION

Traffic Simulator/Monitor Station Program (Continued)
The conditional compilation parameter 'EPROM27128' determines board ROM size. If it is true, the 80186's wait state generator is programmed to 0 wait state for upper 64K-byte memory locations. If it is false, the wait state generator is programmed to 0 wait state for upper 1M-byte memory locations. 

```
1 #include
2 do;
3 declare hib_ir label public;
4 declare main label external;
5 declare lit literally 'literally',
   UMCS_reg  lit 'OFFA0H',
   LMCS_reg  lit 'OFFA2H',
   PACS_reg  lit 'OFFA4H',
   MPCS_reg  lit 'OFFA6H',
   INT_MASK_reg lit 'OFFA8H',
   ISCPLOCLO  lit '03FBH',
   ISCPLOCLOHI lit '07H',
   SCC_CH_B_CMD lit '8300H',
   SCC_CH_A_CMD lit '8304H',
   SCC_CH_A_DATA lit '8306H',
   NMU    lit '0',
   CR     lit '0DH',
   LF     lit '0AH',
   BS     lit '0BH',
   SP     lit '20H',
   GM     lit '3FH',
   DEL    lit '07FH',
   BEL    lit '07H';
```
# System Configuration Pointer

```c
declare scp structure
(
    sysbus byte,
    unused (3) byte,
    iscp@addr@lo word,
    iscp@addr@hi word
)
at (OFFF6H) data (0, 0, 0, 0, 0, ISCP@LOC@LO, ISCP@LOC@HI);
```

```c
init@int@clt: procedure;
output(INT_mask_reg) = OFFH; /* mask all interrupts */
end init@int@clt;
```

```c
rra: procedure (reg_no byte);
declare reg_no byte;
if (reg_no and OFH) <> 0 then output(SCC_CH_A_CMD) = reg_no and OFH;
return input(SCC_CH_A_CMD);
end rra;
```

```c
rrb: procedure (reg_no byte);
declare reg_no byte;
if (reg_no and OFH) <> 0 then output(SCC_CH_B_CMD) = reg_no and OFH;
return input(SCC_CH_B_CMD);
end rrb;
```

```c
wra: procedure (reg_no, value);
declare (reg_no, value) byte;
if (reg_no and OFH) <> 0 then output(SCC_CH_A_CMD) = reg_no and OFH;
output (SCC_CH_A_CMD) = value;
end wra;
```

```c
wrb: procedure (reg_no, value);
declare (reg_no, value) byte;
if (reg_no and OFH) <> 0 then output(SCC_CH_B_CMD) = reg_no and OFH;
output (SCC_CH_B_CMD) = value;
end wrb;
```

```c
init@Scc@B: procedure;
call wrb(09, 01000000b); /* channel B reset */
```

186/586 High Integration Board Initialization Routine (Continued)
36  ccall wrb(04, 01001110b);  /* 2 stop, no parity, brf = 16x */
37  ccall wrb(03, 11000000b);  /* rx 8 bits/char, no auto-enable */
38  ccall wrb(05, 01100000b);  /* tx 8 bits/char */
39  ccall wrb(10, 00000000b);
40  ccall wrb(11, 01010110b);  /* rxc = txc = BRG, trxc = BRG out */
41  ccall wrb(12, 00001011b);  /* baud rate = 9600 */
42  ccall wrb(13, 00000000b);
43  ccall wrb(14, 00000011b);  /* BRG source = SYS CLK, enable BRG */
44  ccall wrb(15, 00000000b);  /* all ext status interrupts off */
45  ccall wrb(03, 11000001b);  /* scc-b receive enable */
46  ccall wrb(05, 11101010b);  /* scc-b transmit enable, dtr on, rts on */
47  end init$SCCB;
48  ccall: procedure byte public,
49  do while (input(SCC_CH_B_CMD) and 1) = 0; end;
50  return (input(SCC_CH_B_DATA));
51  end ccall;
52  ccall: procedure (char) public;
53  do while (input(SCC_CH_B_CMD) and 4) = 0; end;
54          output(SCC_CH_B_DATA) = char;
55  end ccall;
56  ccall: procedure (file@id, msg@ptr, count, actual@ptr, status@ptr) public;
57  declare file@id word,
58          msg@ptr pointer,
59          count word,
60          actual@ptr pointer,
61          status@ptr pointer,
62          msg based msg@ptr (1) byte,
63          buf (200) byte,
64          actual based actual@ptr word,
65          status based status@ptr word,
66          i word,
67          ch byte;
68  /* This procedure implements the ISIS read procedure. All control characters */
69  /* except LF, BS, and DEL are ignored. If BS or DEL is encountered, a */
70  /* backspace is done. */
71  status = 0;
72  i, ch = 0;
73  do while (ch <> CR) and (ch <> LF) and (i < 198);
74      ch = ccall and 07FH;
75  if (ch = BS) or (ch = DEL) then
76      do;
77  186/586 High Integration Board Initialization Routine (Continued)
if \( i > 0 \) then

do:
\[ i = i - 1; \]
call c\@out(DEL);
call c\@out(BS);
call c\@out(SP);
call c\@out(DEL);
call c\@out(BS);
end;
else
\[ \text{call c\@out(BEL);} \]
end;
else
\[ \text{if } ch \geq \text{ SP then} \]
do:
call c\@out(ch);
buf(i) = ch;
i = i + 1;
end;
else
\[ \text{if } (ch = \text{ CR}) \text{ or } (ch = \text{ LF}) \text{ then} \]
do:
\[ \text{buf}(i) = \text{ CR}; \]
\[ \text{buf}(i + 1) = \text{ LF}; \]
i = i + 2;
end;
else
\[ \text{call c\@out(BEL);} \]
call c\@out(CR);
if \( i > \text{ count} \) then i = \text{ count};
do i = 0 to actual - 1;
msg(i) = buf(i);
end;
end read;
csts: \text{procedure byte public,}
return ((\text{input}(\text{SCC_CH_B_CMD}) \text{ and } 1) \not\equiv 0),
csts:
write: \text{procedure (file\$id, msg\$ptr, count, status\$ptr) public,}
declare (file\$id, count) word,
(msg\$ptr, status\$ptr) pointer,
msg based msg\$ptr (1) byte,
status based status\$ptr word,
ch byte,
i word,
/* This procedure implements the ISIS write */
status = 0;
do while \( i < \text{count} \),
\[ \text{ch} = \text{msg}(i), \]
if \((\text{ch} \geq \text{SP})\) and \((\text{ch} < \text{DEL})\) or \((\text{ch} = \text{CR})\) or \((\text{ch} = \text{LF})\) or \((\text{ch} = \text{NUL})\)
then
\[ \text{call c@out(ch);} \]
else
\[ \text{call c@out(TM);} \]
i = i + 1;
end;
end write:

**hib_ir:**

*IF EPROM27128*
\[ \text{output(UMCS\_reg)} = \text{OF03BH}; \]
/\* Starting Address = OF0000H, no wait state */

*ELSE*
\[ \text{output(UMCS\_reg)} = \text{OE03BH}; \]
/\* Starting Address = OE0000H, no wait state */

*ENDIF*

\[ \text{output(LMCS\_reg)} = \text{03FCH}; \]
/\* 16K, no wait state */
\[ \text{output(PACS\_reg)} = \text{03ECH}; \]
/\* PBA = 8000H, no wait state for PSC0-2 */
\[ \text{output(MPCS\_reg)} = \text{0BFH}; \]
/\* Peripherals in I/O space, no Al & A2 provided, 3 wait states for PSC4-6 */

\[ \text{call init@int@clt;} \]
\[ \text{call init@SCC@B;} \]
\[ \text{go to main;} \]
\[ \text{end inI186;} \]

---

186/586 High Integration Board Initialization Routine (Continued)
APPENDIX C
THE 82530 SCC - 80186 INTERFACE AP BRIEF

INTRODUCTION

The object of this document is to give the 82530 system designer an in-depth worst case design analysis of the typical interface to a 80186 based system. This document has been revised to include the new specifications for the 6 MHz 82530. The new specifications yield better margins and a 1 wait state interface to the CPU (2 wait states are required for DMA cycles). These new specifications will appear in the 1987 data sheet and advanced specification information can be obtained from your local Intel sales office. The following analysis includes a discussion of how the interface TTL is utilized to meet the timing requirements of the 80186 and the 82530. In addition, several optional interface configurations are also considered.

INTERFACE OVERVIEW

The 82530 - 80186 interface requires the TTL circuitry illustrated in Figure 1. Using five 14 pin TTL packages, 74LS74, 74AS74, 74AS08, 74AS04, and 74LS32, the following operational modes are supported:

- Polling
- Interrupt in vectored mode
- Interrupt in non-vectored mode
- Half-duplex DMA on both channels
- Full-duplex DMA on channel A

A brief description of the interface functional requirements during the five possible BUS operations follows below.

Figure 1. 82530-80186 Interface
READ CYCLE: The 80186 read cycle requirements are met without any additional logic, Figure 2. At least one wait state is required to meet the 82530 tAD access time.

WRITE CYCLE: The 82530 requires that data must be valid while the WR pulse is low, Figure 3. A D Flip-Flop delays the leading edge of WR until the falling edge of CLOCKOUT when data is guaranteed valid and WR is guaranteed active. The CLOCKOUT signal is inverted to assure that WR is active low before the D Flip-Flop is clocked. No wait states are necessary to meet the 82530's WR cycle requirements, but one is assumed from the RD cycle.

INTA CYCLE: During an interrupt acknowledge cycle, the 80186 provides two INTA pulses, one per bus cycle, separated by two idle states. The 82530 expects only one long INTA pulse with a RD pulse occurring only after the 82530 IEI/IEO daisy chain settles. As
illustrated in Figure 4, the INTA signal is sampled on the rising edge of CLK (82530). Two D Flip-Flops and two TTL gates, U2 and U5, are implemented to generate the proper INTA and RD pulses. Also, the INT signal is passively pulled high, through a 1 k resistor, and inverted through U3 to meet the 80186's active high requirement.

DMA CYCLE: Conveniently, the 80186 DMA cycle timings are the same as generic read and write operations. Therefore, with two wait states, only two modifications to the DMA request signals are necessary. First, the RDYREQA signal is inverted through U3 similar to the INT signal, and second the DTR/REQA signal is conditioned through a D Flip-Flop to prevent inadvertent back to back DMA cycles. Because the 82530 DTR/REQA signal remains active low for over five CLK (82530)'s, an additional DMA cycle could occur. This uncertain condition is corrected when U4 resets the DTR/REQ signal inactive high. Full Duplex on both DMA channels can easily be supported with one extra D Flip-Flop and an inverter.

RESET: The 82530 does not have a dedicated RESET input. Instead, the simultaneous assertion of both RD and WR causes a hardware reset. This hardware reset is implemented through U2, U3, and U4.

ALTERNATIVE INTERFACE CONFIGURATIONS

Due to its wide range of applications, the 82530 interface can have many varying configurations. In most of these applications the supported modes of operation need not be as extensive as the typical interface used in this analysis. Two alternative configurations are discussed below.

8288 BUS CONTROLLER: An 80186 based system implementing an 8288 bus controller will not require the preconditioning of the WR signal through the D Flip-Flop U4. When utilizing an 8288, the control signal IOWC does not go active until data is valid, therefore, meeting the timing requirements of the 82530. In such a configuration, it will be necessary to logically OR the IOWC with reset to accommodate a hardware reset operation.

NON-VECTORED INTERRUPTS: If the 82530 is to be operated in the non-vectored interrupt mode (B step only), the interface will not require U1 or U5. Instead, INTA on the 82530 should be pulled high, and pin 3 of U2 (RD AND RESET) should be fed directly into the RD input of the SCC.

Obviously, the amount of required interface logic is application dependent and in many cases can be considerably less than required by the typical configuration, supporting all modes of SCC operation.

DESIGN ANALYSIS

This design analysis is for a typical microprocessor system, pictured in Figure 5. The Timing analysis assumes an 8 MHz 80186 and a 4 MHz 82530. Also, included in the analysis are bus loading, and TTL-MOS compatibility considerations.
Figure 5. Typical Microprocessor System

Bus Loading and Voltage Level Compatibilities

The data and address lines do not exceed the drive capability of either 80186 or the 82530. There are several control lines that drive more than one TTL equivalent input. The drive capability of these lines are detailed below.

**WR:** The WR signal drives U3 and U4.
- \( I_{OL} (2.0 \text{ mA}) > I_{IL} (-0.4 \text{ mA} + -0.5 \text{ mA}) \)
- \( I_{OH} (-400 \mu A) > I_{OIH} (20 \mu A + 20 \mu A) \)

**PCS5:** The PCS5 signal drives U2 and U4.
- \( I_{OL} (2.0 \text{ mA}) > I_{IL} (-0.5 \text{ mA} + -0.5 \text{ mA}) \)
- \( I_{OH} (-400 \mu A) > I_{OIH} (20 \mu A + 20 \mu A) \)

**INTA:** The INTA signal drives 2(U2) and U5.
- \( I_{OL} (2.0 \text{ mA}) > I_{IL} (-0.4 \text{ mA} + -0.8 \text{ mA} + -0.4 \text{ mA}) \)
- \( I_{OH} (-400 \mu A) > I_{OIH} (20 \mu A + 40 \mu A + 20 \mu A) \)

All the 82530 I/O pins are TTL voltage level compatible.

TIMING ANALYSIS

Certain symbolic conventions are adhered to throughout the analysis below and are introduced for clarity.

1. All timing variables with a lower case first letter are 82530 timing requirements or responses (i.e., \( t_{RR} \)).
2. All timing variables with Upper case first letters are 80186 timing responses or requirements unless preceded by another device’s alpha-numeric code (i.e., \( T_{CE} \) or \( '373 T_{PD} \)).
3. In the write cycle analysis, the timing variable \( T_{PDWR186-WR530} \) represents the propagation delay between the leading or trailing edge of the WR signal leaving the 80186 and the WR edge arrival at the 82530 WR input.

Read Cycle

1. **tAR:** Address valid to RD active set up time for the 82530. Since the propagation delay is the worst case path in the assumed typical system, the margin is calculated only for a propagation delay constrained and not an ALE limited path. The spec value is 0 ns minimum.

   \[
   1 T_{ACL} - T_{CL}(\text{max}) - '245 T_{PD}(\text{max}) + T_{CIR}(\text{min}) + 2(U2) T_{PD}(\text{min}) - t_{AR}(\text{min})
   \]

   \[
   = 125 - 55 - 20.8 + 10 + 2(2) - 0 = 63.2 \text{ ns margin}
   \]
2. \( t_{RA} \): Address to RD inactive hold time. The ALE delay is the worst case path and the 82530 requires 0 ns minimum.

\[
\begin{align*}
& 1 \text{Tclcl} - T_{CIRh} (max) + T_{CHlh} (min) + '373 LE \\
& Tpd(min) - 2(U2) Tpd(max) \\
= & 55 - 55 + 5 + 8 - 2(5.5) = 2 \text{ ns margin}
\end{align*}
\]

3. \( t_{CLR} \): CS active low to RD active low set up time. The 82530 spec value is 0 ns minimum.

\[
\begin{align*}
& 1 \text{Tclcl} - T_{CIRh} (max) - T_{CSv} (min) - U2 \\
& \text{skew}(RD - CS) + U2 Tpd(max) \\
= & 125 - 66 - 10 - 1 + 2 = 50 \text{ ns margin}
\end{align*}
\]

4. \( t_{RCS} \): RD inactive to CS inactive hold time. The 82530 spec calls for 0 ns minimum.

\[
\begin{align*}
& 1 \text{Tclcl} + 1 \text{Tchcl} - T_{CSv} (max) + T_{CIRh} (min) - U2 \\
& \text{skew} (RD - CS) + U2 Tpd(min) - t_{CHR} \\
= & 125 + 55 - 35 - 10 - 1 + 2 = 131 \text{ ns margin}
\end{align*}
\]

5. \( t_{RR} \): RD pulse active low time. One 80186 wait state is included to meet the 150 ns minimum timing requirements of the 82530.

\[
\begin{align*}
& T_{CIRh} (min) - (T_{CIRh} \text{wait state}) - 2(U2 \text{skew}) - t_{RR} \\
= & (250 - 50) + (125) - 2(1) - 150 = 173 \text{ ns margin}
\end{align*}
\]

7. \( t_{RDV} \): RD active low to data valid maximum delay for 80186 read data set up time (Tdvc1 = 20 ns). The margin is calculated on the Propagation delay path (worst case).

\[
\begin{align*}
& 2 \text{Tclcl} + \text{Tclocv} (max) - T_{CIR} (max) + T_{Dvc1} (min) - '245 Tpd(max) - 82530 TRDV(max) - 2(U2) Tpd(max) \\
= & 2(125) + (125) - 70 - 20 - 14.2 - 105 - 2(5.5) = 154 \text{ ns margin}
\end{align*}
\]

8. \( t_{DF} \): RD inactive to data output float delay. The margin is calculated to DEN active low of next cycle.

\[
\begin{align*}
& 2 \text{Tclcl} + T_{CIRh} (max) + T_{Chctv} (min) - 2(U2) Tpd(max) - 82530 t_{DF} (max) \\
= & 250 + 55 - 55 + 10 - 11 - 70 = 179 \text{ ns margin}
\end{align*}
\]

9. \( t_{AD} \): Address required valid to read data valid maximum delay. The 82530 spec value is 325 ns maximum.

\[
\begin{align*}
& 3 \text{Tclcl} + 1(T_{CIRh} \text{wait state}) - T_{Dvc1} (max) - 373 LE \\
& Tpd(max) - '245 Tpd - Tdvc1 (min) - t_{AD} \\
= & 375 + 125 - 55 - 20.8 - 14.2 - 20 - 325 = 65 \text{ ns margin}
\end{align*}
\]

**Write Cycle**

1. \( t_{AW} \): Address required valid to WR active low set up time. The 82530 spec is 0 ns minimum.

\[
\begin{align*}
& T_{CIR} - T_{CIRh} (max) - T_{Ctvct} (min) - '373 Tpd(max) \\
& + TpdWR186 - WR530(LOW) [Tclcl - Tcvctv(min) + U3 Tpd(min) + U4 Tpd(min)] - t_{AW} \\
= & 125 - 55 - 5 - 20.8 + [125 - 5 + 1 + 4.4] = 170.6 \text{ ns margin}
\end{align*}
\]

2. \( t_{WA} \): WR inactive to address invalid hold time. The 82530 spec is 0 ns.

\[
\begin{align*}
& T_{CIRh} (min) - T_{Ctvct} (max) + T_{CHlh} (min) + '373 LE \\
& Tpd(min) - TpdWR186 = WR530(HIGH) [U2 Tpd(max) + U3 Tpd(max) + U4 Tpd(max)] \\
= & 55 - 55 + 5 + 8 - [5.5 + 3 + 7.1] = -2.6 \text{ ns margin}
\end{align*}
\]

5. \( t_{CHW} \): Chip Select inactive high to WR active low set up time. The 82530 spec is 5 ns.

\[
\begin{align*}
& 1 \text{Tclcl} - T_{CIRh} (min) - U2 Tpd(max) \\
& - TpdWR186 = WR530(HIGH) [U2 Tpd(max) + U3 Tpd(max) + U4 Tpd(max)] \\
= & 35 + 1.5 - [5.5 + 3 + 7.1] = 20.9 \text{ ns margin}
\end{align*}
\]

**Write Cycle**

1. \( t_{AW} \): Address required valid to WR active low set up time. The 82530 spec is 0 ns minimum.

\[
\begin{align*}
& T_{CIR} - T_{CIRh} (max) - T_{Ctvct} (min) - '373 Tpd(max) \\
& + TpdWR186 - WR530(LOW) [Tclcl + Tcvctv(min) + U3 Tpd(min) + U4 Tpd(min)] - t_{AW} \\
= & 125 - 55 - 5 - 20.8 + [125 - 5 + 1 + 4.4] = 170.6 \text{ ns margin}
\end{align*}
\]

2. \( t_{WA} \): WR inactive to address invalid hold time. The 82530 spec is 0 ns.

\[
\begin{align*}
& T_{CIRh} (min) - T_{Ctvct} (max) + T_{CHlh} (min) + '373 LE \\
& Tpd(min) - TpdWR186 = WR530(HIGH) [U2 Tpd(max) + U3 Tpd(max) + U4 Tpd(max)] \\
= & 55 - 55 + 5 + 8 - [5.5 + 3 + 7.1] = -2.6 \text{ ns margin}
\end{align*}
\]

3. \( t_{CLW} \): Chip select active low to WR active low hold time. The 82530 spec is 0 ns.

\[
\begin{align*}
& 1 \text{Tclcl} + 1(T_{CIRh} \text{wait state}) - T_{Dvc1} (max) - 373 LE \\
& Tpd(max) - '245 Tpd - Tdvc1 (min) - t_{AD} \\
= & 375 + 125 - 55 - 20.8 - 14.2 - 20 - 325 = 65 \text{ ns margin}
\end{align*}
\]

4. \( t_{WCS} \): WR invalid to Chip Select invalid hold time. 82530 spec is 0 ns.

\[
\begin{align*}
& T_{CIRh} (min) - U2 Tpd(max) \\
& - TpdWR186 = WR530(HIGH) [U2 Tpd(max) + U3 Tpd(max) + U4 Tpd(max)] \\
= & 35 + 1.5 - [5.5 + 3 + 7.1] = 20.9 \text{ ns margin}
\end{align*}
\]

5. \( t_{CHW} \): Chip Select inactive high to WR active low set up time. The 82530 spec is 5 ns.

\[
\begin{align*}
& 1 \text{Tclcl} + T_{CIRh} (min) - T_{Ctvct} (min) - T_{CHlh} (max) \\
& - U2 Tpd(max) + TpdWR186 = WR530(LOW) [Tclcl - Tcvctv(min) + U3 Tpd(min) + U4 Tpd(min)] - t_{CHW} \\
= & 125 - 55 - 5 - 35 - 5.5 + [125 - 5 + 1 + 4.4] = 20.9 \text{ ns margin}
\end{align*}
\]

6. \( t_{WW} \): WR active low pulse. 82530 requires a minimum of 60 ns from the falling to the rising edge of WR. This includes one wait state.
Must be corrected for tWWh 2Tclc - 40 + 1 (Tclclwait state) - TpdWR186 - WR530(LOW) [Tccl - Tcvctx(min) + U3 Tpd(max) + U4 Tpd(max)] + TpdWR186 - WR530(HIGH) [U2 Tpd(min) + U3 Tpd(min) + U4 Tpd(min)] = 120 + 1(129) - 125 - 5 + 4.5 + 9.2] - [1.5 + 1.3] - 60 = 35.6 ns margin

7. tDW: Data valid to WR active low setup time. The 82530 spec requires 0 ns.

* Tcvctx(min) - Tclcl - Tcvctx(min) + U2 Tpd(max) + U3 Tpd(min) + U4 Tpd(min)]

8. tWD: Data valid to WR inactive high hold time. The 82530 requires a hold time of 0 ns.

* Tclcl - skew [Tcvctx(max) + Tcvctx(min)] + TpdWR186 - WR530(HIGH) [U2 Tpd(max) + U3 Tpd(max) + U4 Tpd(max)]

INTA Cycle:

1. tIC: This 82530 spec implies that the INTA signal is latched internally on the rising edge of CLK (82530). Therefore the maximum delay between the 80186 asserting INTA active low or inactive high and the 82530 internally recognizing the new state of INTA is the propagation delay through U1 plus the 82530 CLK period.

* U1 Tpd(max) + 82530 CLK period

2. tCI: rising edge of CLK to INTA hold time. This spec requires that the state of INTA remains constant for 100 ns after the rising edge of CLK. If this spec is violated any change in the state of INTA may not be internally latched in the 82530. tCI becomes critical at the end of an INTA cycle when INTA goes inactive. When calculating margins with tCI, an extra 82530 CLK period must be added to the INTA inactive delay.

3. tIW: INTA inactive high to WR active low minimum setup time. The spec pertains only to 82530 WR cycle and has a value of 55 ns. The margin is calculated assuming an 82530 WR cycle occurs immediately after an INTA cycle. Since the CPU cycles following an 82530 INTA cycle are devoted to locating and executing the proper interrupt service routine, this condition should never exist. 82530 drivers should insure that at least one CPU cycle separates INTA and WR or RD cycles.

4. tWI: WR inactive high to INTA active low minimum hold time. The spec is 0 ns and the margin assumes CLK coincident with INTA.

* Tccl - Tcvctx(max) - TpdWR186 - WR530(HIGH) [U3 Tpd(max) + U4 Tpd(max)] + Tcvctx(min) + U1 Tpd(min)

5. tIR: INTA inactive high to RD active low minimum setup time. This spec pertains only to 82530 RD cycles and has a value of 55 ns. The margin is calculated in the same manner as tIW.

6. tRI: RD inactive high to INTA active low minimum hold time. The spec is 0 ns and the margin assumes CLK coincident with INTA.

7. tID: INTA active low to RD active low minimum setup time. This parameter is system dependent. For any SCC in the daisy chain, tID must be greater than the sum of tCEQ for the highest priority device in the daisy chain, tE1 for this particular SCC, and tEIEO for each device separating them in the daisy chain. The typical system with only 1 SCC requires tID to be greater than tCEQ. Since tE1 occurs coincidently with tCEQ and it is smaller it can be neglected. Additionally, tEIEO does not have any relevance to a system with only one SCC. Therefore tID > tCEQ = 250 ns.

8. tDV: RD active low to interrupt vector valid delay. The 80186 expects the interrupt vector to be valid on the data bus a minimum of 20 ns before T4 of the second acknowledge cycle (Tdvcl). tDV spec is 100 ns maximum.

* 3 Tclc - Tcvctx(max) - U5 Tpd(max) - U2 Tpd(max) - tDV(max) - '245 Tpd(max) - Tdvcl(min)

148 ns margin
9. **tII:** RD pulse low time. The 82530 requires a minimum of 125 ns.

\[
3 \cdot T_{cll} - T_{cvcx(max)} - U_3 T_{pd(max)} - U_2 T_{pd(max)} + T_{cvcx(min)} + U_5 T_{pd(min)} + U_2 T_{pd(min)} - t{l}(min) = 375 - 70 - 25 - 5.5 + 5 + 6 + 1.5 - 125 = 162 \text{ ns margin}
\]

**DMA Cycle**

Fortunately, the 80186 DMA controller emulates CPU read and write cycle operation during DMA transfers. The DMA transfer timings are satisfied using the above analysis. Because of the 80186 DMA request input requirements, two wait states are necessary to prevent inadvertent DMA cycles. There are also CPU DMA intracycle timing considerations that need to be addressed.

1. **tDRD:** RD inactive high to DTRREQ (REQUEST) inactive high delay. Unlike the READYREQ signal, DTRREQ does not immediately go inactive after the requested DMA transfer begins. Instead, the DTRREQ remains active for a maximum of 5 tCY + 300 ns. This delayed request pulse could trigger a second DMA transfer. To avoid this undesirable condition, a D Flip Flop is implemented to reset the DTRREQ signal inactive low following the initiation of the requested DMA transfer. To determine if back to back DMA transfers are required in a source synchronized configuration, the 80186 DMA controller samples the service request line 25 ns before T1 of the deposit cycle, the second cycle of the transfer.

\[
4 \cdot T_{cll} - T_{cvcx(max)} - U_4 T_{pd(max)} - T_{drqcl(min)} = 500 - 66 - 10.5 - 25 = 398.5 \text{ ns margin}
\]

2. **tRRI:** 82530 RD active low to REQ inactive high delay. Assuming source synchronized DMA transfer, the 80186 requires only one wait state to meet the tRRI spec of 200 ns. Two are included for consistency with tWRI.

\[
2 \cdot T_{cll} + 2(T_{cll} \text{ wait state}) - T_{cvcx(max)} - 2(U_2) T_{pd(max)} - T_{drqcl} = 2(125) + 2(125) - 70 - 2(5.5) - 200 = 219 \text{ ns margin}
\]

3. **tWRI:** 82530 WR active low to REQ inactive high delay. Assuming destination synchronized DMA transfers, the 80186 needs two wait states to meet the tWRI spec. This is because the 80186 DMA controller samples requests two clocks before the end of the deposit cycle. This leaves only 1 Tccl + n(wait states) minus WR active delay for the 82530 to inactivate its REQ signal.

\[
T_{cll} + 2(T_{cll} \text{ wait state}) - T_{cvcx(min)} - T_{pd(85086-85030 \text{ LOW})} + U_3 T_{pd(max)} + U_4 T_{pd(max)} - T_{drqcl} - t_{WRI} = 375 - 5 - [125 - 5 + 4.5 + 9.2] - 25 - 200 = 11.3 \text{ ns margin}
\]

**NOTE:**

If one wait state DMA interface is required, external logic, like that used on the DTRREQ signal, can be used to force the 82530 REQ signal inactive.

4. **tREC:** CLK recovery time. Due to the internal data path, a recovery period is required between SCC bus transactions to resolve metastable conditions internal to the SCC. The DMA request lines are marked from requesting service until after the tREC has elapsed. In addition, the CPU should not be allowed to violate this recovery period when interleaving DMA transfers and CPU bus cycles. Software drivers or external logic should orchestrate the CPU and DMA controller operation to prevent tREC violation.

**Reset Operation**

During hardware reset, the system RESET signal is asserted high for a minimum of four 80186 clock cycles (1000 ns). The 82530 requires WR and RD to be simultaneously asserted low for a minimum of 250 ns.

\[
4 \cdot T_{cll} - U_3 T_{pd(max)} - 2(U_2) T_{pd(max)} + U_4 T_{pd(min)} - t_{REC} = 1000 - 17.5 - 2(5.5) + 3.5 - 250 = 725 \text{ ns margin}
\]
Implementing StarLAN with the Intel 82588
1.0 INTRODUCTION

Personal computers have become the most prolific workstation in the office, serving a wide range of needs such as word processing, spreadsheets, and data bases. The need to interconnect PCs in a local environment has clearly emerged, for purposes such as the sharing of file, print, and communication servers; downline loading of files and application programs; electronic mail; etc. Proliferation of the PC makes it the workstation of choice for accessing the corporate mainframe/s; this function can be performed much more efficiently and economically when clusters of PCs are already interconnected through Local Area Networks (LANs). According to market surveys, the installed base of PCs in business environments reached about 10 million units year-end '85, with only a small fraction connected via LANs. The installed base is expected to double by 1990. There is clearly a great need for locally interconnecting these machines; furthermore, end users expect interconnectability across vendors. Thus, there is an urgent need for industry standards to promote cost effective PC LANs.

A large number of proprietary PC LANs have become available for the office environment over the past several years. Many of these suffer from high installed cost, technical deficiencies, non-conformance to industry standards, and general lack of industry backing. StarLAN, in Intel's opinion, is one of the few networks which will emerge as a standard. It utilizes a proven network access method, it is implemented with proven VLSI components; it is cost effective, easily installable and reconfigurable; it is technically competent; and it enjoys the backing of a large cross section of the industry which is collaborating to develop a standard (IEEE 802.3, type 1BASE5).

1.1 StarLAN

StarLAN is a 1 Mb/s network based on the CSMA/CD access method (Carrier Sense, Multiple Access with Collision Detection). It works over standard, unshielded, twisted pair telephone wiring. Typically, the wiring connects each desk to a wiring closet in a star topology (from which the IEEE Task Force working on the standard derived the name StarLAN in 1984). In fact, telephone and StarLAN wiring can coexist in the same twisted pair bundle connecting a desk to the wiring closet. Abundant quantities of unused phone wiring exist in most office environments, particularly in the U.S. The StarLAN concept of wiring and networking concepts was originated by AT&T Information Systems.

1.2 The 82588

The 82588 is a single-chip LAN controller designed for CSMA/CD networks. It integrates in one chip all functions needed for such networks. Besides implementing the standard CSMA/CD functions like framing, deferring, backing off and retrying on collisions, transmitting and receiving frames, it performs data encoding and decoding in Manchester or NRZI format, carrier sensing and collision detection, all up to a speed of 2 Mb/s (independent of the chosen encoding scheme). These functions make it an optimum controller for a StarLAN node. The 82588 has a very conventional microcomputer bus interface, easing the job of interfacing it to any processor.

1.3 Organization of the Application Note

This application note has two objectives. One is to describe StarLAN in practical terms to prospective implementers. The other is to illustrate designing with 82588, particularly as related to StarLAN which is expected to emerge as its largest application area.

Section 2 of this Application Note describes the StarLAN network, its basic components, collision detection, signal propagation and network parameters. Sections 3 and 4 describe the 82588 LAN controller and its role in the StarLAN network. Section 5 goes into the details of designing a StarLAN node for the IBM PC. Section 6 describes the design of the HUB. Both these designs have been implemented and operated in an actual StarLAN environment. Section 7 documents the software used to drive the 82588. It gives the actual procedures used to do operations like, configure, transmit and receive frames. It also shows how to use the DMA controller and interrupt controller in the IBM PC and goes into the details of doing I/O on the PC using DOS calls. Appendix A shows oscilloscope traces of the signals at various points in the network. Appendix B describes the multiple point extension (MPE) being considered by IEEE. Appendixes C and D talk about advanced usages of the 82588; working with only one DMA channel, and measuring network delays with the 82588.

1.4 References

For additional information on the 82588, see the Intel Microcommunications Handbook. StarLAN specifications are currently available in draft standard form through the IEEE 802.3 Working Group.

2.0 StarLAN

StarLAN is a low cost 1 Mb/s networking solution aimed at office automation applications. It uses a star
topology with the nodes connected in a point-to-point fashion to a central HUB. HUBs can be connected in a hierarchical fashion. Up to 5 levels are supported. The maximum distance between a node and the adjacent HUB or between two adjacent HUBs is 800 ft. (about 250 meters) for 24 gauge wire and 600 ft. (about 200 meters) for 26 gauge wire. Maximum node-to-node distance with one HUB is 0.5 km, hence IEEE 802.3 designation of type 1BASE5. 1 stands for 1 Mb/s and BASE for baseband. (StarLAN doesn’t preclude the use of more than 800 ft wiring provided 6.5 dB maximum attenuation is met, and cable propagation delay is no more than 4 bit times).

One of the most attractive features of StarLAN is that it uses telephone grade twisted pair wire for the transmission medium. In fact, existing installed telephone wiring can also be used for StarLAN. Telephone wiring is very economical to buy and install. Although use of telephone wiring is an obvious advantage, for small clusters of nodes, it is possible to work around the use of building wiring.

Factors contributing to low cost are:
1) Use of telephone grade, unshielded, 24 or 26 gauge twisted pair wire transmission media.
2) Installed base of redundant telephone wiring in most buildings.
3) Buildings are designed for star topology wiring. They have conduits leading to a central location.
4) Availability of low cost VLSI LAN controllers like the 82588 for low cost applications and the 82586 for high performance applications.
5) Off-the-shelf, Low cost RS-422, RS-485 drivers/receivers compatible with the StarLAN analog interface requirements.

2.1 StarLAN Topology

StarLAN, as the name suggests, uses a star topology. The nodes are at the extremities of a star and the central point is called a HUB. There can be more than one HUB in a network. The HUBs are connected in a hierarchical fashion resembling an inverted tree, as shown in Figure 1, where nodes are shown as PCs. The HUB at the base (at level 3) of the tree is called the Header Hub (HHUB) and others are called Intermediate HUBs (IHUB). It will become apparent, later in this section, that topologically, this entire network of nodes and HUBs is equivalent to one where all the nodes are connected to a single HUB. Also StarLAN doesn’t limit the number of nodes or HUBs at any given level.

2.1.1 TELEPHONE NETWORK

StarLAN is structured to run parallel to the telephone network in a building. The telephone network has, in fact, exactly the same star topology as StarLAN. Let us now examine how the telephone system is typically laid out in a building in the USA. Figure 2 shows how a typical building is wired for telephones. 24 gauge unshielded twisted pair wires emanate from a Wiring Closet. The wires are in bundles of 25 or 50 pairs. The bundle is called D inside wiring (DIW). The wires in these cables end up at modular telephone jacks in the wall. The telephone set is either connected directly to

*Maximum of 5 HUB levels.
*PCs or DTEs can connect directly at any level.

Figure 1. StarLAN Topology
the jack or through an extension cable. Each telephone generally needs one twisted pair for voice and another for auxiliary power. Thus, each modular jack has 2 twisted pairs (4 wires) connected to it. A 25 pair DIW cable can thus be used for up to 12 telephone connections. In most buildings, not all pairs in the bundle are used. Typically, a cable is used for only 4 to 8 telephone connections. This practice is followed by telephone companies because it is cheaper to install extra wires initially, rather than retrofitting to expand the existing number of connections. As a result, a lot of extra, unused wiring exists in a building. The stretch of cable between the wiring closet and the telephone jack is typically less than 800 ft. (250 meters). In the wiring closet the incoming wires from the telephones are routed to another wiring closet, a PABX or to the central office through an interconnect matrix. Thus, the wiring closet is a concentration point in the telephone network. There is also a redundancy of wires between the wiring closets.

2.1.2 StarLAN AND THE TELEPHONE NETWORK

StarLAN does not have to run on building wiring, but the fact that it can significantly adds to its attractiveness. Figure 3 shows how StarLAN piggybacks on telephone wiring. Each node needs two twisted pair wires to connect to the HUB. The unused wires in the 25 pair DIW cables provide an electrical path to the wiring closet, where the HUB is located. Note that the telephone and StarLAN are electrically isolated. They only use the wires in the same bundle cable to connect to the wiring closet. Within the wiring closet, StarLAN wires connect to a HUB and telephone wires are routed to a different path. Similar cable sharing can occur in connecting HUBs to one another. See Figure 4 for a typical office wired for StarLAN through telephone wiring.

Figure 2. Telephone Wiring in a Building

Figure 3. Coexistence of Telephone and StarLAN

*StarLAN and telephones share the same bundle, but are electrically isolated.
*StarLAN uses the unused wires in existing bundles.
Figure 4. A Typical Office Using Telephone Wiring for StarLAN
2.1.3 StarLAN AND Ethernet

StarLAN and Ethernet are similar CSMA/CD networks. Since Ethernet has existed longer and is better understood, a comparison of Ethernet with StarLAN is worthwhile.

1. The data rate of Ethernet is 10Mb/s and that of StarLAN is 1 Mb/s.

2. Ethernet uses a bus topology with each node connected to a coaxial cable bus via a 50 meter transceiver cable containing four shielded twisted pair wires. StarLAN uses a star topology, with each node connected to a central HUB by a point to point link through two pairs of unshielded twisted pair wires.

3. Collision detection in Ethernet is done by the transceiver connected to the coaxial cable. Electrically, it is done by sensing the energy level on the coax cable. Collision detection in StarLAN is done in the HUB by sensing activity on more than one input line connected to the HUB.

4. In Ethernet, the presence of collision is signalled by the transceiver to the node by a special collision detect signal. In StarLAN, it is signalled by the HUB using a special collision presence signal on the receive data line to the node.

5. Ethernet cable segments are interconnected using repeaters in a non-hierarchical fashion so that the distance between any two nodes does not exceed 2.8 kilometers. In StarLAN, the maximum distance between any two nodes is 2.5 kilometers. This is achieved by wiring a maximum of five levels of HUBs in a hierarchical fashion.

2.2 Basic StarLAN Components

A StarLAN network has three basic components:

1. StarLAN node interface
2. StarLAN HUB
3. Cable

![Figure 5. Ethernet and StarLAN Similarities](image)
2.2.1 A StarLAN Node Interface

Figure 6 shows a typical StarLAN node interface. It interfaces to a processor on the system side. The processor runs the networking software. The heart of the node interface is the LAN controller which does the job of receiving and transmitting the frames in adherence to the IEEE 802.3 standard protocol. It maintains all the timings—like the slot time, interframe spacing etc.—required by the network. It performs the functions of framing, deferring, backing-off, collision detection which are necessary in a CSMA/CD network. It also does Manchester encoding of data to be transmitted and clock separation—or decoding—of the Manchester encoded data that is received. These signals before going to the unshielded twist pair wire, may undergo pulse shaping (optional) pulse shaping basically slows down the fall/rise times of the signal. The purpose of that is to diminish the effects of cross-talk and radiation on adjacent pairs sharing the same bundle (digital voice, T1 trunks, etc). The shaped signal is sent on to the twisted pair wire through a pulse transformer for DC isolation. The signals on the wire are thus differential, DC isolated from the node and almost sinusoidal (due to shaping and the capacitance of the wire).

NOTE:
Work done by the IEEE 802.3 committee has shown that no slew rate control on the drivers is required. Shaping by the transformer and the cable is sufficient to avoid excessive EMI radiation and crosstalk.

The squelch circuit prevents idle line noise from affecting the receiver circuits in the LAN controller. The squelch circuit has a 600 mv threshold for that purpose. Also as part of the squelch circuitry an envelope detector is implemented. Its purpose is to generate an envelope of the transitions of the RXD line. Its output serve as a carrier sense signal. The differential signal from the HUB is received using a zero-crossing RS-422 receiver. Output of the receiver, qualified by the squelch circuit, is fed to the RxD pin of the LAN controller. The RxD signal provides three kinds of information:
1) Normal received data, when receiving the frame.
2) Collision information in the form of the collision presence signal from the HUB.
3) Carrier sense information, indicating the beginning and the end of frame. This is useful during transmit and receive operations.

2.2.2 StarLAN HUB

HUB is the point of concentration in StarLAN. All the nodes transmit to the HUB and receive from the HUB. Figure 7 shows an abstract representation of the HUB. It has an upstream and a downstream signal processing unit. The upstream unit has N signal inputs and 1 signal output. And the downstream unit has 1 input and N output signals. The inputs to the upstream unit come from the nodes or from the intermediate HUBs (IHUBs) and its output goes to a higher level HUB. The downstream unit is connected the other way around; input from an upper level HUB and the outputs to nodes or lower level IHUBs. Physically each input and output consist of one twisted pair wire carrying a differential signal. The downstream unit essentially just re-times the signal received at the input, and sends it to all its outputs. The functions performed by the upstream unit are:
1. Collision detection
2. Collision Presence signal generation
3. Signal Retiming
4. Jabber Function
5. Start of Idle protection timer

![Diagram of 82588 Based StarLAN Node](image-url)
The collision detection in the HUB is done by sensing the activity on the inputs. If there is activity (or transitions) on more than one input, it is assumed that more than one node is transmitting. This is a collision. If a collision is detected, a special signal called the Collision Presence Signal is generated. This signal is generated and sent out as long as activity is sensed on any of the input lines. This signal is interpreted by every node as an occurrence of collision. If there is activity only on one input, that signal is re-timed—or cleaned up of any accumulated jitter—and sent out. Figure 8 shows the input to output relations of the HUB as a black box.

If a node transmits for too long the HUB exercises a Jabber function to disable the node from interfering with traffic from other nodes. There are two timers in the HUB associated with this function and their operation is described in section 6.

The last function implemented by the HUB is the start of Idle protection timer. During the end of reception, the HUB will see a long undershoot at its input port. This undershoot is a consequence of the transformer discharging accumulated charge during the 2 microseconds of high of the idle pattern. The HUB should implement a protection mechanism to avoid the undesirable effects of that undershoot.

Figure 9 shows a block diagram of the HUB. A switch position determines whether the HUB is an IHUB or a HHUB (Header HUB). If the HUB is an IHUB, the switch decouples the upstream and the downstream units. HHUB is the highest level HUB; it has no place to send its output signal, so it returns its output signal (through the switch) to the outputs of the downstream unit. There is one and only one HHUB in a StarLAN network and it is always at the base of the tree. The returned signal eventually reaches every node in the network through the intermediate nodes (if any). StarLAN specifications do not put any restrictions on the number of IHUBS at any level or on number of inputs to any HUB. The number of inputs per HUB are typically 6 to 12 and is dictated by the typical size of clusters in a given networking environment.
2.2.3 StarLAN CABLE

Unshielded telephone grade twisted pair wires are used to connect a node to a HUB or to connect two HUBs. This is one of the cheapest types of wire and an important factor in bringing down the cost of StarLAN.

Although the 24 gauge wire is used for long stretches, the actual connection between the node and the telephone jack in the wall is done using extension cable, just like connecting a telephone to a jack. For very short StarLAN configurations, where all the nodes and the HUB are in the same room, the extension cable with plugs at both ends may itself be sufficient for all the wiring. (Extension cables must be of the twisted pair kind, no flat cables are allowed).

The telephone twisted pair wire of 24 gauge has the following characteristics:

- **Attenuation**: 42.55 db/mile @ 1 MHz
- **DC Resistance**: 823.69 Ω/mile
- **Inductance**: 0.84 mH/mile
- **Capacitance**: 0.1 μF/mile
- **Impedance**: 92.6Ω, −4 degrees @ 1 MHz

Experiments have shown that the sharing of the telephone cable with other voice and data services does not cause any mutual harm due to cross-talk and radiation, provided every service meets the FCC limits.

Although it is outside the scope of the IEEE 802.3 1BASE5 standard, there is considerable interest in using fiber optics and coaxial cable for node to HUB or HUB to HUB links especially in noisy and factory environments. Both these types of cables are particularly suited for point-to-point connections. Even mixing of different types of cables is possible (this kind of environments are not precluded).

**NOTE:**
StarLAN IEEE 802.3 1BASE5 draft calls for a maximum attenuation of 6.5 dB between the transmitter and the corresponding receiver at all frequencies between 500 KHz to 1 MHz. Also the maximum allowed cable propagation delay is 4 microseconds.

2.3 Framing

Figure 10 shows the format of a 802.3 frame. The beginning of the frame is marked by the carrier going active and the end marked by carrier going inactive. The preamble has a 56 bit sequence of 101010 . . . ending in a 0. This is followed by 8 bits of start of frame delimiter (sfd) − 10101011. These bits are transmitted with the MSB (leftmost bit) transmitted first. Source and destination fields are 6 bytes long. The first byte is the least significant byte. These fields are transmitted with LSB first. The length field is 2 bytes long and gives the length of data in the Information field. The entire information field is a minimum of 46 bytes and a maximum of 1500 bytes. If the data content of the Informa-
The frames can be directed to a specific node (LSB of address must be 0), to a group of nodes (multicast or group—LSB of address must be 1) or all nodes (broadcast—all address bits must be 1).

2.4 Signal Propagation and Collision

Figure 11 will be used to illustrate three typical situations in a StarLAN with two IHUBs and one HHUB. Nodes A and B are connected to HUB1, nodes C and D to HUB2 and node E to HUB3.
Situation #1. A Transmitting

Situation #2. A & B Transmitting

Situation #3. A, B & C Transmitting

HUB1, HUB2 are IHUBs
HUB3 is the HHUB
Fa, Fb, Fc—Frames from nodes A, B & C
Fx—Collision Presence Signal

Figure 11. Signal Propagation and Collisions
2.4.1 Situation #1

Whenever node A transmits a frame Fa, it will reach HUB1. If node B is silent, there is no collision. HUB1 will send Fa to HUB3 after re-timing the signal. If nodes C, D and E are also silent, there is no collision at HUB2 or HUB3. Since HUB3 is the HHUB, it sends the frame Fa to HUB1, HUB2 and to node E after re-timing. HUB1 and HUB2 send the frame Fa to nodes A, B and C, D. Thus, Fa reaches all the nodes on the network including the originator node A. If the signal received by node A is a valid Manchester signal and not the Collision Presence Signal (CPS) for the entire duration of the slot time, then the node A assumes that it was a successful transmission.

2.4.2 Situation #2

If both nodes A and B were to transmit, HUB1 will detect it as a collision and will send signal Fx (the Collision Presence Signal) to the HUB3—Note that HUB1 does not send Fx to nodes A and B yet. HUB 3 receives a signal from HUB1 but nothing from node E or HUB2, thus it does not detect the situation as a collision and simply re-times the signal Fx and sends it to node E, HUB2 and HUB1. Fx ultimately reach all the nodes. Nodes A and B detect this signal as CPS and call it a collision.

2.4.3 Situation #3

In addition to nodes A and B, if node C were also to transmit, the situation at HUB1 will be the same as in situation #2. HUB2 will propagate Fc from C towards HUB3. HUB3 now sees two of its inputs active and hence generates its own Fx signal and sends it towards each node.

These situations should also illustrate the point made earlier in the chapter that, the StarLAN network, with nodes connected to multiple HUBs is, logically, equivalent to all the nodes connected to a single HUB (Yet there are some differences between stations connected at different HUB levels, those are due to different delays to the header hub HHUB).

2.5 StarLAN System and Network Parameters

Preamble length (incl. sfd) .................. 64 bits
Address length ................................ 6 bytes
FCS length CRC (Autodin II) .............. 32 bits
Maximum frame length ...................... 1518 bytes
Minimum frame length ...................... 64 bytes
Slot time .................................. 512 bit times
Interframe spacing .......................... 96 bit times
Minimum jam timing ......................... 32 bit times
Maximum number of collisions ............. 16
Backoff limit .......................... 10

Backoff method .................... Truncated binary exponential
Encoding ................................ Manchester
Clock tolerance .................... ± 0.01% (100 ppm)
Maximum jitter per segment .......... ± 62.5 ns

3.0 LAN CONTROLLER FOR StarLAN

One of the attractive features of StarLAN is the availability of the 82588, a VLSI LAN controller, designed to meet the needs of a StarLAN node. The main requirements of a StarLAN node controller are:

1. IEEE 802.3 compatible CSMA/CD controller.
2. Configurable to StarLAN network and system parameters.
3. Generation of all necessary clocks and timings.
4. Manchester data encoding and decoding.
6. Carrier Sensing.
7. Squelch or bad signal filtering.
8. Fast and easy interface to the processor.

82588 performs all these functions in silicon, providing a minimal hardware interface between the system processor and the StarLAN physical link. It also reduces the software needed to run the node, since a lot of functions, like deferring, back off, counting the number of collisions etc., are done in silicon.

3.1 IEEE 802.3 Compatibility

The CSMA/CD control unit on the 82588 performs the functions of deferring, maintaining the Interframe Space (IFS) timing, reacting to collision by generating a jam pattern, calculating the back-off time based on the number of collisions and a random number, decoding the address of the incoming frame, discarding a frame that is too short, etc. All these are performed by the 82588 in accordance to the IEEE 802.3 standards. For inter-operability of different nodes on the StarLAN network it is very important to have the controllers strictly adhere to the same standards.

3.2 Configurability of the 82588

Almost all the networking parameters are programmable over a wide range. This means that the StarLAN parameters form a subset of the total potential of the 82588. This is a major advantage for networks whose standards are being defined and are in a flux. It is also an advantage when carrying over the experience gained with the component in one network to other applications, with differing parameters (leveraging the design).

The 82588 is initialized or configured to its working environment by the CONFIGURE command. After the execution of this command, the 82588 knows its system and network parameters. A configure block in
memory is loaded into the 82588 by DMA. This block contains all the parameters to be programmed as shown in Figure 12. Following is a partial list of the parameters with the programmable range and the StarLAN value:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Range</th>
<th>StarLAN Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Preamble length</td>
<td>2, 4, 8, 16 bytes</td>
<td>8</td>
</tr>
<tr>
<td>Address length</td>
<td>0 to 6 bytes</td>
<td>6</td>
</tr>
<tr>
<td>CRC type</td>
<td>16, 32 bit</td>
<td>32</td>
</tr>
<tr>
<td>Minimum frame length</td>
<td>6 to 255 bytes</td>
<td>64</td>
</tr>
<tr>
<td>Interframe spacing</td>
<td>12 to 255 bit times</td>
<td>96</td>
</tr>
<tr>
<td>Slot time</td>
<td>1 to 2047 bit times</td>
<td>512</td>
</tr>
<tr>
<td>Number of retries</td>
<td>0 to 15</td>
<td>15</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Range</th>
<th>StarLAN Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data encoding</td>
<td>NRZI, Man.,</td>
<td>Manch.</td>
</tr>
</tbody>
</table>

Besides these, there are many other options available, which may or may not apply to StarLAN:

- Data sampling rate of 8 or 16
- Operating in Promiscuous mode
- Reception of Broadcast frames
- Internal loopback operation
- External loopback operation
- Transmit without CRC
- HDLC Framing

![Figure 12. Configuration Block](231422-15)
3.3 Clocks and Timers

The 82588 requires two clocks; one for the operation of the system interface and another for the serial side. Both clocks are totally asynchronous to each other. This permits transmitting and receiving frames at data rates that are virtually independent of the speed at which the system interface operates.

The serial clock can be generated on chip using just an external crystal of a value 8 or 16 times the desired bit rate. An external clock may also be used.

The 82588 has a set of timers to maintain various timings necessary to run the CSMA/CD control unit. These are timings for the Slot time, Interframe spacing time, Back off time, Number of collisions, Minimum frame length, etc. These timers are started and stopped automatically by the 82588.

3.4 Manchester Data Encoding and Decoding

In StarLAN the data transmitted by the node must be encoded in Manchester format. The node should also be able to decode Manchester encoded data when receiving a frame—a process also known as clock recovery. The 82588 does the encoding and decoding of data bits on chip for data rates up to 2 Mb/s.

Besides Manchester, the 82588 can also do encoding and decoding in NRZI and Differential Manchester formats. Figure 13 shows samples of encoding in

<table>
<thead>
<tr>
<th>Encoding Method</th>
<th>Mid Bit Cell Transitions</th>
<th>Bit Cell Boundary Transitions</th>
</tr>
</thead>
<tbody>
<tr>
<td>NRZ</td>
<td>Do not exist.</td>
<td>Identical to original data.</td>
</tr>
<tr>
<td>NRZI</td>
<td>Do not exist.</td>
<td>Exist only if original data bit equals 0. Dependent on present encoded signal level: to 0 if 1, to 1 if 0.</td>
</tr>
<tr>
<td>Manchester</td>
<td>Exist for every bit of the original data: from 0 to 1 for 1, from 1 to 0 for 0.</td>
<td>Exist for consequent equal bits of original data: from 1 to 0 for 1, from 0 to 1 for 0.</td>
</tr>
<tr>
<td>Differential Manchester</td>
<td>Exist for every bit of the original data. Dependent on present encoded signal level: to 0 if 1, to 1 if 0.</td>
<td>Exist only if original data bit equals 0. Dependent on present Encoded signal level: to 0 if 1, to 1 if 0.</td>
</tr>
</tbody>
</table>

Figure 13. 82588 Data Encoding Rules
these three formats. The main advantage of NRZI over the other two is that NRZI requires half the channel bandwidth, for any given data rate. On the other hand, since the NRZI signal does not have as many transitions as the other two, clock recovery from it is more difficult. The main advantage of Differential Manchester over straight Manchester is that for any given data rate.

On the other hand, since the NRZI signal does not have as many transitions as the other two, clock recovery from it is more difficult. The main advantage of Differential Manchester over straight Manchester is that for any given data rate.

The main advantage of Differential Manchester over straight Manchester is that for a signal that is differentially driven (as in RS 422), crossing of the two wires carrying the data does not change the data received at the receiver. In other words, NRZI and Differential Manchester encoding methods are polarity insensitive (Even though NRZI, Differential Manchester are polarity insensitive, the 82588 expects a high level in the RXD line to detect carrier inactive at the end of frames).

### 3.5 Detection of the Collision Presence Signal

In a StarLAN network, HUB informs the nodes that a collision has occurred by sending the Collision Presence Signal (CPS) to the nodes. The CPS signal is a special signal which contains violations in Manchester encoding. Figure 14 shows the CPS signal. It has a 5 ms period, looking very much like a valid Manchester signal except for missing transitions (or violations) at periodic intervals. When the 82588 decodes this signal, it fails to see mid-cell transitions repeatedly at intervals of 2.5 bit times and hence calls it a code violation. The edges of CPS are marked for illustration as a, b, c, d, . . . . Let us see how the 82588 interprets the signal if it starts calling the edge 'a' as the mid-cell transition for '1'. Then edge at 'b' is '0'. Now the 82588 expects to see an edge at '*' but since there is none, it is a Manchester code violation. The edge that eventually does occur at 'd' is then used to re-synchronize and, since it is a falling edge, it is taken as a mid-cell transition for '0'. The edge at 'e' is for a '1' and then again there is no edge at '*'. This goes on, with the 82588 flagging code violation and re-synchronizing again every 2.5 bit times. When a transmitting node sees this CPS signal being returned by the HUB (instead of a valid Manchester signal it transmitted), it assumes that a collision occurred. The 82588 has two built-in mechanisms to detect collisions. These mechanisms are very general and can be used for a very broad class of applications to detect collisions in a CSMA/CD network. Using these mechanisms, the 82588 can detect collisions (two or more nodes transmitting simultaneously) by just receiving the collided signal during transmission, even if there was no HUB generating the CPS signal.

![Figure 14. 82588 Decoding the Collision Presence Signal](image_url)
3.5.1 COLLISION DETECTION BY CODE VIOLATION

If during transmission, the 82588 sees a violation in the encoding (Manchester, NRZI or Differential Manchester) used, then it calls it a collision by aborting the transmission and transmitting a 32 bit jam pattern. The algorithm used to detect collisions, and to do the data decoding, is based on finding the number of sampling clocks between an edge to the next one. Suppose an edge occurred at time 0, the sampling instant of the next edge determines whether it was a collision (C), a long pulse (L)—with a nominal width of 1 bit time—or a short pulse (S)—nominal width of half a bit time. The following two charts show the decoding and collision detection algorithm for sampling rates of 8 and 16 when using Manchester encoding. The numbers at the bottom of the line indicate sampling instances after the occurrence of the last edge (at 0). The alphabets on the top show what would be inferred by the 82588 if the next edge were to be there.

Sampling rate = 8 (clock is 8x bit rate)

<table>
<thead>
<tr>
<th>C</th>
<th>C</th>
<th>S</th>
<th>S</th>
<th>S</th>
<th>S</th>
<th>L</th>
<th>L</th>
<th>L</th>
<th>L</th>
<th>L</th>
<th>L</th>
<th>C</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>10</td>
<td>11</td>
<td>12</td>
<td>13</td>
</tr>
</tbody>
</table>

Collision also if:
- RxD stays low for 13 samples or more
- A mid cell transition is missing

Sampling rate = 16 (clock is 16x bit rate)

| C | C | C | C | C | C | C | S | S | S | S | S | S | S | S | S | C | C | C | C | C | C | C | C |
| 0 | 2 | 4 | 6 | 8 | 10 | 12 | 14 | 16 | 18 | 20 | 22 | 24 | 26 |

A single instance of code violation can qualify as collision. The 82588 has a parameter called collision detection filter (CDT Filter) that can be configured from 0 to 7. This parameter determines for how many bit times the violation must remain active to be flagged as a collision. For StarLAN CDT Filter must be configured to 0—that is disabled.

3.5.2 COLLISION DETECTION BY SIGNATURE (OR BIT) COMPARISON

This method of collision detection compares a signature of the transmitted data with that of the data received on the RxD pin while transmitting. Figure 15 shows a block diagram of the logic. As the frame is transmitted it flows through the CRC generation logic. A timer, called the Tx slot timer, is started at the same time that the CRC generation starts. When the count in the timer reaches the slot time value, the current value of the CRC generator is latched in as the transmit signature. As the frame is returned back (through the HUB) it flows through the CRC checker. Another timer—Rx slot timer—is started at the same time as the CRC checker starts checking. When this timer reaches the slot time value, the current value of the CRC checker is latched in as the receive signature. If the received signature matches the transmitted one, then it is assumed that there was no collision. Whereas, if the signatures do not match, a collision is assumed to have occurred.

![Figure 15. Collision Detection by Signature Comparison](image-url)
Note that, even if the collision were to occur in the first few bits of the frame, a slot time must elapse before it is detected. In the code violation method, collision is detected within a few bit times. However, since the signature method compares the signatures, which are characteristic of the frame being transmitted, it is more robust. The code violation method can be fooled by returning a signal to the 82588 which is not the same as the transmitted signal but is a valid Manchester signal—like a 1 MHz signal. Both methods can be used simultaneously giving a combination of speed and robustness.

NOTE:
In order to reliably detect a collision using the collision by bit comparison mode, the transmitter must still be transmitting up to the point where the receiver has seen enough bits to complete its signature. Otherwise, the transmitter may be done before the RX signature is completed resulting in an undetected collision. A sufficient condition to avoid this situation is to transmit frames with a minimum length of 1.5 * slot-time (see Figure 16).

3.5.3 ADDITIONAL COLLISION DETECTION MECHANISM

In addition to the collision detection mechanisms described in the preceding sections, the 82588 also flags collision when after starting a transmission any of the following conditions become valid:

a) Half a slot time elapses and the carrier sense of 82588 is not active.

b) Half a slot time + 16 bit times elapse and the opening flag (sfd) is not detected.

c) Carrier sense goes inactive after an opening flag is received with transmitter still active.

These mechanisms add a further robustness to the collision detection mechanism of the 82588. It is also possible to OR an externally generated collision detect signal to the internally generated condition by bit comparison (see Figure 17).

3.6 Carrier Sensing

A StarLAN network is considered to be busy if there are transitions on the cable. Carrier is supposed to be active if there are transitions. Every node controller needs to know when the carrier is active and when not. This is done by the carrier sensing circuitry. On the 82588 this circuit is on chip. It looks at the RxD (receive data) pin and if there are transitions, it turns on an internal carrier sense signal. It turns off the carrier sense signal if RxD remains in idle (high) state for 13/8 bit times. This carrier sense information is used to mark the start of the interframe space time and the back off time. The 82588 also defers transmission when the carrier sense is active.

When operating in the NRZI encoded mode, carrier sense is turned off if RxD pin is in the idle state for 8 bit times or more (see Figure 18).

![Figure 16. Limitation of CDBBC Mechanism](image-url)
3.7 Squelching the Input

Squelch circuit is used to filter idle noise on the receiver input. Basically two types of squelch may be used: Voltage and time. Voltage squelch is done to filter out signals whose strength is below a defined voltage threshold (0.6 volts for StarLAN). It prevents idle line noise from disturbing the receive circuits on the controller. The voltage squelch circuit is placed right after the receiving pulse transformer. It enables the input to the RxD pin of the 82588 only when the signal strength is above the threshold.

If the signal received has the proper level but not the proper timing, it should not bother the receiver. This is accomplished by the time squelch circuit on the 82588. Time squelching is essential to weed out spikes, glitches and bad signal especially at the beginning of a frame. The 82588 does not turn on its carrier sense (or receive enable) signal until it receives three consecutive edges, each separated by time periods greater than the fast time clock high time but less than 13/8 bit-times as shown in Figure 18.
The carrier sense activation can be programmed for a further delay by up to 7 bit times by a configuration parameter called carrier sense filter.

### 3.8 System Bus Interface

The 82588 has a conventional bus interface making it very easy to interface to any processor bus. Figure 19 shows that it has an 8 bit data bus, read, write, chip select, interrupt and reset pins going to the processor bus. It also needs an external DMA controller for data transfer. A system clock of up to 8 MHz is needed. The read and write access times of the 82588 are very short—95 ns—as shown by Figure 20. This further facilitates interfacing the controller to almost any processor.

---

**Figure 19. Chip Interface**

**Figure 20. Access Times**
The 82588 has over 50 bytes of registers, and most are accessed only indirectly. Figure 21 shows the register access mechanism of the 82588. It has one I/O port and 2 DMA channel ports. These are the windows into the 82588 for the CPU and the DMA controller. An external CPU can write into the Command register and read from the Status registers using I/O instructions and asserting chip select and write or read lines. Although there is just one I/O port and 4 status registers, they can be read out in a round robin fashion through the same port as shown in Figure 22. Other registers like the Configuration, Individual Address registers can be accessed only through DMA. All the internal registers can be dumped into memory by DMA using the Dump command. The execution of some of the commands is described in section 4. See the 82588 Reference Manual for details on these commands.

3.9 Debug and Diagnostic Aids

Besides the standard functions that can be used directly for StarLAN, the 82588 offers many debug and diag-

![Figure 21. Register Access](image-url)
4 Status registers are accessed through one read port

The pointer can be changed using a command or can be automatically incremented.

```plaintext
READ_STATUS_588: PROCEDURE;
  /* COMMAND 15 */
  OUTPUT (CS_588) = 15;
  /* RELEASE POINTER, INITIAL = 00 */
  STATUS_588(0) = INPUT (CS_588);
  /* REFRESH STATUS REGISTER IMAGE */
  STATUS_588(1) = INPUT (CS_588);
  /* IN MEMORY. */
  STATUS_588(2) = INPUT (CS_588);
  STATUS_588(3) = INPUT (CS_588);
RETURN
END READ_STATUS_588;
```

Figure 22. Reading the Status Register

Figure 23. 82588 Jitter Performance

3.10 Jitter Performance

When the 82588 receives a frame from the HUB, the signal has jitter. Jitter is the shifting of the edges of the signal from their nominal position due to the transmission over a length of cable. Many factors like, intersymbol interference (pulses of different widths have different delays through the transmission media), rise and fall times of drivers and receivers, cross talk etc., contribute to the jitter. StarLAN specifies a maximum jitter of ±62.5 ns whenever the signal goes from a NODE/HUB or HUB/HUB. Figure 23 shows that the jitter tolerance of the 82588 is exactly the required ±62.5 ns at 1 Mbs for both 8X, 16X Manchester encoded data.

<table>
<thead>
<tr>
<th>Code Violations</th>
<th>x8</th>
<th>x16</th>
</tr>
</thead>
<tbody>
<tr>
<td>NRZI</td>
<td>±1/16 BT</td>
<td>±3/32 BT</td>
</tr>
<tr>
<td>(Code Violations Enabled)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>NRZI</td>
<td>±3/16 BT</td>
<td>±3/16 BT</td>
</tr>
<tr>
<td>(Code Violations Disabled)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

3.10 Jitter Performance

When the 82588 receives a frame from the HUB, the signal has jitter. Jitter is the shifting of the edges of the signal from their nominal position due to the transmission over a length of cable. Many factors like, intersymbol interference (pulses of different widths have different delays through the transmission media), rise and fall times of drivers and receivers, cross talk etc., contribute to the jitter. StarLAN specifies a maximum jitter of ±62.5 ns whenever the signal goes from a NODE/HUB or HUB/HUB. Figure 23 shows that the jitter tolerance of the 82588 is exactly the required ±62.5 ns at 1 Mbs for both 8X, 16X Manchester encoded data.

4.0 THE 82588

This chapter describes the basic 82588 operations. Please refer to the 82588 reference manual in Intel Microcommunications Handbook for a detailed description. Basic operations like transmitting a frame, receiving a frame, configuring the 82588 and dumping the register contents are discussed here to give a feel for how the 82588 works.
4.1 Transmit and Retransmit Operations

To transmit a frame, the CPU prepares a block in the memory called the transmit data block. As shown in Figure 24, this block starts with a byte count field, indicating how long the rest of the block is. The destination address field contains the node address of the destination. The rest of the block contains the information or the data field of the frame. The CPU also programs the DMA controller with the start address of the transmit data block. The DMA byte count must be equal to or greater than the block length. The 82588 is then issued a TRANSMIT command—an OUT instruction to the command port of the 82588. The 82588 starts generating DMA requests to read in the transmit data block by DMA. It also determines whether and how long it must defer on the link and after that, it starts transmitting the preamble. The 82588 constructs the frame on the fly. It takes the destination address from the memory, source address from its own individual address memory (previously programmed), data field from the memory and the CRC, is generated on chip, at the end of the frame.

1. Prepare Transmit Data—Block in Memory
2. Program DMA Controller
3. Issue Transmit Command on the Desired Channel

4. Interrupt is received on completion of command or if the command was aborted or there was a collision. The status bytes 1 and 2 indicate the result of the operation.

4.2 Configuring the 82588

To initialize the 82588 and program its network and system parameters, a configure operation is performed. It is very similar to the transmit operation. Instead of a transmit data block as in transmit command, a configure data block—shown in Figure 12—is prepared by the CPU in the memory. The first two bytes of the block specify the length of the rest of the block, which specify the network and system parameters for the 82588. The DMA controller is then programmed by the CPU to the beginning of this block and a CONFIGURE command is issued to the 82588. The 82588 reads in the parameters by DMA and loads the parameters in the on-chip registers.

Similarly, for programming the INDIVIDUAL ADDRESS and MULTICAST ADDRESSes, the DMA controller is used to load the 82588 registers.

4.3 Frame Reception

Before enabling the 82588 for reception the CPU must make a buffer available for the frame to be received. The CPU must program the DMA controller with the starting address of the buffer and then issue the RX ENABLE command to the 82588. When a frame arrives at the RxD pin of the 82588, it starts being received. Only if the address in the destination address matches either the Individual address, Multicast address or if it is a broadcast address, is the frame deposited into memory by the 82588 using DMA. The format of storage in the memory is shown in Figure 25. At the end, a two byte field is attached which shows the status of the received frame. If CRC, alignment or overrun errors are encountered, they are reported. An inter-
1. Prepare a Buffer for Reception
2. Program DMA Controller
3. Issue Receiver Enable Command

When a frame is received, it is deposited in the memory. Receive status bytes (2) are appended to the frame in the memory, byte count written in the status registers 1, 2, and an interrupt is generated.

![Figure 25. Receive Operation (Single Buffer)](image)

rupt from 82588 occurs when all the bytes have been transferred to the memory. This informs the CPU that a new frame has been received.

If the received frame has errors, the CPU must recover (or re-use) the buffer. Note that the entire frame is deposited into one buffer. The 82588 when NOT configured for the external loopback mode, will detect collisions (code violations) during receptions. If a collision is detected, the reception is aborted and status updated. CPU is then informed by an interrupt (if the collided frame fragment is shorter than the address length, no reception will be started), and no interrupt will happen.

4.3.1 Multiple Buffer Frame Reception

It is also possible to receive a frame into a number of fixed size buffers. This is particularly economical if the received frames vary widely in size. If the single buffer scheme were used as described above, the buffer required would have to be bigger than the longest expected frame and would be very wasteful for very short (typically acknowledge or control) frames. The multiple buffer reception is illustrated in Figure 26. It uses two DMA channels for reception.

![Figure 26. Multiple Buffer Reception](image)
As in single buffer reception, the one channel, say channel 0, of the DMA controller is programmed to the start of buffer 1, and the 82588 is enabled for reception with the chaining bit set. As soon as the first byte is read out of the 82588 by the DMA controller and written into the first location of buffer 1, the 82588 generates an interrupt, saying that it is filling up its last available buffer and one more buffer must be allocated. The filling up of the buffer 1 continues. The CPU responds to the interrupt by programming the other DMA channel—channel 1—with the start address of the second buffer and issuing an ASSIGN ALTERNATE buffer command with an INTACK (interrupt acknowledge). This informs the 82588 that one more buffer is available on the other channel. When buffer 1 is filled up (the 82588 knows the size of buffers from the configuration command), the 82588 starts generating the DMA requests on the other channel. This automatically starts filling up buffer 2. As soon as the first byte is written into buffer 2, the 82588 interrupts the CPU again asking for one more buffer. The CPU programs the channel 0 of the DMA controller with the start address of buffer 3, issues an ASSIGN ALTERNATE buffer command with INTACK. This keeps the buffer 3 ready for the 82588. This switching of channels continues until the entire frame is received generating an end of frame interrupt. The CPU maintains the list of pointers to the buffers used.

Since a new buffer is allocated at the time of filling up of the last buffer, the 82588 automatically switches to the new buffer to receive the next frame as soon as the last frame is completely received. It can start receiving the new frame almost immediately, even before the end of frame interrupt is serviced and acknowledged by the CPU. If a new frame comes in, and the previous frame interrupt is not yet acknowledged, another interrupt needed for new buffer allocation is buffered (and not lost). As soon as the first one is acknowledged, the interrupt line goes active again for the buffered one.

If by the time a buffer fills up no new buffer is available, the 82588 keeps on receiving. An overrun will occur and will be reported in the received frame status. However, ample time is available for the allocation of a new buffer. It is roughly equal to the time to fill up a buffer. For 128 byte buffers it is $128 \times 8 = 1024$ ms or approximately 1 millisecond. You get 1 ms to assign a new buffer after getting the interrupt for it. Hence the process of multiple buffer reception is not time critical for the system performance.

This method of reception is particularly useful to guarantee the reception of back-to-back frames separated by IFS time. This is because a new buffer is always available for the new frame after the current frame is received.

Although both the DMA channels get used up in receiving, only one channel is kept ready for reception and the other one can be used for other commands until the reception starts. If an execution command like transmit or dump command is being executed on a channel which must be allocated for reception, the command gets automatically aborted when the ASSIGN ALTERNATE BUFFER command is issued to the channel used for the execution command. The interrupt for command abortion occurs after the end of frame interrupt.
4.4 Memory Dump of Registers
All the 82588 internal registers can be dumped in the memory by the DUMP command. A DMA channel is used to transfer the register contents to the memory. It is very similar to reception of a frame; instead of data from the serial link, the data from the registers gets written into the memory. This provides a software debugging and diagnostic tool.

4.5 Other Operations
Other 82588 operations like DIAGNOSE, TDR, ABORT, etc. do not require any parameter or data transfer. They are executed by writing a command to the 82588 command register and knowing the results (if any) through the status registers.

5.0 StarLAN NODE FOR IBM PC
This chapter deals with the hardware—the StarLAN board—to interface the IBM PC to a StarLAN Network. This is a slave board which takes up one slot on the I/O channel of the IBM PC. Figure 27 shows an abstract block diagram of the board. It requires the IBM PC resources of the CPU, memory, DMA and interrupt controller on the system board to run it. Such a board has two interfaces. The IBM PC I/O Channel on the system or the parallel side and the telephone grade twisted pair wire on the serial side. Figures 28, 29 show the circuit diagram of the board.

![Figure 27. 82588 Based StarLAN Node](image-url)
5.1 Interfacing to the IBM PC I/O Channel

IBM PC has 8 slots on the system board to allow expansion of the basic system. All of them are electrically identical and the I/O channel is the bus that links them all to the 8088 system bus. The I/O channel contains an 8 bit bidirectional data bus, 20 address lines, 6 levels of interrupt, 3 channels of DMA control lines and other control lines to do I/O and memory read/write operations. Figure 30 shows the signals and the pin assignment for the I/O Channel.

<table>
<thead>
<tr>
<th>Hex Range</th>
<th>Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>000-00F</td>
<td>DMA Chip 8237A-5</td>
</tr>
<tr>
<td>020-021</td>
<td>Interrupt 8259A</td>
</tr>
<tr>
<td>040-043</td>
<td>Timer 8253-5</td>
</tr>
<tr>
<td>060-063</td>
<td>PPI 8255A-5</td>
</tr>
<tr>
<td>080-083</td>
<td>DMA Page Registers</td>
</tr>
<tr>
<td>0AX*</td>
<td>NMI Mask Register</td>
</tr>
<tr>
<td>0CX</td>
<td>Reserved</td>
</tr>
<tr>
<td>0EX</td>
<td>Reserved</td>
</tr>
<tr>
<td>200-20F</td>
<td>Game Control</td>
</tr>
<tr>
<td>210-217</td>
<td>Expansion Unit</td>
</tr>
<tr>
<td>220-224F</td>
<td>Reserved</td>
</tr>
<tr>
<td>278-27F</td>
<td>Reserved</td>
</tr>
<tr>
<td>2F0-2F7</td>
<td>Reserved</td>
</tr>
<tr>
<td>2F8-2FF</td>
<td>Asynchronous Communications (Secondary)</td>
</tr>
<tr>
<td>300-31F</td>
<td>Prototype Card</td>
</tr>
<tr>
<td>320-32F</td>
<td>Fixed Disk</td>
</tr>
<tr>
<td>378-37F</td>
<td>Printer</td>
</tr>
<tr>
<td>380-38C**</td>
<td>SDLC Communications</td>
</tr>
<tr>
<td>380-389**</td>
<td>Binary Synchronous Communications (Secondary)</td>
</tr>
<tr>
<td>3A0-3A9</td>
<td>Binary Synchronous Communications (Primary)</td>
</tr>
<tr>
<td>3B0-3BF</td>
<td>IBM Monochrome Display/Printer</td>
</tr>
<tr>
<td>3C0-3CF</td>
<td>Reserved</td>
</tr>
<tr>
<td>3D0-3DF</td>
<td>Color/Graphics</td>
</tr>
<tr>
<td>3E0-3E7</td>
<td>Reserved</td>
</tr>
<tr>
<td>3F0-3F7</td>
<td>Diskette</td>
</tr>
<tr>
<td>3F8-3FF</td>
<td>Asynchronous Communications (Primary)</td>
</tr>
</tbody>
</table>

* At power-on time, the Non Mask Interrupt into the 8088 is masked off. This mask bit can be set and reset through system software as follows:
  Set mask: Write hex 80 to I/O Address hex A0 (enable NMI)
  Clear mask: Write hex 00 to I/O Address hex A0 (disable NMI)

** SDLC Communications and Secondary Binary Synchronous Communications cannot be used together because their hex addresses overlap.

Figure 30. I/O Channel Diagram

5.1.1 REGISTER ACCESS AND DATA BUS INTERFACE

The CPU accesses the StarLAN adapter card through 2 I/O address windows. Address 300H is used to access to 82588 for commands and status, address 301H accesses an on board control port that enables the various interrupt and DMA lines. Even though only two addresses are needed, the card uses all the 16 addresses spaces from 300H to 30FH. This was done to keep simplicity and minimum component count. Registers address decoding is done using a PAL (16L8) and an external NAND gate (U8).
The signal CS__ decodes address 300H, it is only active when AEN is inactive meaning CPU and not DMA cycles. LDPORT__ has exactly the same logic for address 301H, but it is only active during I/O write cycles. The I/O port sitting on address 301H is write only. The data BUS lines D0 to D7 are buffered from the 82588 to the PC bus using an 74LS245 transceiver chip.

The Bus transceiver is enabled if: A DMA access is taking place, or I/O ports 300H to 30FH are being accessed.

5.1.2 Control Port

As mentioned the StarLAN adapter port has a 4-bit write only control port. The purpose of this port is to selectively enable the DMA and INTERRUPT request lines. Also it can completely disable the transmitter.

Data Bus Interface

<table>
<thead>
<tr>
<th>ENDRQ1</th>
<th>ENDRQ3</th>
<th>ENINTER</th>
<th>TXEN</th>
</tr>
</thead>
<tbody>
<tr>
<td>“1”</td>
<td>“1”</td>
<td>“1”</td>
<td>“1”</td>
</tr>
</tbody>
</table>

ENDRQ1, ENDRQ2 : “1” Enable DMA requests.
ENINTER : “1” Enable INTERRUPT request.
TXEN : “1” Enable the transmitter.

On power up all bits default to “0”.

Register Access

Format of Following Equations Will Be According To The Following Specifications:

\[
\begin{align*}
&! \text{ INVERT} \\
&{_{-}} \text{ SIGNAL ACTIVE LOW} \\
& & \& \text{ LOGIC AND} \\
& & \# \text{ LOGIC OR} \\
&A9\text{NANDA8} = {! (A9 \& A8) } \\
&CS_ = {! (AEN \& A9\text{NANDA8} \& A7 \& A6 \& A5 \& A4 \& A0) } \\
&LDPORT_ = {! (AEN \& A9\text{NANDA8} \& A7 \& A6 \& A5 \& A4 \& A0 \& IOWR_ ) } \\
&BUSEN_ = \text{ DACK1}_ & \text{ DACK2} _ & ( {! (AEN \& A9\text{NANDA8} \& A7 \& A6 \& A5 \& A4) } )
\end{align*}
\]
5.1.3 CLOCK GENERATION

The 82588 requires two clocks for operation. The system clock and the serial clock. The serial clock can be generated on chip by putting a crystal across X1 and X2 pins. Alternatively, an externally generated clock can be fed in at pin X1 (with X2 left open). In both cases, the frequency must be either 8 or 16 times (sampling factor) the desired bit rate. For StarLAN, 8 or 16 MHz are the correct values to generate 1 Mb/s data rate. A configuration parameter is used to tell the 82588 what the sampling factor is. An externally supplied clock must have MOS levels (0.6V – 3.9V). Specifications for the crystal and the circuit are shown in Figure 32.

The system clock has to be supplied externally. It can be up to 8 MHz. This clock runs the parallel side of the 82588. Its frequency does not have any impact on the read and write access times but on the rate at which data can be transferred to and from the 82588 (Maximum DMA data rate is one byte every two system clocks). This clock doesn’t require MOS levels.

The I/O channel of the IBM PC supplies a 4.77 MHz signal of 33% duty cycle. This signal could be used as a system clock. It was decided, however, to generate a separate clock on the StarLAN board to be independent of the I/O channel clock so that this board can also be used in other IBM PCs and also in some other compatibles. The 8 MHz system clock is generated using a DIP OSCILLATOR which have the required 50 ppm tolerance to meet StarLAN. This clock is converted to MOS levels by 74HCT00 and fed into both the system and serial clock inputs.

5.1.4 DMA INTERFACE

The 82588 requires either one or two DMA channels for full operation. In this application, one channel is dedicated for reception and the other is used for transmissions and the other commands. Use of only one DMA channel is possible but may require more complex software, also some RX frames may be lost during switches of the DMA channel from the receiver to the transmitter (Those frames will be recovered by higher layers of the protocol). Also using only one DMA channel will limit the 82588 loopback functionality. So the recommendation is to operate with two DMA channels if available. Appendix C describes a method of operating with only one DMA channel without losing RX frames.

The IBM PC system board has one 8237A DMA controller. Channel 0 is used for doing the refresh of DRAMs. Channels 1, 2 and 3 are available for add-on boards on the I/O Channel. The floppy disk controller board uses the DMA channel 2 leaving exactly two channels (1 and 3) for the 82588. The situation is worse if the IBM PC/XT is used, since it uses channel 3 for the Winchester hard disk leaving just the channel 1 for

<table>
<thead>
<tr>
<th>Series Resonance</th>
</tr>
</thead>
<tbody>
<tr>
<td>- Frequency Will Drift by About 400 PPM from Nominal</td>
</tr>
<tr>
<td>- No Capacitors Needed</td>
</tr>
<tr>
<td>- Doesn’t Meet StarLAN Requirements</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Meeting StarLAN 100 PPM Requirements</th>
</tr>
</thead>
<tbody>
<tr>
<td>- Use Parallel Resonance Crystal</td>
</tr>
<tr>
<td>- Recommended For Precise Frequencies</td>
</tr>
<tr>
<td>- 82588 X-TAL Oscillator Stability ± 35 PPM (0–70°C)</td>
</tr>
</tbody>
</table>

Crystal: Load Capacitance = 20 pF
Shunt Capacitance = 7 pF Maximum
Series Resistance = 30Ω Maximum
Frequency Tolerance = 50 PPM (0–70°C)

C1, C2 → 27 pF or 39 pF, 5%

Figure 32. Crystal Specifications
the 82588. On the other hand, the IBM PC/AT has 5 free DMA channels. We will assume that 8237A DMA channels 1 and 3 are available for the 82588 as in the case of the IBM PC.

Since the channel 0 of 8237A is used to do refresh of DRAMs all the channels should be operated in single byte transfer mode. In this mode, after every transfer for any channel the bus is granted to the current highest priority channel. In this way, no channel can hog the bus bandwidth and, more important, the refresh of DRAMs is assured every 15 microseconds since the refresh channel (number 0) has the highest priority. This mode of operation is very slow since the HOLD is dropped by the 8237A and then asserted again after every transfer. Demand mode of operation is a lot more suitable to 82588 but it cannot be used because of the refresh requirements.

Whenever the 82588 interfaces to the 8237A in the single transfer mode, there is a potential 8237A lock-up problem. The 82588 may deactivate its DMA request line (DREQ) before receiving an acknowledge from the DMA controller. This situation may happen during command abortions, or aborted receptions. The 8237A under those circumstances may lock-up. In order to solve this potential problem, an external logic must be used to insure that DREQ to the DMA controller is never deactivated before the acknowledge is received. Figure 33 shows the logic to implement this function. This logic is implemented in the 16L8 PAL.

The 82588 DREQ lines are connected to the IBM/PC bus through tri-state buffers which are enabled by writing to I/O port 301H. This function enables the use of either one or two DMA channels and also the sharing of DMA channels with other adapter boards.

### 5.1.5 INTERRUPT CONTROLLER

The 82588 interrupts the CPU after the execution of a command or on reception of a frame. It uses the 8259A interrupt controller on the system board to interrupt the CPU. There are 6 interrupt request lines, IRQ2 to IRQ7, on the I/O channel. Figure 34 shows the assignment of the lines. In fact, none of the lines are completely free for use. To add any new peripheral which uses a system board interrupt, this interrupt needs to have the capability to share the specific line, by driving the line with a tri-state driver. The 82588 StarLAN adapter board can optionally drive interrupt lines IRQ3, IRQ4 or IRQ5 (An 74LS125 driver is used).

<table>
<thead>
<tr>
<th>Number</th>
<th>Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>NMI</td>
<td>Parity</td>
</tr>
<tr>
<td>0</td>
<td>Timer</td>
</tr>
<tr>
<td>1</td>
<td>Keyboard</td>
</tr>
<tr>
<td>2</td>
<td>Reserved</td>
</tr>
<tr>
<td>3</td>
<td>Asynchronous Communications (Secondary)</td>
</tr>
<tr>
<td></td>
<td>SDLC Communications</td>
</tr>
<tr>
<td></td>
<td>BSC (Secondary)</td>
</tr>
<tr>
<td>4</td>
<td>Asynchronous Communications (Primary)</td>
</tr>
<tr>
<td></td>
<td>SDLC Communications</td>
</tr>
<tr>
<td></td>
<td>BSC (Primary)</td>
</tr>
<tr>
<td>5</td>
<td>Fixed Disk</td>
</tr>
<tr>
<td>6</td>
<td>Diskette</td>
</tr>
<tr>
<td>7</td>
<td>Printer</td>
</tr>
</tbody>
</table>

**Figure 34. IBM PC Hardware Interrupt Listing**

*Figure 33. DMA Request Logic*
5.2 Serial Link Interface

A typical StarLAN adapter board is connected to the twisted pair wiring using an extension cable (typically up to 8 meters—25 ft.). See Figure 35. One end of the cable plugs into the telephone modular jack on the StarLAN board and the other end into a modular jack in the wall. The twisted pair wiring starts at the modular jack in the wall and goes to the wiring closet. In the wiring closet, another telephone extension cable is used to connect to a StarLAN HUB. The transmitted signal from the 82588 reach the on-board telephone jack through a RS-422 driver with pulse shaping and a pulse transformer. The received signals from the telephone jack to the 82588 come through a pulse transformer, squelch circuit and a receive enable circuit.

5.2.1 TRANSMIT PATH

The single ended transmit signal on the TxD pin is converted to a differential signal and the rise and fall times are increased to 150 to 200 ns before feeding it to the pulse transformer (this pulse shaping is not a requirement, but proves to give good results). Am26LS30 is a RS-422 driver which converts the TxD signal to a differential signal. It also has slew rate control pins to increase to rise and fall times. A large rise and fall time reduces the possibility of crosstalk, interference and radiation. By the other hand a slower edge rate increases the jitter. In the StarLAN adapter card, the first approach was used. The 26LS30 converts a square pulse to a trapezoidal one—see Figure 36. The filtering effect of the cable further adds to reduce the higher frequency components from the waveform so that on the cable the signal is almost sinusoidal. The pulse transformer is for DC isolation. The pulse transformers from Pulse Engineering—type PE 64382—was used in this design. This is a dual transformer package which introduces an additional rise and fall time of about 70–100 ns on the signal, helping the former discussed waveshaping.

5.2.2 IDLE PATTERN GENERATION

StarLAN requires transmitters to generate an IDLE pattern after the last transmitted data bit. The IDLE pattern is defined to be a constant high level for 2–3 microseconds. The purpose of this pattern is to insure that receivers will decode properly the last transmitted data bits before signal decay. Currently the 82588 needs one external component to generate the IDLE. The operation principle is to have an external shift register (74LS164) that will kind of act as an envelope detector of the TXD line. Whenever the TXD line goes low...
(first preamble bit), the output of the shift register (third cell) will immediately go low, enabling the RS-422 driver, the shift register being clocked by TCLK—will time the duration of the TXD high times. If the high time is more than 2 microseconds, meaning that the 82588 has gone idle, the transmitter will be disabled (See Figure 37). Another piece of this logic is the OR-ing of the output of the shift register with TXEN—signal which comes from the board control port. This signal completely disables the transmitter. The other purpose of this enable signal, is to make sure that after power-up, before the 82588 is configured, the RS-422 drivers won't be enabled (TCLK__ is not active before the configure command). See Figures 28, 29 for the complete circuit.

5.3 RECEIVE PATH

The signal coming from the HUB over the twisted pair wire is received on the StarLAN board through a 100Ω line termination resistor and a pulse transformer. The pulse transformer is of the same type as for the transmit side and its function is dc isolation. The received signal which is differential and almost sinusoidal is fed to the Am26LS32 RS-422 receiver. As seen from Figure 38 the pulse transformer feeds two RS-422 receivers. The one on the bottom is for squelch filtering and the one above is the real receiver which does real zero crossing detection on the signal and regenerates a square digital waveform from the sinusoidal signal that is received. Proper zero crossing detection is very essential; if the edges of the regenerated signal are not at zero crossings, the resulting signal may not be a proper Manchester encoded signal (self introduced jitter) even if the original signal is valid Manchester. The resistors in the lower receiver keep its differential inputs at a voltage difference of 600 mV. These bias resistors ensure that the output remains high as long as the input signal is more than −600 mV. It is very important that the RxD pin remains HIGH (not LOW or floating) whenever the receive line is idle. A violation of this may cause the 82588 to lock-up on transmitting. Remember, that based on the signal on the RxD pin, the 82588 extracts information on the data being received, Carrier Sense and Collision Detect. This squelch of 600 mV keeps the idle line noise from getting to the 82588. Figure 39 shows that when the differential input of the receiver crosses zero, a transition occurs at the output. It also shows that if the signal strength is higher than −600 mV, the output does not change. (This kind of squelching is called negative squelching, and it is done due to the fact that the preamble pattern starts with a going low transition). Note that the differential voltage at the upper receiver input is zero when the line is idle. The output of the squelch goes to a pulse stretcher which generates an envelope of the received frame. The envelope is a receive enable signal and is used to AND the signal from the real zero crossing receiver before feeding it to the RxD pin of the 82588.

![Figure 37. Idle Generation](231422-83)
FILTERING OF HIGH FREQUENCY NOISE

TERMINATION

HIGH FREQUENCY FILTER

300Ω

2.2 KΩ

VOLTAGE SQUELCH (600 mV)

TIME SQUELCH

ENVELOPE IDLE DETECT

DATA

CARRIER

INPUT DATA

CARRIER

3 EDGES

1.6 μs

Figure 38. Input Ports

Figure 39. Squelch Circuit Output
5.4 80188 Interface to 82588

Although the 82588 interfaces easily to almost any processor, no processor offers as much of the needed functionality as the 80186 or its 8 bit cousin, the 80188. The 80188 is 8088 object code compatible processor with DMA, timers, interrupt controller, chip select logic, wait state generator, ready logic and clock generator functions on chip. Figure 40 shows how the 82588, in a StarLAN environment interfaces to the 80188. It uses the clock, chip select logic, DMA channels, interrupt controller directly from the 80188. The interface components between the CPU and the 82588 are totally eliminated.

5.5 iSBX Interface to StarLAN

Figure 41 shows how to interface the 82588 in a StarLAN environment to the iSBX bus. It uses 2 DMA channels—tapping the second DMA channel from a neighboring iSBX connector. Such a board can be used to make a StarLAN to an Ethernet or a SNA or DECNET gateway when it is placed on an appropriate SBC board. It may also be used to give a StarLAN access to any SBC board (with an iSBX connector) independent of the type of processor on the board.
Figure 40. 80188 Interface to 82588
6.0 THE StarLAN HUB

The function of a StarLAN HUB is described in section 2.0. Figure 42 shows a block diagram of a HUB. It receives signals from the nodes (or lower level HUBs) detects if there is a collision, generates the collision presence signal, re-times the signal and sends it out to the higher level HUB. It also receives signals from the higher level HUB, re-times it and sends it to all the nodes and lower level HUBs connected to it. If there is no higher level HUB, a switch on the HUB routes the upstream received signal down to all the lower nodes. The functions performed by a HUB are:

*Receiving signals, squelch
*Carrier Sensing
*Collision Detection
*Collision Presence Signal Generation
*Signal Retiming
*Driving signals on the cable
*Jabber Function
*Receive protection Timer

6.1 A StarLAN Hub for the IBM/PC

Figure 43 shows the implementation of a 5/6 port HUB for the IBM/PC.

The idea of the following design is to show a HUB that plugs into the IBM/PC backplane. This HUB not only gets its power from the backplane, but also enables the host PC to be one NODE into the StarLAN network. This embedded node scheme enables further savings due to the fact that all the analog interface for this port is saved (receiver, transmitter, transformer, etc).

This kind of board would suit very much a small cluster topology (very typical in departments and small offices) where the HUB board would be plugged into the FILE SERVER PC (PC/XT, PC/AT).

The HUB design doesn’t implement the Jabber and the protection timers as called by the IBASE5 draft standard. Those functions are optional and were not closed during the writing of this AP-NOTE. This HUB does implement the RETIMING circuit which is an essential requirement of StarLAN.

Figures 44 to 49 show a complete set of schematics for the HUB design.

Figure 42. StarLAN HUB
* Low Cost HUB, Uses IBM/PC Power Supply
* 82588, Embedded Port Savings
  Transformers
  422 Drivers
* Functional StarLAN Cluster, For Low Cost/Small Topologies

Figure 43. IBM/PC Resident HUB
Figure 47

DCO TECHNICAL MARKETING, INTEL

TITLE: RETIMING LOGIC (UP PATH)

DRAWN BY: ADI GOLBERT

DATE: 86-11-96 SHEET 4 OF 6

231422-91
6.1.1 HUB INPUT PORTS

Figure 38 shows a block diagram of an input port. Differently than the implementation in Figure 29 the HUB input port is potentially more complex than the NODE input port. The reason being that the HUB is a central resource and much more sensitive to noise. For example, if the NODE input port would falsely interpret noise on an IDLE line as valid signal, the worst case situation would be that this noise would be filtered out by the 82588 time squelch circuitry, on the HUB by the other hand, this false carrier sense could trigger a COLLISION and a good frame (on another input) potentially discarded.

As shown in Figure 38 immediately after the termination resistor, there is a HIGH FREQUENCY FILTER circuit. The purpose of this circuit is to eliminate high frequency noise components keeping noise jitter into the allocated budget (about ±30 ns). A 4 MHz two pole butterworth filter is being recommended by the IEEE 802.3 1BASE5 task force (see Figure 50).

The time squelch for the NODE board is implemented by the 82588 (see section 3.7) this circuit makes sure that pulses that are shorter than a specified duration will be filtered out.

The other components of the block diagram were explained in section 3.0.

The HUB design doesn’t implement the HIGH FREQUENCY FILTER and TIME SQUELCH. In the HUB design as an output of each input port, two signals are available: Rn, En, (RA, RB ..., EA, EB ... ). The Rn signals are the receive data after the zero crossing receivers. The En lines are CARRIER SENSE signals. The HUB design supports either 5 or 6 input ports, dependent upon if it is configured as IHUB or HHUB. Port RE, EE (Figure 49) is bidirectional, configurable for either input or output. Port RF, EF is the embedded 82588 port, and doesn’t require the analog circuitry (EF is inverted, being generated from the RTS__ signal).

Figure 50. Receiver High Frequency Filter
6.1.2 COLLISION DETECTION

Rn and En signals from each channel are fed to a 16L8 PAL, where the collision detection function is performed.

COLLISION DETECTION:

\[
CDT = \neg \left( (EA \land \neg EB \land \neg EC \land \neg ED \land EE \land EF) \right)
\]

- \((\neg EA \land \neg EB \land \neg EC \land \neg ED \land \neg EE \land EF)\)
- \((\neg EA \land \neg EB \land \neg EC \land \neg ED \land EE \land \neg EF)\)
- \((\neg EA \land \neg EB \land \neg EC \land ED \land \neg EE \land \neg EF)\)
- \((\neg EA \land \neg EB \land \neg EC \land \neg ED \land EE \land \neg EF)\)
- \((\neg EA \land \neg EB \land \neg EC \land \neg ED \land \neg EE \land \neg EF)\)

(COLLISION DETECTION SR-FF):

\[
COLLEN = \neg \left( \text{CDT} \land \text{COLLEN} \right)
\]

(only EA active)

(only EB active)

(only EC active)

(only ED active)

(only EE active)

(only EF active)

(none of the inputs active)

Collision Detection in the StarLAN HUB is performed by detecting the presence of activity on more than one input channels. This means if the signal En is active for more than one channel, a collision is said to occur. This translates to the PAL equations:

RECEIVE DATA OUTPUT:

\[
RCVDAT = \left( (RA \land \neg EA) \land (RB \land \neg EB) \land (RC \land \neg EC) \land (RD \land \neg ED) \land (RE \land \neg EE) \land (RF \land \neg EF) \right)
\]

(output is high if no active input)
The COLLEN signal once triggered will stay active until all inputs go quiet. This signal is used externally to either enable passing RCVDAT or the collision presence signal (CPS) to the retiming logic. An external multiplexer using 3 nand gates is used for this function. Note that in this specific implementation the CPS/RCVDAT multiplexer is before the retiming logic, which is different from Figure 42 diagram. StarLAN provides enough BIT-BUDGET delay to allow the CPS signal to be generated through the retiming FIFO. In this HUB implementation it was decided to use this option to make sure that the CPS startup is synchronized with the previously transmitted bit as required by the 1BASE5 draft.

### 6.1.3 THE LOCAL 82588

As described before, the purpose of the local 82588 is to enable the Host IBM/PC to also be a node into the StarLAN network. The interface of this 82588 is exactly similar to the one explained in section 5. The RTS signal serves as the carrier EF signal, and TXD as RF signal. This local node interfaces to the HUB without any analog interface which is a significant saving.

### 6.1.4 THE COLLISION PRESENCE SIGNAL

The Collision Presence Signal (CPS) is generated by the HUB whenever the HUB detects a collision. It then propagates the CPS to the higher level HUB. The CPS signal pattern is shown in Figure 51. Whenever a StarLAN node receives this signal, it should be able to detect within a very few bit times that a collision occurred. Since the nodes detect the occurrence of a collision by detecting violations in Manchester encoding, the CPS must obviously be a signal which violates Manchester encoding. Section 3.5 shows that the CPS has missing mid-cell transitions occurring every two and a half bit cells. These are detected as Manchester code violations. Thus, the StarLAN node is presented with collision detection indications every two and a half ms. This results in fast and reliable detection of collisions. CPS has a period of 5 ms.

One may wonder why such a strange looking signal was selected for CPS. The rationale is that this CPS looks very much like a valid Manchester signal—edges are 0.5 or 1.0 microsec. apart—resulting in identical radiation, cross-talk and jitter characteristics as a true Manchester. This also makes the re-timing logic for the signals simpler—it need not distinguish between valid Manchester and CPS. Moreover, this signal is easy to generate.

A few important requirements for CPS signal are: a) it should be generated starting synchronized with the last transmitted bit cell. CPS is allowed to start either low or high, but no bit cell of more than 1 microsecond is allowed (Avoid false idles, very long “low” bits). b) once it starts, it should continue until all the input lines to the HUB die out. Typically, when the collision occurs, the multiplexer in the HUB switches from RCV signal to the CPS. This switch is completely asynchronous to the currently being transmitted data, and by such may violate the requirement of not having bit cells longer than 1 μs. In order to avoid those long pulses, the output of the CPS/RCVDAT multiplexer is passed through the retiming circuitry which will correct those long pulses to their nominal value. The reason for restriction b) is to ensure that the CPS is seen by all nodes on the network since it is generated until every node has finished generating the Jam pattern.

![Figure 51. Collision Presence Signal](image-url)
CPS is generated using a 4-bit shift register and a flip-flop as shown in Figure 52. It works off a 2 MHz clock. A closer look at the CPS waveform shows that it is inverse symmetric within the 5 μs period. The circuit is a 5-hit shift register with a complementary feedback from the last to the first bit. The bits remain in defined states (01100) till collision occurs. On collision the bits start rotating around generating the pattern of 0011011001, 0011011001, 00110... with each state lasting for 0.5 μs.

![Figure 52. Collision Presence Signal Generation](image)

### 6.1.5 SIGNAL RETIMING

Whenever the signal goes over a cable it suffers jitter. This means that the edges are no longer separated by the same 0.5 or 1.0 μs as at the point of origin. There are various causes of jitter. Drivers, receivers introduce some shifting of edges because of differing rise and fall times and thresholds. A random sequence of bits also produces a jitter which is called intersymbol interference, which is a consequence of different propagation delays for different frequency harmonics in the cable. Meaning short pulses have a longer delay than long ones. A maximum of 62.5 ns of jitter can accumulate in a StarLAN network from a node to a HUB or from a HUB to another HUB. The following values show what are the jitter components:

<table>
<thead>
<tr>
<th>Component</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Transmitter skew</td>
<td>± 10 ns</td>
</tr>
<tr>
<td>Cable Intersymbol interference</td>
<td>± 9 ns</td>
</tr>
<tr>
<td>Cable Reflections</td>
<td>± 8 ns</td>
</tr>
<tr>
<td>Reflections due to receiver</td>
<td>± 5 ns</td>
</tr>
<tr>
<td>termination mismatch</td>
<td>± 5 ns</td>
</tr>
<tr>
<td>HUB fan-in, fan-out</td>
<td>± 5 ns</td>
</tr>
<tr>
<td>Noise</td>
<td>± 25.5</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td>± 62.5 ns</td>
</tr>
</tbody>
</table>

It is important for the signal to be cleaned up of this jitter before it is sent on the next stretch of cable because if too much jitter accumulates, the signal is no longer meaningful. A valid Manchester signal would, as a result of jitter, may no longer be decodable. The process of either re-aligning the edges or reconstructing the signal or even re-generating the signal so that it once again “looks new” is called re-timing. StarLAN requires for the signal to be re-timed after it has travelled on a segment of cable. In a typical HUB two re-timing circuits are necessary; one for the signals going upstream towards the higher level HUB and the other for signals going downstream towards the nodes.

### 6.1.6 RETIMING CIRCUIT, THEORY OF OPERATION

This section will discuss the principles of designing a re-timing circuit. Figure 53 shows the block diagram of a re-timing circuit. The data coming in is synchronized using an 8 MHz sampling clock. Edges in the waveform are detected doing an XOR of two consecutive samples. A counter counts the number of 8 MHz clocks between two edges. This gives an indication of long (6 to 10 clocks) or short (3 to 5 clocks) pulses in the received waveform. Pulses shorter than 3 clocks are filtered out. Every time an edge occurs, the length—(S)hort or (L)ong—of the pulse is fed into the FIFO. Retiming of the waveform is done by actually generating a new waveform based on the information being pumped into the FIFO. The signal regeneration unit reads the FIFO and generates the output waveform out of 8 MHz clock pulses based on what it reads, either short or longs. In summary every time a bit is read from the fifo, it indicates that a transition needs to occur, and when to fetch the next bit. When idle the output of the retiming logic starts with a “high” level.

<table>
<thead>
<tr>
<th>FIFO</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>empty</td>
<td>...... 1111</td>
</tr>
<tr>
<td>S</td>
<td>0000</td>
</tr>
<tr>
<td>S</td>
<td>1111</td>
</tr>
<tr>
<td>L</td>
<td>00000000</td>
</tr>
<tr>
<td>L</td>
<td>11111111</td>
</tr>
</tbody>
</table>

It can be seen that the output always has edges separated by 4 or 8 clock pulses—0.5 or 1.0 μs.

The FIFO is primarily needed to account for a difference of clock frequencies at the source and regeneration end. Due to this difference, data can come in faster or slower than the regeneration circuit expects. A 16 deep FIFO can handle frequency deviations of up to 200 ppm for frame lengths up to 1600 bytes. The FIFO also overcomes short term variations in edge separation. It is essential that the FIFO fills in up to about half before the process of regeneration is started. Thus, if the regeneration is done at a clock slightly faster than the source clock, there is always data in the FIFO to work from. That is why the FIFO threshold detect logic is necessary, which counts 8 edges and then enables the signal regeneration logic.
Example:
Input Waveform \( \ldots 111100001111000000011111111110001111100 \ldots \)
Input into the FIFO
\(<S><S><L><L\times S><S>\)
Regenerated Output:
Output: \( \ldots 111110000111100000000111111110000111 \ldots \)
FIFO: \(<S><S><L><S><S>\)

![Block Diagram](image)

**Figure 53. Retiming Block Diagram**

### 6.1.7 RETIMING CIRCUIT IMPLEMENTATION

The retiming circuit implementation can be seen in Figures 47, 48. Both figures implement exactly the same function, one for the upstream, and the other for the downstream. The retiming circuit was implemented using about 8 SSI, MSI TTL components, one fifo chip and one PAL. The purpose of implementing this function with discrete components was to show the implementation details. The discussion of the implementation will refer to Figure 47 for unit numbers.

The signal UPIMP which is an output of the HUB multiplexing logic, is asynchronous to the local clock. This signal is synchronized by two flip-flops and fed into an edge generation logic (basically an XOR gate that compares the present sample with the previous one). On every input transition a 125 ns pulse will be generated at the output of the edge detector (U28). This pulse will reset the 74LS161 counter that is responsible for measuring pulse widths (in X8 clock increments). The output of the pulse discriminator will reflect the previous pulse width every time a new edge is detected. The following events will take place on every detected edge:

1. U26 which is the threshold detector will shift one "1" in. The outputs of U26 will be used by the control PAL to start the reconstruction process.
2. The output of U23 which specifies the last pulse width will be input into the control PAL for determining if it was a long or short pulse. The result of this evaluation will be the LSIN signal which will be loaded into the fifo (U22).

U22 is the retiming FIFO, it is 16x4 fifo, but only one bit is necessary to store the SHORT/LONG information.
CONTROL LOGIC PAL functions (U25):

Signals definition:

INPUTS:

PD0..PD3: Outputs of the pulse descriminator, indicate the width of the last measured pulse.

EDD_: Output of the edge detector, pulse of 125 ns width, indicates the occurrence of an edge in the input data.

THRESH: Output of the threshold logic, indicates at least one bit was already received.

CNTEN: Output of the Threshold logic, indicates 7 bits have been loaded into the FIFO, and that signal reconstruction can begin.

CNTEND: The same signal as before delayed by one clock.

OUTDAT: Output of the retiming logic, is feedback into the PAL to implement a clocked T-FF.

RESET_: Resets the retiming logic.

OUTPUTS:

LDFIFO_: Loads SHORT/LONG indications into the FIFO.

LSIN: Indicates SHORT/LONG

CNTPE_: Loads FIFO SHORT/LONG output into the reconstruction counter.

ODAT: Together with the external U21 flip-flop and OUTDAT implement a clocked T-FF.

Loading the FIFO will be done every time there is an edge, we have passed the one bit filter threshold level, and the pulse width is longer than two 8X clocks. This one bit threshold level serves as a time domain filter discarding the first received preamble bit.

\[ \text{LDFIFO}_\neg = ! ( PD1 \# PD2 \# PD3 ) \& !EDD_ \& \text{THRESH} ; \]

Whenever there is an edge, we are above the first received bit threshold and the pulse width is longer than "1" the fifo is loaded.

\[ \text{LSIN} = ! (PD3 \# (PD2 \& PD0) \# (PD2 \& PD1)) ; \]

Every pulse longer than 6 is considered to be a long pulse.

\[ \text{CNTPE}_\neg = ! (\text{CNTEN} \& \neg \text{CNTEND}) \# \text{CNTTC} ; \]

The reconstruction counter is loaded in two conditions:

Whenever CNTEN comes active, meaning the FIFO threshold of seven was exceeded.
Whenever the terminal count of U24 is active meaning a new pulse is going to be reconstructed.

\[ \text{ODAT} = !\text{RESET} \# (\neg \text{CNTPE} \& \neg \text{OUTDAT}) \quad (A) \]
\[ # (\text{CNTPE} \& \text{OUTDAT}) \quad (B) \]
\[ # (\neg \text{CNTPE} \& \neg \text{OR}) \quad (C) \]

Minterm (A) and (B) implement a T-FF, whenever CNTPE is "low" ODAT will toggle. The external U21 is part of this flip-flop.
Minterm (C) insures the output of the flip-flop will go inactive "high" when the FIFO is empty. RESET causes the output to go "high" on initialization.
U24 as mentioned is the reconstruction counter. This counter is loaded by the control logic with either 8 or 12, it counts up and is reloaded on terminal count. Essentially generating at the output nominal length longs and shorts.

U22 is the retiming FIFO, and its function as mentioned is to accommodate frequency skews between the incoming and outgoing signal.

U27 is the IDLE generation logic. The purpose of this logic is to detect when the FIFO is empty, meaning that no more data needs to be transmitted. On detection of this event this component will generate 2 ms of IDLE time. On the end of IDLE the whole retiming logic will be reset.

6.1.8 DRIVER CIRCUITS

The signal coming out of the RETIMING LOGIC is fed into 26LS30s and pulse transformers to drive the twisted pair lines (See section 5.0 for details).

6.1.9 HEADER/INTERMEDIATE HUB SWITCH

As seen on Figure 43 this hub can be configured as either an intermediate hub, or a Header one. One of the phone jacks, more specifically JACK #5 is either an input port or an output one. In order to implement this function, an 8 position DIP SWITCH (SW1) is used. The phone jacks are marked with UD, DD notation, meaning upstream data, and downstream data respectively. As specified in the StarLAN 1BASE5 draft NODES transmit data on UD pair, and HUBS on the DD pair. Switch SW1 has the function to invert UD, DD in PHONE JACK #5 to enable it to be either input or output port.

6.1.10 JABBER FUNCTION

This design does not implement the jabber unit but it is described here for completeness. IEEE 802.3 does not mandate this feature, but it is “Strongly Recommended”. The jabber function in the HUB protects the network from abnormally long transmissions by any node.

Two timers T1, T2 are used by the JABBER function. They may be implemented either as local timers (one for each HUB port) or as global timers shared by all ports. After detecting an input active, timers T1, T2 will be started, and T1 will time out after 25 to 50 ms. T2 will time-out after 51 to 100 ms. During T2 time, after T1 expired, the HUB will send the CP-PATTERN informing any jamming stations to quit their transmissions. If on T2 time-out there are still jamming ports, their input is going to be disabled. A disabled port, will be reenabled whenever its input becomes again active and the downward side is idle.

The following is an explanation of the requirement that the downward side be idle to reenable an input port. Consider the case of Figure 54. The figure shows a two port HUB. Port A has two wires A_u, A_d for the up and down paths. Port B has B_u, B_d respectively. Port C is the output port, that broadcasts to the other HUBs higher in the hierarchy. Consider the case as shown, where B_u and B_d are shorted together. Suppose the case that port A_u is active. Its signal will propagate up in the hierarchy through C_u and come down from C_d to A_d, and B_d. Due to the short between B_d and B_u the signal will start a loop, that will first cause a collision and jam the network forever. This kind of fault is taken care of by the jabber circuitry. T1 and T2 will expire, causing the jabber logic to disable B_u input. Upon this disabling B_u is going to go Idle and be a candidate for future enabling. Suppose now that A_u is once again active. If the reenable condition would not require C_d to be IDLE, B_u would be reenabled causing the same loop to happen once again. Note that in this case C_d will be active before B_u causing this port to continue to be disabled and avoiding the jamming situation (Figure 55) gives a formal specification of the jabber function.

![Figure 54. Jabber Function](image-url)
6.1.11 HUB RECEIVER PROTECTION TIMER

On the end of a transmission, during the transition from IDLE to high impedance state, the transmitter will exhibit an undershoot and/or ringing, as a consequence of transformer discharge. This undershoot/ringing will be transmitted to the receiver which needs to protect itself from false carriers due to this effect. One way of implementing this protection mechanism is to implement a blind timer, which upon IDLE detection will “blind” the receiver for a few microseconds.

Causes of the transmitter undershoot/ringing:
1. Difference in the magnitudes of the differential output voltage between the high and the low output stages.
2. Waveform asymmetry due to transmitter jitter.
3. Transmitter and receiver inductance (transformer L).
4. Two to three microseconds of IDLE pattern.

All the described elements will contribute to energy storage into the transformer inductor, which will discharge during the transition of the driver to high impedance.

The blinding timer is currently defined to be from 20 to 30 microseconds for the HUBs, being from 0 to 30 microseconds for the nodes (optional). The 82588 has built-in this function. It won’t receive any frames for an inter-frame-spacing (IFS) from the idle detection.

6.1.12 HUB RELIABILITY

Since the StarLAN HUBs form focal points in the network, it is important for them to be very reliable, since they are single points of failure which can affect a number of nodes or can even bring down the whole network. StarLAN 1BASE5 draft requires HUBs to have a mean time between failures (MTBF) of at least 5 years of continuous operation.
7.0 SOFTWARE DRIVER

The software needed to drive the 82588 in a StarLAN environment is not different from that needed in a generic CSMA/CD environment. This section goes into specific procedures used for operations like TRANSMIT, RECEIVE, CONFIGURE, DUMP, ADDRESS SET-UP, etc. A special treatment will be given to interfacing with the IBM PC—DMA, interrupt and I/O.

Since all the routines were written and tried out in PLM-86 and ASM-86, all illustrations are in these languages.

The following software examples are pieces of an 82588 exerciser program. This program's main purpose was to exercise the 82588 functionality and provide the functions of traffic generation and monitoring. By such the emphasis was on speed and accuracy of statistics gathering.

7.1 Interfacing to IBM PC

The StarLAN board interfaces to the CPU, DMA controller and the interrupt controller on the IBM PC system board. The software to operate the 82588 runs on the system board CPU. The illustrated routines in this section show exactly how the software interface works between the system resources on the IBM PC and the StarLAN board.

7.1.1 DOING I/O ON IBM PC

The safest way to use the PC monitor as an output device and the keyboard as the input device is to use them through DOS system calls. The following is a set of routines which are handy to do most of the I/O:

- key$stat —to find out if a new key has been pressed
- keyin$noecho —to read a key from the keyboard
- char$out —to display a character on the screen
- msg$out —to display a character string on the screen
- line$in —to read in a character string from the keyboard

The exact semantics and the protocol for doing these functions through DOS system calls is shown in Figure 56. Refer to the DOS Manual for a more detailed description. To make a DOS system call, register AH of 8088 is loaded with the call Function Number and then, a software interrupt (or trap) 21 hex is executed. Other 8088 registers are used to transfer any parameters between DOS and the calling program. The code is written in Assembly language for register access. Let us see an example of the 'msg$out' routine:

```
lds dx,STRING_POINTER ; load pointer to string in reg. ds:dx
mov ah,09h
int 21h ; 9 = function number for string o/p
```

These procedures are called from another module, written in a higher level language like PLM-86. The parameters are transferred to the ASM-86 routines on the stack.

Examples of using the I/O routines:

```
KEY_STATUS = key$stat; /* INQUIRE KEYBOARD STATUS */
NEW_KEY = keyin$noecho; /* INPUT NEW KEY */
call line$in(@LINE_BUFFER); /* STRING INPUT */
call char$out(CHAR_OUT); /* TO OUTPUT CHAR_OUT ON SCREEN */
call msg$out('@('THIS IS A MESSAGE.$')'); /* OUTPUT STRING */
call msg$out('@('NOTE $ TERMINATOR */
```
key$stat: procedure byte external; /* key status routine */
end key$stat;

key$in$noecho: procedure byte external; /* console input routine */
end key$in$noecho;

char$out: procedure(char) external; /* console output routine */
declare char byte;
end char$out;

msg$out: procedure(msg$ptr) external; /* console string output routine */
declare msg$ptr pointer;
end msg$out;

line$in: procedure(line$ptr) external; /* console string input routine */
declare line$ptr pointer;
end line$in;

Assembly Language implementation of the routines
$TITLE(IBM/PC DOS CALLS PROCEDURES)

NAME DOSPROCS

; DECLARE CHAR BYTE;
; END CHAR$OUT;
; Outputs character to the screen.
; DOS system call 2
;
CHAR

CHAR$OUT: PROCEDURE(CHAR) EXTERNAL;
end CHAR$OUT;

CHAR$OUT PROC NEAR
PUBLIC CHAR$OUT
PUBLIC CHAR

PUSH BP
MOV BP,SP
MOV DL,CHAR
MOV AH,2
INT DOS
POP BP
RET
X

CHAR$OUT ENDP

; KEYIN$NOECHO: procedure byte external;
; end KEYIN$NOECHO;
; Reads character without echoing to display;
;
KEYIN$NOECHO PROC NEAR
PUBLIC KEYIN$NOECHO
MOV AH,6
INT DOS
RET

KEYIN$NOECHO ENDP

Figure 7-56. I/O Routines for IBM/PC (continued)
### 7.2 Initialization and Declarations

Figure 56 shows the initialization routines for the IBM PC and for the 82588. It also shows some of the typical values taken by the memory buffers for Configure, IA_Set, Multicast and transmit buffers.

Figure 57 shows some declarations describing what addresses the devices have and also some literals to help understand the other routines in this section.

```assembly
    ; MSGOUT: PROCEDURE(MSG$PTR) EXTERNAL;
    ; DECLARE MSG$PTR POINTER;
    ; END MSGOUT;
    ; /* NOTE: MESSAGE IS TERMINATED WITH A DOLLAR SIGN */
    ; MSG$PTR is double word pointer SEG:OFFSET

    MSG_L EQU [BP+4]
    MSG_H EQU [BP+8]

    MSGOUT PUBLIC PROC NEAR
    
    PUSH BP
    MOV BP,SP
    MOV DX,MSG_L
    PUSH DS
    MOV AX,MSG_H
    MOV DS,AX
    MOV AX,9
    Int DOS
    POP DS
    POP BP
    RET 4

    MSGOUT ENDP

    ; LINE$IN: PROCEDURE(LINE$PTR) EXTERNAL;
    ; DECLARE LINE$PTR POINTER;
    ; END LINE$IN

    LINE_L EQU [BP+4]
    LINE_H EQU [BP+8]

    LINEIN PUBLIC PROC NEAR
    
    PUSH BP
    MOV BP,SP
    PUSH DS
    MOV AX,LININ
    MOV DS,AX
    MOV AX,10
    Int DOS
    POP DS
    POP BP
    RET 4

    LINEIN ENDP

    ; KEY$STAT: PROCEDURE BYTE EXTERNAL;
    ; END KEY$STAT;
    ; Indicates whether any keyboard key was pressed.

    KEYSTAT PUBLIC PROC NEAR
    
    PUSH AX
    MOV AH,11
    Int DOS
    RET

    KEYSTAT ENDP

    CODE END
```

Figure 56. I/O Routines for IBM/PC (Continued)
Following are some literal declarations that are used in the procedure examples:

```
Following are some literal declarations that are used in the procedure examples.

declare

os_588         literally '0300h' /* 82588 COMMAND/STATUS */
brd_port       literally '0301h' /* DMA/INTERUPT ENABLE PORT */
pio_mask       literally '0311h' /* 8258A MASK REGISTER */
pio_cowd       literally '0302h' /* 8258A COMMAND WORD */
dma_mask       literally '0304h' /* 8257A MASK REGISTER */
dma_mode       literally '0305h' /* 8257A MODE REGISTER */
dma_fifr       literally '0306h' /* 8257A INT/END BYTE FLOP */
dma_addr_1     literally '0307h' /* 8257A CHANNEL 1 ADDR. REG. */
dma_bo_1       literally '0308h' /* 8257A CHANNEL 1 BYTE COUNT */
dma_addr_3     literally '0309h' /* 8257A CHANNEL 3 ADDR. REG. */
dma_bo_3       literally '030Ah' /* 8257A CHANNEL 3 BYTE COUNT */
dma_addr_5     literally '030Bh' /* 8257A CHANNEL 5 ADDR. REG. */
dma_bo_5       literally '030Ch' /* 8257A CHANNEL 5 BYTE COUNT */
dma_on_1       literally '0311h' /* START CHANNEL 1 */
dma_on_3       literally '0312h' /* START CHANNEL 3 */
dma_off_1      literally '0313h' /* STOP CHANNEL 1 */
dma_off_3      literally '0314h' /* STOP CHANNEL 3 */
seoi_pico      literally '0365h' /* SPECIFIC EO LEVEL 5 */
tx_dir         literally '01h' /* MEMORY TO 82588 */
rx_dir         literally '01h' /* 82588 TO MEMORY */
dma_rx_mode_1  literally '0458h' /* RX ON CHANNEL # 1 */
dma_rx_mode_3  literally '045Ch' /* RX ON CHANNEL # 3 */
dma_tx_mode_1  literally '0454h' /* TX ON CHANNEL # 1 */
dma_tx_mode_3  literally '045Oh' /* TX ON CHANNEL # 3 */
```

Figure 57. Literal Declarations

Initialization Routines

```
Initialization routines
/* SYSTEM INITIALIZE */
syst_init: procedure;
call setinterrupt (15, intr_588); /* BASE 8. LEVEL 5 */
output(pio_mask) = input(pio_mask) and enable_588; /* ENABLE 588 INTR. */
output(pio_cowd) = seoi_pico; /* ACKS PENDING INTR. */
wr_ptr.rd_ptrfifoout = 0; /* RESET STATUS FIFO */

/* CONVERT SEG:OFFSET FORMAT TO 20 BIT ADDRESSES */
/* FOR ALL THE BUFFERS */
/* COUNTERS */
/* DMA SEGMENT ADDRESS */
/* DMA CRITICAL ADDRESS */
/* DMA FIF0 ADDRESS */
/* DMA CHANNEL ADDRESS */
/* DMA CHANNEL ADDRESS */
/* DMA CHANNEL ADDRESS */
/* DMA CHANNEL ADDRESS */
/* DMA CHANNEL ADDRESS */
/* DMA CHANNEL ADDRESS */
/* DMA CHANNEL ADDRESS */
/* DMA CHANNEL ADDRESS */
/* DMA CHANNEL ADDRESS */
/* DMA CHANNEL ADDRESS */
/* DMA CHANNEL ADDRESS */
/* DMA CHANNEL ADDRESS */

iset_dma_addr = convert_32bit_addr(dma_set_buff_588(0));
cnf_dma_addr = convert_32bit_addr(dmaconfigure_588(0));
dmp_dma_addr = convert_32bit_addr(dma_dump_buff_588(0));
mo_dma_addr = convert_32bit_addr(dma masturbation_buff_588(0));
tx_dma_addr = convert_32bit_addr(tx_buffer_588(0));
do i=0 to 7;
   tx_dma_addr(1) = convert_32bit_addr(txr_buffer(i).buff(0));
end;
output(brd_port) &= offh; /* ENABLE DMA AND INTERRUPT DRIVERS */
end syst_init;

82588 initialization
init_588: procedure;
config_588(00) = 10; /* TO CONFIGURE ALL 10 PARAMETERS */
config_588(01) = 00;
config_588(08) = 00010000b; /* MODE 0, 8 MSEG CLOCKS, 1 KB/S */
config_588(09) = 00001100b; /* RECEIVE BUFFER LENGTH */
config_588(0A) = 00101100b; /* NO LOOPBACK, ADDR LEN = 6, PREAMBLE = 8 */
config_588(0B) = 00000000b; /* DIFFERENTIAL MANCHESTER = OFF */
config_588(0C) = 00; /* PARITY = 0 */
config_588(0D) = 10011011b; /* SLOT TIME = 512 TCLK */
config_588(0E) = 11101010b; /* MAX. NO. RETRIES = 15 */
config_588(0F) = 00000000b; /* MANCHESTER ENCODING */
config_588(10) = 10001001b; /* INTERNAL CRST AND CTZ CRSF = 0 */
config_588(11) = 64; /* MIN FRAME LENGTH = 64 BYTES = 512 BITS */
```

Figure 58. Initialization Routines
Figure 58. Initialization Routines (Continued)

7.3 General Commands

Operations like Transmit, Receive, Configure, etc. are done by a simple sequence of loading the DMA controller with the necessary parameters and then writing the command to the 82588.

Example: Configure Command

To configure the operating environment of the 82588. This command must be the first one to be executed after a RESET.

call
DMA_LOAD(1,1,12,@CONFIG_588_ADDR);
output (CS_588) = 12h;

The first statement is the prologue to the configure command to the 82588 which calls a routine to load and initialize the DMA controller for the desired operation. This routine is described in section 7.4. The parameters for DMA_LOAD are:

first parameter = 82588 channel number ( = 1)
second parameter = direction ( = 1, memory >> 82588)
third parameter = length of DMA transfer ( = 12)
fourth parameter = pointer to a 20 bit address of the memory buffer (=@CONFIG_588_ADDR)

The second statement writes 12h to the command register of the 82588 to execute a Configure command on channel 1.

When the command execution is complete (successfully or not), 82588 interrupts the 8088 CPU through the 8259A, on the system board. This executes the interrupt service routine, described in section 7.5, which takes the epilogue action for the command.

Most operations are very similar in structure to Configure. The 82588 Reference Manual describes them in detail. Figure 59 shows a listing of the most commonly used operations like:

CONFIGURE    INDIVIDUAL-ADDRESS (IA)
TRANSMIT     MULTICAST-ADDRESS (MC)
DIAGNOSE     RECEIVE (RCV)-ENABLE
DUMP         RECEIVE (RCV)-DISABLE
TDR          RECEIVE (RCV)-STOP
RETRANSMIT    READ-STATUS
ia_set: procedure public;  /* COMMAND - 01 */
    call dma_load(cmd_channel, tx_dir, 8, @iaset_dma_addr);
    /* SET DMA CHANNEL 0 OR 1 TO TRANSFER FROM MEMORY
    TO THE 82558. iaset_dma_addr VARIABLE STORES THE
    20 BIT POINTER TO THE INDIVIDUAL ADDRESS BUFFER */
    if cmd_channel then output (os_888) = 11h;
    else output(os_888) = 01h;
    /* EVERY COMMAND CAN BE EXECUTED IN EITHER DMA CHANNEL 0 OR 1.
    THE VARIABLE cmd_channel INDICATES THE REQUIRED CHANNEL */
end ia_set;
/*--------------------------------------------------------------------------*/
config: procedure public;  /* COMMAND - 02 */
    call dma_load(cmd_channel, tx_dir, 12, @conf_dma_addr);
    if cmd_channel then output (os_888) = 12h;
    else output(os_888) = 02h;
end config;
/*--------------------------------------------------------------------------*/
multicast: procedure public;  /* COMMAND - 03 */
    call dma_load(cmd_channel, tx_dir, 14, @mco_dma_addr);
    if cmd_channel then output (os_888) = 13h;
    else output(os_888) = 03h;
end multicast;
/*--------------------------------------------------------------------------*/
transmit: procedure(buffer_len) public;  /* COMMAND - 04 */
    declare buffer_len word;
    tx_buffer_888(00) = low(buffer_len);
    tx_buffer_888(01) = high(buffer_len);
    call dma_load(cmd_channel, tx_dir, 1586, @tx_dma_addr);
    if cmd_channel then output (os_888) = 14h;
    else output(os_888) = 04h;
end transmit;

Figure 59. General Commands
tdr: procedure public;  /* COMMAND - 08 */
    if cmd_channel then output (os_588) - 16h;
    else output(os_588) - 05h;
end tdr;

/*--------------------------------------------------------------------------------*/
dump_588: procedure public;  /* COMMAND - 06 */
call dma_load(cmd_channel, rx_dir, 64, &dump_dma_addr);
    if cmd_channel then output (os_588) - 16h;
    else output(os_588) - 05h;
end dump_588;

/*--------------------------------------------------------------------------------*/
diagnose: procedure public;  /* COMMAND - 07 */
    if cmd_channel then output (os_588) - 17h;
    else output(os_588) - 07h;
end diagnose:

/*--------------------------------------------------------------------------------*/
rov_enable: procedure(channel, buffer_no, len) public;  /* COMMAND - 08 */
declare channel byte;
declare len word;
declare buffer_no byte;
call dma_load(channel, rx_dir, len, &rx_dma_addr(buffer_no));
    if rx_channel then output (os_588) - 16h;
    else output(os_588) - 05h;
end rov_enable;

/*--------------------------------------------------------------------------------*/
rov_disable: procedure public;  /* COMMAND - 10 */
    enable_rov:0;
    output(os_588)-0ah;
end rov_disable;

/*--------------------------------------------------------------------------------*/
rov_stop: procedure public;  /* COMMAND - 11 */
    enable_rov:0;
    output(os_588)-08h;
end rov_stop;

/*--------------------------------------------------------------------------------*/
retransmit: procedure public;  /* COMMAND - 12 */
call dma_load(cmd_channel, tx_dir, 1536, &tx_dma_addr);
    if cmd_channel then output (os_588) - 16h;
    else output(os_588) - 06h;
end retransmit;

/*--------------------------------------------------------------------------------*/
abort: procedure public;  /* COMMAND - 13 */
    output(os_588) - 1dh;
    call new_status(1);
end abort;

/*--------------------------------------------------------------------------------*/
reset_588: procedure public;  /* COMMAND - 14 */
    enable_rov:0;
    output(os_588) - 1eh;
    call config;
end reset_588;

Figure 59. General Commands (Continued)
7.4 DMA Routines

DMA_LOAD procedure is used to program the 8237A DMA controller for all the operations requiring DMA service. It also starts or enables the programmed DMA channel after programming it. Figure 60 shows the listing of this procedure. It accepts 4 parameters from the calling routine to decide the programming configuration for the 8237A. The parameters for DMA_LOAD are: Channel, direction, buff_len, and buff_addr.

Converting a pointer SEG:OFFSET to a 20 bit address
convert_20bit_addr:procedure(ptr) dword public;
    declare ptr pointer,
        ptr_addr pointer,
        ptr_20bit dword (wrd based ptr_addr)(2) word;
    ptr_addr=ptr;
    ptr_20bit=shl((ptr_20bit:-wrd(1)):4)+wrd(0);
    return(ptr_20bit);
end convert_20bit_addr;

IBM/PC DMA loading procedure
dma_load: procedure(channel,direction,buff_len,buff_addr) reentrant public;
    declare channel byte;
    declare direction byte; /* RX, 586 --> MEM, TX, MEM --> 586 */
    declare buff_len word; /* BYTE COUNT */
    declare buff_addr pointer; /* BUFFER ADDR IN 20 BITS FORM */
    declare (wrd based buff_addr)(2) word;
    channel=channel and 1; /* GET LEAST SIGNIFICANT BIT */
    if channel=0 then /* EXECUTE COMMAND ON CHANNEL 1 */
        do;
            output(dma_ff1) = 0; /* CLEAR FIRST/LAST FLIP-FLOP */
            if direction=0 then output(dma_mode)=dma_rx_mode_1; /* DIRECTION BIT, TELLS */
                else output(dma_mode)=dma_tx_mode_1; /* TRANSMIT OR RECEIVE */
                output(dma_addr_1) = low(wrd(0)); /* LOAD LSB ADDRESS BYTE */
                output(dma_addr_1) = high(wrd(0)); /* LOAD PAGE REGISTER */
                output(dma_bo_1) = low(buff_len); /* LOAD LSB BYTE COUNT */
                output(dma_bo_1) = high(buff_len); /* LOAD MSB BYTE COUNT */
                output(dma_mask) = dma_on_1; /* START CHANNEL 1 */
                end;
            else
do;
            output(dma_ff1) = 0; /* SAME AS BEFORE FOR CHANNEL 3 */
            if direction=0 then output(dma_mode)=dma_rx_mode_3;
                else output(dma_mode)=dma_tx_mode_3;
                output(dma_addr_3) = low(wrd(0));
                output(dma_addr_3) = high(wrd(0));
                output(dma_addr_3) = low(wrd(1));
                output(dma_addr_3) = high(wrd(1));
                output(dma_bo_3) = low(buff_len);
                output(dma_bo_3) = high(buff_len);
                output(dma_mask) = dma.on_3;
                end;
        end dma_load;

Figure 60. DMA Routine
One peculiarity about this procedure is that in order to speed up the DMA step-up, this procedure doesn’t get a pointer to the buffer, but a pointer to a 20 bit address in the 8237 format. The 8088/8086 architecture define pointers as 32 bits seg:offset entities, where seg and offset are 16 bit operands. By the other hand the IBM/PC uses an 8237A and a page register, requiring a memory address to be a 20 bit entity. The process of converting a seg:offset pointer to a 20 bit address is time consuming and could negatively affect the performance of the 82588 driver software. The decision was to make the pointer/address conversions during initialization, considering that the buffers are static in memory (essentially removing this calculation from the real time response loops).

Figure 61 is a listing of the DMA_LOAD procedure for the 80188 or 80188 on-chip DMA controller. It has the same caller interface as the 8237A based one.

```
dma_load: procedure(channel,direction,trans_len,buff_addr) reentrant;
/* To load and start the 80186 DMA controller for the desired operation */
declare dma_rx_mode literally '1010001001000000b'; /* rx channel */
   /* src=IO, dest=M(inc), sync=src, TC, noint, priority, byte */
declare dma_tx_mode literally '0000110100000000b'; /* tx channel */
   /* src=M(inc), dest=IO, sync=dest, TC, noint, noprior, byte */
declare channel byte;    /* channel # */
declare direction byte;   /* 0 = rx, 588 -> mem; 1 = tx, mem -> 588 */
declare trans_len word;  /* byte count */
declare buff_addr pointer; /* buffer pointer in 20 bit addr. form */
declare (wrd based buff_addr)(2) word;
do case channel and 00000001b;
do case direction and 00000001b;
do;
   /* channel 0, 588 to memory */
   output(dma_0_dpl) = wrd(0);
   output(dma_0_dph) = wrd(1);
   output(dma_0_spl) = ch_a_588;
   output(dma_0_sph) = 0;
   output(dma_0_tc) = trans_len;
   output(dma_0_cw) = dma_rx_mode or 0006h; /* Start DMA chl 0 */
end;
do;
   /* channel 0, memory to 588 */
   output(dma_0_dpl) = ch_a_588;
   output(dma_0_dph) = 0;
   output(dma_0_spl) = wrd(0);
   output(dma_0_sph) = wrd(1);
   output(dma_0_tc) = trans_len;
   output(dma_0_cw) = dma_tx_mode or 0006h; /* Start DMA chl 0 */
end;
end;
```

Figure 61. 80186 DMA Routines
do case direction and 000000001b;
do;  /* channel 1, 588 to memory */
output(dma_1_dpl) = wrd(0);
output(dma_1_dph) = wrd(1);
output(dma_1_spl) = ch_b_588;
output(dma_1_sph) = 0;
output(dma_1_tc) = trans_len;
output(dma_1_cw) = dma_rx_mode or 0006h; /* Start DMA chl 1 */
end;
do;  /* channel 1, memory to 588 */
output(dma_1_dpl) = ch_b_588;
output(dma_1_dph) = 0;
output(dma_1_spl) = wrd(0);
output(dma_1_sph) = wrd(1);
output(dma_1_tc) = trans_len;
output(dma_1_cw) = dma_tx_mode or 0006h; /* Start DMA chl 1 */
end;
end;
ed dma_load;

Figure 61. 80186 DMA Routines (Continued)

7.5 Interrupt Routine

The interrupt service routine, 'intr_588', shown in Figure 62, is invoked whenever the 82588 interrupts. The main difficulty in designing this interrupt routine was to speed its performance. Fast status processing was a basic requirement to be able to handle back to back frames.

The interrupt handler will read 82588 status, and put them into a 64 byte long EVENT_FIFO. Those statuses are going to be used in the main loop for updating screen counters. All the statistics are updated as fast as possible in the interrupt handler to fulfill the back-to-back frame processing requirement.

The interrupt handler is not reentrant, interrupts are disabled at the beginning and reenabled on exit.
Interrupt service routine

The code snippet provided is written in assembly language, specifically targeting the Intel 8086 processor. It outlines an interrupt service routine for managing data transfers and handling collisions. Here’s a breakdown of the key components:

1. **Initialization**: The routine begins by declaring variables necessary for the interrupt service. These include `stat`, `event`, `st0`, `st1`, `st2`, and `st3`, each of type `byte`.

2. **Interrupt Handling**: The routine is designed to handle interrupts selectively. It checks the status flags for the channels to determine if an interrupt is related to a DMA operation or a collision.

3. **Input Handling**: When an interrupt is received, the routine reads the status flags from the DMA controller. These status flags indicate whether the DMA operation was successful or if there was a collision.

4. **Data Processing**: Depending on the status, the routine performs actions such as updating the DMA controller's status, checking for collisions, and processing the DMA requests.

5. **Collision Handling**: If a collision is detected, the routine updates the collision counter and handles the situation accordingly.

6. **Data Transmission**: The routine also handles the transmission of data, ensuring that it is completed successfully.

The provided code snippet is a part of a larger assembly language program designed for the Intel 8086 processor, and it reflects typical interrupt handling in early computer systems where DMA and collision detection were critical for efficient data transfer.

---

**Figure 62. Interrupt Routine**

[Image: Diagram of the interrupt service routine]
/* EV_09 REQUESTS ASSIGNMENT OF A NEW BUFFER */
ev_09: call allocate_new_buffer(scr(scr(st1 and 00000001) and 00000001b));
ev_09: stop_rx_dma; /* STOP RECEIVE */
ev_10: stop_rx_dma; /* STOP RECEIVE */
ev_11: do; /* RE-TRANSMIT DONE */
   if (stat=s0h) /* RETRANSMIT */
   then do; /* RETRANSMIT */
      call dma_load(1,tx_dir,1556,tx_dma_addr);
      issue_rx_cmd;
      coll_cnt(17) = coll_cnt(17) + 1;
      total_rx_count = total_rx_count + 1;
      bad_rx_count = bad_rx_count + 1;
      end;
   else do; /* LOOP RETRANSMISSIONS */
      call dma_load(cmd_channel,tx_dir,1556,tx_dma_addr);
      issue_rx_cmd;
      total_tx_count = total_tx_count + 1;
      end;
end;
/* MAX COLLISION */
if (stat=0ACh) then do:
   call dma_load(1,tx_dir,1556,tx_dma_addr);
   issue_rx_cmd;
   coll_cnt(16) = coll_cnt(16) + 1;
   bad_rx_count = bad_rx_count + 1;
   end;
/* UPDATE SPECIFIC COLLISION COUNTER */
else coll_cnt(st1 and 0Ch) = coll_cnt(st1 and 0Ch) + 1;
end;
/* EXECUTION ABORTED */
ev_15: stop_cmd_dma;
/* DIAGNOSIS FAILED */
ev_16: stop_cmd_dma;
/* DIAGNOSIS FAILED */
end;
/* ACKNOWLEDGE 8259A INTERRUPT */
output(pio,cmd) = 8259A_CMD; /* SPECIFIC EOI FOR 8259 */
end intr_588;

Figure 62. Interrupt Routine (Continued)
APPENDIX A
STARLAN SIGNALS
Figure 63. StarLAN Signals
Figure 64. Received Signal Eye Diagram
APPENDIX B
802.3 1BASE5 MULTI-POINT EXTENSION (MPE)

As previously stated, one of the most important advantages of StarLAN is being able to work on already installed phone wires. This advantage is considerably diminished in Europe where numerous constraints exist to the using of those wires:

1. Wire belongs to local PTTs.
2. Not enough spare wires.

This same issue is raised when talking about small businesses where in a lot of cases no wiring closets and/or spare wires are available.

In summary, in a lot of cases rewiring will be necessary, in which case the STAR topology may not be the most economical one.

Recently the StarLAN 802.3 1BASE5 task force has been considering the extension of the StarLAN base topology. This extension called MULTI POINT EXTENSION (MPE) is going to be developed to address the previously described marketing requirements.

Currently no agreement has been reached by the StarLAN task force on the MPE exact topology and implementation. Multiple approaches have been presented, but no consensus met. It was decided though that the MPE is going to be an addendum to the STAR topology, and that its final specification will happen after the approval of the current 1BASE5 STAR topology (July 1986).
HUB COST ELIMINATED IN SMALL TOPOLOGIES. LOWER COST PER PORT (UP TO 8 STATIONS PER PORT)

THROUGH A HUB UPGRADEABLE TO THE FULL STARLAN TOPOLOGY (2500 m. MAX END-TO-END)

CONNECTION OPTIONAL, NOT NEEDED FOR SMALL TOPOLOGIES

LOWER COST. TERMINALS ATTRACTIVE

USER INSTALLABLE INTERCONNECT

FEWER CONNECTIONS TO WIRING CLOSETS
APPENDIX C
SINGLE DMA CHANNEL INTERFACE

In a typical system, the 82588 needs 2 DMA channels to operate in a manner that no received frames are lost as discussed in section 5.1.3. If an existing system has only one DMA channel available, it is still possible to operate the 82588 in a way that no frames are lost. This method is recommended only in situations where a second DMA channel is impossible to get.

Figure 66 shows how the 82588 DMA logic is interfaced to one channel of a DMA controller. Two DRQ lines are ORed and go to the DMA controller DRQ line and the DACK line from the DMA controller is connected to DACK0 and DACK1 of the 82588. The 82588 is configured for multiple buffer reception (chaining), although the entire frame is received in a single buffer. Let us assume that channel CH-0 is used as the first channel for reception. After the ENABLE RECEIVE command, CH-0 is dedicated to reception. As long as no frame is received, the other channel, CH-1, can be used for executing any commands like transmit, multicast address, dump, etc., by programming the DMA channel for the execution command. The status register should be checked for any ongoing reception, to avoid issuing an execution command when reception is active.

![Diagram of 82588 Using One DMA Channel](image)

Figure 66. 82588 Using One DMA Channel

If a frame is received, an interrupt for additional buffer occurs immediately after an address match is established, as shown in Figure 67. After this, the received bytes start filling up the on-chip FIFO. The 82588 activates the DRQ line after 15—FIFO LIMIT + 3 bytes are ready for transfer in the FIFO (about 80 microseconds after the interrupt). The CPU should react to the interrupt within 80 μs and disable the DMA controller. It should also issue an ASSIGN ALTERNATE BUFFER command with INTACK to abort any execution command that may be active. The FIFO fills up in about 160 μs after interrupt. To prevent an underrun, the CPU must reprogram the DMA controller for frame reception and re-enable the DMA controller within 160 μs after the interrupt (time to receive about 21 bytes). No buffer switching actually takes place, although the 82588 generates request for alternate buffer every time it has no additional buffer. The CPU must respond to these interrupts with an ASSIGN ALTERNATE BUFFER command with INTACK. To keep the CPU overhead to a minimum, the buffer size must be configured to the maximum value of 1 kbyte.

If a frame transmission starts deferring due to the reception occurring just prior to an issued transmit command, the transmission can start once the link is free after reception. A maximum of 19 bytes are transmitted (stored in the FIFO and internal registers) followed by a jam pattern and then an execution aborted interrupt occurs. The aborted frame can be transmitted again.

If the transmit command is issued and the 82588 starts transmitting just prior to receiving a frame then transmit wins over receive—but this will obviously lead to a collision.

Note that the interrupt for additional buffer is used to abort an ongoing execution command and to program the DMA channel for reception just when a frame is received. This scheme imposes real time interrupt handling requirements on the CPU and is recommended only when a second DMA channel is not available.
Figure 67. Timing at the Beginning of Frame Reception for Single DMA Channel Operation
APPENDIX D
MEASURING NETWORK DELAYS WITH THE 82588

Knowing networks round-trip delays in local area networks is an important capability. The round-trip delay very much defines the slot time parameter which by itself has a direct relationship to network efficiency and throughput. Very often the slot-time parameter is not flexible, due to standards requirements. Whenever it is flexible, optimization of this number may lead to significant improvement in network performance.

Another possible usage of the network delay knowledge is in balancing the inter-frame-spacing (IFS) on broadband networks. On those networks, stations nearer to the HEAD-END hear themselves faster than farther ones. Effectively having a shorter IFS than stations far from the HEAD-END. This difference causes an imbalance in network access time for different stations at different distances from the HEAD-END. Knowing the STATION/HEAD-END delay allows the user to reprogram the 82588 IFS accordingly, and by that balance the effective IFS for all the stations.

The 82588 has an internal mechanism that allows the user to measure this delay in BIT-TIME units. The method is based on the fact that the 82588 when configured for internal collision detection, requires that the carrier sense be active within half a slot-time after transmission has started. If this requirement is not fulfilled the 82588 notifies that a collision has occurred. Thus it is possible to configure the 82588 to different slot-time values, then transmit a long frame (of at least half a slot-time). If the transmission succeeds, the network round-trip delay is less than half the programmed slot-time. If a collision is reported, the delay is longer. The value of the round-trip delay can be found by repeating this experiment process while scanning the slot-time configuration parameter value and searching the threshold. A binary search algorithm is used for that purpose. First the slot-time is configured for the maximum (2048 bits) and according if there was a collision or not, the number changed for the next try. (See Figure 68)
The scheme is based on the fact that the 82588 expects RX carrier to be active after 1/2 slot time.

Figure 68. Network Delay Measurement using the 82588
Using the Intel 82592 to Integrate a Low-Cost Ethernet Solution into a PC Motherboard

MICHAEL ANZILOTTI
TECHNICAL MARKETING ENGINEER
1.0 INTRODUCTION

During the past several years office networking has become an increasingly efficient method of resource sharing for companies looking to increase productivity while reducing cost. Networking allows multiuser access to a data base of files or programs via a network file server; it allows sharing of expensive peripherals; e.g., laser printers; and it offers a greater degree of data security by centralizing the hard disk and backup facilities. This type of network allows a user to concentrate his resources; e.g., a high-capacity, high-performance hard disk, at the network file server, allowing the other nodes, or PC workstations, on the network to function with limited or no mass data storage capability.

As Local Area Networks (LANs) have become more common in the office and in industry, some clear market development trends have emerged. Possibly the most significant development in the LAN marketplace is the concern for cost reduction. This need is driven by intense competition between network vendors for market share. Today's LAN marketplace requires low-cost, simple network solutions that do not sacrifice performance. Another significant development in the LAN marketplace is the acceptance of Ethernet, or a derivative (e.g., Cheapernet or Twisted Pair Ethernet), as the industry standard for high-performance LANs. Because of Ethernet's popularity, there is a great need for cost reduction in this market.

Personal computers (PCs) have also seen significant changes over the past several years. PCs have become firmly entrenched in the office. Their popularity, coupled with a highly competitive market, has compelled PC vendors to both reduce costs for their LAN solutions and to attempt to distinguish their product from the competition's. The means of this cost reduction range from eliminating expensive hardware, such as disk drives and their associated hardware, to using highly integrated VLSI devices that implement the functions of a PC in a combination chip set containing several devices. Differentiation has been achieved by integrating peripheral functions, normally contained on an external adapter card, into the main processor board, or motherboard, of the PC. Video Graphics Array (VGA) and LAN connections are examples of this strategy.

The Intel 82592 LAN controller is uniquely suited for integration into a PC AT style motherboard. It meets the demands of today's market by providing the PC vendor (1) a means of reducing cost while maintaining high performance, and (2) a path for differentiation. An 82592 integrated into a PC motherboard provides a very low cost and very simple implementation because it uses the host system's existing resources to complete the LAN solution; e.g., system memory and DMA. This leaves the 82592, the serial interface, and some control logic as the only components required to complete a mother-board LAN solution.

1.1 Objective

This Application Note presents the general concept of integrating a Local Area Networking into a PC motherboard, and how the 82592 suits this purpose. The design of the 82592 Embedded LAN Module, which plugs into an Intel SYP301 motherboard (or any standard PC AT style motherboard), is explained in detail—providing a demonstration of an integrated Ethernet LAN solution.

1.2 Acknowledgements

For their contributions to this Application Note, and for their work in developing the architecture of the 82592 Embedded LAN Module, I would like to acknowledge, and thank, Uri Elzur, Dan Gavish, and Haim Sadger, of the Intel Israel System Validation group; and Joe Dragony, of Intel's (Folsom) Data Communications Focus Group.

2.0 THE EVOLUTION OF LAN SOLUTION ARCHITECTURES

LAN solutions have undergone an evolution in architecture—from expensive and complex to more cost-efficient and streamlined. A definite trend in office networking can be seen, as these solutions permit the host system to perform functions that were previously included in the LAN solution.

The first LAN solutions were usually intelligent buffered adapter cards, with a CPU, large memory requirements (up to 512 kBytes), firmware, a LAN controller, and a serial interface. As networking became more prevalent in the office environment—linking PCs and workstations via Ethernet—this complex architecture evolved into simpler and more streamlined nonintelligent, buffered adapters. In this architecture the CPU is no longer part of the LAN solution; its processing power is supplied by the host system. This architecture does not need memory to support a local CPU. Memory is only needed to supply a buffer space to store data before moving it to system memory or onto the serial link. The memory requirement for nonintelligent, buffered architectures is typically 8 kBytes to 32 kBytes. The firmware to boot the CPU is also no longer needed. The evolution to a nonintelligent, buffered architecture has resulted in significant cost savings and reduced complexity.
Significant increases in speed and processing power have been made to PCs during the past several years. This trend to higher performance host systems has allowed further streamlining of the LAN solution's architecture, resulting in even greater cost reduction and simplification. This is accomplished by using host system resources whenever possible. A nonintelligent, non-buffered architecture is the result. In this architecture, the host system's memory and DMA are used by the LAN controller. The complexity associated with buffered LAN solutions (e.g., supplying a dual-port arbitration scheme for local memory access by both the CPU and the LAN controller) is reduced; this complexity is removed from the LAN solution and returned to the host system, which is designed for these complex tasks. The result of this architectural optimization is a very simple, low component count, cost-efficient solution for a LAN connection. The 82592 Embedded LAN Module is the realization of this optimization. The trend to optimization of LAN architectures is shown in Figure 1.

![Figure 1. Architectural Optimization of LAN Solutions](image-url)
3.0 THE 82592 LAN CONTROLLER

3.1 General Features

The 82592 is a second generation, CMOS, advanced CSMA/CD LAN controller with a 16-bit data path. Along with its 8-bit version, the 82590, it is the follow-on design to the 82588 LAN controller. The 82592 is upwards software compatible from the 82588. The 82592 has two modes of serial operation, High Speed Mode and High Integration Mode. In High Speed Mode (up to 20 Mb/s) the 82592 couples with the Intel 82C501 to provide an all CMOS kit for IEEE 802.3 Ethernet applications. In this mode the 82592 can also serve as the controller for Twisted Pair Ethernet (TPE) applications. In High Integration Mode (up to 4 Mb/s) the 82592 performs Manchester and NRZI encoding/decoding, collision detection, transmit clocking, and receive clock recovery on chip; in this mode it can serve as a controller for StarLAN and other midrange LANs.

The 82592 provides several features that allow an efficient system interface to a wide variety of Intel microprocessors (e.g., iAPX 188, 186, 286, and 386) and industry standard buses (e.g., the IBM PC I/O channel or the PS/2 Micro Channel™). To issue a command to the 82592 (e.g., TRANSMIT or CONFIGURE) the CPU only needs to set up a block in memory that contains the parameters to be transferred to the 82592, program the DMA controller to point to that location and issue the proper opcode to the 82592. The 82592 and DMA controller perform the functions needed to complete the command, with the 82592 interrupting the CPU when the command is complete. The 82592 has a high-performance, 16-bit bus interface, operating at up to 16 MHz. It also implements a specialized hardware handshake with industry standard DMA controllers (e.g., the Intel 8237, 82380, and 82370) or the Intel 82560. This allows for back-to-back frame reception, and automatic retransmission on collision—without CPU intervention. The 82592 FIFOs (Rx and Tx) can have their 64 bytes divided into combinations of 32/32, 16/48, 48/16, or 16/16.

The 82592 features a Deterministic Collision Resolution (DCR) mode. When a collision is detected while in this mode, all nodes in a deterministic network enter into a time-division-multiplexed algorithm where each node has its own unique slot in which to transmit. This ensures that the collision is resolved within a calculated worst-case time. The 82592 also features a number of network management and diagnostic capabilities; for example,

- Monitor mode
- A 24-bit timer
- Three 16-bit event counters

• Internal and external loopback
• Internal register dump
• A TDR mechanism
• Internal diagnostics

For further information on the 82592, please refer to the Intel Microcommunications Handbook.

3.2 Unique Features for Embedded LAN Applications

The 82592 has several unique features that enable implementing a high-performance embedded LAN solution with minimal cost and complexity.

Peripherals on a motherboard must compete for access to the system bus. Because there is no local buffer for intermediate buffering of data, data transfers take place in real-time over the system bus to the system memory. A LAN controller must have a large internal data storage area to be able to wait for access to the system bus while serial data is being received or transmitted. Without sufficient internal data storage, a LAN controller cannot take advantage of the cost efficiency and simplicity of a non-buffered architecture. The 82592 has a total of 64 bytes of FIFOs. This expanded FIFO section allows the 82592 to tolerate long system bus latencies. For example, during a Receive (with the Rx FIFO length configured to 48 bytes) the 82592 can tolerate up to 38.4 µs of bus latency—the time from a DMA request to reception of a DMA Acknowledge from the DMA controller—before the possibility of a data overrun occurring in a 10 Mb/s Ethernet application. Once access to the system bus has been obtained, the 82592's high-performance, 16-bit bus interface provides efficient data transfer over the system bus, thus reducing the bus utilization load for a LAN connection on the host system.

The 82592 features a specialized hardware handshake with industry standard DMA controllers. This hardware handshake between the 82592 and the DMA controller (on signal lines DRQ and EOP) relays the status of a Receive or Transmit and allows for back-to-back frame reception and automatic retransmission on collision without CPU intervention. This allows the 82592 and the DMA controller to perform these time-critical operations in real-time without depending on the CPU via an interrupt service routine, and without the time delays inherent in such routines. For the 82592 Embedded LAN Module, this hardware handshake is enabled by configuring the 82592 to the Tightly Coupled Interface (TCI) mode. Figure 2 shows details of the 82592's TCI signals.
Transmit/Receive Status Encoding on DRQ and EOP

<table>
<thead>
<tr>
<th>DRQ</th>
<th>EOP</th>
<th>Status Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Hi-Z</td>
<td>Idle</td>
</tr>
<tr>
<td>1</td>
<td>Hi-Z</td>
<td>DMA Transfer</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>Transmission or Reception Terminated OK</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>Transmission or Reception Aborted</td>
</tr>
</tbody>
</table>

Tightly Coupled Interface Timings

**Table:**

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Parameter Description</th>
<th>Min</th>
<th>Max</th>
<th>Units</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>t23</td>
<td>WR or RD Low to DRQ0 or DRQ1 Inactive</td>
<td>45</td>
<td>ns</td>
<td></td>
<td>C&lt;sub&gt;L&lt;/sub&gt; = 50 pF</td>
</tr>
<tr>
<td>t104</td>
<td>WR or RD High to DRQ0 or DRQ1 Inactive</td>
<td>2.5</td>
<td>65</td>
<td>ns</td>
<td>C&lt;sub&gt;L&lt;/sub&gt; = 50 pF</td>
</tr>
<tr>
<td>t105</td>
<td>WR or RD Low to EOP Active</td>
<td>45</td>
<td>ns</td>
<td>Open Drain I/O Pin</td>
<td></td>
</tr>
<tr>
<td>t106</td>
<td>EOP Float after DACK0 or DACK1 Inactive</td>
<td>40</td>
<td>ns</td>
<td>Open Drain I/O Pin</td>
<td></td>
</tr>
</tbody>
</table>

**Figure 2.** TCI Encoding and Timings
These three features (FIFO depth, high-performance bus interface, and TCI) allow the 82592 to operate successfully in a high-performance motherboard LAN application. The application of these features will be discussed further in Section 4.

4.0 SYP301 INTERFACE

This section will discuss the details of the interface of the 82592 Embedded LAN Module to the Intel SYP301. The basic architecture will be presented, demonstrating that the 82592 Embedded LAN Module is a low-cost, low component count Ethernet solution for networking office PCs or workstations.

The Intel SYP301 is compatible with the IBM PC AT™. It features an Intel 80386™ microprocessor, running at 16 MHz, as its CPU. Its system bus is compatible with the standard PC AT I/O-channel bus.

4.1 Basic Architecture

Figure 3 shows the basic architecture of the 82592 Embedded LAN Module, and Figure 4 shows the module's schematics. The module consists of an 82592, two 20L10 PALs, and two 8-bit LS573 address latches that combine to provide a 16-bit address latch. The module contains no DMA unit or local memory.

The 82592 Embedded LAN Module is a simple, low-cost, low component count solution because it uses the available system resources (DMA and memory) to provide for those functions normally added to a LAN solution. Removing DMA and local memory from a LAN solution reduces cost and complexity. Two host DMA channels, one for receive and one for transmit, are needed to support the module. The DMA interface from the 82592 (through PAL B) is the standard combination of DRQ, DACK and EOP. These three signals also provide the TCI between the 82592 and the DMA controller. The size of the memory buffer needed to support the module depends on the specific application and the amount of free memory available; the buffer size can be specified by the programmer.
The two PALs (PAL A and B) provide two major junctions for the module: (1) address decode (PAL A), and (2) interpreting the TCI from the 82592 (PAL B). PAL A decodes addresses for CS to the 82592, OE for the address latches, and an Enable/Disable of the LAN module. PAL B interprets the TCI of the 82592. When PAL B detects EOP from the 82592 during reception of a frame (EOP indicates the last byte of the receive frame) it loads the memory address of the last byte of the receive frame (the byte count) into the Address Latch at the time it is written into memory. This allows back-to-back frame reception without CPU intervention, and will be covered in detail in Section 4.2. For Auto-Retransmit on collision, PAL B passes the EOP signal from the 82592 to the DMA controller, reinitializing the DMA controller for retransmission. This process will be discussed in more detail in Section 4.3. Both sets of PAL equations are listed in Table 1.

**Table 1. PAL Equations**

**PAL20L10 MMI—PAL A (Version 1.1)**

<table>
<thead>
<tr>
<th>AEN</th>
<th>A2</th>
<th>NC</th>
<th>AO</th>
<th>IOWBAR</th>
<th>A5</th>
<th>A6</th>
<th>A7</th>
<th>A9</th>
<th>GND</th>
<th>IOWBAR</th>
<th>592CTS</th>
<th>OE2BAR</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>501LB</td>
<td>592CTS</td>
<td></td>
</tr>
</tbody>
</table>

\[
\text{IF (VCC) } 501LB = 592CTS
\]

**PAL20L10 MMI—PAL B (Version 1.1)**

| 592DRQO | RESET | DACK7BAR | DACK6BAR | 10RBAR | 592DRQ1 | 592E0PBAR | ENLANBAR | AEN | NC | IOWBAR | GND | 592INT | NC | DRQ6BAR | DRQ7 | DISDACK | IRQ10 | NC | MSEOBAR | LTCW | 592DACKBAR | VCC |
|---------|-------|----------|----------|--------|---------|-----------|----------|-----|----|--------|-----|--------|----|----------|------|----|----------|------|------------|-----|
|         |       |          |          |        |         |           |          |     |   |        |     |         |    |          |      |   |          |      |            |     |

\[
\text{IF (VCC) } \overline{\text{LTCW}} = \overline{\text{IORBAR}} + \overline{592E0PBAR} + \overline{\text{DACK7BAR}}
\]

**NOTE:**

The suffix BAR added to the above signal names indicates an active low signal. A signal in these equations with a line drawn above it indicates this signal is to be in a low state for the equation.
4.2 Back-to-Back Frame Reception

The architecture of the 82592 Embedded LAN Module allows it to receive back-to-back frames without CPU intervention. It uses a contiguous Receive Frame Area (RFA) buffer in host system memory where receive frames can be continuously stored. This sequential storage of receive frames can continue until the buffer space is exhausted. The size of the RFA buffer can be specified by the programmer. Its size will be programmed as the byte count of the Rx DMA channel. The Base Address Register contents of that channel serve as the start address of the RFA buffer. The receive frames will be stored sequentially in the RFA buffer based on the contents of the Current Address Register of the Rx DMA channel. The module's architecture, and the 82592 receive frame memory structure, allows the CPU to recover the addresses of each Receive frame in memory for processing. The CPU can also reinitialize the RFA buffer (by reinitializing the Rx DMA channel) as the RFA buffer fills up and its contents are processed. Alternatively, configuring the Rx DMA channel to Auto-Initialize mode will allow the Rx buffer to automatically wrap around, back to the beginning of the buffer, when its end is reached. This creates a virtual "endless" circular buffer. When using this approach, care must be taken to avoid writing over unprocessed Rx frames—either by the addition of a hardware Stop Register, or by guaranteeing that the Rx frames can be processed faster than the buffer can wrap around.

Back-to-back frame reception without CPU intervention—and eventual recovery of the frames for processing by the CPU—is based on PAL B's decoding of the TCI signals of the 82592 (PAL B loads the address latch with the address of the last byte of the received frame) and the structure of the received frame transferred from the 82592 to memory. Figure 5 shows the format of an 82592 receive frame in TCI mode. After the information fields are written to memory, the Status and byte count of the received frame are appended to the frame in memory. These four bytes (two bytes of Status and two bytes of byte count) are the last four bytes of the receive frame written to memory. The high byte of the byte count is the last byte transferred from the 82592 to memory. As this last byte is transferred to memory, the 82592 asserts the EOP signal. When PAL B detects the assertion of EOP by the 82592, it loads the address of the last byte of the receive frame into the Address Latch as this byte is written into memory. This action ensures that there will always be a pointer (the contents of the Address Latch) to the byte count of the last frame stored in the RFA buffer in system memory. Based on the value of the byte count, the beginning address of the receive frame in memory can be calculated; i.e., Byte Count Address Pointer = Byte Count + Beginning of Frame. The byte count of a previous receive frame would reside one address location before the first byte of the current receive frame. That frame, and any additional receive frames that may have preceded it, can have their start addresses recovered by the same calculation used to recover the last frame received. This process allows frames to be continually stored in the RFA buffer without CPU intervention, and to be recovered by the CPU for processing. Figure 6 illustrates the process of back-to-back frame reception.

---

### Figure 5. Receive Format for the 82592 in 16-Bit Mode (Tightly Coupled Interface Enabled)

<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>DESTINATION ADDRESS SECOND BYTE</td>
<td>DESTINATION ADDRESS FIRST BYTE</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SOURCE ADDRESS LAST BYTE</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SOURCE ADDRESS LAST BYTE</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>INFORMATION (LENGTH FIELD, HIGH)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>INFORMATION (LENGTH FIELD, LOW)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>INFORMATION LAST BYTE</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CRC BYTE 1*</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CRC BYTE 0*</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CRC BYTE 3*</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CRC BYTE 2*</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>SHORT FRAME</td>
<td>NO EOF</td>
<td>TOO LONG</td>
<td>1</td>
<td>NO SFD</td>
<td>NO ADD MATCH</td>
<td>I-A MATCH</td>
<td>Rx CLD</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>0</td>
<td>0</td>
<td>Rx</td>
<td>LEN ERR</td>
<td>CRC ERROR</td>
<td>ALG ERROR</td>
<td>0</td>
</tr>
<tr>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>BYTE COUNT LOW</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BYTE COUNT HIGH</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

*The CRC bytes are transferred to memory only when the device is so configured.*
**NOTES:**
The 82592 appends the byte count to the end of each RCV frame. 
PAL 'B' loads the latch with the memory address of the last byte of each RCV frame. 
Based on latch contents and the byte count of each frame, the CPU recovers the RCV frames.

**Figure 6. Back-to-Back Frame Reception**

### 4.3 Automatic Retransmission on Collision

Automatic retransmission on collision detection is accomplished by the TCI between the 82592 and the host 8237 DMA controller and requires no CPU intervention. The transmit channel of the 8237 should be configured for Auto-Initialize mode. The transmit block (data to be transmitted) starts at the location pointed to by the Base Address Register of the Tx DMA channel. During a Transmit command, the 82592 DMA requests begin at the start of the transmit block and work sequentially through the block (by incrementing the contents of the 8237's Current Address Register) until the transmission is complete. Should a collision occur, the 82592 asserts the EOP signal and DRQ* to the 8237 (these signals pass through PAL B) causing the 8237 to auto-initialize back to the beginning of the transmit block (the Current Address Register is loaded with the value in the Base Address Register). Internally, the 82592 generates a Retransmit command and begins making DMA requests to the 8237, which is now pointing to the beginning of the transmit block. The 82592 also enters into a back-off algorithm (counting to a random number to resolve the collision). When the back-off algorithm is complete, and the 82592 regains access to the serial link, retransmission is attempted. The 82592 will repeat this process until the retransmission is completed successfully or until the maximum allowable number of collisions per Transmit command is reached—at that point all retransmit attempts stop. No CPU involvement is required to carry out a retransmission. The process of automatic retransmission is shown in Figure 7.

**NOTE:**
*For Auto-Initialization of the 8237, the signal DRQ must be asserted to the 8237 along with assertion of EOP. With the 82380 and 82370 DMA controllers, Auto-Initialization can be triggered by asserting the EOP signal alone.*
4.4 Target Systems for Integration

The 82592 Embedded LAN Module is designed to be implemented on an Intel SYP301 motherboard; thereby demonstrating a low-cost LAN connection for a workstation. The SYP301 has an IBM PC AT style bus architecture with a 32-bit Intel 80386 as the main processor. The interface between the 82592 LAN Module and the SYP301 is based on standard interface signals (DRQ, DACK, EOP, IRQ, IOR, IOW, etc.) so the basic architecture of the module can be implemented on PC AT based systems. This design has been successfully tested in PC AT style systems produced by several manufacturers. For some PC AT based systems, and PS/2 Micro Channel systems, the module’s design may require some modification. IBM PC and PC XT based systems do not have sufficient DMA bandwidth to support the non-buffered architecture of this module.

4.4.1 PC AT BASED DESIGNS

High-integration chip sets replace a large number of discrete VLSI, LSI, and TTL components with several integrated VLSI devices that duplicate a large portion of the PC’s functionality. PC AT compatible systems using such chip sets may lack support for the automatic retransmission feature of the 82592 LAN Module. This is because many manufacturers of such chip sets have integrated the EOP function but eliminated the EOP input. This lack of an EOP input disables auto-initialization of the DMA controller for retransmission. In this case retransmission can be performed in one of two ways.

- Should a collision occur while transmitting the preamble, the 82592 (when configured to automatic retransmission mode) will automatically retransmit without CPU intervention or auto-initialization of the DMA. This is effective for shorter network topologies where collisions are normally detected early in the frame.

- Should a collision occur after the preamble, the 82592 will interrupt the CPU and the CPU will initiate the retransmission.

For a PC AT style architecture, logic must be implemented to accommodate DRAM refresh. DRAM refresh cycles typically occur at 15 μs intervals. In a standard PC AT, any DMA user should limit the time of a DMA burst to 15 μs; this is to ensure that the system bus is free for the refresh to take place. Any designer using burst mode DMA must consider this requirement when implementing a design.

4.4.2 PS/2 MICRO CHANNEL ARCHITECTURE DESIGNS

The IBM PS/2 and other compatibles using the Micro Channel architecture have a different host interface to the 82592 Embedded LAN Module; however, the basic architecture of the module is still applicable. As in the SYP301 solution, the TCI between the 82592 and a
control PAL loads the address latch with a pointer to the last receive frame. Based on the contents of the latch and the 82592 receive memory structure, the frames are recovered for processing by the CPU. The differences between a PC AT architecture and a Micro Channel architecture require different control signal decoding. The Micro Channel requires a 24-bit address latch, as opposed to a 16-bit latch in the 301, and to acquire the system DMA it requires different arbitration logic to drive a 4-bit arbitration level on the Micro Channel. The Micro Channel also does not have an EOP input; therefore, auto-initialization of the Tx DMA channel and support of automatic retransmission without CPU intervention must be provided by using one of the alternative methods recommended in the previous section.

4.4.3 EMBEDDED CONTROL DESIGNS

The 82592 Embedded LAN Module architecture can also be applied to an embedded control application that contains some DMA functions. For an embedded application using an 8237, 82380 or 82370 DMA controller, the basic architecture of the 82592 Embedded LAN Module can be used. For an interface to DMA devices that do not feature the EOP signal as an input (for example, DMA units on board a CPU), the alternative methods for retransmission given earlier can be used.

5.0 SERIAL INTERFACE MODULE

The serial interface for the Intel SYP301 82592 Embedded LAN Module is implemented as a separate module. Since the 82592 Embedded LAN Module is intended to be integrated into a system motherboard, implementing the serial interface as a separate module—perhaps as a very small PC board that plugs into a socket—allows for easy interchangeability between different serial interface media. This modularity allows the system board manufacturer to avoid committing his motherboard to only one type of medium, and thus requiring a major redesign for each different serial interface.

Modularity in the data communications field is encouraged by the Open Systems Interconnect (OSI) reference model. The 82592 is designed to operate through the lower half of the Data Link Layer (see Figure 8), implementing CSMA/CD Medium Access Control and interfacing directly with the Physical layer below it. By interfacing the 82592's standard CSMA/CD interface signals to a serial module (TxD, RxD, TxC, RxC, CDT, CRS, and others) different Physical Link modules can be implemented without any change to software. Examples of serial interface modules that could be interchanged by simply plugging a new module into the motherboard are Ethernet/Cheapernet, Twisted Pair Ethernet (TPE), StarLAN, Broadband Ethernet, and many proprietary CSMA serial media. Figure 9 shows the schematics of an Ethernet module; and Figure 10 those of an Ethernet/Cheapernet module.
6.0 PERFORMANCE COMPARISON

Figure 11 compares the performance of the 82592 Embedded LAN Module with the PC586E nonintelligent, buffered adapter. The PC586E is an Intel evaluation board based on the Intel 82586 LAN Coprocessor. It contains 16 kbytes of local memory, has a 16-bit bus interface, and has a high-performance arbitration scheme providing both the CPU and the 82586 LAN controller zero wait state access to local memory. The PC586E has been characterized in the industry as one of the highest performance nonintelligent, buffered adapters available.

A performance comparison, using Novell's Perform 2 utility, shows that the 82592 Embedded LAN Module, operating as a workstation accessing a file server, outperforms the PC586E. For all tests the host system was an Intel SYP301. The SYP301 was run in both standard mode, a nominal 16 MHz, and in its reduced speed mode, 6 MHz. In all cases the SYP301 system DMA operates at 4 clocks per cycle at 4 MHz. The file server was a Novell 286A, an 8 MHz, zero wait state system, using a PC586E as the LAN adapter. The tests recorded are for one node on the network (the workstation under test). For write tests to the file server's hard disk, the performance numbers are generally the same. This is due to limitations in accessing the file server's hard disk. This slow access causes a bottleneck. For the read tests the workstations are accessing files stored in cache memory, thus removing the bottleneck for this test. Without this limitation, the 82592 Embedded LAN Module accesses the file server at a higher rate than the PC586E: at full speed, 318 kbytes/s vs 282.3 kbytes/s; and at reduced speed, 202.8 kbytes/s vs 195.2 kbytes/s.

7.0 SOFTWARE EXAMPLES

The following examples are from a driver written for an 82592 Embedded LAN Module operating in an Intel SYP301. The driver was written by Joe Dragony, Intel Data Communications Technical Marketing Engineer. The excerpts will cover (1) declarations of program constants and variables, (2) initializing the Embedded LAN Module hardware and buffer space, (3) assembly and transmission of a frame, and (4) processing received frames. A brief description of each of these processes is followed by excerpts from the code. The driver uses the Xerox Internetwork Packet Exchange (IPX) protocol and serves as a software interface between the 82592 Embedded LAN Module hardware and the IPX.

Exerciser Software for the 82592 Embedded LAN Module is also available from Intel. Detailed documentation for both the exerciser program and the network driver are available upon request from Intel.

7.1 Declarations

Table 2 shows declarations of program variables and equates of program constants. This section is included to help the reader understand the following program excerpts.

*NOTE:
The benchmark program Landmark CPU Speed Test, © 1986 by Landmark Software, shows an effective throughput of 14.3 MHz for a SYP301 in standard mode; and 5.4 MHz in reduced speed mode.
Table 2. Declarations

```assembly
; External declarations

name LANOMotherboardModule
assume cs: CGroup, ds: CGroup

Code segment word public 'CODE'
public DriverSendPacket
public DriverBroadcastPacket
public DriverPoll
public LANOptionName
extrn IPXGetECB: NEAR
extrn IPXReturnECB: NEAR
extrn IPXReceivePacket: NEAR
extrn IPXReceivePacketEnabled: NEAR
extrn IPXHoldEvent: NEAR
extrn IPXServiceEvents: NEAR
extrn IPXIntervalMarker: word
extrn MaxPhysPacketSize: word
extrn ReadWriteCycles: byte
extrn IPXStartCriticalSection: NEAR
extrn IPXEndCriticalSection: NEAR
```

290189-16
### Table 2. Declarations (Continued)

| Equates |  |
| CR       | equ 0Dh |
| LF       | equ 0Ah |
| BAD      | equ OFFh |
| BPORT    | equ 0 |
| IRQLOC   | equ 19 |
| DMA0LOC  | equ 23 |
| DMA1LOC  | equ 25 |
| TransmitHardwareFailure | equ OFFh |
| PacketUndeliverable     | equ 0FEh |
| PacketOverflow          | equ 0F7h |
| ECBProcessing          | equ 0FAh |
| TxTimeOutTicks         | equ 20 |

; Latch definitions
TanCantLo  equ 301h
TanCantHi  equ 302h

; Enables for 10cent
EnLAN      equ 303h
DisLAN     equ 304h

; 8259 definitions
InterruptControlPort equ 020h
InterruptMaskPort    equ 0A1h ; for secondary 8259A
ExtraInterruptControlPort equ 0A0h
EOI       equ 020h

; 8237 definitions
DMAcmdstat equ 0D0h
DMAreq    equ 0D2h
DMAangmask equ 0D4h
DMAmode   equ 0D6h
DMAeff    equ 0D8h
DMAtempolr equ 0DAh
DMAclmask equ 0DCh
DMAallmask equ 0DEh
DMA@gpage equ 089h
DMA@gaddr equ 0C8h
DMA@gwcount equ 0CAh
DMA@gpage equ 08Ah
DMA7@gaddr equ 0CCh
DMA7@gwcount equ 0CEh
DMA@x6    equ 01Ah ; demand mode, autoinit, read transfer
DMA@x7    equ 01Bh ; demand mode, autoinit, read transfer
DMA@x6    equ 006h ; demand mode, no autoinit, write transfer
DMA@x7    equ 007h ; demand mode, no autoinit, write transfer
DMA@mask  equ 006h
DMA@unmask equ 002h
DMA7mask  equ 007h
DMA7unmask equ 003h
DMA@ena   equ 0h

290189-17
### Table 2. Declarations (Continued)

<table>
<thead>
<tr>
<th>Declaration</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>NetWareType</td>
<td>1111h</td>
</tr>
<tr>
<td>; 82592 Commands</td>
<td></td>
</tr>
<tr>
<td>C_NOP</td>
<td>00h</td>
</tr>
<tr>
<td>C_SW1</td>
<td>10h</td>
</tr>
<tr>
<td>C_SELAST</td>
<td>0Fh</td>
</tr>
<tr>
<td>C_SW0</td>
<td>01h</td>
</tr>
<tr>
<td>C_IASET</td>
<td>01h</td>
</tr>
<tr>
<td>C_CONFIG</td>
<td>02h</td>
</tr>
<tr>
<td>C_MCSET</td>
<td>03h</td>
</tr>
<tr>
<td>C_TX</td>
<td>04h</td>
</tr>
<tr>
<td>C_TDR</td>
<td>05h</td>
</tr>
<tr>
<td>C_DUMP</td>
<td>16h</td>
</tr>
<tr>
<td>C_DIAG</td>
<td>07h</td>
</tr>
<tr>
<td>C_RXENB</td>
<td>18h</td>
</tr>
<tr>
<td>C_ALZNBUF</td>
<td>09h</td>
</tr>
<tr>
<td>C_RXDISR</td>
<td>1Ah</td>
</tr>
<tr>
<td>C_STPRX</td>
<td>18h</td>
</tr>
<tr>
<td>C_RETX</td>
<td>0Ch</td>
</tr>
<tr>
<td>C_ABORT</td>
<td>0Dh</td>
</tr>
<tr>
<td>C_RST</td>
<td>0Eh</td>
</tr>
<tr>
<td>C_RLSPTR</td>
<td>0Fh</td>
</tr>
<tr>
<td>C_FIXPTR</td>
<td>1Fh</td>
</tr>
<tr>
<td>C_INTACK</td>
<td>80h</td>
</tr>
</tbody>
</table>

```plaintext
::: Data Structures
::: even hardware_structure struc
    io_addr1 dw ?
    io_range1 dw ?
    io_addr2 dw ?
    decode_range2 dw ?
    mem_addr1 dw ?
    mem_range1 dw ?
    mem_addr2 dw ?
    mem_range2 dw ?
    int_used1 db ?
    int_line1 db ?
    int_used2 db ?
    int_line2 db ?
    dma_used1 db ?
    dma_chan1 db ?
    dma_used2 db ?
    dma_chan2 db ?

hardware_structure ends

::: ech_structure struc
    link dd 0
    esr_address dd 0
    in_use db 0
    completion_code db 0
```
Table 2. Declarations (Continued)

<table>
<thead>
<tr>
<th>Variable</th>
<th>Type</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>socket_number</td>
<td>dw</td>
<td>0</td>
</tr>
<tr>
<td>ipx_workspace</td>
<td>db</td>
<td>4 dup (0)</td>
</tr>
<tr>
<td>driver_workspace</td>
<td>db</td>
<td>12 dup (0)</td>
</tr>
<tr>
<td>immediate_address</td>
<td>db</td>
<td>6 dup (0)</td>
</tr>
<tr>
<td>fragment_count</td>
<td>dw</td>
<td>1</td>
</tr>
<tr>
<td>fragment_descriptor_list</td>
<td>dw</td>
<td>6 dup (?)</td>
</tr>
<tr>
<td>fragment_descriptor_struc</td>
<td>dd</td>
<td>?</td>
</tr>
<tr>
<td>fragment_address</td>
<td>dd</td>
<td>?</td>
</tr>
<tr>
<td>fragment_length</td>
<td>dw</td>
<td>?</td>
</tr>
<tr>
<td>fragment_descriptor_struc</td>
<td>ends</td>
<td></td>
</tr>
<tr>
<td>rx_buf_structure_struc</td>
<td></td>
<td></td>
</tr>
<tr>
<td>rx_dest_addr</td>
<td>db</td>
<td>6 dup (?)</td>
</tr>
<tr>
<td>rx_source_addr</td>
<td>db</td>
<td>6 dup (?)</td>
</tr>
<tr>
<td>rx_physical_length</td>
<td>dw</td>
<td>?</td>
</tr>
<tr>
<td>rx_checksum</td>
<td>dw</td>
<td>?</td>
</tr>
<tr>
<td>rx_length</td>
<td>dw</td>
<td>?</td>
</tr>
<tr>
<td>rx_trans_control</td>
<td>db</td>
<td>?</td>
</tr>
<tr>
<td>rx_hdr_type</td>
<td>db</td>
<td>?</td>
</tr>
<tr>
<td>rx_dest_net</td>
<td>db</td>
<td>4 dup (?)</td>
</tr>
<tr>
<td>rx_dest_node</td>
<td>db</td>
<td>6 dup (?)</td>
</tr>
<tr>
<td>rx_dest_socket</td>
<td>dw</td>
<td>?</td>
</tr>
<tr>
<td>rx_source_net</td>
<td>db</td>
<td>4 dup (?)</td>
</tr>
<tr>
<td>rx_source_node</td>
<td>db</td>
<td>6 dup (?)</td>
</tr>
<tr>
<td>rx_source_socket</td>
<td>dw</td>
<td>?</td>
</tr>
<tr>
<td>rx_buf_structure_struc</td>
<td>ends</td>
<td></td>
</tr>
<tr>
<td>tci_status_struc</td>
<td></td>
<td></td>
</tr>
<tr>
<td>status0</td>
<td>db</td>
<td>?</td>
</tr>
<tr>
<td>dead1</td>
<td>db</td>
<td>?</td>
</tr>
<tr>
<td>status1</td>
<td>db</td>
<td>?</td>
</tr>
<tr>
<td>dead2</td>
<td>db</td>
<td>?</td>
</tr>
<tr>
<td>bc_lo</td>
<td>db</td>
<td>?</td>
</tr>
<tr>
<td>dead3</td>
<td>db</td>
<td>?</td>
</tr>
<tr>
<td>bc_hi</td>
<td>db</td>
<td>?</td>
</tr>
<tr>
<td>tci_status_struc</td>
<td>ends</td>
<td></td>
</tr>
<tr>
<td>ipx_header_structure_struc</td>
<td></td>
<td></td>
</tr>
<tr>
<td>checksum</td>
<td>dw</td>
<td>?</td>
</tr>
<tr>
<td>packet_length</td>
<td>dw</td>
<td>?</td>
</tr>
<tr>
<td>transport_control</td>
<td>db</td>
<td>?</td>
</tr>
<tr>
<td>packet_type</td>
<td>db</td>
<td>?</td>
</tr>
<tr>
<td>destination_network</td>
<td>db</td>
<td>4 dup (?)</td>
</tr>
<tr>
<td>destination_node</td>
<td>db</td>
<td>6 dup (?)</td>
</tr>
<tr>
<td>destination_socket</td>
<td>dw</td>
<td>?</td>
</tr>
<tr>
<td>source_network</td>
<td>db</td>
<td>4 dup (?)</td>
</tr>
<tr>
<td>source_node</td>
<td>db</td>
<td>6 dup (?)</td>
</tr>
<tr>
<td>source_socket</td>
<td>dw</td>
<td>?</td>
</tr>
<tr>
<td>ipx_header_structure_struc</td>
<td>ends</td>
<td></td>
</tr>
</tbody>
</table>

; Variables

even
### Table 2. Declarations (Continued)

<table>
<thead>
<tr>
<th>Declaration</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>tx_start_time dw 0</td>
<td>Time at which transmission started</td>
</tr>
<tr>
<td>adapter_io dw ?</td>
<td>Adapter I/O register</td>
</tr>
<tr>
<td>config dw ?</td>
<td>Configuration register</td>
</tr>
<tr>
<td>send_list dd 0</td>
<td>Points to list of ECBs to be sent</td>
</tr>
<tr>
<td>buffer_segment dw ?</td>
<td>Buffer segment address</td>
</tr>
<tr>
<td>rx_ecb dd ?</td>
<td>Receiver ECB pointer</td>
</tr>
<tr>
<td>tx_ecb dd ?</td>
<td>Transmitter ECB pointer</td>
</tr>
<tr>
<td>config_block db</td>
<td>Configuration block</td>
</tr>
<tr>
<td>0Fh, 00h, 48h, 08h, 26h, 00h, 00h, 00h, 0F2h, 00h, 00h, 40h, 0F5h, 00h, 3Fh, 87h, 0F0h, 0DFh</td>
<td></td>
</tr>
<tr>
<td>temp_flag db</td>
<td>Temporary flag</td>
</tr>
<tr>
<td>int_mask dw ?</td>
<td>Integer mask</td>
</tr>
<tr>
<td>int_vector_addr dw ?</td>
<td>Integer vector address</td>
</tr>
<tr>
<td>int_bit db ?</td>
<td>Integer bit</td>
</tr>
<tr>
<td>int_mask db ?</td>
<td>Integer mask</td>
</tr>
<tr>
<td>command_reg dw 300h</td>
<td>Command register</td>
</tr>
<tr>
<td>read_in_length dw ?</td>
<td>Read in length</td>
</tr>
<tr>
<td>configdma0_loc db ?</td>
<td>Configuration DMA 0 location</td>
</tr>
<tr>
<td>config dma1_loc db ?</td>
<td>Configuration DMA 1 location</td>
</tr>
<tr>
<td>config_irq_loc db ?</td>
<td>Configuration IRQ location</td>
</tr>
<tr>
<td>config_Iport dw ?</td>
<td>Configuration Iport</td>
</tr>
<tr>
<td>tx_active_flag db 0</td>
<td>Transmission active flag</td>
</tr>
<tr>
<td>frame_status db 0</td>
<td>Frame status</td>
</tr>
<tr>
<td>status10 db 0</td>
<td>Status 10</td>
</tr>
<tr>
<td>status11 db 0</td>
<td>Status 11</td>
</tr>
<tr>
<td>status20 db 0</td>
<td>Status 20</td>
</tr>
<tr>
<td>status21 db 0</td>
<td>Status 21</td>
</tr>
<tr>
<td>even</td>
<td>Even</td>
</tr>
<tr>
<td>gp_buf dw 5000 dup (0)</td>
<td>General Purpose Buffer EA</td>
</tr>
<tr>
<td>gp_length dw 1388h</td>
<td>General Purpose Buffer length</td>
</tr>
<tr>
<td>gp_buf_offset dw gp_group:gp_buf</td>
<td>Offset of gp_buf in memory address</td>
</tr>
<tr>
<td>gp_buf_start dw 0</td>
<td>Start of General Purpose Buffer EA</td>
</tr>
<tr>
<td>gp_buf_page dw 0</td>
<td>Page of General Purpose Buffer EA</td>
</tr>
<tr>
<td>tx_byte_cnt dw 0</td>
<td>Transmit byte count</td>
</tr>
<tr>
<td>rx_buf_start dw 0</td>
<td>Start of General Purpose Buffer EA</td>
</tr>
<tr>
<td>rx_buf_page dw 0</td>
<td>Page of General Purpose Buffer EA</td>
</tr>
<tr>
<td>rx_buf_head dw 0</td>
<td>Head address of General Purpose Buffer EA</td>
</tr>
<tr>
<td>even</td>
<td>Even</td>
</tr>
<tr>
<td>rx_buf_tail dw 0</td>
<td>Tail address of General Purpose Buffer EA</td>
</tr>
<tr>
<td>rx_buf_ptr dw 0</td>
<td>Pointer to transmit buffer</td>
</tr>
<tr>
<td>rx_buf_stop dw 0</td>
<td>Stop address of General Purpose Buffer EA</td>
</tr>
<tr>
<td>rx_buf_length dw 0</td>
<td>Length of General Purpose Buffer EA</td>
</tr>
<tr>
<td>rx_buf_segment dw 0</td>
<td>Segment of General Purpose Buffer EA</td>
</tr>
<tr>
<td>curr_rx_length dw 0</td>
<td>Current length of General Purpose Buffer EA</td>
</tr>
<tr>
<td>rx_list dw 180 dup (0)</td>
<td>Rx list</td>
</tr>
<tr>
<td>num_of_frames dw 0</td>
<td>Number of frames</td>
</tr>
<tr>
<td>reset_rx_buf dw 0</td>
<td>Reset of transmit buffer</td>
</tr>
<tr>
<td>padding dw 0</td>
<td>Padding</td>
</tr>
<tr>
<td>;</td>
<td>Define Hardware Configuration</td>
</tr>
<tr>
<td>;</td>
<td></td>
</tr>
</tbody>
</table>
### Table 2. Declarations (Continued)

<table>
<thead>
<tr>
<th>ConfigurationID</th>
<th>db</th>
<th>'NetWareDriverLAN WS</th>
</tr>
</thead>
<tbody>
<tr>
<td>SDriverConfiguration</td>
<td>LABEL</td>
<td>byte</td>
</tr>
<tr>
<td>reserved1</td>
<td>db</td>
<td>4 dup (0)</td>
</tr>
<tr>
<td>node_addr</td>
<td>db</td>
<td>6 dup (0)</td>
</tr>
<tr>
<td>reserved2</td>
<td>db</td>
<td>0 ; non-zero means is a real driver.</td>
</tr>
<tr>
<td>node_addr_type</td>
<td>db</td>
<td>0 ; address is determined at initialization</td>
</tr>
<tr>
<td>max_data_size</td>
<td>dw</td>
<td>1024 ; largest read data request will handle</td>
</tr>
<tr>
<td>(512, 1024, 2048, 4096)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>lan_desc_offset</td>
<td>dw</td>
<td>LANO Option Name</td>
</tr>
<tr>
<td>lan_hardwa_id</td>
<td>db</td>
<td>000h ; Bogus Type Code</td>
</tr>
<tr>
<td>transport_time</td>
<td>dw</td>
<td>1 ; transport time</td>
</tr>
<tr>
<td>reserved_3</td>
<td>db</td>
<td>11 dup (0)</td>
</tr>
<tr>
<td>major_version</td>
<td>db</td>
<td>01h ; Bogus version number</td>
</tr>
<tr>
<td>minor_version</td>
<td>db</td>
<td>00h</td>
</tr>
<tr>
<td>flag Bits</td>
<td>db</td>
<td>0</td>
</tr>
<tr>
<td>selected_configuration</td>
<td>db</td>
<td>0 ; board configuration (interrupts, IO addresses, etc.)</td>
</tr>
<tr>
<td>number_of_configs</td>
<td>db</td>
<td>01</td>
</tr>
<tr>
<td>config_pointers</td>
<td>dw</td>
<td>configuration0</td>
</tr>
<tr>
<td>LANO Option Name</td>
<td>db</td>
<td>'Intel LAN-On-Motherboard Module', 0, '$'</td>
</tr>
<tr>
<td>configuration0</td>
<td>dw</td>
<td>300h, 16, 0, 0 ; IO ports and ranges</td>
</tr>
<tr>
<td>db</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>dw</td>
<td>0 , 0</td>
<td></td>
</tr>
<tr>
<td>db</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>dw</td>
<td>0 , 0 ; memory decode</td>
<td></td>
</tr>
<tr>
<td>db</td>
<td>OFFH, 10, 0, 0 ; interrupt level 10</td>
<td></td>
</tr>
<tr>
<td>db</td>
<td>OFFH, 6, OFFH, 7 ; DMA channels 6 and 7</td>
<td></td>
</tr>
<tr>
<td>db</td>
<td>0, 0</td>
<td></td>
</tr>
<tr>
<td>db</td>
<td>'IRQ 10, IO Addr = 300h, DMA 6 and 7, For Evaluation Only', 0</td>
<td></td>
</tr>
</tbody>
</table>

;*****************************************************************
| Error Counters |
| ;*****************************************************************

Public DriverDiagnosticTable LABEL, DriverDiagnosticText

<table>
<thead>
<tr>
<th>DriverDiagnosticTable</th>
<th>LABEL</th>
<th>byte</th>
</tr>
</thead>
<tbody>
<tr>
<td>DriverDebugCount</td>
<td>dw</td>
<td>DriverDebugEnd - DriverDiagnosticTable</td>
</tr>
<tr>
<td>DriverVersion</td>
<td>db</td>
<td>01, 00</td>
</tr>
<tr>
<td>StatisticsVersion</td>
<td>db</td>
<td>01, 00</td>
</tr>
<tr>
<td>TotalTxPacketCount</td>
<td>dw</td>
<td>0, 0</td>
</tr>
<tr>
<td>TotalRxPacketCount</td>
<td>dw</td>
<td>0, 0</td>
</tr>
<tr>
<td>NoCEBAvailableCount</td>
<td>dw</td>
<td>0</td>
</tr>
<tr>
<td>PacketTxTooBigCount</td>
<td>dw</td>
<td>-1 ; not used</td>
</tr>
<tr>
<td>PacketTxTooSmallCount</td>
<td>dw</td>
<td>-1 ; not used</td>
</tr>
<tr>
<td>PacketRxOverflowCount</td>
<td>dw</td>
<td>0</td>
</tr>
<tr>
<td>PacketRxTooBigCount</td>
<td>dw</td>
<td>0</td>
</tr>
<tr>
<td>PacketRxTooSmallCount</td>
<td>dw</td>
<td>0</td>
</tr>
<tr>
<td>PacketTxMiscErrorCount</td>
<td>dw</td>
<td>-1 ; not used</td>
</tr>
<tr>
<td>PacketRxMiscErrorCount</td>
<td>dw</td>
<td>-1 ; not used</td>
</tr>
<tr>
<td>RetryTxCount</td>
<td>dw</td>
<td>0</td>
</tr>
<tr>
<td>ChecksumErrorCount</td>
<td>dw</td>
<td>-1 ; not used</td>
</tr>
</tbody>
</table>

290189-21
Table 2. Declarations (Continued)

<table>
<thead>
<tr>
<th>HardwareRxMismatchCount</th>
<th>dw 0</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>NumberOfCustomVariables</td>
<td>dw (DriverDiagnosticText-DriverDebugEnd1)/2</td>
<td></td>
</tr>
<tr>
<td>DriverDebugEnd1 LABEL</td>
<td>byte</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>rx_errors</td>
<td>dw 0</td>
<td></td>
</tr>
<tr>
<td>underruns</td>
<td>dw 0</td>
<td></td>
</tr>
<tr>
<td>no_cts</td>
<td>dw 0</td>
<td></td>
</tr>
<tr>
<td>no_crs</td>
<td>dw 0</td>
<td></td>
</tr>
<tr>
<td>rx_aborts</td>
<td>dw 0</td>
<td></td>
</tr>
<tr>
<td>no_S90_int</td>
<td>dw 0</td>
<td></td>
</tr>
<tr>
<td>false_S90_int</td>
<td>dw 0</td>
<td></td>
</tr>
<tr>
<td>lost_rx</td>
<td>dw 0</td>
<td></td>
</tr>
<tr>
<td>stop_rx</td>
<td>dw 0</td>
<td></td>
</tr>
<tr>
<td>tent_cen_latch_crash</td>
<td>dw 0</td>
<td></td>
</tr>
<tr>
<td>rx_disb_failure</td>
<td>dw 0</td>
<td></td>
</tr>
<tr>
<td>tx_abort_failure</td>
<td>dw 0</td>
<td></td>
</tr>
<tr>
<td>rx_bufv_ovflw</td>
<td>dw 0</td>
<td></td>
</tr>
<tr>
<td>tx_timeout</td>
<td>dw 0</td>
<td></td>
</tr>
<tr>
<td>DriverDiagnosticText LABEL</td>
<td>byte</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>db 'RxErrorCount', 0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>db 'UnderrunCount', 0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>db 'LostCTSCount', 0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>db 'LostCRSCount', 0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>db 'RxAbortCount', 0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>db 'NoS90InterruptCount', 0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>db 'False590InterruptCount', 0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>db 'LostOurReceiverCount', 0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>db 'QuitTransmittingCount', 0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>db 'TencentLatchCrashCount', 0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>db 'TxDisableFailureCount', 0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>db 'TxMontAbort', 0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>db 'ReceiveBufferOverflow', 0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>db 'TxTimeoutErrorCount', 0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>db 0, 0</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

DriverDebugEnd LABEL word 290189-22
7.2 Initialization Routine

This routine, Driver Initialize, initializes the Embedded LAN Module hardware and the system hardware needed to support the module. It also sets up the system memory structure to support the module.

7.2.1 HARDWARE INITIALIZATION AND 82592 CONFIGURATION

Initialization of the Embedded LAN Module hardware begins with generating an individual address for the station, initializing the interrupt line and interrupt vector, and enabling the module by writing to port address 303h. After initializing the memory structure, the 82592 is directly programmed. This programming includes configuring the 82592 and initializing it with the station's individual address. The 82592 is configured in two steps. The first specifies a 16-bit-wide system bus interface by issuing a Configure command to the 82592, with 00h as the byte count; i.e., no parameters passed to the device. Then a second Configure command is issued; it does the following.

- The 82592 is put in High Speed Mode to support Ethernet serial bit rates.
- It is placed in TCI mode for interface to the Embedded LAN Module architecture.
- All network parameters (e.g., Frame Length, Slot Time, and Preamble Length) are set up for default Ethernet values.

Following this initialization and configuration of the module’s hardware, the 8259A Programmable Interrupt Controller's interrupt line for the module is enabled, allowing the interrupt-driven events frame reception and completed transmission. Then a Receive Enable command is issued to the 82592. Table 3 contains the code for hardware initialization.

<table>
<thead>
<tr>
<th>Table 3. Hardware Initialization</th>
</tr>
</thead>
</table>

```
monbo_init    segment 'CODE'
    public DriverInitialize, DriverUnHook
    no_card_message    db CR,LF,'No adapter installed in PC$
    config_failure_message    db CR,LF,'Configuration Failure$
    asset_failure_message    db CR,LF,'2A Setup Failure$
    ConfigDataUnderrunMass    db CR,LF,'Configuration underrun$

    ;     Driver Initialize
    ;     assumes:
    ;     DS, BS are set to GGroup (= CS)
    ;     DX points to where to stuff node address
    ;     Interrupts are ENABLED
    ;     The Real Time Ticks variable is being set, and the entire AES system is initialized.
    ;
    ;     returns:
    ;     If initialization is done OK:
    ;      AX has a 0
    ;     If board malfunction:
    ;      AX gets offset (in GGroup) of '$'-terminated error string
    ;
    DriverInitialize PROC     NEAR
    mov MaxBytesPacketSize, 1024
    cli
    cli
    mov ax, cs
    mov ds, ax
    mov es, ax
    ; get DOS time and use for address.
    mov ah,02Ch
    int 21h
    mov bx,OFFSET GGroup:node_addr
    mov byte ptr GGroup:[bx], 00h
    mov byte ptr GGroup:[bx+1], 0Ah
    mov byte ptr GGroup:[bx+2], 0h
    mov byte ptr GGroup:[bx+3], 01h
    mov byte ptr GGroup:[bx+4], 0h
    mov byte ptr GGroup:[bx+5], 7Eh
    mov ax, bx
    ;stuff address at point IPX indicated
    movsw
    movsw
    movsw
    sti
    ; initialize the configuration table
    mov al,selected_configuration
    cbw
    shl ax,1 ; multiply by two
    add ax,OFFSET GGroup:config_pointers ;ax contains the offset value
```

290189-23
Table 3. Hardware Initialization (Continued)

```assembly
mov bx, ax        ; of the default configuration
mov bx, [bx]     ; list
mov Config, bx
mov al, [bx+DMA0LOC]   
mov al, [bx+DMA0LOC]
mov config_dma0_loc, al
mov config_dma0_loc, al
mov al, [bx+DMA1LOC]   
mov config_dma1_loc, al
mov config_dma1_loc, al
mov ax, [bx+IPORT]       
mov command_reg, 300h

SetTheInterruptVector:
    push di
    mov al, config_irq_loc
    mov bx, OFFSET CGroup: DriverISR
    call SetInterruptVector
    pop di
    mov dx, EnLAN
    out dx, al        ; enable LAN on MB module
    slow
    mov dx, command_reg
    mov al, C_RST
    out dx, al        ; reset the 82592 controller

; generate 20 bit address for DMA controller from configure block location
; this is necessary to accommodate the page register used in the PC DMA

    call set_up_buffers
    ; set up DMA channel for configure command
    xor ax, ax
    out DMAff, al      ; data is don’t care
    slow
    mov al, DMAena
    out DMAcmdstat, al
    mov ax, gp_buf_start
    slow
    out DMA6addr, al
    mov al, ah
    slow
    out DMA6addr, al
    mov ax, gp_buf_page
    slow
    out DMA6page, al      ; DMA page value
    mov ax, 1
    slow
    out DMA6wrcount, al  ; make two transfers
    mov al, ah
    slow
    out DMA6wrcount, al
    mov al, DMAtr6       ; setup channel 6 for tx mode
    slow
    out DMAmode, al
    mov al, DMAfmsk
```

290189-24
Table 3. Hardware Initialization (Continued)

```assembly
;slow
out    DMA$nglmask, al
xor    ax, ax
the
mov    di, gp_buf_offset
;mov zeroes into the byte count field of the
stow   ;buffer to put the 82592 into 16 bit mode
stow
%slow
mov    dx, command_reg
mov    al, C_CONFIG
;configure the 82592 for 16 bit mode
out    dx, al
;issue configure command
%slow

wide_mode_wait_loop:
xor    ax, ax
%slow
out    dx, al
;point to register 0
%slow
in     al, dx
;read register 0
and    al, ODFh
;disregard exec bit
cmp    al, 82h
; is configure finished?
je     do_config
loop   wide_mode_wait_loop
mov    ax, OFFSET OGroup: no_card_message
jmp    init_exit

do_config:
    mov   al, C_INTACK
    out   dx, al
;clear interrupt
xor    ax, ax
%slow
out    DMAff, al
;data is don’t care
mov    ax, gp_buf_start
%slow
out    DMA addr, al
mov    al, ah
%slow
out    DMA addr, al
mov    ax, gp_buf_page
%slow
out    DMA page, al
;DMA page value
%slow
mov    al, DMA tx6
;setup channel 1 for tx mode
out    DMA mode, al
%slow
mov    ax, 8
out    DMA$wdcount, al
%slow
mov    al, ah
out    DMA$wdcount, al
%slow
mov    al, DMA$unmask
out    DMA$nglmask, al
mov    ax, ds
mov    ax, ax
mov    si, offset OGroup: config_block
mov    di, gp_buf_offset
```

290189-25
Table 3. Hardware Initialization (Continued)

```assembly
mov cx, 18
rep movsb
mov dx, command_reg
mov al, C_CONFIG ; configure the 82592
out dx, al
	slow
xor cx, cx

config_wait_loop:
	slow
xor al, al
	slow
out dx, al ; point to register 0
	slow
in dx, dx ; read register 0
and al, 0DFh ; discard extraneous bits
cmp al, 82h ; is configure finished?
je config_done
loop config_wait_loop
mov ax, OFFSET CGroup: config_failure_message
jmp init_exit

config_done: ; clear interrupt caused by configuration
mov al, C_INTACK
out dx, al

do an IA_setup
mov di, gp_buf_offset
mov al, 06h ; address byte count
stosb
mov al, 00h
stosb
mov ax, OFFSET CGROUP node_addr
mov cx, SIZE node_addr
rep movsb
	slow
out DMAff, al ; data is don't care
	slow
mov ax, gp_buf_start
out DMA6addr, al
mov al, ah
	slow
out DMA6addr, al
mov ax, gp_buf_page
	slow
out DMA6page, al ; DMA page value
	slow
mov al, DMAx6 ; setup channel 1 for tx mode
out DMAmode, al
	slow
mov ax, 3
out DMA6wdcount, al
	slow
mov al, ah
out DMA6wdcount, al
	slow
mov al, DMA6unmsk
out DMAunmsk, al
```
### Table 3. Hardware Initialization (Continued)

```
mov dx, command_reg
mov al, C_IASET ; set up the 82392 individual address
out dx, al
xor cx, cx ; cx is used by the loop instruction below, this
; causes the loop to be executed 64k times max

ia_wait_loop:
    xor al, al
    out dx, al
    slow
    in al, dx
    and al, 0DFh ; discard extraneous bits
    cmp al, 81h ; is command finished?
    js ia_done
    loop ia_wait_loop
mov ax, OFFSET CGroup: iaset_failure_message
jmp init_exit

ia_done:
    mov al, C_INTACK
    out dx, al ; clear interrupt from iaset
    ; initialize the receive DMA channel
    xor al, al
    out DMAff, al
    mov ax, rx_buf_start ; set dma up to point to the beginning of rx buf
    slow
    out DMAaddr, al
    mov al, ah
    slow
    out DMAaddr, al
    mov ax, rx_buf_page ; set rx page register
    slow
    out DMA7page, al
    mov al, DMArx?
    slow
    out DMAmode, al
    mov ax, rx_buf_length ; set word count to proper value
    slow
    out DMA7wdcount, al
    mov al, ah
    slow
    out DMA7wdcount, al
    mov al, dma7unmsk ; unmask receive DMA channel
    slow
    out DMA7unmsk, al
    ; unmask our interrupt channel
    in al, InterruptMaskPort
    mov bl, OFBH
    and al, bl
    slow
    out InterruptMaskPort, al
    ; enable the receiver
    mov dx, command_reg ; enable receives
    mov al, C_RXENB
    out dx, al
    xor ax, ax
```
### Table 3. Hardware Initialization (Continued)

```
mov cx, 1
init_exit:
    ret

ConfigDataUnderrun:
    mov ax, OFFSET CGroup: ConfigDataUnderrunMess
    jmp init_exit

IASetupDataUnderrun:
    mov ax, OFFSET CGroup: IIASetupDataUnderrun
    jmp init_exit

DriverInitialize   endp

; SetInterruptVector
; Set the interrupt vector to the interrupt procedure’s address
; save the old vector for the unhook procedure
; assumes: bx has the ISR offset
; al has the IRQ level
; interrupts are disabled

SetInterruptVector PROC NEAR
    ; mask on the appropriate interrupt mask
    push ax
    xchg ax, cx
    mov dl, 1
    shl dl, cl ; get the appropriate bit location
    mov int_bit, dl ; set the interrupt bit variable
    not dl
    mov int_mask, dl ; set the interrupt mask variable
    mov ax, InterruptMaskPort
    mov int_mask_register, ax
    pop ax
    std
    cbw
    xor cx, cx
    mov es, cx
    add al, 68h ; adding 8 converts int number to int type, i.e.,
               ; int 4 = type 12, int 5 = type 13 etc.
    shl ax, 1
    shl ax, 1 ; two shifts = mul by 4 to create offset of vector
    xchg ax, di
    mov int_vector_addr, di ; save this address for unhook
    mov ax, es: [di] ; save old interrupt vector
    mov word ptr old_irq_vector, ax
    mov ax, es: [di] + 2
    mov word ptr old_irq_vector + 2, ax
    xchg ax, bx ; bx has the ISR offset
    stosw
    mov ax, cs

    stosw
    ret

SetInterruptVector endp
```
7.2.2 Initializing System Memory

A buffer is constructed in system memory to support the Embedded LAN Module architecture. This buffer is divided into a receive buffer area and a transmit/general-purpose buffer area. This buffer (Tx/GP) is used as the transmit buffer and as the parameter block for 82592 commands that require parameters.

The combined size of the buffer areas requested by the program is 10 kB. The Tx/GP buffer should be at least 1200 bytes long. The Rx buffer should be at least 5 kB long. The amount of memory requested is twice the size of the minimum Rx buffer length because of the possibility of a DMA page break occurring at some point in the 10 kB buffer area. A page break can occur because the SYP301 (or any PC AT based architecture) uses a static page register to supply the upper address bits (A17–A23 for a 16-bit DMA channel) during a DMA cycle. These upper bits of the address cannot be incremented. The software checks for a page break and adjusts the buffer size if one is found. There are three possible page break scenarios.

- No page break occurs. The buffer size is not adjusted, the Tx/GP buffer area will be in the first 1200 bytes of the 10 kB buffer, and the Rx area will use the remainder.
- A page break occurs, and the buffer is divided so that one fragment is smaller than 1200 bytes. This fragment is too small to be used and both the Tx/GP and Rx areas will be placed in the larger segment.
- A page break occurs that divides the 10 kB buffer into two segments both larger than 1200 bytes. The software then places the Tx/GP area in the smaller segment, and the Rx area in the larger.

These three scenarios are shown in Figure 12. In no case is the Rx area less than 5 kB—half the total buffer size. Once these calculations are made, the transmit and receive DMA channels, along with their page registers, are programmed to point to their respective areas in the buffer (Tx/GP and Rx). With the memory now initialized, configuration and initialization of the 82592 can begin.

![Figure 12. DMA Page Break Affect on Buffer Size](image-url)
The Rx buffer area is implemented as a restartable linear buffer. As frames are stored in this buffer they are processed by the IPX routine IPXReceivePacket. A variable called RX_BUF_STOP points to a location 1200 bytes from the end of the Rx area. On reaching (or passing) this location in the Rx area, frame reception is temporarily disabled, and the remainder of the receive frames are processed. After the last frames have been processed, the receive area is reinitialized, the receive channel DMA is initialized to point back to the beginning of the receive area, and frame reception is reenabled. Table 4 contains the code that initializes the buffer memory. Section 7.3 gives further information on receive frame processing.

### Table 4. Buffer Memory Initialization

```assembly
set_up_buffers proc near

mov ax, offset group: gp_buf
mov gp_buf_offset, ax
mov bx, cs
shr ax, 1
mov cx, 3
shl bx, cl
rol dx, cl  ; get upper 3 bits for page register
and dx, 0007h  ; clear all but the lowest 3 bits
add ax, bx  ; ax contains EA of first location in buffer
adc dx, 0  ; if addition caused a carry add it to page
mov cx, 0FFFFh  ; uf buffer to page break
sub cx, ax  ; cx contains the number of bytes to page break
cmp cx, 01388h
ja low_ok
    jmp copacetic
int intel_hop:
    ; it's cool, whole buffer space is in one page

low_ok:
    cmp cx, 0258h
    ja low_ok
    add ax, cx  ; move pointer past the page break to discard fragment
    sub gp_length, cx  ; adjust length variable to reflect shorter length
    mov gp_offset_adjust, cx
    shl gp_offset_adjust, 1  ; convert to byte format
    mov cx, gp_offset_adjust
    add gp_buf_offset, cx  ; adjust gp_buf starting point to reflect change
    jmp copacetic  ; both buffers will be in the same page, rx buf

shortened

high_ok:
    ; now since both fragments are usable we have to find the
    cmp cx, 09C4h  ; actual page break, the large half will be the receive
    ja rx_first
    mov gp_buf_page, dx
    shl gp_buf_page, 1
```

290189-30
Table 4. Buffer Memory Initialization (Continued)

<table>
<thead>
<tr>
<th>Instruction</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>mov    gp_buf_start, ax</td>
<td>Setup first address of buffer space</td>
</tr>
<tr>
<td>mov    gp_buf_start, 0000h</td>
<td></td>
</tr>
<tr>
<td>mov    gp_buf_stop, 0000h</td>
<td></td>
</tr>
<tr>
<td>add    dx, 1</td>
<td>:next page</td>
</tr>
<tr>
<td>mov    rx_buf_page, dx</td>
<td></td>
</tr>
<tr>
<td>shr    rx_buf_page, 1</td>
<td></td>
</tr>
<tr>
<td>shr    ax, 1</td>
<td></td>
</tr>
<tr>
<td>adc    dx, 0</td>
<td>:save number of bytes to page break</td>
</tr>
<tr>
<td>mov    bx, cx</td>
<td></td>
</tr>
<tr>
<td>mov    cx, 12</td>
<td></td>
</tr>
<tr>
<td>shr    dx, cx</td>
<td></td>
</tr>
<tr>
<td>mov    rx_buf_length, bx</td>
<td></td>
</tr>
<tr>
<td>sub    cx, 258h</td>
<td></td>
</tr>
<tr>
<td>add    cx, ax</td>
<td></td>
</tr>
<tr>
<td>mov    rx_buf_stop, cx</td>
<td></td>
</tr>
<tr>
<td>jmp    buffers_set</td>
<td></td>
</tr>
<tr>
<td>rc_first:</td>
<td></td>
</tr>
<tr>
<td>mov    rx_buf_start, dx</td>
<td></td>
</tr>
<tr>
<td>shr    rx_buf_start, 1</td>
<td></td>
</tr>
<tr>
<td>mov    rx_buf_head, ax</td>
<td></td>
</tr>
<tr>
<td>mov    rx_buf_segment, dx</td>
<td></td>
</tr>
<tr>
<td>mov    gp_length, bx</td>
<td></td>
</tr>
<tr>
<td>mov    cx, gp_length</td>
<td></td>
</tr>
<tr>
<td>mov    rx_buf_stop, 0F89EH</td>
<td>:1200 bytes from end of buffer</td>
</tr>
<tr>
<td>mov    gp_buf_start, 0000h</td>
<td></td>
</tr>
<tr>
<td>add    dx, 1</td>
<td>:next page</td>
</tr>
<tr>
<td>mov    gp_buf_page, dx</td>
<td></td>
</tr>
<tr>
<td>shr    gp_buf_page, 1</td>
<td></td>
</tr>
<tr>
<td>add    cx, 1</td>
<td></td>
</tr>
<tr>
<td>shr    dx, cx</td>
<td></td>
</tr>
<tr>
<td>mov    gp_offset_adjust, cx</td>
<td></td>
</tr>
<tr>
<td>add    gp_buf_offset, cx</td>
<td></td>
</tr>
<tr>
<td>sub    dx, 1</td>
<td></td>
</tr>
<tr>
<td>shr    ax, 1</td>
<td></td>
</tr>
<tr>
<td>adc    dx, 0</td>
<td></td>
</tr>
<tr>
<td>mov    cx, 12</td>
<td></td>
</tr>
<tr>
<td>shr    dx, cx</td>
<td></td>
</tr>
<tr>
<td>mov    rx_buf_segment, dx</td>
<td></td>
</tr>
<tr>
<td>jmp    buffers_set</td>
<td></td>
</tr>
<tr>
<td>copyset:</td>
<td></td>
</tr>
<tr>
<td>mov    gp_buf_start, ax</td>
<td>:All-AS of gp buffer, gp buffer is first</td>
</tr>
<tr>
<td>add    ax, 258h</td>
<td>:1200 bytes for gp buffer at front of buffer space</td>
</tr>
<tr>
<td>mov    rx_buf_start, ax</td>
<td>:rx buffer starts 1200 bytes in</td>
</tr>
<tr>
<td>mov    rx_buf_head, ax</td>
<td></td>
</tr>
<tr>
<td>shr    rx_buf_head, 1</td>
<td></td>
</tr>
<tr>
<td>sub    gp_length, 258h</td>
<td></td>
</tr>
<tr>
<td>mov    cx, gp_length</td>
<td></td>
</tr>
<tr>
<td>mov    rx_buf_length, cx</td>
<td></td>
</tr>
<tr>
<td>shr    dx, 1</td>
<td>:convert segment to byte address</td>
</tr>
<tr>
<td>mov    rx_buf_page, dx</td>
<td></td>
</tr>
<tr>
<td>mov    gp_buf_page, dx</td>
<td></td>
</tr>
<tr>
<td>shl    ax, 1</td>
<td>:convert offset to byte address</td>
</tr>
<tr>
<td>adc    dx, 0</td>
<td>:adjust segment for shift</td>
</tr>
<tr>
<td>mov    cx, 12</td>
<td></td>
</tr>
<tr>
<td>shr    dx, cx</td>
<td></td>
</tr>
<tr>
<td>mov    rx_buf_segment, dx</td>
<td>:load variable for transfers to IPX</td>
</tr>
<tr>
<td>mov    cx, rx_buf_length</td>
<td></td>
</tr>
<tr>
<td>sub    cx, 258h</td>
<td>:setup marker for low rx buffer space, &gt;600 words</td>
</tr>
<tr>
<td>shr    cx, 1</td>
<td></td>
</tr>
<tr>
<td>add    ax, cx</td>
<td></td>
</tr>
<tr>
<td>mov    rx_buf_stop, ax</td>
<td></td>
</tr>
<tr>
<td>buffers_set:</td>
<td>ret</td>
</tr>
<tr>
<td>set_up_buffers endp</td>
<td></td>
</tr>
</tbody>
</table>

290189-31

290189-32
7.3 Assembly and Transmission of Frames

Frame assembly and transmission is accomplished by the interaction of the software driver and IPX through the use of IPX Event Control Blocks (ECBs). To transmit a frame, a transmit ECB is prepared that contains address information and a list of fragments in memory containing the frame to be transmitted. This ECB is placed in a queue for assembly and transmission of the frame. If the queue is empty, or when the ECB reaches the front of the queue, a routine is called that processes the ECB for transmission. This routine determines the length of the frame (padding the frame if necessary) and then constructs the frame in the Tx/GP buffer area. The construction of the frame is based on the ECB's address information and fragment list. The transmit DMA channel is now initialized to point to the beginning of the transmit frame in the Tx/GP area, and the byte count for that channel is also initialized. A Transmit command is now issued to the 82592. A separate routine monitors the transmission for a time-out error. When an interrupt from the 82592 indicates that the transmission attempt is complete (whether successful or unsuccessful), or if a time-out error has occurred, the proper completion code is inserted into the frame's ECB, and the ECB is passed back to IPX. If additional ECBs remain in the transmit queue the processing of the next ECB will begin. Table 5 contains the code used for assembly and transmission of frames.

Table 5. Assembly and Transmission of Frames

```assembly
; DriverSendPacket PROC NEAR
; Assumes
; ES:SI points to a fully prepared Event Control Block
; DS = CS
; Interrupts are DISABLED but may be reenabled temporarily if necessary
; don't need to save any registers

DriverSendPacket PROC NEAR
    cli ; disable the interrupts
    mov cx, word ptr send_list + 2
    jmp AddToListEndFound

AddToListEndFound:
    mov es:word ptr [si].link, cx ;move null pointer to newest ECB's
    mov es:word ptr [si].link + 2, cx ;link field
    mov ds:word ptr [dl].link, si
    mov ds:word ptr [dl].link + 2, es
    mov ax, cs
    mov ds, ax ;set ds back to entry condition
    ret

AddToListEndFound:
    mov es:word ptr [si].link, cx
    mov es:word ptr [si].link + 2, cx
    mov word ptr send_list, si
    mov word ptr send_list + 2, es
    ; drop through to Start Send

DriverSendPacket endp

; Start Send
; assumes:
; ES:SI points to the ECB to be sent.
; interrupts are disabled

start_send PROC NEAR
    public start_sen
    cli ; disable the interrupts
    ret
```

290189-33
Table 5. Assembly and Transmission of Frames (Continued)

<table>
<thead>
<tr>
<th>Line</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>cld</td>
<td>save SCB address in variable tx_ecb to liberate registers</td>
</tr>
<tr>
<td>mov</td>
<td>word ptr tx_ecb, si</td>
</tr>
<tr>
<td>mov</td>
<td>word ptr tx_ecb + 2, as</td>
</tr>
<tr>
<td>push ds</td>
<td>save ds for future use</td>
</tr>
<tr>
<td>; get IPX packet length out of the first fragment (IPX header)</td>
<td></td>
</tr>
<tr>
<td>lds bx, es: dword ptr [si].fragment_descriptor_list</td>
<td></td>
</tr>
<tr>
<td>mov ax, ds: [bx].packet_length</td>
<td></td>
</tr>
<tr>
<td>pop ds</td>
<td>; restore ds to Group</td>
</tr>
<tr>
<td>push ax</td>
<td>; save length for later use in 590 length field</td>
</tr>
<tr>
<td>xchg al, ah</td>
<td>; byte swap for 592 length field calculation</td>
</tr>
<tr>
<td>add ax, 18</td>
<td>; add in the overhead bytes DA, SA, CRC, length</td>
</tr>
<tr>
<td>mov padding, 0</td>
<td></td>
</tr>
<tr>
<td>cmp ax, 64</td>
<td></td>
</tr>
<tr>
<td>js long_enough</td>
<td></td>
</tr>
<tr>
<td>mov padding, 64</td>
<td></td>
</tr>
<tr>
<td>sub padding, ax</td>
<td></td>
</tr>
<tr>
<td>mov ax, 64</td>
<td></td>
</tr>
<tr>
<td>long_enough:</td>
<td></td>
</tr>
<tr>
<td>sub ax, 10</td>
<td>; SA and CRC are done automatically</td>
</tr>
<tr>
<td>inc ax</td>
<td></td>
</tr>
<tr>
<td>and ax, 0F0H</td>
<td>; frame must be even</td>
</tr>
<tr>
<td>mov tx_byte_cnt, ax</td>
<td></td>
</tr>
<tr>
<td>mov dl, gpbuf_offset</td>
<td></td>
</tr>
<tr>
<td>mov bx, cs</td>
<td></td>
</tr>
<tr>
<td>mov ea, bx</td>
<td></td>
</tr>
<tr>
<td>; move the byte count into the transmit buffer</td>
<td></td>
</tr>
<tr>
<td>stosw</td>
<td></td>
</tr>
<tr>
<td>mov the destination address from the tx ECB to the tx buffer</td>
<td></td>
</tr>
<tr>
<td>mov bx, si</td>
<td></td>
</tr>
<tr>
<td>lea si, [bx].immediate_address</td>
<td></td>
</tr>
<tr>
<td>mov ds, word ptr tx_ecb + 2</td>
<td></td>
</tr>
<tr>
<td>movsw movsw</td>
<td></td>
</tr>
<tr>
<td>mov ax, cs</td>
<td>; get back to the code (Dgroup) section</td>
</tr>
<tr>
<td>mov ds, ax</td>
<td></td>
</tr>
<tr>
<td>; now the 590 length field</td>
<td></td>
</tr>
<tr>
<td>pop ax</td>
<td></td>
</tr>
<tr>
<td>xchg ah, al</td>
<td></td>
</tr>
<tr>
<td>inc ax</td>
<td></td>
</tr>
<tr>
<td>and ax, 0F0H</td>
<td>; make sure E-Net length field is even</td>
</tr>
<tr>
<td>xchg ah, al</td>
<td></td>
</tr>
<tr>
<td>stosw</td>
<td></td>
</tr>
<tr>
<td>lds si, tx_ecb</td>
<td></td>
</tr>
<tr>
<td>mov ax, ds: [si].fragment_count</td>
<td></td>
</tr>
<tr>
<td>lea bx, [si].fragment_descriptor_list</td>
<td></td>
</tr>
<tr>
<td>move_frag_loop:</td>
<td></td>
</tr>
<tr>
<td>push ds</td>
<td>; save the segment</td>
</tr>
<tr>
<td>mov cx, ds: [bx].fragment_length</td>
<td></td>
</tr>
<tr>
<td>lds si, ds: [bx].fragment_address</td>
<td></td>
</tr>
<tr>
<td>tfastcopy</td>
<td></td>
</tr>
<tr>
<td>pop ds</td>
<td>; get the segment back</td>
</tr>
<tr>
<td>add bx, 6</td>
<td></td>
</tr>
<tr>
<td>dec ax</td>
<td></td>
</tr>
<tr>
<td>jnz move_frag_loop</td>
<td></td>
</tr>
</tbody>
</table>
Table 5. Assembly and Transmission of Frames (Continued)

```assembly
; start transmitting
mov cx, cs
mov ds, cx

; add any required padding
mov cx, 4     ; make sure frame ends with a NOP
add cx, padding
shr cx, 1
rep stosw
mov tx_active_flag, 1
xor ax, ax
out DMAff, al ; data is don't care, AX has been zeroed
mov ax, gp_buf_start

%slow
out DMA$addr, al
mov al, ah

%slow
out DMA$addr, al
mov ax, gp_buf_page
%slow
out DMA$page, al ; DMA page value

%slow
mov al, DMA$tx6 ; setup channel 1 for tx mode
out DMA$mode, al
mov ax, tx$byte_cnt
add ax, 4 ; add two for byte count, two for tx chain fetch
shr ax, 1 ; convert to word value and account for odd
adc ax, 0 ; byte DMA transfer
out DMA$wdcount, al

%slow
mov al, ah
out DMA$wdcount, al

%slow
mov al, DMA$unmask
out DMA$cancel, al
mov dx, command_reg
mov al, C_TX
out dx, al
mov ax, IPXIntervalMarker
mov tx_start_time, ax
%inc32 TotalTxPacketCount
ret

start_send   endp

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
; DriverPoll
;
; Poll the driver to see if there is anything to do
;
; Is there a transmit timeout? If so, abort transmission and return
; ECB with bad completion code. Check to see if frames are queued.
; If they are set up ES:SI and call DriverSendPacket.
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

DriverPoll PROC    NEAR
cli

```

---
Table 5. Assembly and Transmission of Frames (Continued)

```assembly
    cmp tx_active_flag, 0
    js NotWaitingOnTx
    mov dx, IPXIntervalMarker
    sub dx, tx_start_time
    cmp dx, TxTimeOutTicks
    jb NotTimedOutYet

    ; This transmit is taking too long so let's terminate it now
    ; Issue an abort to the 82592
    mov dx, command_reg
    mov al, C_ABORT ; abort transmit
    out dx, al

    inc tx_timeout
    les si, tx_tcb
    mov es: [si].completion_code, TransmitHardwareFailure ; stuff completion code of a failed tx
    mov ax, es: word ptr [si].link
    mov word ptr send_list, ax
    mov ax, es: word ptr [si].link + 2
    mov word ptr send_list + 2, ax

    ; Finish the transmit
    mov es: [si].in_use, 0
    call IPXHoldEvent

    ; Make sure that execution unit didn't lock up because of abort errata
    mov dx, command_reg
    mov al, C_SW1
    out dx, al
    mov al, C_SELRS
    %slow
    out dx, al
    mov al, C_SW0
    %slow
    out dx, al
    mov al, C_RXEB
    %slow
    out dx, al
    mov tx_active_flag, 0

    ; See if any frames are queued
    mov cx, word ptr send_list + 2
    jcxz queue_empty
    mov es, cx
    mov si, word ptr send_list
    call start_send

    queue_empty:
    NotWaitingOnTx:
    NotTimedOutYet:
    ret
```
Table 5. Assembly and Transmission of Frames (Continued)

DriverPoll endp

;****************************************************************
; Interrupt Procedure
;****************************************************************

even

R setErrorCheck:

BufSetOverflow:
  inc 5
  jmp int_exit

not 5 int:
  inc 5
  jmp int_exit

DriverISR PROC far
public DriverISR

push ax
push bx
push cx
push dx
push si
push di
push bp

push ds
push es

cli

; interrupt

call PXStartCriticalSection ; tell AES we're busy
mov al, EOI
out InterruptControlPort, al
out ExtraInterruptControlPort, al
mov ax, cs
mov ds, ax ; DS points to C/DGroup
mov dx, command_reg
mov al, 0
out dx, al ; set status reg to point to reg 0

; poll

in 5, dx

test al, 80h
jz not 5 int

and al, NOT 20h ; ignore the EXEC bit
mov ah, al ; save the status in AH
cmp ah, 0DBh ; did I receive a frame?
Table 5. Assembly and Transmission of Frames (Continued)

```assembly
js rcdv_packet
cmp ah, $eh ; did I finish a transmit?
sent_packet_jmp:
jmp int_exit

sent_packet_jmp:
js rcdv_packet
jmp sent_packet

sent_packet:
cli
cmp tx_active_flag, 0
je false_tx_int ; shouldn't have been transmitting
in al, dx
mov status10, al

false_tx_int:
jmp sent_packet

tx_error:
test status10, 20h
je tx_error
mov al, status10 ; extract the total number of retries from
and ax, 0fh ; the status register and add to retry count
add RetryTxCount, ax
xor ax, ax ; status = 0, good transmit

FinishUpTransmit:
lea si, tx_ecb
mov ax, [si].completion_code, al
mov ax, ax, word ptr [si].link
mov word ptr send_list, ax
mov ax, ax, word ptr [si].link + 2
mov word ptr send_list + 2, ax
mov ax, [si].in_use, 0
call IPXHoldEvent
push cx
pop dx
mov cx, word ptr send_list + 2
mov tx_active_flag, cl
jcxz int_exit_jmpl
mov bx, dx ; segment of next SCB in list
mov si, word ptr send_list ; offset of next SCB in list
call start_send
jmp finish_exit

int_exit_jmpl:
jmp int_exit

false_tx_int:
jmp int_exit

tx_error:
test status10, 20h ; Max collisions??
jnz QuitTransmitting
int exit

test status11, 01h ; Tx underrun??
jz lost_cts
inc underruns

lost_cts:
test status11, 02h ; did we lose clear to send??
jz lost_cts
inc no_cts

lost_crs:
test status11, 04h ; did we lose carrier sense??
jz hmmm
inc no_crs

hmmm:
lea si, tx_ecb
call start_send
mov al, TransmitHardwareFailure
jmp FinishUpTransmit

QuitTransmitting:
mov al, status10
and ax, 0fh
add RetryTxCount, ax
inc stop_tx
mov al, TransmitHardwareFailure
jmp FinishUpTransmit

DriverISR endp
```

290189–38

290189–39
7.4 Receive Frame Processing

Receive frame processing is triggered by an interrupt from the 82592. If the status read from the 82592 by the Interrupt Service Routine (ISR) indicates that a frame has been received, a jump is made to the beginning of the code that services frames. The receive buffer area is managed by using several variables. These variables are listed below. Please refer to Section 4.1 and 4.2 for a review of receive frame processing.

- RX_BUF_TAIL. Contains the contents of the 16-bit latch. They point to the byte count of the last frame written into memory.
- RX_BUF_PTR. Keeps track of the current position in the buffer while the CPU recovers locations of the received frames in the buffer processing.
- RX_BUF_HEAD. Contains the pointer to the byte count of the last frame that was processed by the CPU. (This differs from rx_buf_tail, which points to the byte count of a frame not yet processed.)
- RX_BUF_STOP. Points to a location that is 1200 bytes from the end of the receive buffer (slightly more than the maximum size of a frame).

After servicing a receive frame, the contents of the 16-bit latch are loaded into RX_BUF_TAIL and RX_BUF_PTR. This value is compared with the value stored in RX_BUF_STOP to determine if most of the buffer has been used and if the buffer must be reinitialized after the current receive frames have been processed (in this case a flag called RESET_RX_BUF is set to indicate that the buffer variables and receive DMA channel must be reinitialized before the Interrupt Service Routine is exited). To process the frame or frames received, both the byte count and status bytes of the frame are used. If the status indicates a receive error the frame is not passed up to IPX. The byte count is used to index back through the chain of received frames, using RX_BUF_PTR to keep track of the current position in the buffer. The frames are checked for length (maximum and minimum), and a check is also made to verify that the Ethernet and IPX length fields agree (including provisions for padding the Ethernet length field). If these checks pass, the frames are added to the list of received frames by storing their location, length, and source address in an array of structures called RX_LIST. When the RX_BUF_PTR contains the same value as RX_BUF_HEAD, all currently received frames have been processed, and a jump is made to a label called HAND_OFF_PACKET. In this routine the frames are handed up to IPX, in the order they were received, using calls to the IPX routine IPXReceivePacket. The value stored in RX_BUF_TAIL is loaded into the RX_BUF_HEAD variable, which now holds the address of the last location in the receive area that was processed, and the execution of the ISR falls through to a routine to exit the ISR. Before exiting the ISR an Interrupt Acknowledge is issued to the 82592; a check for additional pending interrupts is made, if one is found the ISR process is repeated; and the flag RESET_RX_BUF is checked, if it is set the receive buffer is reinitialized. The machine states of the previous routines are restored to their original states, and the ISR is exited. Table 6 contains the code used for receive frame processing.
Table 6. Receive Frame Processing

;*******************************************************
; Interrupt Procedure
;*******************************************************

even
RxErrorTypeCheck:

BufferOverflow:
    inc rx_buff_ovflw
    jmp int_exit

not_590_int:
    inc no_590_int
    jmp int_exit

DriverISR PROC far
public DriverISR

push ax
push bx
push cx
push dx
push si
push di
push bp
push ds
push es
cld

int_poll_loop:
cld
    call IPXStartCriticalSection ;tell ARS we're busy
    mov al, EOI
    out InterruptControlPort, al
    out ExtraInterruptControlPort, al
    mov ax, cs
    mov ds, ax ; DS points to C/DSGroup
    mov dx, command_reg
    mov al, 0
    out dx, al ; set status reg to point to reg 0
%s
    in al, dx
    test al, 80h
    js not_590_int
Table 6. Receive Frame Processing (Continued)

; int_poll_loop:
    and al, NOT 20h ;ignore the EXEC bit
    mov ah, al ; save the status in AH
    cmp ah, 0D8h ; did I receive a frame?
    jz rcvd_packet
    cmp ah, 84h ; did I finish a transmit?
    jz sent_packet_jmp
    cmp ah, 8Ch ; did I finish a retransmit?
    jz sent_packet_jmp
    inc false_S9_int ; unwanted interrupt
    jmp int_exit

    sent_packet_jmp:
        jmp sent_packet
    bad_rcv:
        inc rx_errors
        jmp RxErrorTypeCheck
    int_exit_jmp:
        jmp int_exit

; When the address bytes are being read it is possible that another frame
; could come in and cause a coherency problem with the ten-cent latches.
; I am dealing with this possibility by reading TenCentHi twice and making
; sure the values match. If they don't the read is redone.

rcvd_packet:
    cli
    mov dx, TenCentHi ; read high address byte of last frame received
    in al, dx
    mov ah, al ; save it in ah
    mov dx, TenCentLo ; read low address byte of last frame received
    in al, dx
    mov rx_buf_tail, ax ; this is the last location containing rx data
    ; Read TenCentHi again to make sure it hasn't changed......
    mov dx, TenCentHi ; read high address byte again
    in al, dx
    cmp al, ah
    jz addr_ok
    jmp rcvd_packet ; read the latches again

addr_ok:
    mov ax, rx_buf_tail ; this is a valid address
    mov rx_buf_ptr, ax ; this is the last location containing rx data
    cmp rx_buf_stop, ax ; is most of the buffer already used?
    ja BufferOK
    mov reset_rx_buf, 1
BufferOK:
    cli
    mov ax, rx_buf_head
    ja process_new_frames
    inc ten_cent_latch_crash
    jmp int_exit

do_next_frame:
process_new_frames:
    mov bx, rx_buf_ptr ; end of current frame to process
    sub bx, 6 ; set bx up to point to beginning of the status
    mov es, rx_buf_segment ; this is necessary because latches hold EA not
                           ; offset relative to CGROUP
Table 6. Receive Frame Processing (Continued)

```assembly
mov al, es:[bx].status1 ;test for good receive
test al, 20h
jnz good_rx
mov cl, es:[bx].bc_lo
mov ch, es:[bx].bc_hi ;cx has actual number of bytes read
dec cx ;toss byte count & status
and cl, 0f8h ;round up
sub bx, cx ;bx points to first location of frame
cmp rx_buf_head, bx
je hand_off_packet_jmp ;this was the first frame in the sequence
mov rx_buf_ptr, bx
sub rx_buf_ptr, 2
to_do_next_frame:
    jmp do_next_frame
hand_off_packet_jmp:
    jmp hand_off_packet

good_rx:
    mov cl, es:[bx].bc_lo
    mov ch, es:[bx].bc_hi ;cx has actual number of bytes read
    mov curr_rx_length, cx
dec cx ;toss byte count & status
    and cl, 0f8h ;round up
    sub bx, cx ;bx points to first location of frame
    mov rx_buf_ptr, bx
    sub rx_buf_ptr, 2 ;rx_buf_ptr = last location of n-1 frame
    sub cx, 14
    cmp cx, 1024 + 64
    jbe not_too_big
    inc PacketRxTooBigCount
    jmp do_next_frame

not_too_big:
    cmp cx, 30
    jae not_too_small
    inc PacketRxTooSmallCount
    jmp do_next_frame

not_too_small:
    mov ax, es:[bx].rx_length ;get IPX length
    xchg al, ah
    inc ax
    and al, 0f8h
    xchg al, ah
    cmp ax, es:[bx].rx_physical_length ;same as 802.3 length ?
    jne to_do_next_frame
    xchg al, ah
    cmp ax, 60 - 14 ;at least min length minus header
    ja len_ok
    ;yes, continue
    mov ax, 60 - 14 ;no, round up
    len_ok:
        cmp ax, cx ;match physical length
        jz not_inconsistent ;yes, continue
        inc HardwareRxMismatchCount
        jmp do_next_frame
    not_inconsistent:
        ;inc32 TotalRxPacketCount ;Double Word Increment
        mov ax, 12
        mul num_of_frames
```

290189–42

1-459
Table 6. Receive Frame Processing (Continued)

<table>
<thead>
<tr>
<th>Instruction</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>mov dl, ax</td>
<td>first location of ethernet frame</td>
</tr>
<tr>
<td>movsx rlist[dli], bx</td>
<td>first location of ipx packet</td>
</tr>
<tr>
<td>add rlist[dli], 14</td>
<td></td>
</tr>
<tr>
<td>mov ax, rx buf segment</td>
<td></td>
</tr>
<tr>
<td>mov rlist[dli + 2], ax</td>
<td></td>
</tr>
<tr>
<td>mov ax, word ptr es:[bx].rx_length</td>
<td></td>
</tr>
<tr>
<td>xor ah, ah</td>
<td></td>
</tr>
<tr>
<td>mov rlist[dli + 4], ax</td>
<td></td>
</tr>
<tr>
<td>mov ax, word ptr es:[bx].rx_source_addr + 0</td>
<td></td>
</tr>
<tr>
<td>mov word ptr rlist[dli + 6], ax</td>
<td></td>
</tr>
<tr>
<td>mov word ptr es:[bx].rx_source_addr + 2</td>
<td></td>
</tr>
<tr>
<td>mov word ptr rlist[dli + 8], ax</td>
<td></td>
</tr>
<tr>
<td>mov ax, word ptr es:[bx].rx_source_addr + 4</td>
<td></td>
</tr>
<tr>
<td>mov word ptr rlist[dli + 10], ax</td>
<td></td>
</tr>
<tr>
<td>add num_of_frames, 1</td>
<td></td>
</tr>
<tr>
<td>cmp rlist[dli], 50</td>
<td></td>
</tr>
<tr>
<td>je hand_off_packet</td>
<td></td>
</tr>
<tr>
<td>jmp do_next_frame</td>
<td></td>
</tr>
<tr>
<td>hand_off_packet:</td>
<td></td>
</tr>
<tr>
<td>mov si, rlist[dli]</td>
<td></td>
</tr>
<tr>
<td>mov as, rlist[dli + 2]</td>
<td></td>
</tr>
<tr>
<td>mov cx, rlist[dli + 4]</td>
<td></td>
</tr>
<tr>
<td>lea bx, rlist[dli + 6]</td>
<td></td>
</tr>
<tr>
<td>cli</td>
<td></td>
</tr>
<tr>
<td>push ds</td>
<td></td>
</tr>
<tr>
<td>call IPXReceivePacket</td>
<td></td>
</tr>
<tr>
<td>pop ds</td>
<td></td>
</tr>
<tr>
<td>sub num_of_frames, 1</td>
<td></td>
</tr>
<tr>
<td>je adjust_rx_head</td>
<td></td>
</tr>
<tr>
<td>sub dl, 12</td>
<td></td>
</tr>
<tr>
<td>jmp hand_off_packet</td>
<td></td>
</tr>
<tr>
<td>adjust_rx_head:</td>
<td></td>
</tr>
<tr>
<td>mov ax, rlist[dli + 12]</td>
<td></td>
</tr>
<tr>
<td>add ax, 2</td>
<td></td>
</tr>
<tr>
<td>mov rlist[dli + 14], ax</td>
<td></td>
</tr>
<tr>
<td>;set rx_buf_head to new value for next receive</td>
<td></td>
</tr>
<tr>
<td>;interrupt</td>
<td></td>
</tr>
<tr>
<td>int 21h</td>
<td></td>
</tr>
<tr>
<td>push cs</td>
<td></td>
</tr>
<tr>
<td>pop ds</td>
<td></td>
</tr>
<tr>
<td>cmp tx_active_flag, 0</td>
<td></td>
</tr>
<tr>
<td>jnz finish_exit</td>
<td></td>
</tr>
<tr>
<td>; verify that our receiver is still going.</td>
<td></td>
</tr>
<tr>
<td>mov dx, command_reg</td>
<td></td>
</tr>
<tr>
<td>mov al, 60h</td>
<td></td>
</tr>
<tr>
<td>out dx, al</td>
<td></td>
</tr>
<tr>
<td>;alow</td>
<td></td>
</tr>
<tr>
<td>in al, dx</td>
<td></td>
</tr>
<tr>
<td>test al, 20h</td>
<td></td>
</tr>
<tr>
<td>jnz finish_exit</td>
<td></td>
</tr>
<tr>
<td>jmp LostOurReceiver</td>
<td></td>
</tr>
<tr>
<td>finish_exit:</td>
<td></td>
</tr>
<tr>
<td>cli</td>
<td></td>
</tr>
</tbody>
</table>

290189-43
Table 6. Receive Frame Processing (Continued)

call IPXEndCriticalSection
mov dx, command_reg
mov al, C_INTACK
out dx, al ;issue interrupt acknowledge to the 590

mov al, al ;set status reg to point to reg 0

in al, dx
test al, 80h
jns int_pending
cmp reset_rx_buf, 1
jns no_rx_buf_reset
mov al, dma7mask ;mask receive DMA channel
out DMAnglemask, al

out DMAaff, al ;data is don't care
mov ax, rx_buf_start ;set dma up to point to the beginning of rx buf
mov rx_buf head, ax
shl rx_buf head, 1
out DMAaddr, al
mov al, ah

out DMA7addr, al
mov al, DMAre7

out DMA7mode, al
mov ax, rx_buf_length ;set up rx buf

out DMA7dcounI. al
mov al, ah

out DMA7dcount, al
mov ax, DMAnglemask
mov al, DMA7angle

out dx, al
mov dx, command_reg
mov al, C_RXENB
out dx, al
mov reset_rx_buf, 0

no_rx_buf_reset:
call IPXServiceEvents
pop es
pop ds

pop bp
pop di
pop si
pop dx
pop cx
pop bx

pop ax

sti
iret

LostOurReceiver:
inc lost_rx
mov al, C_RXENB
mov dx, command_reg
out dx, al
jmp finish_exit

too_big:
inc PacketRxOverflowCount
jmp int_exit

int_pending:
jmp int_poll_loop

290189-44

290189-45
Expanding the 82592 Embedded LAN Module Architecture to a Low-Cost Non-Buffered Adapter

The basic architecture of the 82592 Embedded LAN Module can be expanded and applied to a low-cost, non-buffered adapter. This requires adding a DMA unit and some logic for a bus master handshake. Such an adapter would contain no local buffer memory. Its cost advantage would come from using existing system memory, as the embedded module does. This adapter is less complex than most existing designs because it does not require arbitration logic for access to local memory. This adapter becomes a bus master when data transfers take place, either to the 82592 (Tx) from system memory or from the 82592 (Rx) into system memory.

The same features of the 82592 that make it successful in embedded applications make it well-suited for non-buffered adapters. As with the embedded module, there is no intermediate buffering of data in a local memory, therefore data transfers to and from system memory take place in real time. The 82592's large FIFO area allows it to tolerate long system bus latencies during memory access. The 82592's high-performance, 16-bit bus interface allows the adapter to efficiently transfer data to and from system memory when it gains access to the system bus. The TCI of the 82592 will interface with the adapter's control logic and DMA unit to provide back-to-back frame reception and automatic retransmission on collision (both without CPU intervention). Figure 13 is a block diagram of the basic architecture of the embedded module modified for a non-buffered adapter application. The block titled "Control PALs and Latch" together with the 82592 is the core of the embedded module architecture. One additional PAL (PAL C) has been added to the basic architecture to offer more logic for decoding additional components added to the adapter. The address latch has also been expanded to 24 bits. The three shaded blocks (DMA Machine, Master Logic, and Control PALs and Latch) show the most likely path for integration on this adapter, providing a three-chip solution of ASIC, 82592, and 82C501. The 82C37 is common in many ASIC cell libraries, offering a migration path for this integration.

ADAPTER BLOCK DESCRIPTIONS

DMA Machine

- **8237 DMA Controller.** Serves as the core for the DMA machine. Performs addressing and control for data transfers between the 82592 and host system memory.
- **8-Bit Page Counter.** Provides the addressing bits for the upper bits of address (A17–A23).
- **8-Bit Register.** Serves as the base register for the upper bits of the Tx DMA channel for reinitialization for automatic retransmission.
- **8-Bit Multiplexer.** Selects between the upper bits of Rx- or Tx-channel DMA.
- **8-Bit Latch.** Latches the upper bits of address from the 8237 (A8–A15).

Master Logic

- **Master PAL.** Implements a "master" handshake with the host system bus to gain access to the bus as a bus master.
- **Timers (2).** Controls the maximum time the adapter can hold the bus, and the minimum time it must wait before attempting to regain bus access.

Control PALs and Latch (Together with 82592 and 82C501)

The basic architecture of the 82592 Embedded LAN Module.

Transceivers

Used to buffer the adapter logic from the host system bus, for drive purposes. Address consists of 24 bits; and Data, 16 bits.
Figure 13. 82592 Non-Buffered Adapter Block Diagram (PC AT Version)
PC586E
CSMA/CD LAN EVALUATION BOARD

- Supports Established CSMA/CD LAN Standards:
  - Ethernet (IEEE 802.3 10BASE5)
  - Cheapernet (IEEE 802.3 10BASE2)
- Interfaces to Popular IBM and IBM Compatible PC Systems:
  - IBM PC, PC-XT, PC-AT (8-Bit Data Transfer)
  - IBM PC-AT (16-Bit Data Transfer)
- Jumper Selection Offers High Degree of Flexibility in System Configuration:
  - Up to 8 Address Decode Ranges
  - Up to 8 Interrupt Lines
  - Ethernet (IEEE 802.3 10BASE5)
  - Cheapernet (IEEE 802.3 10BASE2)
  - Number of Wait-States
- Auto-Configuring for either 8-Bit or 16-Bit Bus Systems
- On-Board Transceiver Provides Direct Coaxial Connection for Cost-Effective Cheapernet Applications
- Pipelined Access in 8-Bit Mode Increase Performance through Reduced Wait-States
- 16 Kbytes of Shared Memory-Mapped SRAM Enables Higher Performance Network Operation
- Reduces Design Complexity because No I/O Address or DMA Channels Required
- High Efficiency Interleaved Memory Access Permits Zero Wait-State Access by Host CPU for Most Cycles
- 8 Kbytes of "Remote Boot" EPROM (Optional) Eliminates Need for Disk Drives
- Provides LAN Designer with a Complete, High-Performance CSMA/CD Ethernet/Cheapernet Solution

The PC586E evaluation board is a non-intelligent, buffered CSMA/CD LAN adapter card designed to demonstrate Intel's high-performance Ethernet/Cheapernet chip set. It provides IEEE 802.3 TYPE 10BASE5 (Ethernet) and TYPE 10BASE2 (Cheapernet or thinwire Ethernet) connections for IBM PC, PC-XT, PC-AT and compatible systems. The PC586E combines the Intel 82586 LAN Coprocessor and the Intel 82C501 Ethernet Serial Interface with an on-board Ethernet Transceiver into a total Ethernet/Cheapernet solution. The card is easily installed in either an 8-bit or 16-bit PC expansion slot and then automatically configures itself for 8-bit or 16-bit data transfers. Its jumpers offer a high degree of flexibility for system-dependent configuration. For Ethernet applications, the 82586/82C501 pair provide the complete transceiver cable interface required by the IEEE 802.3 standard. In addition, the PC586E's on-board transceiver provides the entire coaxial cable interface for convenient, cost-effective Cheapernet systems.

The PC586E is provided solely as an evaluation tool for use in designing with Intel's 82586 chip set. It has not been tested for compliance to FCC requirements for EMI (Part 15, subpart j). Intel is not responsible for any misuse of this evaluation board.

Figure 1. PC586E Block Diagram

290196-1
The PC586E is part of Intel's LAN Evaluation Board program. The board is intended to demonstrate the high-performance characteristics of the 82586 chip set in an adapter card application. The PC586E gives LAN engineers a head start in finding the best solution for their specific network problem. PC586E boards are shipped with detailed design documentation (artwork and PAL equations also available).

The PC586E is based on an Interleaved Local Memory Access scheme with Static RAM dual-ported between the 82586 LAN Coprocessor and the Host System CPU. Access to the board is purely Memory-Mapped, and therefore, no I/O ports or DMA channels are required. In addition to the shared SRAM, the system supports a "Remote-Boot" EPROM and 32 bytes of Address PROM. The 82586 has access only to the Static RAM.

**MEMORY**

The Local Memory consists of 16 Kbytes of Static RAM, 32 bytes of Address PROM, 16 Command Registers, and up to 8 Kbytes of "Remote Boot" EPROM (Optional). All of the Local Memory is mapped into unused memory space of the Host System. Commands are issued to the PC586E by transferring the instruction to a Command Register. The Command Registers are used for issuing the Reset and Channel Attention signals to the 82586, enabling interrupts and configuring the board.

**CONFIGURATION**

There are up to 8 jumper-selectable locations for the Local Memory and the Command Registers (four of these locations are mapped above the 1 Mbyte boundary, FFFFh). In addition, the jumpers are used for the Interrupt Request Signal which may be assigned to any one of eight Interrupt Request lines.

The PC586E automatically detects if it is placed in an 8-bit or 16-bit expansion slot. When the PC586E is in a 16-bit slot, a Command Register is used to program the PC586E for either 8-bit or 16-bit data transfers. One of the Command Registers can also be used to disable the interrupt signal.

**INTERLEAVED MEMORY ACCESS**

The PC586E uses Interleaved Memory Access between the 82586 LAN Coprocessor and the Host System CPU to increase system performance. One read or write access is allowed by the Host System for every read or write access by the 82586. In this way, high utilization of local memory is achieved. The logic used is a "cycle-stealing" approach in which the 82586 is never given wait-states. This precludes the need for wait-state logic for the 82586 and allows the 82586 to run at 6 MHz.

When the 82586 is inactive, the interleaving logic becomes transparent and the Host System may access the Local Memory with no wait-states (16-bit buses only). This provides about a 15% to 20% boost in bus performance.

**DESCRIPTION OF INTERLEAVE LOGIC**

Since the 82586's READY and HOLD ACKNOWLEDGE signals are always active, only a simple arbiter is required. The Control Logic merely interleaves Host System accesses with 82586 accesses. When the 82586 is active, the Host System access will occur during the first half of the 82586 "read/write" cycle. When the 82586 is inactive, the Host System access will occur at the speed of the Host Bus.

If the Host System initiates access to the static RAM during T1 or T2 of the 82586 "read/write" cycle, it will complete operation without any additional wait-states. If the Host System should initiate access during T2 or T3 of the 82586 "read/write" cycle, a maximum of three wait-states will be inserted for an 8 MHz AT system. The maximum number of wait-states depends on the width and frequency of the Host System.

**WORD ASSEMBLY/DISASSEMBLY**

For systems with 8-bit data buses, the PC586E has a special Word Assembly/Disassembly function. Access to the Static RAM may be made either as 8-bit or 16-bit operations. If 8-bit transfers are made, the Word Assembly/Disassembly logic is used to increase performance.

**WORD DISASSEMBLY**

An 8-bit "read" operation to an even address causes 16 bits of data to be read from the Static RAM. The first 8 bits are transferred onto the Host bus and the second 8 bits (corresponding to the odd address) are temporarily stored in a latch. When the subsequent "read" is made to the odd address, the data stored in the latch is copied onto the Host Bus. In this way, access to the Static RAM by the Host CPU is reduced by 50%.

**WORD ASSEMBLY**

An 8-bit "write" operation to an even address causes the data stored at this location to be temporarily
transferred to a latch. When the subsequent 8-bit "write" operation is made (corresponding to the odd address), the two 8-bit bytes are combined into a 16-bit word which is then transferred to the Static RAM.

In order to take advantage of this scheme, all access to the Static RAM must be made on a 16-bit word basis to even addresses. Since the 82586 data structures are naturally designed to be 16-bits wide, this requirement has little or no impact on software. The bus interface of systems with 8-bit data busses will automatically break 16-bit operations into two 8-bit operations. The same software can thus be used for both 8-bit and 16-bit systems.

The Word Assembly/Disassembly function is only used for access to the Static RAM. All accesses to the Address PROM, Remote Boot EPROM and Command Registers are made as 8-bit transfers only.

REMOTE BOOT EPROM

An optional 8192 byte EPROM may be installed for either "Remote Boot" operation or general purpose ROM. Upon booting the system, the Host CPU searches for a 55AAh data pattern starting at address C8000h. If the pattern is not found, additional attempts will be made at subsequent addresses in 2 Kbyte increments. If the pattern is found, the Host will then search for a jump instruction and a Cyclic Redundancy Check (CRC). If these are found, the CPU will begin executing the code at the location specified by the jump instruction. In order to take advantage of the "Remote Boot" option, the software on the EPROM must be able to configure the PC586E and copy the operating system through the network. This ability removes the need for disk drives. The EPROM may be used for general purpose storage instead of remote booting. In either case, only 8-bit "read" operations are permitted from this device.

ETHERNET/CHEAPERNET SELECTION

The PC586E Board is jumper-selectable to operate in either Ethernet (IEEE 802.3 10BASE5) or Cheapernet (IEEE 802.3 10BASE2) mode.

ETHERNET

In Ethernet mode, the 82586 LAN Coprocessor is used in conjunction with the Intel 82C501 Ethernet Serial Interface. Functions of the 82C501 include Manchester encoding/decoding of transmit and receive data, generation of the transmit and receive clock and interface to the AUI/Transceiver cable. In addition, the 82C501 has a built in watchdog timer, internal loopback diagnostics and collision detection circuitry. The 82586/82C501 thus provide the complete transceiver cable interface required by IEEE 802.3.

CHEAPERNET

In Cheapernet mode, the Ethernet Transceiver is located on-board. The transceiver works in conjunction with the 82586 and 82C501 to provide the complete, on-board, coaxial cable interface.

COMPONENT DESCRIPTION

82586 LAN Coprocessor

- Implements a Complete CSMA/CD Data Link
- Incorporates all Logic for Executing Time Critical Functions Independently of Host System
- High-Level Command Interface Simplifies Software Programming
- Supporting Industry CSMA/CD LAN Standards Ethernet (IEEE 802.3 10BASE5) Cheapernet (IEEE 802.3 10BASE2)
- Provides On-Chip Memory Management with Automatic Buffer Chaining and Reclaiming
- Interfaces to Industry Standard 8-Bit and 16-Bit Microprocessors
- Powerful System Interface
  - On-Chip DMA Control Allows Up to 5 Mbytes/Sec Bus Capacity
  - 8-Bit or 16-Bit Data Bus
  - Back-to-Back Frame Reception at 10 Mb/s
- Built-In Network Management and Diagnostics
  - Transmission/Reception Error Reporting
  - Network Activity and Error Statistics
  - Station Diagnostics (External Loopback)
  - Self Test Diagnostics

The 82586 is an intelligent peripheral that completely manages the processes of transmitting and receiving frames of data over the network, thus off-loading the Host CPU of communication management tasks. The 82586 features an on-chip DMA controller which allows it to access the local memory through an efficient buffer chaining mechanism. Other features of the 82586 are the ability to perform network management activities including error and collision tallies and diagnostic capabilities via the internal and external loopback function. Control of the 82586 is through high level commands such as TRANSMIT and CONFIGURE.
All information passed between the 82586 and the Host board is made through shared local memory. The Host may load the memory with a command and prompt the 82586 to execute. While receiving a packet, the 82586 loads receive buffers in local memory and, after completing the reception, interrupts the Host board to indicate that a packet has been received.

**82C501 ETHERNET SERIAL INTERFACE**

- Direct Interface to the 82586 LAN Coprocessor and Ethernet Transceiver
- Conforms to IEEE 802.3 10BASE5 (Ethernet) and IEEE 802.3 10BASE2 (Cheapernet) Specifications
- 10 Mb/s Serial Data Rate
- Manchester Encoding/Decoding and Receive Clock Recovery
- 10 MHz Transmit Clock Generation
- Drives and Receives IEEE 802.3 AUI (Transceiver) Cable
- Optional Watchdog Timer Prevents Babbling
- Internal Diagnostic Loopback for Fault Detection and Isolation
- Functionally Compatible with the SEEQ 8023A

The 82C501 provides the Ethernet (IEEE 802.3 10BASE5) or Cheapernet (IEEE 802.3 10BASE2) Serial Interface for the 82586 LAN Coprocessor. Major functions of the 82C501 include generation of the transmit and receive clock (10 MHz for Ethernet and Cheapernet), Manchester encoding/decoding of transmit and receive data, and interfacing the 10BASE5 Access Unit Interface (AUI/Transceiver) cable. In addition, the 82C501 provides for fault isolation with internal diagnostic loopback. An on-chip watchdog timer prevents the station from locking up in the continuous transmit mode (jabber control).

**PC586E Specifications***

**Software:**
- Network Software Drives are Currently Available for the Following Applications:
  - UNIX/TCP-IP
  - Novell/Netware
  - (Additional Drivers to be Announced)

**Hardware:**
- IBM PC, PC-XT, PC-AT and Compatible Systems

**Cable Connections:**
- DB-15 Connector (Ethernet)
- BNC Connector (Cheapernet)

**System Components:**
- Intel 82586 LAN Coprocessor
- Intel 82C501 Ethernet Serial Interface

**Memory Capacity:**
- Static RAM 16 Kbytes
- General Address PROM 32 bytes
- Bootable EPROM 8 Kbytes

**Memory Address Ranges:**
1. 0C0000h–0C7FFFh
2. 0C8000h–0CFFFFh
3. 0D0000h–0D7FFFh
4. 0D8000h–0DFFFFh
5. F00000h–F3FFFFh
6. F40000h–F7FFFFh
7. F80000h–FBFFFFh
8. FC0000h–FFFFFFh

**Frequency:**
- Board Master Clock 24 MHz
- 82586-6 6 MHz

8-Bit PC Bus
- Frequency (Max.): 4.77 MHz 0 Additional Wait-States
- 8 MHz 0 Additional Wait-States
- > 8 MHz Not Supported

16-Bit AT Bus
- Frequency (Max.): 8 MHz 0 Additional Wait-States
- 10 MHz 0 Additional Wait-States
- 12 MHz 1 Additional Wait-States
- > 12 MHz Not Supported

**Voltage Limits:**
- +5V Input ±5%
- +12V Input ±5%

**Current Requirements:**
- +5V Input 3.0A*
- +12V Input 300 mA*

**Power Dissipation:**
- Maximum 18.6W*

**Temperature Range:**
- Operating 0°C to +55°C
- Storage 0°C to +70°C

**DIMENSIONS (Not Including Mounting Bracket)**

Length: 8.2 in. (20.8 cm)
Height: 4.2 in. (10.7 cm)
Width: 0.7 in. (1.8 cm)

*Preliminary, subject to change
Wide Area Networks
The Intel® 8251A is the industry standard Universal Synchronous/Asynchronous Receiver/Transmitter (USART), designed for data communications with Intel’s microprocessor families such as MCS-48, 80, 85, and iAPX-66, 88. The 8251A is used as a peripheral device and is programmed by the CPU to operate using virtually any serial data transmission technique presently in use (including IBM "bi-sync"). The USART accepts data characters from the CPU in parallel format and then converts them into a continuous serial data stream for transmission. Simultaneously, it can receive serial data streams and convert them into parallel data characters for the CPU. The USART will signal the CPU whenever it can accept a new character for transmission or whenever it has received a character for the CPU. The CPU can read the complete status of the USART at any time. These include data transmission errors and control signals such as SYNDET, TxEMPTY. The chip is fabricated using Intel’s high performance HMOS technology.
FEATURES AND ENHANCEMENTS

The 8251A is an advanced design of the industry standard USART, the Intel® 8251. The 8251A operates with an extended range of Intel microprocessors and maintains compatibility with the 8251. Familiarization time is minimal because of compatibility and involves only knowing the additional features and enhancements, and reviewing the AC and DC specifications of the 8251A.

The 8251A incorporates all the key features of the 8251 and has the following additional features and enhancements:

- The 8251A has double-buffered data paths with separate I/O registers for control, status, Data In, and Data Out, which considerably simplifies control programming and minimizes CPU overhead.
- In asynchronous operations, the Receiver detects and handles “break” automatically, relieving the CPU of this task.
- A refined Rx initialization prevents the Receiver from starting when in “break” state, preventing unwanted interrupts from a disconnected USART.
- At the conclusion of a transmission, TxD line will always return to the marking state unless SBRK is programmed.
- Tx Enable logic enhancement prevents a Tx Disable command from halting transmission until all data previously written has been transmitted. The logic also prevents the transmitter from turning off in the middle of a word.
- When External Sync Detect is programmed, Internal Sync Detect is disabled, and an External Sync Detect status is provided via a flip-flop which clears itself upon a status read.
- Possibility of false sync detect is minimized by ensuring that if double character sync is programmed, the characters be contiguously detected and also by clearing the Rx register to all ones whenever Enter Hunt command is issued in Sync mode.
- As long as the 8251A is not selected, the RD and WR do not affect the internal operation of the device.
- The 8251A Status can be read at any time but the status update will be inhibited during status read.
- The 8251A is free from extraneous glitches and has enhanced AC and DC characteristics, providing higher speed and better operating margins.
- Synchronous Baud rate from DC to 64K.

FUNCTIONAL DESCRIPTION

General

The 8251A is a Universal Synchronous/Asynchronous Receiver/Transmitter designed for a wide range of Intel microcomputers such as 8048, 8080, 8085, 8086 and 8088. Like other I/O devices in a microcomputer system, its functional configuration is programmed by the system’s software for maximum flexibility. The 8251A can support most serial data techniques in use, including IBM “bi-sync”.

In a communication environment an interface device must convert parallel format system data into serial format for transmission and convert incoming serial format data into parallel system data for reception. The interface device must also delete or insert bits or characters that are functionally unique to the communication technique. In essence, the interface should appear “transparent” to the CPU, a simple input or output of byte-oriented system data.

Data Bus Buffer

This 3-state bidirectional, 8-bit buffer is used to interface the 8251A to the system Data Bus. Data is transmitted or received by the buffer upon execution of Input or Output instructions of the CPU. Control words, Command words and Status information are also transferred through the Data Bus Buffer. The Command Status, Data-In and Data-Out registers are separate, 8-bit registers communicating with the system bus through the Data Bus Buffer.

This functional block accepts inputs from the system Control bus and generates control signals for overall device operation. It contains the Control Word Register and Command Word Register that store the various control formats for the device functional definition.

RESET (Reset)

A “high” on this input forces the 8251A into an “Idle” mode. The device will remain at “Idle” until a new set of control words is written into the 8251A to program its functional definition. Minimum RESET pulse width is 6 tCY (clock must be running).

A command reset operation also puts the device into the “Idle” state.
CLK (Clock)
The CLK input is used to generate internal device timing and is normally connected to the Phase 2 (TTL) output of the Clock Generator. No external inputs or outputs are referenced to CLK but the frequency of CLK must be greater than 30 times the Receiver or Transmitter data bit rates.

WR (Write)
A “low” on this input informs the 8251A that the CPU is writing data or control words to the 8251A.

RD (Read)
A “low” on this input informs the 8251A that the CPU is reading data or status information from the 8251A.

C/D (Control/Data)
This input, in conjunction with the WR and RD inputs, informs the 8251A that the word on the Data Bus is either a data character, control word or status information.

<table>
<thead>
<tr>
<th>C/D</th>
<th>RD</th>
<th>WR</th>
<th>CS</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>X</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>X</td>
<td>X</td>
<td>X</td>
<td>1</td>
</tr>
</tbody>
</table>

8251A DATA → DATA BUS
DATA BUS → 8251A DATA
STATUS → DATA BUS
DATA BUS → CONTROL
DATA BUS → 3-STATE
DATA BUS → 3-STATE
CS (Chip Select)
A "low" on this input selects the 8251A. No reading or writing will occur unless the device is selected. When CS is high, the Data Bus is in the float state and RD and WR have no effect on the chip.

Modem Control
The 8251A has a set of control inputs and outputs that can be used to simplify the interface to almost any modem. The modem control signals are general purpose in nature and can be used for functions other than modem control, if necessary.

DSR (Data Set Ready)
The DSR input signal is a general-purpose, 1-bit inverting input port. Its condition can be tested by the CPU using a Status Read operation. The DSR input is normally used to test modem conditions such as Data Set Ready.

DTR (Data Terminal Ready)
The DTR output signal is a general-purpose, 1-bit inverting output port. It can be set "low" by programming the appropriate bit in the Command Instruction word. The DTR output signal is normally used for modem control such as Data Terminal Ready.

RTS (Request to Send)
The RTS output signal is a general-purpose, 1-bit inverting output port. It can be set "low" by programming the appropriate bit in the Command Instruction word. The RTS output signal is normally used for modem control such as Request to Send.

CTS (Clear to Send)
A "low" on this input enables the 8251A to transmit serial data if the Tx Enable bit in the Command byte is set to a "one". If either a Tx Enable off or CTS off condition occurs while the Tx is in operation, the Tx will transmit all the data in the USART, written prior to Tx Disable command before shutting down.

Transmitter Buffer
The Transmitter Buffer accepts parallel data from the Data Bus Buffer, converts it to a serial bit stream, inserts the appropriate characters or bits (based on the communication technique) and outputs a composite serial stream of data on the TxD output pin on the falling edge of TxC. The transmitter will begin transmission upon being enabled if CTS = 0. The TxD line will be held in the marking state immediately upon a master Reset or when Tx Enable or CTS is off or the transmitter is empty.

Transmitter Control
The Transmitter Control manages all activities associated with the transmission of serial data. It accepts and issues signals both externally and internally to accomplish this function.

TxRDY (Transmitter Ready)
This output signals the CPU that the transmitter is ready to accept a data character. The TxRDY output pin can be used as an interrupt to the system, since it is masked by TxEnable; or, for Polled operation, the CPU can check TxRDY using a Status Read operation. TxRDY is automatically reset by the leading edge of WR when a data character is loaded from the CPU.

Note that when using the Polled operation, the TxRDY status bit is not masked by TxEnable, but will only indicate the Empty/Full Status of the Tx Data Input Register.

TxE (Transmitter Empty)
When the 8251A has no characters to send, the TxEMPTY output will go "high". It resets upon receiving a character from CPU if the transmitter is enabled. TxEMPTY remains high when the transmitter is disabled. TxEMPTY can be used to indicate the end of a transmission mode, so that the CPU "knows" when to "turn the line around" in the half-duplex operational mode.

In the Synchronous mode, a "high" on this output indicates that a character has not been loaded and the SYNC character or characters are about to be or are being transmitted automatically as "fillers". Tx EMPTY does not go low when the SYNC characters are being shifted out.

2-4
**Figure 4. 8251A Block Diagram Showing Modem and Transmitter Buffer and Control Functions**

**TxC (Transmitter Clock)**

The Transmitter Clock controls the rate at which the character is to be transmitted. In the Synchronous transmission mode, the Baud Rate (1x) is equal to the \( \text{TxC} \) frequency. In Asynchronous transmission mode, the baud rate is a fraction of the actual \( \text{TxC} \) frequency. A portion of the mode instruction selects this factor; it can be 1, \( \frac{1}{16} \), or \( \frac{1}{64} \) the \( \text{TxC} \).

For Example:

- If Baud Rate equals 110 Baud, 
  \( \text{TxC} \) equals 110 Hz in the 1x mode.
  \( \text{TxC} \) equals 1.72 kHz in the 16x mode.
  \( \text{TxC} \) equals 7.04 kHz in the 64x mode.

The falling edge of \( \text{TxC} \) shifts the serial data out of the 8251A.

**Receiver Control**

This functional block manages all receiver-related activities which consists of the following features.

The RxD initialization circuit prevents the 8251A from mistaking an unused input line for an active low data line in the "break condition". Before starting to receive serial characters on the RxD line, a valid "1" must first be detected after a chip master Reset. Once this has been determined, a search for a valid low (Start bit) is enabled. This feature is only active in the asynchronous mode, and is only done once for each master Reset.

The False Start bit detection circuit prevents false starts due to a transient noise spike by first detecting the falling edge and then strobing the normal center of the Start bit (RxD = low).

Parity error detection sets the corresponding status bit.

The Framing Error status bit is set if the Stop bit is absent at the end of the data byte (asynchronous mode).

**Receiver Buffer**

The Receiver accepts serial data, converts this serial input to parallel format, checks for bits or characters that are unique to the communication technique and sends an "assembled" character to the CPU. Serial data is input to RxD pin, and is clocked in on the rising edge of RxC.

**2-5**
RxRDY (Receiver Ready)

This output indicates that the 8251A contains a character that is ready to be input to the CPU. RxRDY can be connected to the interrupt structure of the CPU or, for polled operation, the CPU can check the condition of RxRDY using a Status Read operation.

RxEnable, when off, holds RxRDY in the Reset Condition. For Asynchronous mode, to set RxRDY, the Receiver must be enabled to sense a Start Bit and a complete character must be assembled and transferred to the Data Output Register. For Synchronous mode, to set RxRDY, the Receiver must be enabled and a character must finish assembly and be transferred to the Data Output Register.

Failure to read the received character from the Rx Data Output Register prior to the assembly of the next Rx Data character will set overrun condition error and the previous character will be written over and lost. If the Rx Data is being read by the CPU when the internal transfer is occurring, overrun error will be set and the old character will be lost.

RxC (Receiver Clock)

The Receiver Clock controls the rate at which the character is to be received. In Synchronous Mode, the Baud Rate (1x) is equal to the actual frequency of RxC. In Asynchronous Mode, the Baud Rate is a fraction of the actual RxC frequency. A portion of the mode instruction selects this factor: 1, 1/16 or 1/64 the RxC.

For Example:

Baud Rate equals 300 Baud, if
RxC equals 300 Hz in the 1x mode;
RxC equals 4800 Hz in the 16x mode;
RxC equals 19.2 kHz in the 64x mode.

Baud Rate equals 2400 Baud, if
RxC equals 2400 Hz in the 1x mode;
RxC equals 38.4 kHz in the 16 mode;
RxC equals 153.6 kHz in the 64 mode.

Figure 5. 8251A Block Diagram Showing Receiver Buffer and Control Functions
Data is sampled into the 8251A on the rising edge of RxC.

**NOTE:**
In most communication systems, the 8251A will be handling both the transmission and reception operations of a single link. Consequently, the Receive and Transmit Baud Rates will be the same. Both TxC and RxC will require identical frequencies for this operation and can be tied together and connected to a single frequency source (Baud Rate Generator) to simplify the interface.

**SYNDET (SYNC Detect/BRKDET Break Detect)**

This pin is used in Synchronous Mode for SYNDET and may be used as either input or output, programmable through the Control Word. It is reset to output mode low upon RESET. When used as an output (internal Sync mode), the SYNDET pin will go “high” to indicate that the 8251A has located the SYNC character in the Receive mode. If the 8251A is programmed to use double Sync characters (bi-sync), then SYNDET will go “high” in the middle of the last bit of the second Sync character. SYNDET is automatically reset upon a Status Read operation.

When used as an input (external SYNC detect mode), a positive going signal will cause the 8251A to start assembling data characters on the rising edge of the next RxC. Once in SYNC, the “high” input signal can be removed. When External SYNC Detect is programmed, Internal SYNC Detect is disabled.

**BREAK (Async Mode Only)**

This output will go high whenever the receiver remains low through two consecutive stop bit sequences (including the start bits, data bits, and parity bits). Break Detect may also be read as a Status bit. It is reset only upon a master chip Reset or Rx Data returning to a “one” state.

---

**Figure 6. 8251A Interface to 8080 Standard System Bus**

---

2-7
DETAILED OPERATION DESCRIPTION

General

The complete functional definition of the 8251A is programmed by the system's software. A set of control words must be sent out by the CPU to initialize the 8251A to support the desired communications format. These control words will program the: BAUD RATE, CHARACTER LENGTH, NUMBER OF STOP BITS, SYNCHRONOUS or ASYNCHRONOUS OPERATION, EVEN/ODD/OFF PARITY, etc. In the Synchronous Mode, options are also provided to select either internal or external character synchronization.

Once programmed, the 8251A is ready to perform its communication functions. The TxRDY output is raised "high" to signal the CPU that the 8251A is ready to receive a data character from the CPU. This output (TxRDY) is reset automatically when the CPU writes a character into the 8251A.

On the other hand, the 8251A receives serial data from the MODEM or I/O device. Upon receiving an entire character, the RxRDY output is raised "high" to signal the CPU that the 8251A has a complete character ready for the CPU to fetch. RxRDY is reset automatically upon the CPU data read operation.

The 8251A cannot begin transmission until the Tx Enable (Transmitter Enable) bit is set in the Command Instruction and it has received a Clear To Send (CTS) input. The TxD output will be held in the marking state upon Reset.

Programming the 8251A

Prior to starting data transmission or reception, the 8251A must be loaded with a set of control words generated by the CPU. These control signals define the complete functional definition of the 8251A and must immediately follow a Reset operation (internal or external).

The control words are split into two formats:

1. Mode Instruction
2. Command Instruction

Mode Instruction

This instruction defines the general operational characteristics of the 8251A. It must follow a Reset operation (internal or external). Once the Mode Instruction has been written into the 8251A by the CPU, SYNC characters or Command Instructions may be written.

Command Instruction

This instruction defines a word that is used to control the actual operation of the 8251A.

Both the Mode and Command Instructions must conform to a specified sequence for proper device operation (see Figure 7). The Mode Instruction must be written immediately following a Reset operation, prior to using the 8251A for data communication.

All control words written into the 8251A after the Mode Instruction will load the Command Instruction. Command Instructions can be written into the 8251A at any time in the data block during the operation of the 8251A. To return to the Mode instruction format, the master Reset bit in the Command Instruction word can be set to initiate an internal Reset operation which automatically places the 8251A back into the Mode Instruction format. Command Instructions must follow the Mode Instruction or Sync characters.

Mode Instruction Definition

The 8251A can be used for either Asynchronous or Synchronous data communication. To understand how the Mode Instruction defines the functional operation of the 8251A, the designer can best view the device as two separate components, one Asynchronous and the other Synchronous, sharing the same package. The format definition can be changed only after a master chip Reset. For explanation purposes the two formats will be isolated.
NOTE:
When parity is enabled it is not considered as one of the data bits for the purpose of programming word length. The actual parity bit received on the Rx Data line cannot be read on the Data Bus. In the case of a programmed character length of less than 8 bits, the least significant Data Bus bits will hold the data; unused bits are “don’t care” when writing data to the 8251A, and will be “zeros” when reading the data from the 8251A.

Asynchronous Mode (Transmission)
Whenever a data character is sent by the CPU the 8251A automatically adds a Start bit (low level) followed by the data bits (least significant bit first), and the programmed number of Stop bits to each character. Also, an even or odd Parity bit is inserted prior to the Stop bit(s), as defined by the Mode Instruction. The character is then transmitted as a serial data stream on the TxD output. The serial data is shifted out on the falling edge of TXC at a rate equal to 1, 1/16, or 1/64 that of the TXC, as defined by the Mode Instruction. BREAK characters can be continuously sent to the TxD if commanded to do so.

When no data characters have been loaded into the 8251A the TxD output remains “high” (marking) unless a Break (continuously low) has been programmed.

Asynchronous Mode (Receive)
The RxD line is normally high. A falling edge on this line triggers the beginning of a START bit. The validity of this START bit is checked by again strobing this bit at its nominal center (16X or 64X mode only). If a low is detected again, it is a valid START bit, and the bit counter will start counting. The bit counter thus locates the center of the data bits, the parity bit (if it exists) and the stop bits. If parity error occurs, the parity error flag is set. Data and parity bits are sampled on the RxD pin with the rising edge of the RXC. If a low level is detected as the STOP bit the Framing Error flag will be set. The STOP bit signals the end of a character. Note that the receiver requires only one stop bit, regardless of the number of stop bits programmed. This character is then loaded into the parallel I/O buffer of the 8251A. The RxRDY pin is raised to signal the CPU that a character is ready to be fetched. If a previous character has not been fetched by the CPU, the present character replaces it in the I/O buffer, and the OVERRUN Error flag

![Figure 8. Mode Instruction Format, Asynchronous Mode](image-url)
is raised (thus the previous character is lost). All of
the error flags can be reset by an Error Reset In-
struction. The occurrence of any of these errors will
not affect the operation of the 8251A.

Synchronous Mode (Transmission)

The TxD output is continuously high until the CPU
sends its first character to the 8251A which usually
is a SYNC character. When the CTS line goes low,
the first character is serially transmitted out. All char-
acters are shifted out on the falling edge of TxC.
Data is shifted out at the same rate as the TxC.

Once transmission has started, the data stream at
the TxD output must continue at the TxC rate. If the
CPU does not provide the 8251A with a data charac-
ter before the 8251A Transmitter Buffers become
empty, the SYNC characters (or character if in single
SYNC character mode) will be automatically inserted
in the TxD data stream. In this case, the TxEMPTY
pin is raised high to signal that the 8251A is empty
and SYNC characters are being sent out. TxEMPTY
does not go low when the SYNC is being shifted out
(see figure below). The TxEMPTY pin is internally
reset by a data character being written into the
8251A.

TRANSMITTER OUTPUT

TxD MARKING
START
BIT
DATA BITS
PARITY
BIT
STOP
BIT

GENERATED
BY 8251A

DOES NOT APPEAR
ON THE DATA BUS

RECEIVER INPUT

RxD
START
BIT
DATA BITS
PARITY
BIT
STOP
BIT

PROGRAMMED
CHARACTER
LENGTH

TRANSMISSION FORMAT

CPU BYTE (5-8 BITS/CHAR)

DATA CHARACTER

ASSEMBLED SERIAL DATA OUTPUT (TxD)

START
BIT
DATA CHARACTER
PARITY
BIT
STOP
BIT

RECEIVE FORMAT

SERIAL DATA INPUT (RxD)

START
BIT
DATA CHARACTER
PARITY
BIT
STOP
BIT

CPU BYTE (5-8 BITS/CHAR)*

DATA CHARACTER

*NOTE:
If character length is defined as 5, 6, or 7 bits the unused bits are set to “zero”.

Figure 9. Asynchronous Mode
Synchronous Mode (Receive)

In this mode, character synchronization can be internally or externally achieved. If the SYNC mode has been programmed, ENTER HUNT command should be included in the first command instruction word written. Data on the RxD pin is then sampled on the rising edge of RxC. The content of the Rx buffer is compared at every bit boundary with the first SYNC character until a match occurs. If the 8251A has been programmed for two SYNC characters, the subsequent received character is also compared; when both SYNC characters have been detected, the USART ends the HUNT mode and is in character synchronization. The SYNDET pin is then set high, and is reset automatically by a STATUS READ. If parity is programmed, SYNDET will not be set until the middle of the parity bit instead of the middle of the last data bit.

In the external SYNC mode, synchronization is achieved by applying a high level on the SYNDET pin, thus forcing the 8251A out of the HUNT mode. The high level can be removed after one RxC cycle. An ENTER HUNT command has no effect in the asynchronous mode of operation.

Figure 10. Mode Instruction Format, Synchronous Mode
Parity error and overrun error are both checked in the same way as in the Asynchronous Rx mode. Parity is checked when not in Hunt, regardless of whether the Receiver is enabled or not.

The CPU can command the receiver to enter the HUNT mode if synchronization is lost. This will also set all the used character bits in the buffer to a "one," thus preventing a possible false SYNDET caused by data that happens to be in the Rx Buffer at ENTER HUNT time. Note that the SYNDET F/F is reset at each Status Read, regardless of whether internal or external SYNC has been programmed. This does not cause the 8251A to return to the HUNT mode. When in SYNC mode, but not in HUNT, Sync Detection is still functional, but only occurs at the "known" word boundaries. Thus, if one Status Read indicates SYNDET and a second Status Read also indicates SYNDET, then the programmed SYNDET characters have been received since the previous Status Read. (If double character sync has been programmed, then both sync characters have been contiguously received to gate a SYNDET indication). When external SYNDET mode is selected, internal Sync Detect is disabled, and the SYNDET F/F may be set at any bit boundary.

Sync characters are loaded (if in Sync Mode) then the device is ready to be used for data communication. The Command Instruction controls the actual operation of the selected format. Functions such as: Enable Transmit/Receive, Error Reset and Modem Controls are provided by the Command instruction.

Once the Mode Instruction has been written into the 8251A and Sync characters inserted, of necessary, then all further "control writes" (C/D = 1) will load a Command Instruction. A Reset Operation (internal or external) will return the 8251A to the Mode Instruction format.

NOTE:
Internal Reset on Power-up:

When power is first applied, the 8251A may come up in the Mode, Sync character or Command format. To guarantee that the device is in the Command Instruction format before the Reset command is issued, it is safest to execute the worst-case initialization sequence (sync mode with two sync characters). Loading three 00Hs consecutively into the device with C/D = 1 configures sync operation and writes two dummy 00H sync characters. An Internal Reset command (40H) may then be issued to return the device to the "idle" state.

COMMAND INSTRUCTION DEFINITION

Once the functional definition of the 8251A has been programmed by the Mode Instruction and the Sync characters are loaded (if in Sync Mode) then the device is ready to be used for data communication. The Command Instruction controls the actual operation of the selected format. Functions such as: Enable Transmit/Receive, Error Reset and Modem Controls are provided by the Command instruction.

Once the Mode Instruction has been written into the 8251A and Sync characters inserted, of necessary, then all further "control writes" (C/D = 1) will load a Command Instruction. A Reset Operation (internal or external) will return the 8251A to the Mode Instruction format.

NOTE:

Internal Reset on Power-up:

When power is first applied, the 8251A may come up in the Mode, Sync character or Command format. To guarantee that the device is in the Command Instruction format before the Reset command is issued, it is safest to execute the worst-case initialization sequence (sync mode with two sync characters). Loading three 00Hs consecutively into the device with C/D = 1 configures sync operation and writes two dummy 00H sync characters. An Internal Reset command (40H) may then be issued to return the device to the "idle" state.

COMMAND INSTRUCTION DEFINITION

Once the functional definition of the 8251A has been programmed by the Mode Instruction and the

Sync characters are loaded (if in Sync Mode) then the device is ready to be used for data communication. The Command Instruction controls the actual operation of the selected format. Functions such as: Enable Transmit/Receive, Error Reset and Modem Controls are provided by the Command instruction.

Once the Mode Instruction has been written into the 8251A and Sync characters inserted, of necessary, then all further "control writes" (C/D = 1) will load a Command Instruction. A Reset Operation (internal or external) will return the 8251A to the Mode Instruction format.

NOTE:

Internal Reset on Power-up:

When power is first applied, the 8251A may come up in the Mode, Sync character or Command format. To guarantee that the device is in the Command Instruction format before the Reset command is issued, it is safest to execute the worst-case initialization sequence (sync mode with two sync characters). Loading three 00Hs consecutively into the device with C/D = 1 configures sync operation and writes two dummy 00H sync characters. An Internal Reset command (40H) may then be issued to return the device to the "idle" state.

COMMAND INSTRUCTION DEFINITION

Once the functional definition of the 8251A has been programmed by the Mode Instruction and the

Sync characters are loaded (if in Sync Mode) then the device is ready to be used for data communication. The Command Instruction controls the actual operation of the selected format. Functions such as: Enable Transmit/Receive, Error Reset and Modem Controls are provided by the Command instruction.

Once the Mode Instruction has been written into the 8251A and Sync characters inserted, of necessary, then all further "control writes" (C/D = 1) will load a Command Instruction. A Reset Operation (internal or external) will return the 8251A to the Mode Instruction format.

NOTE:

Internal Reset on Power-up:

When power is first applied, the 8251A may come up in the Mode, Sync character or Command format. To guarantee that the device is in the Command Instruction format before the Reset command is issued, it is safest to execute the worst-case initialization sequence (sync mode with two sync characters). Loading three 00Hs consecutively into the device with C/D = 1 configures sync operation and writes two dummy 00H sync characters. An Internal Reset command (40H) may then be issued to return the device to the "idle" state.

COMMAND INSTRUCTION DEFINITION

Once the functional definition of the 8251A has been programmed by the Mode Instruction and the

Sync characters are loaded (if in Sync Mode) then the device is ready to be used for data communication. The Command Instruction controls the actual operation of the selected format. Functions such as: Enable Transmit/Receive, Error Reset and Modem Controls are provided by the Command instruction.

Once the Mode Instruction has been written into the 8251A and Sync characters inserted, of necessary, then all further "control writes" (C/D = 1) will load a Command Instruction. A Reset Operation (internal or external) will return the 8251A to the Mode Instruction format.

NOTE:

Internal Reset on Power-up:

When power is first applied, the 8251A may come up in the Mode, Sync character or Command format. To guarantee that the device is in the Command Instruction format before the Reset command is issued, it is safest to execute the worst-case initialization sequence (sync mode with two sync characters). Loading three 00Hs consecutively into the device with C/D = 1 configures sync operation and writes two dummy 00H sync characters. An Internal Reset command (40H) may then be issued to return the device to the "idle" state.
NOTE: Error Reset must be performed whenever RxEnable and Enter Hunt are programmed.

Figure 12. Command Instruction Format

STATUS READ DEFINITION

In data communication systems it is often necessary to examine the "status" of the active device to ascertain if errors have occurred or other conditions that require the processor’s attention. The 8251A has facilities that allow the programmer to "read" the status of the device at any time during the functional operation. (Status update is inhibited during status read.)

A normal "read" command is issued by the CPU with \( C/D = 1 \) to accomplish this function.

Some of the bits in the Status Read Format have identical meanings to external output pins so that the 8251A can be used in a completely polled or interrupt-driven environment. TxRDY is an exception.
Note that status update can have a maximum delay of 28 clock periods from the actual event affecting the status.

---

**Figure 13. Status Read Format**

- **DSR**
- **SYNDET/BRKDET**
- **FE**
- **OE**
- **PE**
- **TxEMPTY**
- **RxRDY**
- **TxRDY**

**Note 1:**

SAME DEFINITIONS AS I/O PINS

**Parity Error**

The PE flag is set when a parity error is detected. It is reset by the ER bit of the Command Instruction. PE does not inhibit operation of the 8251A.

**Overrun Error**

The OE flag is set when the CPU does not read a character before the next one becomes available. It is reset by the ER bit of the Command Instruction. OE does not inhibit operation of the 8251A, however, the previously overrun character is lost.

**Framing Error** (Async only)

The FE flag is set when a valid Stop bit is not detected at the end of every character. It is reset by the ER bit of the Command Instruction. FE does not inhibit the operation of the 8251A.

**Data Set Ready**

Indicates that the DSR is at a zero level.

---

**NOTE:**

1. TxRDY status bit has different meanings from the TxRDY output pin. The former is not conditioned by CTS and TxEN; the latter is conditioned by both CTS and TxEN.
2. TxRDY status bit = DB Buffer Empty
3. TxRDY output = DB Buffer Empty • (CTS = 0) • (TxEN = 1)
APPLICATIONS OF THE 8251A

Figure 14. Asynchronous Serial Interface to CRT Terminal, DC—9600 Baud

Figure 15. Synchronous Interface to Terminal or Peripheral Device
APPLICATIONS OF THE 8251A (Continued)

Figure 16. Asynchronous Interface to Telephone Lines

Figure 17. Synchronous Interface to Telephone Lines

NOTES:
1. AC timings measured $V_{OH} = 2.0 \, V_{OL} = 0.8$, and with load circuit of Figure 18.
2. Chip Select (CS) and Command/Data (C/D) are considered as Addresses.
3. Assumes that Address is valid before Rd.
4. This recovery time is for Mode initialization only. Write Data is allowed only when TxRDY = 1. Recovery Time between Writes for Asynchronous Mode is 8 $t_{CY}$ and for Synchronous Mode is 16 $t_{CY}$.
5. The $f_{TX}$ and $f_{RX}$ frequencies have the following limitations with respect to CLK: For 1x Baud Rate, $f_{TX} \text{ or } f_{RX} \leq 1/(30 \, t_{CY})$; For 16x and 64x Baud Rate, $f_{TX} \text{ or } f_{RX} \leq 1/(4.5 \, t_{CY})$. This applies to Baud Rates less than or equal to 64K Baud.
6. Reset Pulse Width = 6 $t_{CY}$ minimum; System clock must be running during Reset.
7. Status update can have a maximum delay of 28 clock periods from the event affecting the status.
8. In external sync mode the test spec. requires the ratio of the system clock (clock) to receive or transmit bit ratios to be greater than 34.
9. A float is defined as the point where the data bus falls below a logic 1 (2.0V @ $I_{OH}$ limit) or rises above a Logic 0 (0.8V @ $I_{OL}$ limit).
ABSOLUTE MAXIMUM RATINGS*  
Ambient Temperature Under Bias ........ 0°C to 70°C  
Storage Temperature ............. −65°C to +150°C  
Voltage on Any Pin  
with Respect to Ground ........... −0.5V to +7V  
Power Dissipation .................. 1W  
*Notice: Stresses above those listed under “Absolute Maximum Ratings” may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions above those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

D.C. CHARACTERISTICS \( T_A = 0°C \) to 70°C, \( V_{CC} = 5.0V \pm 10\% \), GND = 0V*  

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Parameter</th>
<th>Min</th>
<th>Max</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>( V_{IL} )</td>
<td>Input Low Voltage</td>
<td>−0.5</td>
<td>0.8</td>
<td>V</td>
</tr>
<tr>
<td>( V_{IH} )</td>
<td>Input High Voltage</td>
<td>2.0</td>
<td>( V_{CC} )</td>
<td>V</td>
</tr>
<tr>
<td>( V_{OL} )</td>
<td>Output Low Voltage</td>
<td>0.45</td>
<td></td>
<td>V</td>
</tr>
<tr>
<td>( V_{OH} )</td>
<td>Output High Voltage</td>
<td>2.4</td>
<td></td>
<td>V</td>
</tr>
<tr>
<td>( I_{OFL} )</td>
<td>Output Float Leakage</td>
<td>±10</td>
<td>( \mu A )</td>
<td></td>
</tr>
<tr>
<td>( I_{IL} )</td>
<td>Input Leakage</td>
<td>±10</td>
<td>( \mu A )</td>
<td></td>
</tr>
<tr>
<td>( I_{CC} )</td>
<td>Power Supply Current</td>
<td>100</td>
<td>ma</td>
<td></td>
</tr>
</tbody>
</table>

CAPACITANCE \( T_A = 25°C \), \( V_{CC} = \) GND = 0V  

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Parameter</th>
<th>Min</th>
<th>Max</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>( C_{IN} )</td>
<td>Input Capacitance</td>
<td>10</td>
<td>pF</td>
<td>( f_c = 1 \text{ MHz} )</td>
</tr>
<tr>
<td>( C_{I/O} )</td>
<td>I/O Capacitance</td>
<td>20 pF</td>
<td></td>
<td>Unmeasured pins returned to GND</td>
</tr>
</tbody>
</table>

A.C. CHARACTERISTICS \( T_A = 0°C \) to 70°C, \( V_{CC} = 5.0V \pm 10\% \), GND = 0V*  

Bus Parameters (Note 1)  

READ CYCLE  

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Parameter</th>
<th>Min</th>
<th>Max</th>
<th>Unit</th>
<th>Test Conditions</th>
</tr>
</thead>
<tbody>
<tr>
<td>( t_{AR} )</td>
<td>Address Stable Before READ (CS, C/D)</td>
<td>0</td>
<td>ns</td>
<td></td>
<td>(Note 2)</td>
</tr>
<tr>
<td>( t_{RA} )</td>
<td>Address Hold Time for READ (CS, C/D)</td>
<td>0</td>
<td>ns</td>
<td></td>
<td>(Note 2)</td>
</tr>
<tr>
<td>( t_{RR} )</td>
<td>READ Pulse Width</td>
<td>250</td>
<td>ns</td>
<td></td>
<td></td>
</tr>
<tr>
<td>( t_{RD} )</td>
<td>Data Delay from READ</td>
<td>200</td>
<td>ns</td>
<td>3, ( C_L = 150 \text{ pF} )</td>
<td></td>
</tr>
<tr>
<td>( t_{DF} )</td>
<td>READ to Data Floating</td>
<td>10</td>
<td>100</td>
<td>ns</td>
<td>(Note 1, 9)</td>
</tr>
</tbody>
</table>

WRITE CYCLE  

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Parameter</th>
<th>Min</th>
<th>Max</th>
<th>Unit</th>
<th>Test Conditions</th>
</tr>
</thead>
<tbody>
<tr>
<td>( t_{AW} )</td>
<td>Address Stable Before WRITE</td>
<td>0</td>
<td>ns</td>
<td></td>
<td></td>
</tr>
<tr>
<td>( t_{WA} )</td>
<td>Address Hold Time for WRITE</td>
<td>0</td>
<td>ns</td>
<td></td>
<td></td>
</tr>
<tr>
<td>( t_{WW} )</td>
<td>WRITE Pulse Width</td>
<td>250</td>
<td>ns</td>
<td></td>
<td></td>
</tr>
<tr>
<td>( t_{DW} )</td>
<td>Data Set-Up Time for WRITE</td>
<td>150</td>
<td>ns</td>
<td></td>
<td></td>
</tr>
<tr>
<td>( t_{WD} )</td>
<td>Data Hold Time for WRITE</td>
<td>20</td>
<td>ns</td>
<td></td>
<td></td>
</tr>
<tr>
<td>( t_{RV} )</td>
<td>Recovery Time Between WRITES</td>
<td>6</td>
<td>( t_{CY} )</td>
<td>(Note 4)</td>
<td></td>
</tr>
</tbody>
</table>
## A.C. CHARACTERISTICS (Continued)

### OTHER TIMINGS

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Parameter</th>
<th>Min</th>
<th>Max</th>
<th>Unit</th>
<th>Test Conditions</th>
</tr>
</thead>
<tbody>
<tr>
<td>t&lt;sub&gt;cy&lt;/sub&gt;</td>
<td>Clock Period</td>
<td>320</td>
<td>1350</td>
<td>ns</td>
<td>(Note 5, 6)</td>
</tr>
<tr>
<td>t&lt;sub&gt;φ&lt;/sub&gt;</td>
<td>Clock High Pulse Width</td>
<td>120</td>
<td>t&lt;sub&gt;cy&lt;/sub&gt; − 90</td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>t&lt;sub&gt;CF&lt;/sub&gt;</td>
<td>Clock Low Pulse Width</td>
<td>90</td>
<td></td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>tr, tf</td>
<td>Clock Rise and Fall Time</td>
<td>20</td>
<td></td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>t&lt;sub&gt;dtx&lt;/sub&gt;</td>
<td>TxD Delay from Falling Edge of TxC</td>
<td>1</td>
<td></td>
<td>μs</td>
<td></td>
</tr>
<tr>
<td>f&lt;sub&gt;tx&lt;/sub&gt;</td>
<td>Transmitter Input Clock Frequency</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>1x Baud Rate</td>
<td>DC</td>
<td>64</td>
<td>kHz</td>
<td></td>
</tr>
<tr>
<td></td>
<td>16x Baud Rate</td>
<td>DC</td>
<td>310</td>
<td>kHz</td>
<td></td>
</tr>
<tr>
<td></td>
<td>64x Baud Rate</td>
<td>DC</td>
<td>615</td>
<td>kHz</td>
<td></td>
</tr>
<tr>
<td>t&lt;sub&gt;tpw&lt;/sub&gt;</td>
<td>Transmitter Input Clock Pulse Width</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>1x Baud Rate</td>
<td>12</td>
<td></td>
<td>t&lt;sub&gt;cy&lt;/sub&gt;</td>
<td></td>
</tr>
<tr>
<td></td>
<td>16x and 64x Baud Rate</td>
<td>1</td>
<td></td>
<td>t&lt;sub&gt;cy&lt;/sub&gt;</td>
<td></td>
</tr>
<tr>
<td>t&lt;sub&gt;tpd&lt;/sub&gt;</td>
<td>Transmitter Input Clock Pulse Delay</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>1x Baud Rate</td>
<td>15</td>
<td></td>
<td>t&lt;sub&gt;cy&lt;/sub&gt;</td>
<td></td>
</tr>
<tr>
<td></td>
<td>16x and 64x Baud Rate</td>
<td>3</td>
<td></td>
<td>t&lt;sub&gt;cy&lt;/sub&gt;</td>
<td></td>
</tr>
<tr>
<td>f&lt;sub&gt;rx&lt;/sub&gt;</td>
<td>Receiver Input Clock Frequency</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>1x Baud Rate</td>
<td>DC</td>
<td>64</td>
<td>kHz</td>
<td></td>
</tr>
<tr>
<td></td>
<td>16x Baud Rate</td>
<td>DC</td>
<td>310</td>
<td>kHz</td>
<td></td>
</tr>
<tr>
<td></td>
<td>64x Baud Rate</td>
<td>DC</td>
<td>615</td>
<td>kHz</td>
<td></td>
</tr>
<tr>
<td>t&lt;sub&gt;rpw&lt;/sub&gt;</td>
<td>Receiver Input Clock Pulse Width</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>1x Baud Rate</td>
<td>12</td>
<td></td>
<td>t&lt;sub&gt;cy&lt;/sub&gt;</td>
<td></td>
</tr>
<tr>
<td></td>
<td>16x and 64x Baud Rate</td>
<td>1</td>
<td></td>
<td>t&lt;sub&gt;cy&lt;/sub&gt;</td>
<td></td>
</tr>
<tr>
<td>t&lt;sub&gt;rpd&lt;/sub&gt;</td>
<td>Receiver Input Clock Pulse Delay</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>1x Baud Rate</td>
<td>15</td>
<td></td>
<td>t&lt;sub&gt;cy&lt;/sub&gt;</td>
<td></td>
</tr>
<tr>
<td></td>
<td>16x and 64x Baud Rate</td>
<td>3</td>
<td></td>
<td>t&lt;sub&gt;cy&lt;/sub&gt;</td>
<td></td>
</tr>
<tr>
<td>t&lt;sub&gt;txrdy&lt;/sub&gt;</td>
<td>TxRDY Pin Delay from Center of Last Bit</td>
<td>14</td>
<td></td>
<td>t&lt;sub&gt;cy&lt;/sub&gt;</td>
<td>(Note 7)</td>
</tr>
<tr>
<td>t&lt;sub&gt;txrdy&lt;/sub&gt; CLEAR</td>
<td>TxRDY ↓ from Leading Edge of WR</td>
<td>400</td>
<td></td>
<td>ns</td>
<td>(Note 7)</td>
</tr>
<tr>
<td>t&lt;sub&gt;rxrdy&lt;/sub&gt;</td>
<td>RxRDY Pin Delay from Center of Last Bit</td>
<td>26</td>
<td></td>
<td>t&lt;sub&gt;cy&lt;/sub&gt;</td>
<td>(Note 7)</td>
</tr>
<tr>
<td>t&lt;sub&gt;rxrdy&lt;/sub&gt; CLEAR</td>
<td>RxRDY ↓ from Leading Edge of RD</td>
<td>400</td>
<td></td>
<td>ns</td>
<td>(Note 7)</td>
</tr>
<tr>
<td>t&lt;sub&gt;s&lt;/sub&gt;</td>
<td>Internal SYNDET Delay from Rising Edge of RXC</td>
<td>26</td>
<td></td>
<td>t&lt;sub&gt;cy&lt;/sub&gt;</td>
<td>(Note 7)</td>
</tr>
<tr>
<td>t&lt;sub&gt;es&lt;/sub&gt;</td>
<td>External SYNDET Set-Up Time After Rising Edge of RXC</td>
<td>16 t&lt;sub&gt;cy&lt;/sub&gt;</td>
<td>t&lt;sub&gt;rpd&lt;/sub&gt; − t&lt;sub&gt;cy&lt;/sub&gt;</td>
<td>ns</td>
<td>(Note 7)</td>
</tr>
<tr>
<td>t&lt;sub&gt;txempty&lt;/sub&gt;</td>
<td>TxEMPTY Delay from Center of Last Bit</td>
<td>20</td>
<td></td>
<td>t&lt;sub&gt;cy&lt;/sub&gt;</td>
<td>(Note 7)</td>
</tr>
<tr>
<td>t&lt;sub&gt;wc&lt;/sub&gt;</td>
<td>Control Delay from Rising Edge of WRITE (TxEn, DTR, RTS)</td>
<td>8</td>
<td></td>
<td>t&lt;sub&gt;cy&lt;/sub&gt;</td>
<td>(Note 7)</td>
</tr>
<tr>
<td>t&lt;sub&gt;cr&lt;/sub&gt;</td>
<td>Control to READ Set-Up Time (DSR, CTS)</td>
<td>20</td>
<td></td>
<td>t&lt;sub&gt;cy&lt;/sub&gt;</td>
<td>(Note 7)</td>
</tr>
</tbody>
</table>

*NOTE:
For Extended Temperature EXPRESS, use MIL 8251A electrical parameters.
A.C. CHARACTERISTICS (Continued)

TYPICAL Δ OUTPUT DELAY VS. Δ CAPACITANCE (pF)

<table>
<thead>
<tr>
<th>Δ OUTPUT DELAY (ns)</th>
</tr>
</thead>
<tbody>
<tr>
<td>+20</td>
</tr>
<tr>
<td>+10</td>
</tr>
<tr>
<td>0</td>
</tr>
<tr>
<td>-10</td>
</tr>
<tr>
<td>-20</td>
</tr>
<tr>
<td>-50</td>
</tr>
<tr>
<td>0</td>
</tr>
<tr>
<td>+50</td>
</tr>
<tr>
<td>+100</td>
</tr>
</tbody>
</table>

Δ CAPACITANCE (pF)

Figure 18

A.C. TESTING INPUT, OUTPUT WAVEFORM

AC Testing: Inputs are driven at 2.4V for a Logic "1" and 0.45V for a Logic "0". Timing measurements are made at 2.0V for a Logic "1" and 0.8V for a Logic "0".

A.C. TESTING LOAD CIRCUIT

C_L = 150 pF

Figure 18

WAVEFORMS

SYSTEM CLOCK INPUT
WAVEFORMS (Continued)

WRITE DATA CYCLE (CPU → USART)

![Write Waveform Diagram](image1)

READ DATA CYCLE (CPU ← USART)

![Read Waveform Diagram](image2)
WAVEFORMS (Continued)

WRITE CONTROL OR OUTPUT PORT CYCLE (CPU → USART)

READ CONTROL OR INPUT PORT (CPU ← USART)

NOTES:
1. TWC includes the response timing of a control byte.
2. TCR includes the effect of CTS on the TxENBL circuitry.
TRANSMITTER CONTROL AND FLAG TIMING (ASYNC MODE)

Example Format = 7 Bit Character With Parity & 2 Stop Bits.

RECEIVER CONTROL AND FLAG TIMING (ASYNC MODE)

Example Format = 7 Bit Character With Parity & 2 Stop Bits
TRANSMITTER CONTROL AND FLAG TIMING (SYNC MODE)

Example Format = 5 Bit Character With Parity, 2 Sync Characters.
NOTES:
1. Internal Sync, 2 Sync Characters, 5 Bits With Parity.
2. External Sync, 5 Bits, With Parity.
The Intel CHMOS 82050 Asynchronous Communications Controller is a low cost, higher performance alternative to the INS 16450—it emulates the INS 16450 and provides 100% compatibility with IBM PC software. Its 28-lead package provides all the functionality necessary for an IBM PC environment while substantially decreasing board space. The 82050's simpler system interface reduces TTL glue—especially for higher frequency PC bus designs. The 82050 is specifically designed to provide a low cost, high-performance integrated modem solution when combined with Intel's 89024 modem chip set. The compact 28-pin 82050 is fabricated using CHMOS III technology for decreased power consumption and increased reliability.
### 82050 Pinout Definition

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Pin No.</th>
<th>Pin Type</th>
<th>Name and Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RESET</td>
<td>17</td>
<td>I</td>
<td><strong>RESET</strong>: A high on this input pin resets the 82050.</td>
</tr>
<tr>
<td>CS</td>
<td>18</td>
<td>I</td>
<td><strong>CHIP SELECT</strong>: A low on this input pin enables the 82050 and allows read or write operations.</td>
</tr>
<tr>
<td>A2–A0</td>
<td>24–22</td>
<td>I</td>
<td><strong>ADDRESS PINS</strong>: These inputs interface with three bits of the system address bus to select one of the internal registers for read or write.</td>
</tr>
<tr>
<td>D7–D0</td>
<td>1–4</td>
<td>I/O</td>
<td><strong>DATA BUS</strong>: Bi-directional, three state, 8-Bit Data Bus. These pins allow transfer of bytes between the CPU and the 82050.</td>
</tr>
<tr>
<td>RD</td>
<td>20</td>
<td>I</td>
<td><strong>READ</strong>: A low on this input pin allows the CPU to read data or status bytes from the 82050.</td>
</tr>
<tr>
<td>WR</td>
<td>19</td>
<td>I</td>
<td><strong>WRITE</strong>: A low on this input pin allows the CPU to write data or control bytes to the 82050.</td>
</tr>
<tr>
<td>INT</td>
<td>5</td>
<td>O</td>
<td><strong>INTERRUPT</strong>: A high on this output pin signals an interrupt request to the CPU. The CPU may determine the particular source and cause of the interrupt by reading the 82050 status registers.</td>
</tr>
<tr>
<td>CLK/X1</td>
<td>9</td>
<td>I</td>
<td><strong>MULTIFUNCTION</strong>: This input pin serves as a source for the internal system clock. The clock may be asynchronous to the serial clocks and to the processor clock. This pin may be used in one of two modes: CLK-in this mode an externally generated clock should be used to drive this input pin; X1-in this mode the clock is generated by a crystal to be connected between this pin (X1) and the X2 pin. (See system clock generation.)</td>
</tr>
<tr>
<td>OUT2/X2</td>
<td>8</td>
<td>O</td>
<td><strong>MULTIFUNCTION</strong>: This is a dual-function pin which may be configured to one of the following functions: OUT2—a general purpose output pin controlled by the CPU is only available when the CLK/X1 pin is driven by an externally generated clock; X2—this pin serves as an output pin for the crystal oscillator. Note: The configuration of pin is done during hardware reset. For more details refer to the system clock generation.</td>
</tr>
</tbody>
</table>
### 82050 PINOUT DEFINITION (Continued)

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Pin No.</th>
<th>Type</th>
<th>Name and Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TXD</td>
<td>6</td>
<td>O</td>
<td>TRANSMIT DATA: Serial data is transmitted via this output pin starting at the least significant bit.</td>
</tr>
<tr>
<td>RXD</td>
<td>13</td>
<td>I</td>
<td>RECEIVE DATA: Serial data is received on this input pin starting at the least significant bit.</td>
</tr>
<tr>
<td>RI</td>
<td>10</td>
<td>I</td>
<td>RING INDICATION: RI - Ring indicator—input, active low. This is a general purpose input accessible by the CPU.</td>
</tr>
<tr>
<td>DTR</td>
<td>15</td>
<td>O</td>
<td>DTR—DATA TERMINAL READY: Output, active low. This is a general purpose output pin controlled by the CPU. During hardware reset, this pin is an input used to determine the system clock mode. (See System Clock Generation.)</td>
</tr>
<tr>
<td>DSR</td>
<td>11</td>
<td>I/O</td>
<td>DSR—DATA SET READY: Input, active low. This is a general purpose input pin accessible by the CPU.</td>
</tr>
<tr>
<td>RTS</td>
<td>16</td>
<td>O</td>
<td>RTS—REQUEST TO SEND: Output, active low. This is a general purpose output pin controlled by the CPU. During hardware reset, this pin is an input used to determine the system clock mode. (See system clock generation)</td>
</tr>
<tr>
<td>CTS</td>
<td>14</td>
<td>I</td>
<td>CLEAR TO SEND: Input active low. This is a general purpose input pin accessible by the CPU.</td>
</tr>
<tr>
<td>DCO</td>
<td>12</td>
<td>I/O</td>
<td>DCD—DATA CARRIER DETECTED: Input, active low. This is a general purpose input pin accessible by the CPU.</td>
</tr>
<tr>
<td>VCC</td>
<td>21</td>
<td>P</td>
<td>VCC: Device power supply.</td>
</tr>
<tr>
<td>VSS</td>
<td>7</td>
<td>P</td>
<td>VSS: Ground.</td>
</tr>
</tbody>
</table>

### SYSTEM INTERFACE

The 82050 has a simple demultiplexed bus interface which consists of a bidirectional, three-state, 8-bit data bus and a 3-bit address bus. The Reset, Chip Select, Read, and Write pins, along with the Interrupt pin, provide the remaining signals necessary to interface to the CPU. The 82050’s system clock can be generated externally and provided through the CLK pin; or its on-chip crystal oscillator can be used by attaching a crystal to the X1 and X2 pins. For compatibility with IBM PC software, a system clock of 18.432 MHz (with divide by two enabled) is recommended. The 82050, along with a transceiver, address decoder, and a crystal, complete the interface to the IBM PC Bus.

### SYSTEM CLOCK OPTIONS

The 82050 has two modes of system clock operation. It can accept an externally generated clock, or use a crystal to internally generate its system clock by using the on-chip oscillator.

The 82050 has an on-chip oscillator which can be used to generate its system clock. The oscillator will take the input from a crystal attached to the X1 and

### CRYSTAL OSCILLATOR

![Crystal Oscillator Diagram](image)

Parallel Resonant Crystal Freq. Max = 18.432 MHz
(Divided by 2)

Figure 4. Crystal Oscillator

X2 pins. The oscillator frequency is divided by two before being inputted into the chip circuitry. If an 18.432 MHz crystal is used, then the actual system clock frequency of the 82050 will be 9.216 MHz. This mode is configured via a strapping option on the RTS pin.

It is very important to distinguish between the clock frequency being supplied into the 82050 and the system clock frequency. The term system clock refers to the clock frequency being supplied to the 82050 circuitry (divided or undivided). The following examples delineate the three options for clock usage and their effect on the 82050 system clock as well as on the BRG source frequency:
1. **Crystal Oscillator:** (Maximum 18.432 MHz)
   - System Clock Frequency = Crystal Frequency/2
   - BRG Source Clock Frequency = Crystal Frequency/10

2. **External Clock (Divide by Two Enabled):** (Maximum 18.432 MHz)
   - System Clock Frequency = External Clock Frequency/2
   - BRG Source Clock Frequency = External Clock Frequency/10

3. **External Clock (Divide by Two Disabled):** (Maximum 9.216 MHz)
   - System Clock Freq. = External Clock Frequency
   - BRG Source Clock Freq. = External Clock Frequency/5

### NOTE:
The use of the Divide by Two strapping option in the crystal oscillator mode is forbidden.

### BAUD RATE GENERATION

The 82050 has a programmable 16-bit Baud Rate Generator (BRG). The 16X baud rate is generated by dividing the source clock with the divisor count from the BRG divisor registers (BAL, BAH). The BRG source clock is the 82050 system clock divided by five. If using an actual 82050 system clock of 9.216 MHz, then the BRG source clock will be 9.216 MHz/5 = 1.8432 MHz, which is compatible with the BRG source clock fed into the IBM PC serial port BRG. This allows the 82050, while using a faster system clock, to maintain full compatibility with software divisor calculations based on the 1.8432 MHz clock used in the IBM PC.

### RESET

The 82050 can be reset by asserting the RESET pin. The RESET pin must be held high for at least 8 system clock cycles. If using crystal oscillator, a reset pulse at least 1 ms should be used to ensure oscillator start up. Upon reset, all 82050 registers (except TXD and RXD) are returned to their default states.

### INTERRUPTS

The INT pin will go high, or active, whenever one of the following conditions occurs provided it is enabled in the interrupt enable register (IER):

a. Receive Machine Error or Break Condition
b. Receive Data Available
c. Transmit Data Register Empty
d. Change in the State of the Modem Input Pins

The INT pin will be reset (low) when the interrupt source is serviced. The Interrupt Identification Register (IIR) along with the Line Status Register (LSR) and the Modem Status Register (MSR) can be used to identify the source requesting service. The IIR register identifies one of the four conditions listed above. The particular event or status, which triggers the interrupt mechanism, can be identified by reading either the Line Status Register or the Modem Status register. If multiple interrupt sources become active at any one time, then highest priority interrupt source is reflected in the IIR register when the interrupt pin becomes active. Once the highest priority interrupt is serviced, then the next highest priority

---

**Figure 5. Strapping**

During the power up or reset the RTS pin is an input; it is weakly pulled high internally and sampled by the falling edge of reset. If it is driven low externally, then the 82050 is configured for a crystal oscillator; otherwise an externally generated clock is expected.

**EXTERNALLY GENERATED SYSTEM CLOCK**

This is the default mode of system clock operation. The system clock is divided by two; however, the user may disable the divide by two by a hardware strapping option on the DTR pin. The strapping option is similar to the one used on the RTS pin.
The 82050 transmission mechanism involves the TX Machine and the TXD Register. The TX Machine reads characters from the TXD Register, serializes the bits, and transmits them over the TXD pin according to signals provided for transmission by the Baud Rate Generator. It also generates parity, and break transmissions upon CPU request.

RECEIVE

The 82050 reception mechanism involves the RX Machine and the RXD Register. The RX Machine assembles the incoming characters, and loads them onto the RXD Register. The RX Machine synchronizes the data, passes it through a digital filter to filter out spikes, and then uses three samples to generate the bit polarity.

SOFTWARE INTERFACE

Like other I/O based peripherals, the 82050 is programmed through its registers to support a variety of functions. The 82050 register set is identical to the 16450 register set to provide compatibility with software written for the IBM PC. The 82050 register set occupies eight addresses and includes control, status, and data registers. The three address lines and the Divisor Latch Access Bit are used to select the 82050 registers.
### REGISTER DESCRIPTION

**Register Map**

<table>
<thead>
<tr>
<th>Register</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
<th>Address</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>TxD</td>
<td>Tx Data Bit 7</td>
<td>Tx Data Bit 6</td>
<td>Tx Data Bit 5</td>
<td>Tx Data Bit 4</td>
<td>Tx Data Bit 3</td>
<td>Tx Data Bit 2</td>
<td>Tx Data Bit 1</td>
<td>Tx Data Bit 0</td>
<td>0</td>
<td>—</td>
</tr>
<tr>
<td>RxD</td>
<td>Rx Data Bit 7</td>
<td>Rx Data Bit 6</td>
<td>Rx Data Bit 5</td>
<td>Rx Data Bit 4</td>
<td>Rx Data Bit 3</td>
<td>Rx Data Bit 2</td>
<td>Rx Data Bit 1</td>
<td>Rx Data Bit 0</td>
<td>0</td>
<td>—</td>
</tr>
<tr>
<td>BAL</td>
<td>BRGA LSB Divide Count (DLAB = 1)</td>
<td>0</td>
<td>02H</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BAH</td>
<td>BRGA MSB Divide Count (DLAB = 1)</td>
<td>1</td>
<td>00H</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>IER</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>Modern Interrupt Enable</td>
<td>Rx Machine Interrupt Enable</td>
<td>Tx Data Interrupt Enable</td>
<td>Rx Data Interrupt Enable</td>
<td>1</td>
<td>00H</td>
</tr>
<tr>
<td>IIR</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>Active Interrupt Bit 1</td>
<td>Active Interrupt Bit 0</td>
<td>Interrupt Pending</td>
<td>2</td>
<td>01H</td>
</tr>
<tr>
<td>LCR</td>
<td>DLAB</td>
<td>Divisor</td>
<td>Latch Access Bit</td>
<td>Set Break</td>
<td>Parity Mode Bit 2</td>
<td>Parity Mode Bit 1</td>
<td>Parity Mode Bit 0</td>
<td>Stop Bit Length Bit 0</td>
<td>Character Length Bit 1</td>
<td>Character Length Bit 0</td>
</tr>
<tr>
<td>MCR</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>Loopback Control Bit</td>
<td>OUT2 Complement</td>
<td>0</td>
<td>RTS Complement</td>
<td>DTR Complement</td>
<td>4</td>
</tr>
<tr>
<td>LSR</td>
<td>0</td>
<td>TxD Status</td>
<td>TxD Empty</td>
<td>Break Detected</td>
<td>Framing Error</td>
<td>Parity Error</td>
<td>Overrun Error</td>
<td>Rx Data Available</td>
<td>5</td>
<td>60H</td>
</tr>
<tr>
<td>MSR</td>
<td>DCD Input Inverted</td>
<td>RI Input Inverted</td>
<td>DSR Input Inverted</td>
<td>CTS Input Inverted</td>
<td>State Change in DCD</td>
<td>State (H → L) Change in RI</td>
<td>State Change in DSR</td>
<td>State Change in CTS</td>
<td>6</td>
<td>00H</td>
</tr>
<tr>
<td>SCR</td>
<td>Scratch-Pad Register</td>
<td>7</td>
<td>00H</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

*Figure 9. Register Description Table*
TRANSMIT DATA REGISTER (TXD)

This register holds the next data byte to be transmitted. When the transmit shift register becomes empty, the contents of the Transmit Data Register are loaded into the shift register and the Transmit Data Register Empty condition becomes true.

RECEIVE DATA REGISTER (RXD)

This register holds the last character received by the RX Machine. The character is right justified and the leading bits are zeroed. Reading the register empties the register and resets the Received Character Available condition.

BRG DIVISOR LOW BYTE (BAL)

This register contains the least significant byte of the Baud Rate Generator's 16-bit divisor. This register is accessible only when the DLAB bit is set in the LCR register.

BRG DIVISOR HIGH BYTE (BAH)

This register contains the most significant byte of the Baud Rate Generator's 16-bit divisor. This register is accessible only when the DLAB bit is set in the LCR register.
INTERRUPT ENABLE REGISTER (IER)

This register enables four types of interrupts which independently activate the INT pin. Each of the four interrupt types can be disabled by resetting the appropriate bit of the IER register. Similarly by setting the appropriate bits, selected interrupts can be enabled. If all interrupts are disabled, then the interrupt requests are inhibited from the IIR register and the INT pin. All other functions, including Status Register and the Line Status Register bits continue to operate normally.

MIE—MODEM Interrupt Enable
RXIE—RX Machine Interrupt Enable
TXDE—TX Data Register Empty
RXDA—RX Data Available

INTERRUPT IDENTIFICATION REGISTER (IIR)

This register holds the highest priority enabled and active interrupt request. The source of the interrupt request can be identified by reading bits 2-1.

B1, B0—Interrupt Bits, 2-1. These two bits reflect the highest priority, enabled and pending interrupt request.
11: RX Error Condition (Highest Priority)
10: RX Character Available
01: TXD Register Empty
00: Modem Interrupt (Lowest Priority)

IPN—Interrupt Pending—This bit is active low, and indicates that there is an interrupt pending. The interrupt logic asserts the INT pin as soon as this bit goes active (NOTE: the IIR register is continuously updated; so while the user is serving one interrupt source, a new interrupt with higher priority may enter IIR and replace the older interrupt vector).
LINE CONTROL REGISTER (LCR)

This is a read/write register which defines the basic configuration of the serial link.

DLAB—Divisor Latch Access Bit—This bit, when set, allows access to the Divisor Count Registers BAL and BAH.

SBK—Set Break—This will force the TXD pin low. The TXD pin will remain low until this bit is reset.

PM2—PM0—Parity Mode Bits—These three bits are used to select the various parity modes of the 82050.

<table>
<thead>
<tr>
<th>PM0</th>
<th>PM2</th>
<th>PM1</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>X</td>
<td>X</td>
<td>No Parity</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>Odd Parity</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>Even Parity</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>High Parity</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>Low Parity</td>
</tr>
<tr>
<td>1</td>
<td>X</td>
<td>X</td>
<td>Software Parity</td>
</tr>
</tbody>
</table>

SBL—Stop Bit Length—This bit defines the Stop Bit lengths for transmission. The RX Machine can identify 3/4 stop bit or more.

<table>
<thead>
<tr>
<th>SBL</th>
<th>Character Length</th>
<th>Stop Bit Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>X</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>5-Bit</td>
<td>1 1/2</td>
</tr>
<tr>
<td>1</td>
<td>(6, 7, or 8-Bit)</td>
<td>2</td>
</tr>
</tbody>
</table>

CL0—CL1—Character Length—These bits define the character length used on the serial link.

<table>
<thead>
<tr>
<th>CL1</th>
<th>CL0</th>
<th>Character Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>5 Bits</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>6 Bits</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>7 Bits</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>8 Bits</td>
</tr>
</tbody>
</table>
LINE STATUS REGISTER (LSR)

This register holds the status of the serial link. When read, all bits of the register are reset to zero.

- **RXDA—RX Data Available**—This bit, indicates that the RXD register has data available for the CPU to read.
- **OE—Overrun Error**—Indicates that a received character was lost because the RXD register was not empty.
- **PE Parity Error**—Indicates that a received character had a parity error.
- **FE—Framing Error**—Indicates that a received character had a framing error.
- **BkD—Break Detected**—This bit indicates that a break condition was detected, i.e., RxD input was held low for two character times.
- **TXDE—TXD Empty**—This indicates that the 82050 is ready to accept a new character for transmission. In addition, this bit causes an interrupt request to be generated if the TXD register Empty interrupt is enabled.
- **TXST—TX Machine Status**—When set, this bit indicates that the TX Machine is Empty, i.e., both the TXD register and the TX Shift Register are empty.

MODEM CONTROL REGISTER (MCR)

This register controls the modem output pins. All the outputs invert the data, i.e., their output will be the complement of the data written into this register.
LC—Loopback Control—This bit puts the 82050 into a Local Loopback mode.

OUT2—OUT2 Output—This bit controls the OUT2 pin. The output signal is the complement of this bit.

**NOTE:**
This bit is only effective when the 82050 is being used with an externally generated clock.

RTS—RTS Output Bit—This bit controls the RTS pin. The output signal is the complement of this bit.

DTR—DTR Output Bit—This bit controls the DTR pin. The output signal is the complement of this bit.

**MODEM STATUS REGISTER (MSR)**
This register holds the status of the modem input pins (CTS, DCD, DSR, RI). It is the source of Modem interrupts (bits 3–0) when enabled in the IER register. If any of the above input pins change levels, then the appropriate bit in MSR is set. Reading MSR will clear the status bits.

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>Complement DCD</td>
</tr>
<tr>
<td>6</td>
<td>Complement RI</td>
</tr>
<tr>
<td>5</td>
<td>Complement DSR</td>
</tr>
<tr>
<td>4</td>
<td>Complement CTS</td>
</tr>
<tr>
<td>3</td>
<td>State Change CTS</td>
</tr>
<tr>
<td>2</td>
<td>State Change DSR</td>
</tr>
<tr>
<td>1</td>
<td>(H --&gt; L) RI</td>
</tr>
<tr>
<td>0</td>
<td>State Change DCD</td>
</tr>
</tbody>
</table>

**DCDC—DCD Complement**—Holds the complement of the DCD pin.

**DRIC—RI Complement**—Holds the complement of the RI pin.

**DSRC—DSR Complement**—Holds the complement of the DSR pin.

**CTSC—CTS Complement**—Holds the complement of the CTS pin.

**DDCD—Delta DCD**—Indicates that the DCD pin has changed state since this register was last read.

**DRI—Delta RI**—Indicates that the RI pin has changed state from high to low since this register was last read.

**DDSR—Delta DSR**—Indicates that the DSR pin has changed state since this register was last read.

**DCTS—Delta CTS**—Indicates that the CTS pin has changed state since this register was last read.

**SCRATCHPAD REGISTER (SCR)**
The 8-bit Read/Write register does not control the ACC. It is intended as a scratch pad register for use by the programmer.
SPECIFICATIONS

D.C. SPECIFICATIONS

ABSOLUTE MAXIMUM RATINGS

Ambient Temperature under Bias ........ 0°C to 70°C
Storage Temperature .................. -65°C to +150°C
Voltage on any Pin (w.r.t. VSS) .......... -0.5V to +0.5V
Voltage on VCC Pin (w.r.t. VSS) ........ -0.5V to +7V
Power Dissipation ..................... 300 mW

D.C. CHARACTERISTICS (T_A = 0°C TO 70°C, VCC = 5V ± 10%)

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Parameter</th>
<th>Notes</th>
<th>Min</th>
<th>Max</th>
<th>Units</th>
</tr>
</thead>
<tbody>
<tr>
<td>VIL</td>
<td>Input Low Voltage</td>
<td>(1)</td>
<td>-0.5</td>
<td>0.8</td>
<td>V</td>
</tr>
<tr>
<td>VIH</td>
<td>Input High Voltage</td>
<td>(1), (7)</td>
<td>2.0</td>
<td>VCC + 0.5</td>
<td>V</td>
</tr>
<tr>
<td>VOL</td>
<td>Output Low Voltage</td>
<td>(2), (9)</td>
<td>0.45</td>
<td>V</td>
<td></td>
</tr>
<tr>
<td>VOH</td>
<td>Output High Voltage</td>
<td>(3), (9)</td>
<td>2.4</td>
<td>V</td>
<td></td>
</tr>
<tr>
<td>ILI</td>
<td>Input Leakage Current</td>
<td>(4)</td>
<td>±10</td>
<td>μA</td>
<td></td>
</tr>
<tr>
<td>ILO</td>
<td>3-State Leakage Current</td>
<td>(5)</td>
<td>±10</td>
<td>μA</td>
<td></td>
</tr>
<tr>
<td>IOHR</td>
<td>Input High for DTR RTS</td>
<td>(10)</td>
<td>0.4</td>
<td>mA</td>
<td></td>
</tr>
<tr>
<td>IOLR</td>
<td>Input Low for DTR, pRTS</td>
<td>(10)</td>
<td>11</td>
<td>mA</td>
<td></td>
</tr>
<tr>
<td>LXTAL</td>
<td>X1, X2 Load</td>
<td></td>
<td>10</td>
<td>pF</td>
<td></td>
</tr>
<tr>
<td>ICC</td>
<td>Power Supply Current</td>
<td>(6)</td>
<td>3.8</td>
<td>mA/MHz</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>35</td>
<td>mA (max)</td>
<td></td>
</tr>
<tr>
<td>Cin</td>
<td>Input Capacitance</td>
<td>(8)</td>
<td>10</td>
<td>pF</td>
<td></td>
</tr>
<tr>
<td>Cio</td>
<td>I/O Capacitance</td>
<td>(8)</td>
<td>10</td>
<td>pF</td>
<td></td>
</tr>
</tbody>
</table>

NOTES:
1. Does not apply to CLK/X1 pin, when configured as crystal oscillator input (X1).
2. @ IOL = 2 mA.
3. @ IOH = -0.4 mA.
4. 0 < VIL < VCC.
5. 0.45V < Vout < (VCC - 0.45).
6. VCC = 5.5V; VIH = 0.5V (max); VIL = VCC - 0.5V (min); IOL = IOH = 0; 9.2 MHz (max).
7. VHI = 2.4V on RD and RXD pins.
8. Freq = 1 MHz.
9. Does not apply OUT2/X2 pin, when configured as crystal oscillator output (X2).
10. Input current for DTR, RTS pins during Reset for Clock Mode Configuration.
A.C. SPECIFICATIONS

Testing Conditions:
- All AC output parameters are under output load of 20 to 100 pF, unless otherwise specified.
- AC testing inputs are driven at 2.4 for logic '1', and 0.45V for logic '0'. Output timing measurements are made at 1.5V for both a logical '0' and '1'.
- In the following tables, the units are ns, unless otherwise specified.

System Interface Specification—System Clock Specification:

The 82050 system clock is supplied via the CLK pin or generated by on-chip crystal oscillator. The clock is optionally divided by two. The CLK parameters are given separately for internal divide-by-two option ACTIVE and INACTIVE.

The system clock (after division by two, if active) must be at least 16X the Tx or Rx baud rate (the faster of the two).

### SYSTEM CLOCK SPECIFICATIONS

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Parameter</th>
<th>Min</th>
<th>Max</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>DIVIDE BY TWO OPTION—ACTIVE</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Tcy/2</td>
<td>CLK Period</td>
<td>54</td>
<td>250</td>
<td>(2)</td>
</tr>
<tr>
<td>TCLCH</td>
<td>CLK Low Time</td>
<td>25</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TCHCL</td>
<td>CLK High Time</td>
<td>25</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TCH1CH2</td>
<td>CLK Rise Time</td>
<td>10</td>
<td>(1)</td>
<td></td>
</tr>
<tr>
<td>TCL2CL1</td>
<td>CLK Fall Time</td>
<td>10</td>
<td>(1)</td>
<td></td>
</tr>
<tr>
<td>FXTAL</td>
<td>External Crystal Frequency Rating</td>
<td>4.0</td>
<td>18.432 MHz</td>
<td></td>
</tr>
<tr>
<td></td>
<td>DIVIDE BY TWO OPTION—INACTIVE</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Tcy</td>
<td>CLK Period</td>
<td>108</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TCLCH</td>
<td>CLK Low Time</td>
<td>54</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TCHCL</td>
<td>CLK High Time</td>
<td>44</td>
<td>250</td>
<td></td>
</tr>
<tr>
<td>TCH1CH2</td>
<td>CLK Rise Time</td>
<td>15</td>
<td>(1)</td>
<td></td>
</tr>
<tr>
<td>TCL2CL1</td>
<td>CLK Fall Time</td>
<td>15</td>
<td>(1)</td>
<td></td>
</tr>
</tbody>
</table>

NOTES:
1. Rise/fall times are measured between 0.8 and 2.0V.
2. Tcy in ACTIVE divide by two option is TWICE the input clock period.

RESET SPECIFICATION

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Parameter</th>
<th>Min</th>
<th>Max</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRSHL</td>
<td>Reset Width—CLK/X1 Configured to CLK</td>
<td>8</td>
<td>Tcy</td>
<td>(1)</td>
</tr>
<tr>
<td>TTLRSL</td>
<td>RTS/DTR LOW Setup to Reset Inactive</td>
<td>6</td>
<td>Tcy</td>
<td>(2)</td>
</tr>
<tr>
<td>TRSLTX</td>
<td>RTS/DTR Low Hold after Reset Inactive</td>
<td>0</td>
<td>Tcy</td>
<td>(2)</td>
</tr>
</tbody>
</table>

NOTES:
1. In case of CLK/X1 configured as X1, additional time is required to guarantee crystal oscillator wake-up.
2. RTS/DTR are internally driven HIGH during RESET active time. The pin should be either left OPEN or externally driven LOW during RESET according to the required configuration of the system clock. These parameters specify the timing requirements on these pins, in case they are externally driven LOW during RESET. The maximum spec on TRSLTX requires that the RTS/DTR pins not be forced later than TRSLTX maximum.
READ CYCLE SPECIFICATIONS

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Parameter</th>
<th>Min</th>
<th>Max</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRLRH</td>
<td>RD Active Width</td>
<td>2 Tcy + 65</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TAVRL</td>
<td>Address/CS Setup Time to RD Active</td>
<td>7</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TRHAX</td>
<td>Address/CS Hold Time after RD Inactive</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TRLDV</td>
<td>Data Out Valid after RD Active</td>
<td>2Tcy + 65 (1)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TCIAD</td>
<td>Command Inactive to Active Delay</td>
<td>Tcy + 15 (2)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TRHDZ</td>
<td>Data Out Float Delay after RD Inactive</td>
<td>40</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

NOTES:
1. C1 = 20 pF to 100 pF.
2. Command refers to either Read or Write signals.

WRITE CYCLE SPECIFICATION

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Parameter</th>
<th>Min</th>
<th>Max</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>TWLWH</td>
<td>WR Active Width</td>
<td>2Tcy + 15</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TAVWL</td>
<td>Address CS Setup Time to WR Active</td>
<td>7</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TWHAX</td>
<td>Address and CS Hold Time after WR</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TDVWH</td>
<td>Data in Setup Time to WR Inactive</td>
<td>90</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TWHDX</td>
<td>Data in Hold Time after WR Inactive</td>
<td>12</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
The Intel CHMOS 82510 is designed to increase system efficiency in asynchronous environments such as modems, serial ports—including expanding performance areas: MCS-51 9-bit format and high speed async. The functional support provided in the 82510 is unparalleled—2 baud rate generators/timers provide independent data rates or protocol timeouts; a crystal oscillator and smart modem I/O simplify system logic. New features, dual FIFOs and Control Character Recognition (CCR), dramatically reduce CPU interrupts and increase software efficiency. The 82510’s software versatility allows emulation of the INS8250A/16450® for IBM PC AT® compatibility or a high performance mode, configured by 35 control registers. All interrupts are maskable at 2 levels. The multi-personality I/O pins are configurable as desired. A DPLL and multiple sampling of serial data improve data reliability for high speed asynchronous communication. The compact 28-pin 82510 is fabricated in CHMOS III technology and includes a software powerdown option.

![82510 Block Diagram](image-url)
### 82510 PINOUT DEFINITION

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Pin No.</th>
<th>Type</th>
<th>Name and Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RESET</td>
<td>17</td>
<td>I</td>
<td>RESET: A high on this input pin resets the 82510 to the Default Wake-up mode.</td>
</tr>
<tr>
<td>CS</td>
<td>18</td>
<td>I</td>
<td>CHIP SELECT: A low on this input pin enables the 82510 and allows read or write operations.</td>
</tr>
<tr>
<td>A2–A0</td>
<td>24-22</td>
<td>I</td>
<td>ADDRESS PINS: These inputs interface with three bits of the System Address Bus to select one of the internal registers for read or write.</td>
</tr>
<tr>
<td>D7–D0</td>
<td>4* 25</td>
<td>I/O</td>
<td>DATA BUS: Bi-directional, three state, eight-bit Data Bus. These pins allow transfer of bytes between the CPU and the 82510.</td>
</tr>
<tr>
<td>RD</td>
<td>20</td>
<td>I</td>
<td>READ: A low on this input pin allows the CPU to read Data or Status bytes from the 82510.</td>
</tr>
<tr>
<td>WR</td>
<td>19</td>
<td>I</td>
<td>WRITE: A low on this input allows the CPU to write Data or Control bytes to the 82510.</td>
</tr>
<tr>
<td>INT</td>
<td>5</td>
<td>O</td>
<td>INTERRUPT: A high on this output pin signals an interrupt request to the CPU. The CPU may determine the particular source and cause of the interrupt by reading the 82510 Status registers.</td>
</tr>
<tr>
<td>CLK/X1</td>
<td>9</td>
<td>I</td>
<td>MULTIFUNCTION: This input pin serves as a source for the internal system clock. The clock may be asynchronous to the serial clocks and to the processor clock. This pin may be used in one of two modes: CLK — in this mode an externally generated TTL compatible clock should be used to drive this input pin; X1 — in this mode the clock is internally generated by an on-chip crystal oscillator. This mode requires a crystal to be connected between this pin (X1) and the X2 pin. (See System Clock Generation.)</td>
</tr>
<tr>
<td>OUT2/X2</td>
<td>8</td>
<td>O</td>
<td>MULTIFUNCTION: This is a dual function pin which may be configured to one of the following functions: OUT2 — a general purpose output pin controlled by the CPU, only available when CLK/X1 pin is driven by an externally generated clock; X2 - this pin serves as an output pin for the crystal oscillator. Note: The configuration of the pin is done only during hardware reset. For more details refer to the System Clock Generation.</td>
</tr>
<tr>
<td>TXD</td>
<td>6</td>
<td>O</td>
<td>TRANSMIT DATA: Serial data is transmitted via this output pin starting at the Least Significant bit.</td>
</tr>
</tbody>
</table>

*Pins 28–25 and Pins 4–1.*
## 82510 PINOUT DEFINITION (Continued)

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Pin No.</th>
<th>Type</th>
<th>Name and Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RXD</td>
<td>13</td>
<td>I</td>
<td><strong>RECEIVE DATA:</strong> Serial data is received on this input pin starting at the Least Significant bit.</td>
</tr>
<tr>
<td>RI/SCLK</td>
<td>10</td>
<td>I</td>
<td><strong>MULTIFUNCTION:</strong> This is a dual function pin which can be configured to one of the following functions. RI - Ring Indicator - Input, active low. This is a general purpose input pin accessible by the CPU. SCLK - This input pin may serve as a source for the internal serial clock(s), RxClk and/or TxClk. See Figure 12, BRG sources and outputs.</td>
</tr>
<tr>
<td>DTR/TB</td>
<td>15</td>
<td>O</td>
<td><strong>MULTIFUNCTION:</strong> This is a dual function pin which may be configured to one of the following functions. DTR - Data Terminal Ready. Output, active low. This is a general purpose output pin controlled by the CPU. TB - This pin outputs the BRGB output signal when configured as either a clock generator or as a timer. When BRGB is configured as a timer this pin outputs a &quot;timer expired pulse.&quot; When BRGB is configured as a clock generator it outputs the BRGB output clock.</td>
</tr>
<tr>
<td>DSR/TA/OUT0</td>
<td>11</td>
<td>I/O</td>
<td><strong>MULTIFUNCTION:</strong> This is a multifunction pin which may be configured to one of the following functions. DSR - Data Set Ready. Input, active low. This is a general purpose input pin accessible by the CPU. TA - This pin is similar in function to pin TB except it outputs the signals from BRGA instead of BRGB. OUT0 - Output pin. This is a general purpose output pin controlled by the CPU.</td>
</tr>
<tr>
<td>RTS</td>
<td>16</td>
<td>O</td>
<td><strong>REQUEST TO SEND:</strong> Output pin, active low. This is a general purpose output pin controlled by the CPU. In addition, in automatic transmission mode this pin, along with CTS, controls the transmission of data. (See Transmit modes for further detail.) During hardware reset this pin is an input. It is used to determine the System Clock Mode. (See System Clock Generation for further detail.)</td>
</tr>
<tr>
<td>CTS</td>
<td>14</td>
<td>I</td>
<td><strong>CLEAR TO SEND:</strong> Input pin, active low. In automatic transmission mode it directly controls the Transmit Machine. (See transmission mode for further details.) This pin can be used as a General Purpose Input.</td>
</tr>
<tr>
<td>DCD/IClk/OUT1</td>
<td>12</td>
<td>I/O</td>
<td><strong>MULTIFUNCTION:</strong> This is a multifunction pin which may be configured to one of the following functions. DCD - Data Carrier Detected. Input pin, active low. This is a general purpose input pin accessible by the CPU. ICLK - This pin is the output of the internal system clock. OUT1 - General purpose output pin. Controlled by the CPU.</td>
</tr>
</tbody>
</table>

**Table 1. Multifunction Pins**

<table>
<thead>
<tr>
<th>Pin #</th>
<th>I/O</th>
<th>Timing</th>
<th>Modem</th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td>*OUT2</td>
<td>X2</td>
<td>—</td>
</tr>
<tr>
<td>9</td>
<td>—</td>
<td>*CLK/X1</td>
<td>—</td>
</tr>
<tr>
<td>10</td>
<td>—</td>
<td>SCLK</td>
<td>*RI</td>
</tr>
<tr>
<td>11</td>
<td>OUT0</td>
<td>TA</td>
<td>*DSR</td>
</tr>
<tr>
<td>12</td>
<td>OUT1</td>
<td>ICLK</td>
<td>*DCD</td>
</tr>
<tr>
<td>14</td>
<td>—</td>
<td>—</td>
<td>*CTS</td>
</tr>
<tr>
<td>15</td>
<td>—</td>
<td>TB</td>
<td>*DTR</td>
</tr>
<tr>
<td>16</td>
<td>—</td>
<td>—</td>
<td>*RTS</td>
</tr>
</tbody>
</table>

*Default
GENERAL DESCRIPTION

The 82510 can be functionally divided into seven major blocks (See Fig 1): Bus Interface Unit, Timing Unit, Modem Module, Tx FIFO, Rx FIFO, Tx Machine, and Rx Machine. Six of these blocks (all except Bus Interface Unit) can generate block interrupts. Three of these blocks can generate second-level interrupts which reflect errors/status within the block (Receive Machine, Timing Unit, and the Modem Module).

The Bus interface unit allows the 82510 to interface with the rest of the system. It controls access to device registers as well as generation of interrupts to the external world. The FIFOs buffer the CPU from the Serial Machines and reduce the interrupt overhead normally required for serial operations. The threshold (level of occupancy in the FIFO which will generate an interrupt) is programmable for each FIFO. The timing unit controls generation of the system clock through either its on-chip crystal oscillator, or an externally generated clock. It also provides two baud Rate Generators/Timers with various options and modes to support serial communication.

FUNCTIONAL DESCRIPTION

CPU Interface

The 82510 has a simple demultiplexed Bus Interface, which consists of a bidirectional three-state eight-bit, data bus and a three-bit address bus. An interrupt pin along with the Read, Write and Chip Select are the remaining signals used to interface with the CPU. The three address lines along with the Bank Pointer register are used to select the registers. The 82510 is designed to interface to all Intel microprocessor and microcontroller families. Like most other I/O based peripherals it is programmed through its registers to support a variety of functions.

Its register set can be used in 8250A/16450 compatibility or High Performance modes. The 8250A/16450 mode is the default wake-up mode in which only the 8250A/16450 compatible registers are accessible. The remaining registers are default configured to support 8250A/16450 emulation.

Software Interface

The 82510 is configured and controlled through its 35 registers which are divided into four banks. Only one bank is accessible at any one time. The bank switching is done by changing the contents of the bank pointer (GIR/BANK–BANK0, BANK1). The banks are logically grouped into 8250A/16450 compatible (0), General Work Bank (1), General Configuration (2), and Modem Configuration (3). The 8250A/16450 compatible bank (Bank 0) is the default bank upon power up.

The 82510 registers can be categorized under the following:

Table 2. 82510 Register/Block Functions

<table>
<thead>
<tr>
<th></th>
<th>Status</th>
<th>Enable</th>
<th>Configuration</th>
<th>Command</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>FIFO</td>
<td>FLR</td>
<td>—</td>
<td>FMD</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>MODEM</td>
<td>MSR</td>
<td>MIE</td>
<td>PMD</td>
<td>MCR</td>
<td>—</td>
</tr>
<tr>
<td>RX</td>
<td>RST, RXF</td>
<td>RIE</td>
<td>RMD</td>
<td>RCM</td>
<td>RXD, RXF</td>
</tr>
<tr>
<td>TX</td>
<td>LSR</td>
<td>LSR</td>
<td>TMD</td>
<td>TCM</td>
<td>TXD, TXF</td>
</tr>
<tr>
<td>TIMER</td>
<td>TMST</td>
<td>TMIE</td>
<td>CLCF, BACF, BBCF</td>
<td>TMCR</td>
<td>BBL, BBH, BAL, BAH</td>
</tr>
<tr>
<td>DEVICE</td>
<td>GSR, GIR</td>
<td>GER</td>
<td>IMD</td>
<td>ICM</td>
<td>—</td>
</tr>
<tr>
<td>8250</td>
<td>LSR, MSR, GIR</td>
<td>GER</td>
<td>LCR, MCR</td>
<td>MCR</td>
<td>TXD, RXD, BAL, BAH</td>
</tr>
</tbody>
</table>
8250 Compatibility

Upon power up or reset, the 82510 comes up in the default wake up mode. The 8250A/16450 compatible bank, bank zero, is the accessible bank and all the other registers are configured via their default values to support this mode. An 18.432 MHz crystal frequency is necessary.

Table 3. 8250A/16450 Compatible Registers

<table>
<thead>
<tr>
<th>Address</th>
<th>82510 Registers (Bank 0)</th>
<th>8250A Registers</th>
</tr>
</thead>
<tbody>
<tr>
<td>00 (DLAB = 0)</td>
<td>RxD</td>
<td>TxD</td>
</tr>
<tr>
<td>01 (DLAB = 0)</td>
<td>GER</td>
<td>GER</td>
</tr>
<tr>
<td>00 (DLAB = 1)</td>
<td>BAL</td>
<td>BAL</td>
</tr>
<tr>
<td>01 (DLAB = 1)</td>
<td>BAH</td>
<td>BAH</td>
</tr>
<tr>
<td>02</td>
<td>GIR/BANK</td>
<td>BANK</td>
</tr>
<tr>
<td>03</td>
<td>LCR</td>
<td>LCR</td>
</tr>
<tr>
<td>04</td>
<td>MCR</td>
<td>MCR</td>
</tr>
<tr>
<td>05</td>
<td>LSR</td>
<td>LSR</td>
</tr>
<tr>
<td>06</td>
<td>MSR</td>
<td>MSR</td>
</tr>
<tr>
<td>07</td>
<td>ACR0</td>
<td>ACR0</td>
</tr>
</tbody>
</table>

Table 4. Default Wake-Up Mode

| RxD | — | ACR1 | 00H | RxF | — |
| TxD | — | RIE | 1EH | TxF | — |
| BAL | 02H | RMD | 00H | TMST | 30H |
| BAH | 00H | CLCF | 00H | TMC | — |
| GER | 00H | BACF | 04H | FLR | 00H |
| GIR/BANK | 01H | BBCF | 84H | RCM | — |
| LCR | 00H | PMD | FCH | TCM | — |
| MCR | 00H | MIE | 0FH | GSR | 12H |
| LSR | 60H | TMIE | 00H | ICM | — |
| MSR | 00H | BBL | 05H | FMD | 00H |
| ACR0 | 00H | BBH | 00H | TMD | 00H |
| RST | 00H | | IMD | 0CH |
Interrupts

There are two levels of interrupt/status reporting within the 82510. The first level is the block level interrupts such as RX FIFO, Tx FIFO, Rx Machine, Tx Machine, Timing unit, and Modem Module. The status of these blocks is reported in the General Status and General Interrupt Registers. The second level is the various sources within each block; only three of the blocks generate second level interrupts (Rx Machine, Timing Unit, and Modem Module). Interrupt requests are maskable at both the block level and at the individual source level within the module. If more than one unmasked block requests interrupt service an on-chip interrupt controller will resolve contention on a priority basis (each block has a fixed priority). An interrupt request from a particular block is activated if one of the unmasked status bits within the status register for the block is set. A CPU service operation, e.g., reading the appropriate status register, will reset the status bits.

ACKNOWLEDGE MODES

The interrupt logic will assert the INT pin when an interrupt is coded into the General Interrupt register. The INT pin is forced low upon acknowledgment. The 82510 has two modes of interrupt acknowledgment:

1. Manual Acknowledge

The CPU must issue an explicit Interrupt Acknowledge command via the Interrupt Acknowledge bit of the Internal Command register. As a result the INT pin is forced low for two clocks and then updated.

2. Automatic Acknowledge

As opposed to the Manual Acknowledge mode, when the CPU must issue an explicit interrupt acknowledge command, an interrupt service operation is considered as an automatic acknowledgment. This forces the INT pin low for two clock cycles. After two cycles the INT pin is updated, i.e., if there is still an active non-masked interrupt request the INT pin is set HIGH.

INTERRUPT SERVICE

A service operation is an operation performed by the CPU, which causes the source of the 82510 interrupt to be reset (it will reset the particular status bit causing the interrupt). An interrupt request within the 82510 will not reset until the interrupt source has been serviced. Each source can be serviced in two or three different ways; one general way is to disable the particular status bit causing the interrupt, via the corresponding block enable register. Setting the appropriate bit of the enable register to zero will mask off the corresponding bit in the status register, thus causing an edge on the input line to the interrupt logic. The same effect can be achieved by masking
of the particular block interrupt request in GSR via the General Enable Register. Another method, which is applicable to all sources, is to issue the Status Clear command from the Internal Command Register. The detailed service requirements for each source are given below:

### Table 5. Service Procedures

<table>
<thead>
<tr>
<th>Interrupt Source</th>
<th>Status Bits &amp; Registers</th>
<th>Interrupt Masking</th>
<th>Specific Service</th>
</tr>
</thead>
<tbody>
<tr>
<td>Timers</td>
<td>TMST (1–0) GSR (5)</td>
<td>TMIE (1–0) GER (5)</td>
<td>Read TMST</td>
</tr>
<tr>
<td>Tx Machine</td>
<td>GSR (4) LSR (6)</td>
<td>GER (4)</td>
<td>Write Character to tx FIFO</td>
</tr>
<tr>
<td>Rx Machine</td>
<td>LSR (4–1) RST (7–1) GSR (2)</td>
<td>RIE (7–1) GER (2)</td>
<td>Read RST or LSR Write 0 to bit in RST/LSR</td>
</tr>
<tr>
<td>Rx FIFO</td>
<td>RST/LSR (0) GSR (0)</td>
<td>GER (0)</td>
<td>Write 0 to LSR/RST Bit zero. Read Character</td>
</tr>
<tr>
<td>Tx FIFO</td>
<td>LSR (5) GSR (1)</td>
<td>GER (1)</td>
<td>Write to FIFO Read GIR(1)</td>
</tr>
<tr>
<td>Modem</td>
<td>MSR (3–0) GSR (3)</td>
<td>MIE (3–0) GER (3)</td>
<td>Read MSR write 0 into the appropriate bits of MSR (3–0).</td>
</tr>
</tbody>
</table>

**NOTE:**
1. Only if pending interrupt is Tx FIFO.

### System Clock Generation

The 82510 has two modes of System Clock Operation. It can accept an externally generated clock, or it can use a crystal to internally generate its system clock.

**Figure 6. Crystal Oscillator**

The 82510 has an on-chip oscillator to generate its system clock. The oscillator will take the inputs from a crystal attached to the X1 and X2 pins. This mode is configured via a hardware strapping option on RTS.

**Figure 7. Strapping Option**

During hardware reset the RTS pin is an input; it is weakly pulled high from within and then checked. If it is driven low externally then the 82510 is configured for the Crystal Oscillator; otherwise an external clock is expected.

**Figure 8. External Clock**

This is the default configuration. Under normal conditions the system clock is divided by two; however, the user may disable divide by two via a hardware strapping option on the DTR pin. The Hardware strapping option is similar to the one used on the RTS pin. It is forbidden to strap both DTR and RTS.

### Transmit

The two major blocks involved in transmission are the Transmit FIFO and the Transmit Machine. The Tx FIFO acts as a buffer between the CPU and the Tx Machine. Whenever a data character is written to the Transmit Data register, it, along with the Transmit Flags (if applicable), is loaded into the Tx FIFO.
The Tx FIFO can hold up to four, eleven-bit characters (nine-bits data, parity, and address flag). It has separate read and write mechanisms. The read and write pointers are incremented after every operation to allow data transfer to occur in a First In First Out fashion. The Tx FIFO will generate a maskable interrupt when the level in the FIFO is below, or equal to, the Threshold. The threshold is user programmable.

For example, if the threshold equals two, and the number of characters in the Tx FIFO decreases from three to two, the FIFO will generate an interrupt. The threshold should be selected with regard to the system's interrupt service latency.

**NOTE:**

There is a one character transmission delay between FIFO empty and Transmitter Idle, so a threshold of zero may be selected without getting an underrun condition. Also if more than four characters are written to the FIFO an overrun will occur and the extra character will not be written to the Tx FIFO. This error will not be reported to the CPU.

**TX MACHINE**

The Tx Machine reads characters from the Tx FIFO, serializes the bits, and transmits them over the TXD pin according to the timing signals provided for transmission. It will also generate parity, transmit break (upon CPU request), and manage the modem handshaking signals (CTS and RTS) if configured so. The Tx machine can be enabled or disabled through the Transmit Command register or CTS. If the transmitter is disabled in the middle of a character transmission the transmission will continue until the end of the character; only then will it enter the disable state.

**TRANSMIT CLOCKS**

There are two modes of transmission clocking, 1X and 16X. In the 1X mode the transmitted data is synchronous to the transmit clock as supplied by the SCLK pin. In this mode stop-bit length is restricted to one or two bits only. In the 16X mode the data is not required to be synchronous to the clock. (Note: The Tx clock can be generated by the BRGs or from the SCLK pin.)

**MODEM HANDSHAKING**

The transmitter has three modes of handshaking.

**Manual Mode**—In this mode the CTS and RTS pins are not used by the Tx Machine (transmission is started regardless of the CTS state, and RTS is not forced low). The CPU may manage the handshake itself, by accessing the CTS and RTS signals through the MODEM CONTROL and MODEM STATUS registers.

**Semi-Automatic Mode**—In this mode the RTS pin is activated whenever the transmitter is enabled. The CTS pin's state controls transmission. Transmission is enabled only if CTS is active. If CTS becomes inactive during transmission, the Tx Machine will complete transmission of the current character and then go to the inactive state until CTS becomes active again.

**Automatic Mode**—This mode is similar to the semi-automatic mode, except that RTS will be activated as long as the transmitter is enabled and there are more characters to transmit. The CPU need only fill the FIFO, the handshake is done by the Tx Machine. When both the shift register and the FIFO are empty RTS automatically goes inactive. (Note: The RTS pin can be forced to the active state by the CPU, regardless of the handshaking mode, via the MODEM CONTROL register.)

**Receive**

The 82510 reception mechanism involves two major blocks; the Rx Machine and the Rx FIFO. The Rx Machine will assemble the incoming character and its associated flags and then LOAD them on to the Rx FIFO. The top of the FIFO may be read by reading the Receive Data register and the Receive Flags Register. The receive operation can be done in two modes. In the **normal** mode the characters are received in the standard Asynchronous format and only control characters are recognized. In the **ulans** mode, the nine bit protocol of the MCS-51 family is supported and the ulan Address characters, rather than Control Characters are recognized.
RX FIFO

The Rx FIFO is very similar in structure and basic operation to the Tx FIFO. It will generate a maskable interrupt when the FIFO level is above, the threshold. The Rx FIFO can also be configured to operate as a one-byte buffer. This mode is used for 8250 compatible software drivers. An overrun will occur when the FIFO is full and the Rx Machine has a new character for the FIFO. In this situation the oldest character is discarded and the new character is loaded from the Rx Machine. An Overrun error bit will also be set in the RECEIVE STATUS and LINE STATUS registers.

The user has the option to disable the loading of incoming characters on to the Rx FIFO by using the UNLOCK/LOCK FIFO commands. (See RECEIVE COMMAND register.) When the Rx FIFO is locked, it will ignore load requests from the Rx Machine, and thus the received characters will not be loaded into the FIFO and may be lost (if another character is received). These two commands are useful when the CPU is not willing to receive characters, or is waiting for specific Control/Address characters. In uLAN mode there are three options of address recognition, each of these options varies in the amount of CPU offload, and degree of FIFO control through OPEN/LOCK FIFO commands.

Automatic Mode—In this mode the Rx Machine will open the FIFO whenever an Address Match occurs; it will LOCK the FIFO if an address mismatch occurs.

Semi-Automatic Mode—In this mode the Rx Machine will open the FIFO whenever an address character is received. It will not lock the FIFO if the Address does not match. The user is responsible for locking the Rx FIFO.

Manual Mode—In this mode the Rx Machine does not control the FIFO automatically; however, the user may UNLOCK/LOCK the FIFO by using the RECEIVE COMMAND register.

RX MACHINE

The RX Machine has two modes of clocking the incoming data—16X or 1X. In 16X synchronization is done internally; in the 1X mode the data must be synchronous to the SCLK pin input. The Rx Machine synchronizes the data, passes it through a digital filter to filter out the spikes, and then uses the voting counter to generate the data bit (multiple sampling of input RXD). Bit polarity decisions are made on the basis of majority voting; i.e., if the majority of the samples are "1" the result is a "1" bit. If all samples are not in agreement then the bit is also reported as a noisy bit in the RECEIVE FLAGS register. The sampling window is programmable for either 3/16 or 7/16 samples. The 3/16 mode is useful for high frequency transmissions, or when serious RC delays are expected on the channel. The 7/16 is best suited for noisy media. The Rx machine also has a DPLL to overcome frequency shift problems; however, using it in a very noisy environment may increase the error, so the user can disable the DPLL via the Receive Mode register. The Rx Machine will generate the parity and the address marker as well as any framing error indications.

Start Bit Detection—The falling edge of the Start bit resets the DPLL counter and the Rx Machine starts sampling the input line (the number of samples is determined by the configuration of the sampling window mode). The Start bit verification can be done through either a majority voting system or an absolute voting system. The absolute voting requires that all the samples be in agreement. If one of the samples does not agree then a false Start bit is determined and the Rx Machine returns to the Start Bit search Mode. Once a Start bit is detected the Rx Machine will use the majority voting sampling window to receive the data bits.

Break Detection—If the input is low for the entire character frame including the stop Bit, then the Rx Machine will set Break Detected as well as Framing Error in the RECEIVE STATUS and LINE STATUS registers. It will push a NULL character onto the Rx FIFO with a framing-error and Break flag (As part of the Receive Flags). The Rx Machine then enters the Idle state. When it sees a mark it will set Break Terminated in RECEIVE STATUS and LINE STATUS registers and resume normal operation.
Control Characters—The Rx machine can generate a maskable interrupt upon reception of standard ASCII or EBCDIC control characters, or an Address marker is received in the uLAN mode. The Rx machine can also generate a maskable interrupt upon a match with programmed characters in the Address/Control Character 0 or Address/Control Character 1 registers.

Table 6. Control Character Recognition

<table>
<thead>
<tr>
<th>CONTROL CHARACTER RECOGNITION</th>
<th>STANDARD SET</th>
<th>USER PROGRAMMED</th>
</tr>
</thead>
<tbody>
<tr>
<td>A) STANDARD SET</td>
<td></td>
<td></td>
</tr>
<tr>
<td>- ASCII: 000X XXXX + 0111 1111 (ASCII DEL) (00 - 1FH + 7 FH)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>OR</td>
<td></td>
<td></td>
</tr>
<tr>
<td>- EBCDIC: 00XX XXXX</td>
<td></td>
<td></td>
</tr>
<tr>
<td>- (00 - 3FH)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>B) User Programmed</td>
<td></td>
<td></td>
</tr>
<tr>
<td>- ACR0, ACR1 XXXX XXXX</td>
<td></td>
<td></td>
</tr>
<tr>
<td>- REGISTERS</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Baud-Rate Generators/Timers

The 82510 has two-on-chip, 16-bit baud-rate generators. Each BRG can also be configured as a Timer, and is completely independent of the other. This can be used when the Transmit and Receive baud rates are different. The mode, the output, and the source of each BRG is configurable, and can also be optionally output to external devices via the TA, TB pins (see Fig. 12. BRG Sources and Outputs).

BAUD RATE GENERATION

The Baud Rate is generated by dividing the source clock with the divisor count. The count is loaded from the divisor count registers into a count down register. A 50% duty cycle is generated by counting down in steps of two. When the count is down to 2 the entire count is reloaded and the output clock is toggled. Optionally the two BRGs may be cascaded to provide a larger divisor. Note that this is the default configuration and used for 8250A/16450 emulation.

\[ f_o = f_{in}/\text{Divisor} \]

where \( f_{in} \) is the input clock frequency and Divisor is the count loaded into the appropriate count registers. System clock frequencies can be selected (4 \( \rightarrow \) 9.216 MHz) to eliminate baud rate error for high baud rates.
### Table 7. Standard Baud Rates

<table>
<thead>
<tr>
<th>Bit Rate</th>
<th>16x Divisor</th>
<th>% Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>110</td>
<td>5236 (1474h)</td>
<td>.007%</td>
</tr>
<tr>
<td>300</td>
<td>1,920 (780h)</td>
<td>—</td>
</tr>
<tr>
<td>1200</td>
<td>480 (1E0h)</td>
<td>—</td>
</tr>
<tr>
<td>2400</td>
<td>240 (F0h)</td>
<td>—</td>
</tr>
<tr>
<td>9600</td>
<td>60 (3Ch)</td>
<td>—</td>
</tr>
<tr>
<td>19,200</td>
<td>30 (1Eh)</td>
<td>—</td>
</tr>
<tr>
<td>38,400</td>
<td>15 (0Fh)</td>
<td>—</td>
</tr>
<tr>
<td>56,000</td>
<td>10 (0Ah)</td>
<td>2.8%</td>
</tr>
<tr>
<td>288,000</td>
<td>2 (02h)</td>
<td>—</td>
</tr>
</tbody>
</table>

Source CLK = Internal Sys. Clk
= 18.432 MHz/2 (Crystal)
= 9.216 MHz (External 1X clock)

**NOTE:**
Internal system clock is 1/4 crystal frequency or 1/2 external clock frequency when using 2 clock option.

The BRG counts down in increments of two and then is divided by two to generate a 50% duty cycle; however, for odd divisors it will count down the first time by one. All subsequent countdowns will then continue in steps of two. In those cases the duty cycle is no longer exactly 50%. The deviation is given by the following equation:

\[
\text{deviation} = \frac{1}{2 \cdot \text{divisor}}
\]

The BRG can operate with any divisor between 1 and 65,535; however, for divisors between 1 and 3 the duty cycle is as follows:

**Table 8. Duty Cycles**

<table>
<thead>
<tr>
<th>Divisor</th>
<th>Duty Cycle</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>33%</td>
</tr>
<tr>
<td>2</td>
<td>50%</td>
</tr>
<tr>
<td>1</td>
<td>Same as Source</td>
</tr>
<tr>
<td>0</td>
<td>FORBIDDEN</td>
</tr>
</tbody>
</table>

### Timer Mode

Each of the 82510 BRGs can be used as Timers. The Timer is used to generate time delays by counting the internal system clock. When enabled the Timer uses the count from the Divisor/Count registers to count down to 1. Upon terminal count a maskable Timer Expired interrupt is generated. The delay between the trigger and the terminal count is given by the following equation:

\[
\text{Delay} = \text{Count} \cdot (\text{System Clock Period})
\]

To start counting, the Timer has to be triggered via the Start Timer Command. To restart the Timer after terminal count or while counting, the software has to issue the trigger command again. While counting the Timer can be enabled or disabled by using a software controlled Gate. It is also possible to output a pulse generated upon terminal count through the TA or TB pins.

In 1X clock mode the only clock source available is the SCLK pin. The serial machines (both Tx Machine and Rx Machine) can independently use one of two clock modes, either 1X or 16X. Also no configuration changes are allowed during operation as each write in the BRG configuration registers causes a reset signal to be sent to the BRG logic. The mode or source clocks may be changed only after a Hardware or Software reset. The Divisor (or count, depending upon the mode) may be updated during operation unless the particular BRG machine is being used as a clock source for one of the serial machines, and the particular serial machine is in operation at the time. Loading the count registers with "0" is forbidden in all cases, and loading it with a "1" is forbidden in the Timer Mode only.

### SERIAL DIAGNOSTICS

The 82510 supports two modes of Loopback operation, Local Loopback and Remote Loopback as well as an Echo mode for diagnostics and improved throughput.

**LOCAL LOOPBACK**

The Tx Machine output and Rx Machine input are shorted internally, TXD pin output is held at Mark. This feature allows simulation of Transmission/Reception of characters and checks the Tx FIFO, Tx Machine, Rx Machine, and Rx FIFO along with the software without any external side effects. The modem outputs OUT1, OUT2, DTR and RTS are internally shorted to RI, DCD, DSR and CTS respectively. OUT0 is held at a mark state.

![Figure 13. Local Loopback](image)
The 82510 powers down when the power down command is issued via the Internal Command Register (ICM). There are two modes of power down, Sleep and Idle.

In Sleep mode, even the system clock of the 82510 is shut down. The system clock source of the 82510 can either be the Crystal Oscillator or an external clock source. If the Crystal Oscillator is being used and the power down command is issued, then the 82510 will automatically enter the Sleep mode. If an external clock is being used, then the user must disable the external clock in addition to issuing the Power Down command, to enter the Sleep mode. The benefit of this mode is the increased savings in power consumption (typical power consumption in the Sleep mode is in the ranges of 100s of microAmps). However, upon wake up, the user must reprogram the device. To exit this mode the user can either issue a Hardware reset, or read the FIFO Level Register (FLR) and then issue a software reset. In either case the contents of the 82510 registers are not preserved and the device must be reprogrammed prior to operation. If the Crystal Oscillator is being used then the user must allow enough time for the oscillator to wake up before issuing the software reset.

The 82510 is in the idle mode when the Power Down command is issued and the system clock is still running (i.e. the system clock is generated externally and not disabled by the user). In this mode the contents of all registers and memory cells are preserved, however, the power consumption in this mode is greater than in the Sleep mode. Reading FLR will take the 82510 out of this mode.

NOTE:
The data read from FLR when exiting Power Down is invalid and should be ignored.

REMOTE LOOPBACK

![Diagram of Remote Loopback]

Figure 14. Remote Loopback

The TXD pin and RXD pin are shorted internally (the data is not sent on to the RX Machine). This feature allows the user to check the communications channel as well as the Tx and Rx pin circuits not checked in the Local Loopback mode.

AUTO ECHO

![Diagram of Auto Echo]

Figure 15. Auto Echo

In Echo Mode the received characters are automatically transmitted back. When the characters are read from the Rx FIFO they are automatically pushed back onto the Tx FIFO (the flags are also included). The Rx Machine baud rate must be equal to, or less than, the Tx Machine baud rate or some of the characters may be lost. The user has an option of preventing echo of special characters; Control Characters and characters with Errors.

Power Down Mode

The 82510 has a “power down” mode to reduce power consumption when the device is not in use.
# DETAILED REGISTER DESCRIPTION

## Table 9. Register Map

<table>
<thead>
<tr>
<th>Bank</th>
<th>Address</th>
<th>Read Register</th>
<th>Write Register</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 (NAS)</td>
<td>0 (DLAB = 0)</td>
<td>RXD</td>
<td>TXD</td>
</tr>
<tr>
<td>8250A/16450</td>
<td>1 (DLAB = 0)</td>
<td>GER</td>
<td>GER</td>
</tr>
<tr>
<td></td>
<td>0 (DLAB = 1)</td>
<td>BAL</td>
<td>BAL</td>
</tr>
<tr>
<td></td>
<td>1 (DLAB = 1)</td>
<td>BAH</td>
<td>BAH</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>GIR/BANK</td>
<td>BANK</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>LCR</td>
<td>LCR</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>MCR</td>
<td>MCR</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>LSR</td>
<td>LSR</td>
</tr>
<tr>
<td></td>
<td>6</td>
<td>MSR</td>
<td>MSR</td>
</tr>
<tr>
<td></td>
<td>7</td>
<td>ACR0</td>
<td>ACR0</td>
</tr>
<tr>
<td>1 (WORK)</td>
<td>0</td>
<td>RXD</td>
<td>TXD</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>RXF</td>
<td>TXF</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>GIR/BANK</td>
<td>BANK</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>TMST</td>
<td>TMCR</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>FLR</td>
<td>MCR</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>RST</td>
<td>RCM</td>
</tr>
<tr>
<td></td>
<td>6</td>
<td>MSR</td>
<td>TCM</td>
</tr>
<tr>
<td></td>
<td>7</td>
<td>GSR</td>
<td>ICM</td>
</tr>
<tr>
<td>2 (GENERAL CONF)</td>
<td>0</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>FMD</td>
<td>FMD</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>GIR/BANK</td>
<td>BANK</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>TMD</td>
<td>TMD</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>IMD</td>
<td>IMD</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>ACR1</td>
<td>ACR1</td>
</tr>
<tr>
<td></td>
<td>6</td>
<td>RIE</td>
<td>RIE</td>
</tr>
<tr>
<td></td>
<td>7</td>
<td>RMD</td>
<td>RMD</td>
</tr>
<tr>
<td>3 (MODEM CONF)</td>
<td>0 (DLAB = 0)</td>
<td>CLCF</td>
<td>CLCF</td>
</tr>
<tr>
<td></td>
<td>1 (DLAB = 0)</td>
<td>BACF</td>
<td>BACF</td>
</tr>
<tr>
<td></td>
<td>0 (DLAB = 1)</td>
<td>BBL</td>
<td>BBL</td>
</tr>
<tr>
<td></td>
<td>1 (DLAB = 1)</td>
<td>BBH</td>
<td>BBH</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>GIR/BANK</td>
<td>BANK</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>BBCF</td>
<td>BBCF</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>PMD</td>
<td>PMD</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>MIE</td>
<td>MIE</td>
</tr>
<tr>
<td></td>
<td>6</td>
<td>TMIE</td>
<td>TMIE</td>
</tr>
<tr>
<td></td>
<td>7</td>
<td>—</td>
<td>—</td>
</tr>
</tbody>
</table>

(1) ACRO is used in INS8250 as a Scratch-Pad Register
(2) DLAB = LCR Bit #7

The 82510 has thirty-five registers which are divided into four banks of register banks. Only one bank is accessible at any one time. The bank is selected through the BANK1-0 bits in the GIR/BANK register. The individual registers within a bank are selected by the three address lines (A2-0). The 82510 registers can be grouped into the following categories.
### BANK ZERO 8250A/16450—COMPATIBLE BANK

<table>
<thead>
<tr>
<th>Register</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
<th>Address</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>TxD (33)</td>
<td>Tx Data bit 7</td>
<td>Tx Data bit 6</td>
<td>Tx Data bit 5</td>
<td>Tx Data bit 4</td>
<td>Tx Data bit 3</td>
<td>Tx Data bit 2</td>
<td>Tx Data bit 1</td>
<td>Tx Data bit 0</td>
<td>0</td>
<td>—</td>
</tr>
<tr>
<td>RxD (35)</td>
<td>Rx Data bit 7</td>
<td>Rx Data bit 6</td>
<td>Rx Data bit 5</td>
<td>Rx Data bit 4</td>
<td>Rx Data bit 3</td>
<td>Rx Data bit 2</td>
<td>Rx Data bit 1</td>
<td>Rx Data bit 0</td>
<td>0</td>
<td>—</td>
</tr>
<tr>
<td>BAL (11)</td>
<td>BRGA LSB Divide Count (DLAB = 1)</td>
<td>0</td>
<td>02H</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BAH (12)</td>
<td>BRGA MSB Divide Count (DLAB = 1)</td>
<td>1</td>
<td>00H</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GIR/BANK (21)</td>
<td>0</td>
<td>0</td>
<td>Timer Interrupt Enable</td>
<td>Tx Machine Interrupt Enable</td>
<td>Modern Interrupt Enable</td>
<td>Rx Machine Interrupt Enable</td>
<td>Tx FIFO Interrupt Enable</td>
<td>Rx FIFO Interrupt Enable</td>
<td>1</td>
<td>00H</td>
</tr>
<tr>
<td>LCR (2)</td>
<td>0</td>
<td>0</td>
<td>BANK Pointer bit 0</td>
<td>BANK Pointer bit 1</td>
<td>0</td>
<td>Active Block Int bit 2</td>
<td>Active Block Int bit 1</td>
<td>Active Block Int bit 0</td>
<td>Interrupt Pending</td>
<td>2</td>
</tr>
<tr>
<td>MCR (32)</td>
<td>0</td>
<td>0</td>
<td>OUT 0 Complement</td>
<td>Loopback Control bit</td>
<td>OUT 2 Complement</td>
<td>OUT 1 Complement</td>
<td>RTS Complement</td>
<td>DTR Complement</td>
<td>4</td>
<td>00H</td>
</tr>
<tr>
<td>LSR (22)</td>
<td>0</td>
<td>0</td>
<td>TxF Idle</td>
<td>TxF FIFO Interrupt</td>
<td>Break Detected</td>
<td>Framing Error</td>
<td>Parity Error</td>
<td>Overrun Error</td>
<td>Rx FIFO Int Req</td>
<td>5</td>
</tr>
<tr>
<td>MSR (27)</td>
<td>0</td>
<td>0</td>
<td>DCD Input Inverted</td>
<td>RI Input Inverted</td>
<td>DSR Input Inverted</td>
<td>CTS Input Inverted</td>
<td>State Change in DCD</td>
<td>State (H → L) Change in DSR</td>
<td>State Change in DSR</td>
<td>6</td>
</tr>
<tr>
<td>ACR0 (5)</td>
<td>0</td>
<td>0</td>
<td>Address or Control Character Zero</td>
<td>7</td>
<td>00H</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### BANK ONE—GENERAL WORK BANK

<table>
<thead>
<tr>
<th>Register</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
<th>Address</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>TxD (33)</td>
<td>Tx Data bit 7</td>
<td>Tx Data bit 6</td>
<td>Tx Data bit 5</td>
<td>Tx Data bit 4</td>
<td>Tx Data bit 3</td>
<td>Tx Data bit 2</td>
<td>Tx Data bit 1</td>
<td>Tx Data bit 0</td>
<td>0</td>
<td>—</td>
</tr>
<tr>
<td>RxD (35)</td>
<td>Rx Data bit 7</td>
<td>Rx Data bit 6</td>
<td>Rx Data bit 5</td>
<td>Rx Data bit 4</td>
<td>Rx Data bit 3</td>
<td>Rx Data bit 2</td>
<td>Rx Data bit 1</td>
<td>Rx Data bit 0</td>
<td>0</td>
<td>—</td>
</tr>
<tr>
<td>RxF (24)</td>
<td>Rx Char Ok</td>
<td>Rx Char Noisy</td>
<td>Rx Char Parity Error</td>
<td>Address or Control Character</td>
<td>Break Flag</td>
<td>Rx Char Framing Error</td>
<td>Ninth Data bit of Rx Char</td>
<td>1</td>
<td>—</td>
<td></td>
</tr>
<tr>
<td>TxF (34)</td>
<td>Address Marker bit</td>
<td>Software Parity bit</td>
<td>Ninth bit of Data Char</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>—</td>
</tr>
<tr>
<td>GIR/BANK (21)</td>
<td>0</td>
<td>0</td>
<td>BANK Pointer bit 0</td>
<td>BANK Pointer bit 1</td>
<td>0</td>
<td>Active Block Int bit 2</td>
<td>Active Block Int bit 1</td>
<td>Active Block Int bit 0</td>
<td>Interrupt Pending</td>
<td>2</td>
</tr>
<tr>
<td>TMST (26)</td>
<td>0</td>
<td>0</td>
<td>Gate B State</td>
<td>Gate A State</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>Timer B Expired</td>
<td>3</td>
</tr>
<tr>
<td>TMCR (31)</td>
<td>0</td>
<td>0</td>
<td>Trigger Gate B</td>
<td>Trigger Gate A</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>Start Timer B</td>
<td>3</td>
</tr>
<tr>
<td>MCR (32)</td>
<td>0</td>
<td>0</td>
<td>OUT 0 Complement</td>
<td>Loopback Control bit</td>
<td>OUT 2 Complement</td>
<td>OUT 1 Complement</td>
<td>RTS Complement</td>
<td>DTR Complement</td>
<td>4</td>
<td>00H</td>
</tr>
</tbody>
</table>

**NOTE:**
The register number is provided as a reference number for the register description.
### BANK ONE—GENERAL WORK BANK (Continued)

<table>
<thead>
<tr>
<th>Register</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
<th>Address</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>FLR (25)</td>
<td>—</td>
<td>—</td>
<td>Rx FIFO Level</td>
<td>—</td>
<td>—</td>
<td>Tx FIFO Level</td>
<td>4</td>
<td>00H</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RST (23)</td>
<td>Address/Control Character Received</td>
<td>Break Terminated</td>
<td>Framing Error</td>
<td>Parity Error</td>
<td>Overrun Error</td>
<td>Rx FIFO Interrupt Requested</td>
<td>5</td>
<td>00H</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RCM (30)</td>
<td>Rx Enable</td>
<td>Rx Disable</td>
<td>Flush RxM</td>
<td>Lock Rx FIFO</td>
<td>Open Rx FIFO</td>
<td>0</td>
<td>0</td>
<td>5</td>
<td>—</td>
<td></td>
</tr>
<tr>
<td>MSR (27)</td>
<td>DCD Complement</td>
<td>RI Input Inverted</td>
<td>DSR Input Inverted</td>
<td>CTS Input Inverted</td>
<td>State Change in DCD</td>
<td>State Change in RI</td>
<td>State Change in DSR</td>
<td>State Change in CTS</td>
<td>6</td>
<td>00H</td>
</tr>
<tr>
<td>TCM (29)</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>Flush Tx Machine</td>
<td>Flush Tx FIFO</td>
<td>Tx Enable</td>
<td>Tx Disable</td>
<td>6</td>
<td>—</td>
</tr>
<tr>
<td>GSR (20)</td>
<td>—</td>
<td>—</td>
<td>Timer Interrupt</td>
<td>RxM Interrupt</td>
<td>Manual Int Acknowledge Command</td>
<td>Status Clear</td>
<td>Power Down Mode</td>
<td>0</td>
<td>7</td>
<td>—</td>
</tr>
</tbody>
</table>

### BANK TWO—GENERAL CONFIGURATION

<table>
<thead>
<tr>
<th>Register</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
<th>Address</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>FMD (4)</td>
<td>0</td>
<td>0</td>
<td>Rx FIFO Threshold</td>
<td>0</td>
<td>0</td>
<td>Tx FIFO Threshold</td>
<td>1</td>
<td>00H</td>
<td></td>
<td></td>
</tr>
<tr>
<td>GIR/BANK (21)</td>
<td>0</td>
<td>BANK Pointer bit 1</td>
<td>BANK Pointer bit 0</td>
<td>0</td>
<td>Active Block Int bit 2</td>
<td>Active Block Int bit 1</td>
<td>Active Block Int bit 0</td>
<td>Interrupt Pending</td>
<td>2</td>
<td>01H</td>
</tr>
<tr>
<td>TMD (3)</td>
<td>Error Echo Disable</td>
<td>Control Character Echo Disable</td>
<td>9-bit Character Length</td>
<td>Transmit Mode</td>
<td>Software Parity Mode</td>
<td>Stop Bit Length</td>
<td>3</td>
<td>00H</td>
<td></td>
<td></td>
</tr>
<tr>
<td>IMD (1)</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>Interrupt Acknowledge Mode</td>
<td>Rx FIFO Depth</td>
<td>Rx FIFO ulan Mode Select</td>
<td>Loopback or Echo Mode of Operation</td>
<td>4</td>
<td>0CH</td>
<td></td>
</tr>
<tr>
<td>ACR1 (6)</td>
<td>Address or Control Character 1</td>
<td>5</td>
<td>00H</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RIE (17)</td>
<td>Address/Control Character Recognition Interrupt Enable</td>
<td>Break Terminated</td>
<td>Framing Error</td>
<td>Parity Error</td>
<td>Overrun Error</td>
<td>0</td>
<td>6</td>
<td>1EH</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RMD (7)</td>
<td>Address/Control Character Mode</td>
<td>Disable DPLL</td>
<td>Sampling Window Mode</td>
<td>Start bit Sampling Mode</td>
<td>0</td>
<td>0</td>
<td>7</td>
<td>00H</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### BANK THREE—MODEM CONFIGURATION

<table>
<thead>
<tr>
<th>Register</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
<th>Address</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLCF (8)</td>
<td>Rx Clock Mode</td>
<td>Rx Clock Source</td>
<td>Tx Clock Mode</td>
<td>Tx Clock Source</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>00H</td>
</tr>
<tr>
<td>BACF (9)</td>
<td>0</td>
<td>BRGA Clock Source</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>BRGA Mode</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>04H</td>
</tr>
<tr>
<td>BBL (13)</td>
<td>BRGB LSB Divide Count (DLAB = 1)</td>
<td>0</td>
<td>05H</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BSH (14)</td>
<td>BRGB MSB Divide Count (DLAB = 1)</td>
<td>1</td>
<td>00H</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
BANK THREE—MODEM CONFIGURATION (Continued)

<table>
<thead>
<tr>
<th>Register</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
<th>Address</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>GIR/BANK (21)</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>01H</td>
</tr>
<tr>
<td>BBCF (10)</td>
<td>BRGB Clock Source</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>3</td>
<td>84H</td>
</tr>
<tr>
<td>PMD (15)</td>
<td>DCD/ICLK/OUT 1</td>
<td>DCD/ICLK/OUT 0</td>
<td>DSR/TA/OUT 0</td>
<td>DSR/TA/OUT 0</td>
<td>RI/SCLK Function</td>
<td>DTR/TB Function</td>
<td>0</td>
<td>0</td>
<td>4</td>
<td>FCH</td>
</tr>
<tr>
<td>MIE (19)</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>5</td>
<td>0FH</td>
</tr>
<tr>
<td>TMIE (18)</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>6</td>
<td>00H</td>
</tr>
</tbody>
</table>

**CONFIGURATION**

These read/write registers are used to configure the device. They may be read at anytime; however, they may be written to only when the device is idle. Typically they are written to only once after system power up. They are set to default values upon Hardware or Software Reset (Default Wake-Up Mode). The default values are chosen so as to allow the 82510 to be fully software compatible with the IBM PC Async Adapter (INS 8250A/16450) when in the default wake-up mode. The 82510 can operate in the High Performance mode by programming the configuration registers as necessary.

The configuration options available to the user are listed below.

**Table 11. Configuration Options**

<table>
<thead>
<tr>
<th>Interrupt Acknowledge Mode</th>
<th>FIFO</th>
<th>Control Character Recognition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Manual</td>
<td>RX FIFO Depth</td>
<td>None</td>
</tr>
<tr>
<td>Receive</td>
<td>RX, TX Threshold</td>
<td>Standard</td>
</tr>
<tr>
<td>Manual, Semi-Automatic, Automatic</td>
<td></td>
<td>ASCII, EBCDIC</td>
</tr>
<tr>
<td>LAN (8051)</td>
<td></td>
<td>Two User Programmed</td>
</tr>
<tr>
<td>Address Recognition</td>
<td></td>
<td>TX Operation</td>
</tr>
<tr>
<td>Manual, Semi-Automatic, Automatic</td>
<td></td>
<td>RTS/CTS Control</td>
</tr>
<tr>
<td>Diagnostics</td>
<td></td>
<td>Manual, Semi-Automatic, Automatic</td>
</tr>
<tr>
<td>Loopback</td>
<td></td>
<td>Parity Mode</td>
</tr>
<tr>
<td>Remote</td>
<td></td>
<td>Stop Bit Length</td>
</tr>
<tr>
<td>Local</td>
<td></td>
<td>Character Size</td>
</tr>
<tr>
<td>Echo</td>
<td></td>
<td>I/O Pins</td>
</tr>
<tr>
<td>Yes/No</td>
<td></td>
<td>Select Function for Each</td>
</tr>
<tr>
<td>Disable Error Echo</td>
<td></td>
<td>Multifunction Pin</td>
</tr>
<tr>
<td>Disable Control/Address</td>
<td></td>
<td>Select Direction for Multi-</td>
</tr>
<tr>
<td>Char. Echo</td>
<td></td>
<td>function Pin (If Applicable)</td>
</tr>
<tr>
<td>I/O Pins</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

2-55
1. IMD—INTERNAL MODE REGISTER

This register defines the general device mode of operation. The bit functions are as follows:

<table>
<thead>
<tr>
<th>Bit</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>7-4</td>
<td>Reserved</td>
</tr>
<tr>
<td>IAM</td>
<td>Interrupt Acknowledge Mode Bit</td>
</tr>
<tr>
<td>0</td>
<td>Manual acknowledgement of pending interrupts</td>
</tr>
<tr>
<td>1</td>
<td>Automatic acknowledgement of pending interrupts (upon CPU service)</td>
</tr>
<tr>
<td>ULM</td>
<td>uLAN Mode</td>
</tr>
<tr>
<td>0</td>
<td>Normal Mode</td>
</tr>
<tr>
<td>1</td>
<td>uLAN Mode</td>
</tr>
<tr>
<td>LEM</td>
<td>Loopback/Echo Mode Select</td>
</tr>
<tr>
<td>0</td>
<td>No Echo</td>
</tr>
<tr>
<td>1</td>
<td>Echo Operation</td>
</tr>
<tr>
<td>RFD</td>
<td>Receive FIFO Depth</td>
</tr>
<tr>
<td>0</td>
<td>Four Bytes</td>
</tr>
<tr>
<td>1</td>
<td>One Byte</td>
</tr>
</tbody>
</table>

This bit, when set, configures the 82510 for the automatic acknowledge mode. This causes the 82510 INT line to go low for two clock cycles upon service of the interrupt. After two clock cycles it is then updated. It is useful in the edge triggered mode. In manual acknowledgement mode the CPU must explicitly issue a command to clear the INT pin. (The INT pin then goes low for a minimum of two clock cycles until another enabled status register bit is set.)

This bit configures the depth of the Rx FIFO. With a FIFO depth of one, the FIFO will act as a 1-byte buffer to emulate the 8250A.
2. LCR—LINE CONFIGURE REGISTER

This register defines the basic configuration of the serial link.

**DLAB**—Divisor Latch Access Bit—This bit, when set, allows access to the Divisor Count registers BAL, BAH, BBL, BBH registers.

**SBK**—Set Break Bit—This bit will force the TxD pin low. The TxD pin will remain low (regardless of all activities) until this bit is reset.

**PM2**–**PM0**—Parity Mode Bits—These three bits combine with the SPF bit of the Transmit Mode register to define the various parity modes. See Table 12.

**SBL0**—Stop Bit Length—This bit, together with SBL1 and SBL2 bits of the Transmit Mode register, defines the stop-bit lengths for transmission. The Rx machine can identify 3/4 stop bit or more. See Table 13.

### Table 12. Parity Modes

<table>
<thead>
<tr>
<th>PM0</th>
<th>SPF</th>
<th>PM2</th>
<th>PM1</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>No Parity</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>Odd Parity</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>Even Parity</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>High Parity</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>Low Parity</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>Software Parity</td>
</tr>
</tbody>
</table>

### Table 13. Stop Bit Length

<table>
<thead>
<tr>
<th>SBL2</th>
<th>SBL1</th>
<th>SBL0</th>
<th>Stop Bit Length</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>16X</td>
<td>1X</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>4/4</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>6/4 or 8/4*</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>3/4</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>4/4</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>5/4</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>6/4</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>7/4</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>8/4</td>
</tr>
</tbody>
</table>

*6/4 if character length is 5 bits; else 8/4

### Table 14. Character Length

<table>
<thead>
<tr>
<th>NBCL</th>
<th>CL1</th>
<th>CL0</th>
<th>Character Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>5 BITS</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>6 BITS</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>7 BITS</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>8 BITS</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>9 BITS</td>
</tr>
</tbody>
</table>
3. TMD—Transmit Machine Mode Register

This register together with the Line Configure Register defines the Tx machine mode of operation.

**EED**—Error Echo Disable—Disables Echo of characters received with errors (valid in echo mode only).

**CED**—Control Character Echo Disable—Disables Echo of characters recognized as control characters (or address characters in uLAN mode). Valid in echo mode only.

**NBCL**—Nine-Bit Length—This bit, coupled with LCR (CL0, CL1), selects Transmit/Receive character length of nine bits. See Table 14.

**TM1-TM0**—Transmit Mode—These bits select one of three modes of control over the CTS and RTS lines.

**00**—Manual Mode—In this mode the CPU has full control of the Transmit operation. The CPU has to explicitly enable/disable transmission, and activate/check the RTS/CTS pins.

**01**—Reserved

**10**—Semi-Automatic Mode—in this mode the 82510 transmits only when CTS input is active. The 82510 activates the RTS output as long as transmission is enabled.

**11**—Automatic Mode—in this mode the 82510 transmits only when CTS input is active. The RTS output is activated only when transmission is enabled and there is more data to transmit.

**SPF**—Software Parity Force—This bit defines the parity modes along with the PM0, PM1, and PM2 bits of the LCR register. When software parity is enabled the software must determine the parity bit through the TxF register on transmission, or check the parity bit in RxF upon reception. See Table 12.

**SBL2-SBL1**—Stop Bit Length—These bits, together with the SBL0 bit of the LCR register define the stop bit length. See Table 13.
4. FMD—FIFO MODE REGISTER

**FMD—FIFO Mode Register**

This register configures the Tx and Rx FIFO's threshold levels—the occupancy levels that can cause an interrupt.

7—6—Reserved

**RFT1—RFT0—Receive FIFO Threshold**—When the Rx FIFO occupancy is greater than the level indicated by these bits the Rx FIFO Interrupt is activated.

3—2—Reserved

**TFT1—TFT0—Transmit FIFO Threshold**—When the TX FIFO occupancy is less than or equal to the level indicated by these bits the Tx FIFO Interrupt is activated.

5. ACR0—ADDRESS/CONTROL CHARACTER REGISTER 0

**ACR0—Address/Control Character Register 0**

This register contains a byte which is compared to each received character. The exact function depends on the configuration of the IMD register.

In normal mode this register may be used to program a special control character; a matched character will be reported in the RECEIVE STATUS register. The maximum length of the control characters is eight bits. If the length is less than eight bits then the character must be right justified and the leading bits be filled with zeros.

In uLAN mode this register contains the eight-bit station address for recognition. In this mode only incoming address characters (i.e., characters with address bit set) will be compared to these register. The PCRF bit in the RECEIVE STATUS register will be set when an Address or Control Character match occurs.

6. ACR1—ADDRESS/CONTROL CHARACTER REGISTER 1

**ACR1—Address/Control Character Register 1**

**NOTE:**
This register is identical in function to ACR0.
7. RMD—RECEIVE MACHINE MODE REGISTER

This register defines the Rx Machine mode of operation.

**uCM0, uCM1—uLAN/Control Character Recognition Mode**—In normal mode it defines the Control Character recognition mode. In uLAN mode they define modes of address recognition.

In uLAN mode: selects the mode of address recognition.

**00—Manual Mode**—Rx Machine reports reception of any address character, via CRF bit of RECEIVE STATUS register, and writes it to the Rx FIFO.

**01—Semi-Automatic Mode**—Operates the same as Manual Mode but, in addition, the Rx Machine OPENS (unlocks) the Rx FIFO upon reception of any address characters. Subsequent received characters will be written into the FIFO. (Note: it is the user’s responsibility to LOCK the FIFO if the address character does not match the station’s address.)

**10—Automatic Mode**—The Rx Machine will OPEN (unlock) the Rx FIFO upon Address Match. In addition the Rx Machine LOCKs the Rx FIFO upon recognition of address mismatch; i.e., it controls the flow of characters into the Rx FIFO depending upon the results of the address comparison. If a match occurs it will allow characters to be sent to the FIFO; if a mismatch occurs it will keep the characters out of the FIFO by LOCKING it.

**11—Reserved**

In normal Mode: selects the mode of Standard Set Control Character Recognition (programmed control characters are always recognized).

00—No Standard Set Control Characters Recognized.

01—ASCII Control Characters (00H—1 FH + 7FH).

10—Reserved.

11—EBCDIC Control Character Recognized (00H—3FH).

**DPD—Disable Digital Phase Locked Loop**—When set, disables the DPLL machine. (Note: using the DPLL in a very noisy media, may increase the error rate.)

**SWM—Sampling Window Mode**—This bit controls the mode of data sampling:

0—Small Window, 3/16 sampling.

1—Large Window, 7/16 sampling.

**SSM—Start Bit Sampling Mode**—This bit controls the mode of Start Bit sampling.

0—Majority Voting for start bit. In this mode a majority of the samples determines the bit.

1—in this mode if one of the bit samples is not ‘0’, the start bit will not be detected.
8. CLCF—CLOCKS CONFIGURE REGISTER

This register defines the Transmit and Receive Code modes and sources.

RxCM—Rx Clock Mode—This bit selects the mode of the receive clock which is used to sample the received data.
0—16X Mode.
1—1X Mode. In this mode the receive data must be synchronous to the Rx Clock; supplied via the SCLK pin.

RxCS—Rx Clock Source—This bit selects the source of the internal receive clock in the case of 16X mode (as programmed by the RxCM bit above).
0—BRGB Output
1—BRGA Output

TxCM—Transmit Clock Mode—This bit selects the mode of the Transmit Data Clock, which is used to clock out the Transmit Data.
0—16X Mode
1—1X Mode. In this mode the Transmit data is synchronous to the Serial Clock; supplied via the SCLK pin.

TxCS—Transmit Clock Source—Selects the source of internal Transmit Clock in case of 16X mode.
0—BRGB Output.
1—BRGA Output.

9. BACF—BRGA CONFIGURATION REGISTER

This register defines the BRGA clock sources and the mode of operation.

BACS—BRGA Clock Source—Selects the input clock source for Baud Rate Generator A.
0—System Clock
1—SCLK Pin

This bit has no effect if BRGA is configured as a timer.
10. BBCF—BRGB CONFIGURATION REGISTER

This register defines the BRGB clock sources and mode of operation. (Note: BRGB can also take its Input Clock from the output of BRGA.)

BBCS1, BBCS0—These two bits together define the input Clock Sources for BRGB. These bits have no effect when in the timer mode.

- 00—System Clock
- 01—SCLK Pin
- 10—BRGA Output
- 11—Reserved

BBM—BRGB Mode of Operation.

- 0—Timer Mode (In this mode the input clock source is always the system clock.)
- 1—BRG Mode

11. BAL—BRGA DIVIDE COUNT LEAST SIGNIFICANT BYTE

This register contains the least significant byte of the BRGA divisor/count.

12. BAH—BRGA DIVIDE COUNT MOST SIGNIFICANT BYTE

This register contains the most significant byte of the BRGA divisor/count.

13. BBL—BRGB DIVIDE COUNT LEAST SIGNIFICANT BYTE

This register contains the least significant byte of the BRGB divisor/count.
This register contains the most significant byte of the BRGB divisor/count.

15. PMD—I/O PIN MODE REGISTER

This register is used to configure the direction and function of the multifunction pins. The following options are available on each pin:

1. Direction: Input or Output Pin.
   - 0—Defines the pin as an output pin (general purpose or special function).
   - 1—Defines the pin as an input pin.

2. Function: General purpose or special purpose pin (no effect if the pin is programmed as an input).
   - 0—special function output pin.
   - 1—general purpose output pin.
     - DIOD—DCD/ICLK/OUT1 Direction.
     - DIOF—DCD/ICLK/OUT1 Function (output mode only).
   - 0—Output: ICLK or OUT1 (depending on bit DIOF)
   - 1—Input: DCD.
     - DIOF—DCD/ICLK/OUT1 Function (output mode only).

   - 0—ICLK (Output of the Internal System Clock).
   - 1—OUT1 general purpose output, controlled by MODEM CONTROL Register
     - DTAD—DSR/TA/OUT0 Direction.
   - 0—Output: TA or OUT0 (Dependent upon DTAF).
   - 1—Input: DSR.
     - DTAF—DSR/TA/OUT0 Direction (output mode only).
   - 0—TA (BRGA Output or Timer A Termination Pulse).
   - 1—OUT0 (general purpose output, controlled by MODEM CONTROL).
     - RRF—RI/SCLK Function
   - 0—SCLK (Receive and/or Transmit Clock)
   - 1—RI
     - DTF—DTR/TB Function
   - 0—TB (BRGB Output Clock on Timer B termination pulse depending upon the mode of BRGB).
   - 1—DTR
INTERRUPT/STATUS REGISTERS

The 82510 uses a two layer approach to handle interrupt and status generation. Device level registers show the status of the major 82510 functional block (MODEM, FIFO, Tx MACHINE, Rx MACHINE, TIMERS, etc.). Each block may be examined by reading its individual block level registers. Also each block has interrupt enable/generation logic which may generate a request to the built-in interrupt controller, the interrupt requests are then resolved on a priority basis.

16. GER—GENERAL ENABLE REGISTER

This register enables or disables the bits of the GSR register from being reflected in the GIR register. It serves as the device enable register and is used to mask the interrupt requests from any of the 82510 block (See Figure 1).

TIE—Timers Interrupt Enable

TxIE—Transmit Machine Interrupt Enable.

17. RIE—RECEIVE INTERRUPT ENABLE REGISTER

This register enables interrupts from the Rx Machine. It is used to mask out interrupt requests generated by the status bits of the RST register.

CRE—Control/uLAN Address Character Recognition Interrupt Enable.—Enables Interrupt when CRF bit of RST register is set.

PCRE—Programmable Control/Address Character Match Interrupt Enable.—Enables Interrupt on PCRF bit of RST.

BkTe—Break Termination Interrupt Enable.

Interrupt Masking

The 82510 has a device enable register, GER, which can be used to enable or mask-out any block interrupt request. Some of the blocks (Rx Machine, Modem, Timer) have an enable register associated with their status register which can be used to mask out the individual sources within the block. Interrupts are enabled when programmed high.

GER—General Enable Register

<table>
<thead>
<tr>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>RFIE—Rx FIFO INTERRUPT ENABLE</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TFIE—FIFO INTERRUPT ENABLE</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RxIE—Rx INTERRUPT ENABLE</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MIE—MODEM INTERRUPT ENABLE</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RESERVED</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TIMERS INTERRUPT ENABLE—TIE</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Tx INTERRUPT ENABLE—TxIE</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

RIE—Receive Interrupt Enable Register

<table>
<thead>
<tr>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>RESERVED</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CONTROL/ADDRESS RECOGNITION</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CONTROL/ADDRESS MATCH</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BREAK TERMINATED</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BREAK DETECTED</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>OVERRUN ERROR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PARITY ERROR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FRAMING ERROR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

BkDE—Break Detection Interrupt Enable—Enable Interrupt on BkD bit of RST.

FEE—Framing Error Enable—Enable Interrupt on FE bit of RST.

PEE—Parity Error Enable—Enable Interrupt on PE bit of RST.

OEE—Overrun Error Enable—Enable Interrupt on OE bit of RST.
18. TMIE—TIMER INTERRUPT ENABLE REGISTER

This is the enable register for the Timer Block. It is used to mask out interrupt requests generated by the status bits of the TMST register.

- **TAIE**—Timer A Expired Interrupt Enable—Enables Interrupt on TAE x bit of TMST.
- **TBIE**—Timer B Expired Interrupt Enable—Enables Interrupt on TBEx bit of TMST.

19. MIE—MODEM INTERRUPT ENABLE REGISTER

This register enables interrupts from the Modem Block. It is used to mask out interrupt requests generated by the status bits of the MODEM STATUS register.

- **DCDE**—Delta DCD Interrupt Enable—Enables Interrupt on DDCD bit of MODEM STATUS.
- **RIE**—Delta RI Interrupt Enable—Enables Interrupt on DRI bit of MODEM STATUS.
- **DSRE**—Delta DSR Interrupt Enable—Enables Interrupt on DSR bit of MODEM STATUS.
- **CTSE**—Delta CTS Interrupt Enable—Enables Interrupt on DCTS bits of MODEM STATUS.

**STATUS/INTERRUPT**

The 82510 has two device status registers, which reflect the overall status of the device, and five block status registers. The two device status registers, GSR and GIR, and supplementary in function. GSR reflects the interrupt status of all blocks, whereas GIR depicts the highest priority interrupt only. GIR is updated after the GSR register; the delay is of approximately two clock cycles.
20. **GSR—GENERAL STATUS REGISTER**

This register reflects all the pending block-level interrupt requests. Each bit in GSR reflects the status of a block and may be individually enabled by GER. GER masks-out interrupts from GIR; it does not have any effect on the bits in GSR. The only way that the bits can be masked out in GSR (i.e., not appear in GSR) is if they are masked out at the lower level.

**TIR—Timers Interrupt Request**—This bit indicates that one of the timers has expired. (See TMST)

**TxFIR—Transmit Machine Interrupt Request**—Indicates that the Transmit Machine is either empty or disabled (Idle).

**MIR—Modem Interrupt Request**—This bit, if set, indicates an interrupt from the Modem Module. (As reflected in MODEM STATUS.)

**RxFIR—Receive Machine Interrupt Request**—(As reflected in RST.)

**TFIR—Transmit FIFO Interrupt Request**—Tx FIFO occupancy is below or equal to threshold.

**RFIR—Receive FIFO Interrupt Request**—Rx FIFO Occupancy is above threshold.

21. **GIR/BANK—GENERAL INTERRUPT REGISTER/BANK REGISTER**

This register holds the highest priority enabled pending interrupt from GSR. In addition it holds a pointer to the current register segment. Writing into this register will update only the BANK bits.

**BANK1, BANK0—Bank Pointer Bits**—These two bits point to the currently accessible register bank. The user can read and write to these bits. The address of this register is always two within all four register banks.

**B12, B11, B10—Interrupt Bits 0–2**—These three bits reflect the highest priority enabled pending interrupt from GSR.

**IPN—Interrupt Pending**—This bit is active low. It indicates that there is an interrupt pending. The interrupt logic asserts the INT pin as soon as this bit goes active. (Note: the GIR register is continuously updated; so that, while the user is serving one interrupt source, a new interrupt with higher priority may enter GIR and replace the older interrupt.)

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>Reserved</td>
</tr>
<tr>
<td>6</td>
<td>Reserved</td>
</tr>
<tr>
<td>5</td>
<td>TIMER INTERRUPT</td>
</tr>
<tr>
<td>4</td>
<td>Tx MACHINE INTERRUPT</td>
</tr>
<tr>
<td>3</td>
<td>Rx FIFO INTERRUPT</td>
</tr>
<tr>
<td>2</td>
<td>Rx MACHINE INTERRUPT</td>
</tr>
<tr>
<td>1</td>
<td>Tx FIFO INTERRUPT</td>
</tr>
<tr>
<td>0</td>
<td>MODEM INTERRUPT</td>
</tr>
</tbody>
</table>

**B0, B1, B2**—Active Block Interrupt

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>Reserved</td>
</tr>
<tr>
<td>6</td>
<td>Reserved</td>
</tr>
<tr>
<td>5</td>
<td>BANK 1</td>
</tr>
<tr>
<td>4</td>
<td>BANK 0</td>
</tr>
<tr>
<td>3</td>
<td>Reserved</td>
</tr>
<tr>
<td>2</td>
<td>IPN—INTERRUPT PENDING</td>
</tr>
<tr>
<td>1</td>
<td>B10</td>
</tr>
<tr>
<td>0</td>
<td>B11</td>
</tr>
<tr>
<td></td>
<td>B12</td>
</tr>
</tbody>
</table>

**IPN—Interrupt Pending**—This bit is active low. It indicates that there is an interrupt pending. The interrupt logic asserts the INT pin as soon as this bit goes active. (Note: the GIR register is continuously updated; so that, while the user is serving one interrupt source, a new interrupt with higher priority may enter GIR and replace the older interrupt.)
22. LSR—LINE STATUS REGISTER

This register holds the status of the serial link. It shares five of its bits with the RST register (Bkd, FE, PE, OE, and RFIR). When this register is read, the RST register (BITS 1–7) and LSR register (BITS 1–4) are cleared. This register is provided for compatibility with the INS8250A.

**TxSt—Transmit Machine Status Bit**—Same as TxIR bit of GSR register. If high it indicates that the Transmit Machine is in Idle State. (Note: Idle may indicate that the TxM is either empty or disabled.

**TFSt—Transmit FIFO Status**—Same as TFIR bit of GSR. It indicates that the Transmit FIFO level is equal to or below the Transmit FIFO Threshold. There are two ways to disable the transmit FIFO status from being reflected in GIR:

1. Writing a “0” to the TFIE bit of the GER register
2. Dynamically by using the Tx FIFO HOLD INTERRUPT logic. When the Tx FIFO is in the Hold State, no interrupts are generated regardless of the TFIR and TFIE bits.

The Transmit FIFO enters the Hold State when the CPU reads the GIR register and the source of the interrupt is Tx FIFO. To Exit, the CPU must drop the TFIR bit of GSR by writing a character to Tx FIFO, or drop TFIE bit of GER (Disable Tx FIFO).

**Bkd—Break Detected**—See Bkd bit in RST register for full explanation. The Bkd bit in RST register is the same as this bit.

**FE—Framing Error Detected**—See FE bit in RST register for a full explanation. The FE bit in RST register is the same as this bit.

**PE—Parity Error Detected**—See PE bit in RST register for full explanation. The PE bit in RST register is the same as this bit.

**OE—Overrun Error**—See OE bit in RST register for full explanation. The OE bit in RST register is the same as this bit.

**RFIR—Receive FIFO Interrupt Request**—This bit is identical to RFIR bit of GSR. It indicates that the RX FIFO level is above the Rx FIFO Threshold. This bit is forced LOW during any READ from the Rx FIFO. A zero written to this bit will acknowledge an Rx FIFO interrupt.
23. RST—RECEIVE MACHINE STATUS REGISTER

This register displays the status of the Receive Machine. It reports events that have occurred since the RST was cleared. This register is cleared when it is read except for BIT0, Rx FIFO interrupt. Each bit in this register, when set, can cause an interrupt. Five bits of this register are shared with the LSR register.

**CRF—Control/Address Character Received**—When enabled, this bit can cause an interrupt if a control character or address character is received.

- In uLAN Mode: indicates that an address character has been received.
- In normal Mode: indicates that a standard control character (either ASCII or EBCDIC) has been received.

**PCRF—Programmed Control/Address Character Received**—This bit, when enabled, will cause an interrupt when an address or control character match occurs.

- In uLAN Mode: indicates that an address character equal to one of the registers ACR0 or ACR1 has been received.
- In normal Mode: indicates that a character which matches the registers ACR0 or ACR1 has been received.

**BkT—Break Terminated**—This bit indicates that a break condition has been terminated.

**BkD—Break Detected**—This bit indicates that a Break Condition has been detected, i.e., RxD input was held low for one character frame plus a stop bit.

**FE—Framing Error**—This bit indicates that a received character did not have a valid stop bit.

**PE—Parity Error**—Indicates that a received character had a parity error.

**OE—Overrun Error**—Indicates that a received character was lost because the Rx FIFO was full.

**RFIR—Receive FIFO Interrupt Request**—Same as the RFIR bit of LSR register.
24. RXF—RECEIVED CHARACTER FLAGS

This register contains additional information about the character in the RXD register. It is loaded by the Rx Machine simultaneously with the RXD register.

**ROK—Received Character OK**—This bit indicates that the character in RXD had no parity or framing error. The parity error is not included in the s/w parity mode.

**RxN—Received Character Noisy**—The character in RXD was noisy. This bit, valid only in 16X sampling mode, indicates that the received character had non-identical samples for at least one of its bits.

**RPE—Receive Character Parity Error**—This bit indicates that the RxD character had a Parity Error. However, in S/W Parity mode it holds the received parity bit as is.

**ACR—Address/Control Character Marker**—This bit indicates that the character in RXD is either: A control Character—in normal Mode. An Address Character in uLAN Mode.

**RFE—Receive Character Framing Error**—Indicates that no Stop bit was found for the character in RXD.

**NOTE:**
A Framing Error will be generated for the first character of the Break sequence.

**RND—Ninth Bit of Received Character**—The most significant bit of the character in RXD is written into this bit. This bit is zero for characters with less than nine bits.

**BKF—Break Flag**—Indicates that the character is part of a “break” sequence.

25. FLR—FIFO LEVEL REGISTER

This register holds the current Receive and Transmit FIFO occupancy levels.

**RFL2, RFL1, RFL0—Receive FIFO Level of Occupancy**—These three bits indicate the level of Occupancy of the Rx FIFO. The valid range is zero (000) to four (100).

**TFL2, TFL1, TFL0—Transmit FIFO Level of Occupancy**—These three bits indicate the level of occupancy in the transmit FIFO. The valid range is zero (000) to four (100).
26. TMST—TIMER STATUS REGISTER

This register holds the status of the timers. Bits TBEx and TAEx generate interrupts which are reflected in bit TIR of GSR. Bits GBS and GAS just display the counting status, they do not generate interrupts.

GBS—Gate B State—This bit does not generate an interrupt. It indicates the counting state of the software gate of Timer B, as written through the TMCR register.

- 0—counting disabled
- 1—counting enabled

GAS—Gate A State—This bit does not generate an interrupt. It reflects the state of the software gate of Timer A, as written through the TMCR register.

- 0—counting disabled
- 1—counting enabled

27. MSR—MODEM/I/O PINS REGISTER

This register holds the status of the Modem input pins (CTS, DCD, DSR, RI). It is the source of interrupts (MSR 0–3) for the MIR bit of GSR. If any of the above inputs change levels the appropriate bit in MODEM STATUS is set. Reading MODEM STATUS will clear the status bits.

DCDC—DCD Complement—Holds the complement of the DCD input pin if programmed as an input in PMD.

DRIC—Holds the complement of the RI input pin if programmed as an input in PMD.

DSRC—DSR Complement—Holds the complement of the DSR input pin if configured as an input in PMD.

CTSC—CTS Complement—Holds the complement of the CTS pin.

DDCD—Delta DCD—Indicates that the DCD input pin has changed state since this register was last read.

DRI—Delta RI—Indicates that there was a high-to-low transition on the RI input pin since the register was last read.

DDSR—Delta DSR—Indicates that the DSR input pin has changed state since this register was last read.

DCTS—Delta CTS—Indicates that the CTS input pin has changed state since this register was last read.

COMMAND REGISTERS

The command registers are write only; they are used to trigger an operation by the device. Once the operation is started the register is automatically reset. There is a device level register as well as four block command registers. It is recommended that only one command be issued during a write cycle.
28. ICM—INTERNAL COMMAND REGISTER

This register activates the device's general functions.

**SRST—Device Software RESET**—Causes a total device reset; the effect is identical to the hardware reset (except for strapping options). The reset lasts four clocks and puts the device into the Default Wake-up Mode.

**INTA—Interrupt Acknowledge**—This command is an explicit acknowledgement of the 82510's interrupt request. It forces the INT pin inactive for two clocks; afterwards, the INT pin may again go active if other enabled interrupts are pending. This command is provided for the Manual Acknowledge mode of the 82510.

**StC—Status Clear**—Clears the following status registers: RST, MSR, and TMST.

**PDM—Power Down**—This command forces the device into the power-down mode. Refer to the functional description for details.

29. TCM—TRANSMIT COMMAND REGISTER

This register controls the operation of the Transmit Machine.

**FTM—Flush Transmit Machine**—Resets the Transmit Machine logic (but not the registers or FIFO) and enables transmission.

**FTF—Flush Transmit FIFO**—Clears the Tx FIFO.

**TxEN—Transmit Enable**—Enables Transmission by the Transmit Machine.

**TxDi—Transmit Disable**—Disables transmission. If transmission is occurring when this command is issued the Tx Machine will complete transmission of the current character before disabling transmission.
This register controls the operation of the Rx machine.

**RxE**—Receive Enable—Enables the reception of characters.

**RxDI**—Receive Disable—Disables reception of data on RXD pin.

**FRM**—Flush Receive Machine—Resets the Rx Machine logic (but not registers and FIFOs), enables reception, and unlocks the receive FIFO.

**FRF**—Flush Receive FIFO—Clears the Rx FIFO.

**LRF**—Locks Rx FIFO—Disables the write mechanism of the Rx FIFO so that characters subsequently received are not written to the Rx FIFO but are lost. However, reception is not disabled and complete status/event reporting continues. (This command may be used in the uLAN mode to disable loading of characters into the Rx FIFO until an address match is detected.)

**ORF**—Open (Unlock) Rx FIFO—This command enables or unlocks the write mechanism of the Rx FIFO.

This register controls the operation of the two 82510 timers. It has no effect when the timers are configured as baud-rate generators. TGA and TGB are not reset after command execution.

**TGB**—Timer-B Gate—This bit serves as a gate for Timer B operation:

1—enables counting
0—disables counting

**TGA**—Timer-A Gate—This bit serves as a gate for Timer-A operation:

1—enables counting
0—disables counting

**STB**—Start Timer B—This command triggers timer B. At terminal count a status bit is set in TMST (TBEx).

**STA**—Start Timer A—This command triggers timer A. At terminal count a status bit is set in TMST (TAEx).
32. MCR—MODEM CONTROL REGISTER

This register controls the modem output pins. With
multi—function pins it affects only the pins config­
ured as general purpose output pins. All the output
pins invert the data, i.e. their output will be the com­
plement of the data written into this register.

OUT0—OUT0 Output Bit—This bit controls the
OUT0 pin. The output signal is the complement of
this bit.

LCB Loopback Control Bit—This bit puts the
82510 into loopback mode. The particular type of
loopback is selected via the IMD register.

OUT2—OUT2 Output Bit—This bit controls the
OUT2 pin. The output signal is the complement of
this bit.

OUT1—OUT1 Output Bit—This bit controls the
OUT1 pin. The output signal is the complement of
this bit.

RTS—RTS Output Bit—This bit controls the RTS
pin. The output signal is the complement of this bit.

DTR—DTR Output Bit—This bit controls the DTR
pin. The output signal is the complement of this bit.

DATA REGISTERS

The data registers hold data or other information
and may be accessed at any time.

33. TXD—TRANSMIT DATA REGISTER

This register holds the next data byte to be pushed
into the Transmit FIFO. For character formats with
more than eight bits of data, or with additional com­
ponents (S/W Parity, Address Marker Bit) the addi­
tional data bits should be written into the TxF regis­
ter. When a byte is written to this register its con­
ten s, along with the contents of the TxF register,
are pushed to the top of the Transmit FIFO. This
register is write only.
34. TXF—TRANSMIT FLAGS REGISTER

This register holds some additional components of the next character to be pushed into the Tx FIFO. The contents of this register are pushed into the Tx FIFO with the Transmit Data register whenever the TxD register is written to by the CPU.

uLAN—uLAN Address Marker Bit—This bit is transmitted in uLAN mode as the address marker bit.

SP—Software Parity Bit—This bit is transmitted in S/W parity mode as the character’s parity bit.

D8—Ninth Bit of Data—in nine-bit character length mode this bit is transmitted as the MSB (D8) bit.

35. RXD—RECEIVE DATA REGISTER

This register holds the earliest received character in the Rx FIFO. The character is right justified and leading bits are zeroed. This register is loaded by the Rx Machine with the first received character. Reading the register causes the next register from the Rx FIFO to be loaded into RxD and RxF registers.
### SPECIFICATIONS

#### ABSOLUTE MAXIMUM RATINGS

- Ambient Temperature under Bias: 0°C to 70°C
- Storage Temperature: -65°C to +150°C
- Voltage on any Pin (w.r.t. VSS): -0.5V to VCC + 0.5V
- Voltage on VCC Pin (w.r.t. VSS): -0.5V to +7V
- Power Dissipation: 300 mW

### D.C. SPECIFICATIONS

#### D.C. CHARACTERISTICS \((T_A = 0^\circ \text{C to } 70^\circ \text{C}, V_{CC} = 5V \pm 10\% )\)

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Parameter</th>
<th>Notes</th>
<th>Min</th>
<th>Max</th>
<th>Units</th>
</tr>
</thead>
<tbody>
<tr>
<td>(V_{IL})</td>
<td>Input Low Voltage</td>
<td>(1)</td>
<td>-0.5</td>
<td>0.8</td>
<td>V</td>
</tr>
<tr>
<td>(V_{IH})</td>
<td>Input High Voltage</td>
<td>(1)</td>
<td>2.0</td>
<td>(V_{CC} - 0.5)</td>
<td>V</td>
</tr>
<tr>
<td>(V_{OL})</td>
<td>Output Low Voltage</td>
<td>(2), (9)</td>
<td>0.45</td>
<td>V</td>
<td></td>
</tr>
<tr>
<td>(V_{OH})</td>
<td>Output High Voltage</td>
<td>(3), (9)</td>
<td>2.4</td>
<td>V</td>
<td></td>
</tr>
<tr>
<td>(I_{LI})</td>
<td>Input Leakage Current</td>
<td>(4)</td>
<td>±10</td>
<td>(\mu)A</td>
<td></td>
</tr>
<tr>
<td>(I_{LO})</td>
<td>3-State Leakage Current</td>
<td>(5)</td>
<td>±10</td>
<td>(\mu)A</td>
<td></td>
</tr>
<tr>
<td>(I_{CC})</td>
<td>Power Supply Current</td>
<td>(6)</td>
<td>3.8</td>
<td>mA/MHz</td>
<td></td>
</tr>
<tr>
<td>(I_{PD})</td>
<td>Power Down Supply</td>
<td>(7)</td>
<td>2</td>
<td>mA</td>
<td></td>
</tr>
<tr>
<td>(I_{STBY})</td>
<td>Standby Supply Current</td>
<td>(10)</td>
<td>500</td>
<td>(\mu)A</td>
<td></td>
</tr>
<tr>
<td>(I_{OH})</td>
<td>RTS, DTR Strapping Current</td>
<td>(11)</td>
<td>0.4</td>
<td>mA</td>
<td></td>
</tr>
<tr>
<td>(I_{OL})</td>
<td>RTS, DTR Strapping Current</td>
<td>(12)</td>
<td>11</td>
<td>mA</td>
<td></td>
</tr>
<tr>
<td>(C_{in})</td>
<td>Input Capacitance</td>
<td>(8)</td>
<td>10</td>
<td>pF</td>
<td></td>
</tr>
<tr>
<td>(C_{io})</td>
<td>I/O Capacitance</td>
<td>(8)</td>
<td>10</td>
<td>pF</td>
<td></td>
</tr>
<tr>
<td>(C_{XTAL})</td>
<td>X1, X2 Load</td>
<td></td>
<td>10</td>
<td>pF</td>
<td></td>
</tr>
</tbody>
</table>

### NOTES:

1. Does not apply to CLK/X1 pin, when configured as crystal oscillator input (X1).
2. \(I_{OL} = 2\) mA.
3. \(I_{OH} = -0.4\) mA.
4. \(0 < V_{IN} < V_{CC}\).
5. \(0.45V < V_{OUT} < (V_{CC} - 0.45)\).
6. \(V_{CC} = 5.5V; V_{IL} = 0.5V\) (max); \(V_{IH} = V_{CC} - 0.5V\) (min); 35 mA (max); Typical value = 2.5 mA/MHz (Not Tested); Ext 1X CLK (9 MHz max); \(I_{OL} = I_{OH} = 0\).
7. \(V_{CC} = 5.5V; V_{IL} = GND; V_{IH} = V_{CC}; I_{OL} = I_{OH} = 0\); device at power down mode, clock running.
8. \(F_{req} = 1\) MHz.
9. Does not apply to OUT2/X2 pin, when configured as crystal oscillator output (X2).
10. Same as 7, but input clock not running.
11. Applies only during hardware reset for clock configuration options. Strapping current for logic HIGH.
12. Applies only during hardware reset for clock configuration. Strapping current for logic LOW.
A.C. SPECIFICATIONS

Testing Conditions:

- All AC output parameters are under output load of 20 to 100 pF, unless otherwise specified.
- AC testing inputs are driven at 2.4 for logic ‘1’, and 0.45V for logic ‘0’. Output timing measurements are made at 1.5V for both a logical ‘0’ and ‘1’.
- In the following tables, the units are ns, unless otherwise specified.

System Interface Specification—System Clock Specification:

The 82510 system clock is supplied via the CLK pin or generated by an on-chip crystal oscillator. The clock is optionally divided by two. The CLK parameters are given separately for internal divide-by-two option ACTIVE and INACTIVE.

The system clock (after division by two, if active) must be at least 16X the Tx or Rx baud rate (the faster of the two).

### SYSTEM CLOCK SPECIFICATIONS

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Parameter</th>
<th>Min</th>
<th>Max</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>DIVIDE BY TWO OPTION—ACTIVE</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Tcy/2</td>
<td>CLK Period</td>
<td>54</td>
<td>250</td>
<td>(2)</td>
</tr>
<tr>
<td>TCLCH</td>
<td>CLK Low Time</td>
<td>25</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TCHCL</td>
<td>CLK High Time</td>
<td>25</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TCH1CH2</td>
<td>CLK Rise Time</td>
<td>10</td>
<td>(1)</td>
<td></td>
</tr>
<tr>
<td>TCL2CL1</td>
<td>CLK Fall Time</td>
<td>10</td>
<td>(1)</td>
<td></td>
</tr>
<tr>
<td>FXTAL</td>
<td>External Crystal Frequency Rating</td>
<td>4.0</td>
<td>18.432 MHz</td>
<td></td>
</tr>
</tbody>
</table>

| DIVIDE BY TWO OPTION—INACTIVE |
| Tcy      | CLK Period                                      | 108 |     |       |
| TCLCH    | CLK Low Time                                    | 54  |     |       |
| TCHCL    | CLK High Time                                   | 44  | 250 |       |
| TCH1CH2  | CLK Rise Time                                   | 15  | (1) |       |
| TCL2CL1  | CLK Fall Time                                   | 15  | (1) |       |

**NOTES:**
1. Rise/fall times are measured between 0.8 and 2.0V.
2. Tcy in ACTIVE divide by two option is TWICE the input clock period.

### RESET SPECIFICATION

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Parameter</th>
<th>Min</th>
<th>Max</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRSHL</td>
<td>Reset Width—CLK/X1 Configured to CLK</td>
<td>8</td>
<td>Tcy</td>
<td>(1)</td>
</tr>
<tr>
<td>TTLRSL</td>
<td>RTS/DTR LOW Setup to Reset Inactive</td>
<td>6</td>
<td>Tcy</td>
<td>(2)</td>
</tr>
<tr>
<td>TRSLTX</td>
<td>RTS/DTR Low Hold after Reset Inactive</td>
<td>0</td>
<td>Tcy - 20</td>
<td>(2)</td>
</tr>
</tbody>
</table>

**NOTES:**
1. In case of CLK/X1 configured as X1, 1 Ms is required to guarantee crystal oscillator wake-up.
2. RTS/DTR are internally driven HIGH during RESET active time. The pin should be either left OPEN or externally driven LOW during RESET according to the required configuration of the system clock. These parameters specify the timing requirements on these pins, in case they are externally driven LOW during RESET.

The maximum spec on TRSLTX requires that the RTS/DTR pins not be forced later than TRSLTX maximum.
READ CYCLE SPECIFICATIONS

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Parameter</th>
<th>Min</th>
<th>Max</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRLRH</td>
<td>RD Active Width</td>
<td>2 Tcy + 65</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TAVRL</td>
<td>Address/CS Setup Time to RD Active</td>
<td></td>
<td>7</td>
<td></td>
</tr>
<tr>
<td>TRHAX</td>
<td>Address/CS Hold Time after RD Inactive</td>
<td></td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>TRLDV</td>
<td>Data Out Valid Delay after RD Active</td>
<td>2Tcy + 65</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TCIAD</td>
<td>Command Inactive to Active Delay</td>
<td>Tcy + 15</td>
<td></td>
<td>(1)</td>
</tr>
<tr>
<td>TRHDZ</td>
<td>Data Out Float Delay after RD Inactive</td>
<td></td>
<td>40</td>
<td></td>
</tr>
</tbody>
</table>

NOTE:
1. Command refers to either Read or Write signals.

WRITE CYCLE SPECIFICATION

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Parameter</th>
<th>Min</th>
<th>Max</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>TWLWH</td>
<td>WR Active Width</td>
<td>2Tcy + 15</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TAVWL</td>
<td>Address CS Setup Time to WR Active</td>
<td></td>
<td>7</td>
<td></td>
</tr>
<tr>
<td>TWHAX</td>
<td>Address and CS Hold Time after WR</td>
<td></td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>TDVWH</td>
<td>Data in Setup Time to WR Inactive</td>
<td>90</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TWHDX</td>
<td>Data in Hold Time after WR Inactive</td>
<td>12</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

NOTE:
Many of the serial interface pins have more than one function; sometimes the different functions have different timings. In such a case, the timing of each function of a pin is given separately.
SCLK PIN SPECIFICATION—16x CLOCKING MODE

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Parameter</th>
<th>Min</th>
<th>Max</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Txcy</td>
<td>SCLK Period</td>
<td>216</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TXLXH</td>
<td>SCLK Low Time</td>
<td>93</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TXHXL</td>
<td>SCLK High Time</td>
<td>93</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TXH1XH2</td>
<td>SCLK Rise Time</td>
<td>15</td>
<td>(1)</td>
<td></td>
</tr>
<tr>
<td>TXL2XL1</td>
<td>SCLK Fall Time</td>
<td>15</td>
<td>(1)</td>
<td></td>
</tr>
</tbody>
</table>

NOTE:
1. Rise/fall times are measured between 0.8V and 2.0V.

SCLK PIN SPECIFICATION—1x CLOCK MODE

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Parameter</th>
<th>Min</th>
<th>Max</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Txcy</td>
<td>SCLK Period</td>
<td>3500</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TXLXH</td>
<td>SCLK Low Time</td>
<td>1650</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TXHXL</td>
<td>SCLK High Time</td>
<td>1650</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TXH1XH2</td>
<td>SCLK Rise Time</td>
<td>15</td>
<td>(1)</td>
<td></td>
</tr>
<tr>
<td>TXL2XL1</td>
<td>SCLK Fall Time</td>
<td>15</td>
<td>(1)</td>
<td></td>
</tr>
</tbody>
</table>

NOTE:
1. Rise/fall times are measured between 0.8V and 2.0V.

RXD SPECIFICATION (1x MODE)

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Parameter</th>
<th>Min</th>
<th>Max</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRPW</td>
<td>RXD Setup Time to SCLK High</td>
<td>250</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TRPD</td>
<td>RXD Hold Time After SCLK High</td>
<td>250</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

TXD SPECIFICATION (1x MODE)

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Parameter</th>
<th>Min</th>
<th>Max</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>TSCLKTXD</td>
<td>TXD Valid Delay after SCLK Low</td>
<td>—</td>
<td>170</td>
<td></td>
</tr>
</tbody>
</table>

REMOTE LOOPBACK SPECIFICATION

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Parameter</th>
<th>Min</th>
<th>Max</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRXDTXD</td>
<td>TXD Delay after RXD</td>
<td>—</td>
<td>170</td>
<td></td>
</tr>
</tbody>
</table>
Receive Logic Diagram

Transmit Logic Diagram
The Intel 8273 Programmable HDLC/SDLC Protocol Controller is a dedicated device designed to support the ISO/CCITT's HDLC and IBM's SDLC communication line protocols. It is fully compatible with Intel's new high performance microcomputer systems such as the MCS 188/186™. A frame level command set is achieved by a unique microprogrammed dual processor chip architecture. The processing capability supported by the 8273 relieves the system CPU of the low level real-time tasks normally associated with controllers.
A BRIEF DESCRIPTION OF HDLC/SDLC PROTOCOLS

General

The High Level Data Link Control (HDLC) is a standard communication link protocol established by International Standards Organization (ISO). HDLC is the discipline used to implement ISO X.25 packet switching systems.

The Synchronous Data Link Control (SDLC) is an IBM communication link protocol used to implement the System Network Architecture (SNA). Both the protocols are bit oriented, code independent, and ideal for full duplex communication. Some common applications include terminal to terminal, terminal to CPU, CPU to CPU, satellite communication, packet switching and other high speed data links. In systems which require expensive cabling and interconnect hardware, any of the two protocols could be used to simplify interfacing (by going serial), thereby reducing interconnect hardware costs. Since both the protocols are speed independent, reducing interconnect hardware could become an important application.

Network

In both the HDLC and SDLC line protocols, according to a pre-assigned hierarchy, a PRIMARY (Control) STATION controls the overall network (data link) and issues commands to the SECONDARY (Slave) STATIONS. The latter comply with instructions and respond by sending appropriate RESPONSES. Whenever a transmitting station must end transmission prematurely it sends an ABORT character. Upon detecting an abort character, a receiving station ignores the transmission block called a FRAME. Time fill between frames can be accomplished by transmitting either continuous frame sequences called FLAGS or an abort character. A time fill within a frame is not permitted. Whenever a station receives a string of more than fifteen consecutive ones, the station goes into an IDLE state.

Frames

A single communication element is called a FRAME which can be used for both Link Control and data transfer purposes. The elements of a frame are the beginning eight bit FLAG (F) consisting of one zero, six ones, and a zero, an eight bit ADDRESS FIELD (A), an eight bit CONTROL FIELD (C), a variable (N-bit) INFORMATION FIELD (I), a sixteen bit FRAME CHECK SEQUENCE (FCS), and an eight bit end FLAG (F), having the same bit pattern as the beginning flag. In HDLC the Address (A) and Control (C) bytes are extendable. The HDLC and the SDLC use three types of frames; an Information Frame is used to transfer data, a Supervisory Frame is used for control purposes, and a Non-sequenced Frame is used for initialization and control of the secondary stations.

Frame Characteristics

An important characteristic of a frame is that its contents are made code transparent by use of a zero bit insertion and deletion technique. Thus, the user can adopt any format or code suitable for his system—it may even be a computer word length or a "memory dump". The frame is bit oriented that is, bits, not characters in each field, have specific meanings. The Frame Check Sequence (FCS) is an error detection scheme similar to the Cyclic Redundancy Checkword (CRC) widely used in magnetic disk storage devices. The Command and Response information frames contain sequence numbers in the control fields identifying the sent and received frames. The sequence numbers are used in Error Recovery Procedures (ERP) and as implicit acknowledgement of frame communication, enhancing the true full-duplex nature of the HDLC/SDLC protocols.

In contrast, BISYNC is basically half-duplex (two way alternate) because of necessity to transmit immediate acknowledgement frames. HDLC/SDLC therefore saves propagation delay times and have a potential of twice the throughput rate of BISYNC.

It is possible to use HDLC or SDLC over half duplex lines but there is a corresponding loss in throughput because both are primarily designed for full-duplex communication. As in any synchronous system, the bit rate is determined by the clock bits supplied by the modem, protocols themselves are speed independent.

A byproduct of the use of zero-bit insertion-deletion technique is the non-return-to-zero invert (NRZI) data transmission/reception compatibility. The latter allows HDLC/SDLC protocols to be used with asynchronous data communication hardware in which the clocks are derived from the NRZI encoded data.
References

*IBM Synchronous Data Link Control General Information*, IBM, GA27-3093-1.

*Standard Network Access Protocol Specification*, DATAPAC, Trans-Canada Telephone System CCG111


*IBM 3650 Retail Store System Loop Interface OEM Information*, IBM, GA 27-3098-0

---

**Figure 3. Frame Format**

**Table 1. Pin Description**

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Pin No.</th>
<th>Type</th>
<th>Name and Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>VCC</td>
<td>40</td>
<td></td>
<td>POWER SUPPLY: +5V Supply.</td>
</tr>
<tr>
<td>GND</td>
<td>20</td>
<td></td>
<td>GROUND: Ground.</td>
</tr>
<tr>
<td>RESET</td>
<td>4</td>
<td>I</td>
<td>RESET: A high signal on this pin will force the 8273 to an idle state. The 8273 will remain idle until a command is issued by the CPU. The modem interface output signals are forced high. Reset must be true for a minimum of 10 TCY.</td>
</tr>
<tr>
<td>CS</td>
<td>24</td>
<td>I</td>
<td>CHIP SELECT: The RD and WR inputs are enabled by the chip select input.</td>
</tr>
<tr>
<td>DB0–DB7</td>
<td>12–19</td>
<td>I/O</td>
<td>DATA BUS: The Data Bus lines are bidirectional three-state lines which interface with the system Data Bus.</td>
</tr>
<tr>
<td>WR</td>
<td>10</td>
<td>I</td>
<td>WRITE INPUT: The Write signal is used to control the transfer of either a command or data from CPU to the 8273.</td>
</tr>
<tr>
<td>RD</td>
<td>9</td>
<td>I</td>
<td>READ INPUT: The Read signal is used to control the transfer of either a data byte or a status word from the 8273 to the CPU.</td>
</tr>
<tr>
<td>TxFNT</td>
<td>2</td>
<td>O</td>
<td>TRANSMITTER INTERRUPT: The Transmitter interrupt signal indicates that the transmitter logic requires service.</td>
</tr>
<tr>
<td>RxFNT</td>
<td>11</td>
<td>O</td>
<td>RECEIVER INTERRUPT: The Receiver interrupt signal indicates that the Receiver logic requires service.</td>
</tr>
<tr>
<td>TxDREQ</td>
<td>6</td>
<td>O</td>
<td>TRANSMITTER DATA REQUEST: Requests a transfer of data between memory and the 8273 for a transmit operation.</td>
</tr>
<tr>
<td>RxDREQ</td>
<td>8</td>
<td>O</td>
<td>RECEIVER DMA REQUEST: Requests a transfer of data between the 8273 and memory for a receive operation.</td>
</tr>
<tr>
<td>Symbol</td>
<td>Pin No.</td>
<td>Type</td>
<td>Name and Function</td>
</tr>
<tr>
<td>--------</td>
<td>---------</td>
<td>------</td>
<td>-------------------</td>
</tr>
<tr>
<td>TXDACK</td>
<td>5</td>
<td>1</td>
<td>TRANSMITTER DMA ACKNOWLEDGE: The Transmitter DMA acknowledge signal notifies the 8273 that the TxDMA cycle has been granted.</td>
</tr>
<tr>
<td>RxDACK</td>
<td>7</td>
<td>1</td>
<td>RECEIVER DMA ACKNOWLEDGE: The Receiver DMA acknowledge signal notifies the 8273 that the RxDMA cycle has been granted.</td>
</tr>
<tr>
<td>A₀–A₁</td>
<td>21–22</td>
<td>I</td>
<td>ADDRESS: These two lines are CPU Interface Register Select lines.</td>
</tr>
<tr>
<td>TxD</td>
<td>29</td>
<td>O</td>
<td>TRANSMITTER DATA: This line transmits the serial data to the communication channel.</td>
</tr>
<tr>
<td>TxC</td>
<td>28</td>
<td>I</td>
<td>TRANSMITTER CLOCK: The transmitter clock is used to synchronize the transmit data.</td>
</tr>
<tr>
<td>RxD</td>
<td>26</td>
<td>I</td>
<td>RECEIVER DATA: This line receives serial data from the communication channel.</td>
</tr>
<tr>
<td>RxC</td>
<td>27</td>
<td>I</td>
<td>RECEIVER CLOCK: The Receiver Clock is used to synchronize the receive data.</td>
</tr>
<tr>
<td>32X CLK</td>
<td>25</td>
<td>I</td>
<td>32X CLOCK: The 32X clock is used to provide clock recovery when an asynchronous modem is used. In loop configuration the loop station can run without an accurate 1X clock by using the 32X CLK in conjunction with the DPLL output. (This pin must be grounded when not used.)</td>
</tr>
<tr>
<td>DPLL</td>
<td>23</td>
<td>O</td>
<td>DIGITAL PHASE LOCKED LOOP: Digital Phase Locked Loop output can be tied to RxC and/or TxC when 1X clock is not available. DPLL is used with 32X CLK.</td>
</tr>
<tr>
<td>FLAG DET</td>
<td>1</td>
<td>O</td>
<td>FLAG DETECT: Flag Detect signals that a flag (01111110) has been received by an active receiver.</td>
</tr>
<tr>
<td>RTS</td>
<td>35</td>
<td>O</td>
<td>REQUEST TO SEND: Request to Send signals that the 8273 is ready to transmit data.</td>
</tr>
<tr>
<td>CTS</td>
<td>30</td>
<td>I</td>
<td>CLEAR TO SEND: Clear to Send signals that the modem is ready to accept data from the 8273.</td>
</tr>
<tr>
<td>CD</td>
<td>31</td>
<td>I</td>
<td>CARRIER DETECT: Carrier Detect signals that the line transmission has started and the 8273 may begin to sample data on RxD line.</td>
</tr>
<tr>
<td>PA₂⁻₄</td>
<td>32–34</td>
<td>I</td>
<td>GENERAL PURPOSE INPUT PORTS: The logic levels on these lines can be Read by the CPU through the Data Bus Buffer.</td>
</tr>
<tr>
<td>PB₁⁻₄</td>
<td>36–39</td>
<td>O</td>
<td>GENERAL PURPOSE OUTPUT PORTS: The CPU can write these output lines through Data Bus Buffer.</td>
</tr>
<tr>
<td>CLK</td>
<td>3</td>
<td>I</td>
<td>CLOCK: A square wave TTL clock.</td>
</tr>
</tbody>
</table>
FUNCTIONAL DESCRIPTION

General

The Intel 8273 HDLC/SDLC controller is a microcomputer peripheral device which supports the International Standards Organization (ISO) High Level Data Link Control (HDLC), and IBM Synchronous Data Link Control (SDLC) communications protocols. This controller minimizes CPU software by supporting a comprehensive frame-level instruction set and by hardware implementation of the low level tasks associated with frame assembly/disassembly and data integrity. The 8273 can be used in either synchronous or asynchronous applications.

In asynchronous applications the data can be programmed to be encoded/decoded in NRZI code. The clock is derived from the NRZI data using a digital phase locked loop. The data transparency is achieved by using a zero-bit insertion/deletion technique. The frames are automatically checked for errors during reception by verifying the Frame Check Sequence (FCS); the FCS is automatically generated and appended before the final flag in transmit. The 8273 recognizes and can generate flags (01111110) Abort, Idle, and GA (EOP) characters.

The 8273 can assume either a primary (control) or a secondary (slave) role. It can therefore be readily implemented in an SDLC loop configuration as typified by the IBM 3650 Retail Store System by programming the 8273 into a one-bit delay mode. In such a configuration, a two wire pair can be effectively used for data transfer between controllers and loop stations. The digital phase locked loop output pin can be used by the loop station without the presence of an accurate Tx clock.

CPU Interface

The CPU interface is optimized for the MCS-80/85™ bus with an 8257 DMA controller. However, the interface is flexible, and allows either DMA or non-DMA data transfers, interrupt or non-interrupt driven. It further allows maximum line utilization by providing early interrupt mechanism for buffered (only the information field can be transferred to memory) Tx command overlapping. It also provides separate Rx and Tx interrupt output channels for efficient operation. The 8273 keeps the interrupt request active until all the associated interrupt results have been read.

The CPU utilizes the CPU interface to specify commands and transfer data. It consists of seven registers addressed via CIA, A1, A0, RD and WR signals and two independent data registers for receive data and transmit data. A1, A0 are generally derived from two low order bits of the address bus. If an 8080 based CPU is utilized, the RD and WR signals may be driven by the 8228 I/OR and I/OW. The table shows the seven register select decoding:

<table>
<thead>
<tr>
<th>A1 A0</th>
<th>TxDACK</th>
<th>RxDACK</th>
<th>CS</th>
<th>RD</th>
<th>WR</th>
<th>Register</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>Command</td>
</tr>
<tr>
<td>0 0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>Status</td>
</tr>
<tr>
<td>0 1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>Parameter</td>
</tr>
<tr>
<td>0 1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>Result</td>
</tr>
<tr>
<td>1 0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>Reset</td>
</tr>
<tr>
<td>1 0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>TxINT Result</td>
</tr>
<tr>
<td>1 1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>—</td>
</tr>
<tr>
<td>1 1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>RxINT Result</td>
</tr>
<tr>
<td>X X</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>Transmit Data</td>
</tr>
<tr>
<td>X X</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>Receive Data</td>
</tr>
</tbody>
</table>
Figure 4. 8273 Block Diagram Showing CPU Interface Functions
Register Description

COMMAND
Operations are initiated by writing an appropriate command in the Command Register.

PARAMETER
Parameters of commands that require additional information are written to this register.

RESULT
Contains an immediate result describing an outcome of an executed command.

TRANSMIT INTERRUPT RESULT
Contains the outcome of 8273 transmit operation (good/bad completion).

RECEIVE INTERRUPT RESULT
Contains the outcome of 8273 receive operation (good/bad completion), followed by additional results which detail the reason for interrupt.

STATUS
The status register reflects the state of the 8273 CPU Interface.

DMA Data Transfers
The 8273 CPU interface supports two independent data interfaces: receive data and transmit data. At high data transmission speeds the data transfer rate of the 8273 is great enough to justify the use of direct memory access (DMA) for the data transfers. When the 8273 is configured in DMA mode, the elements of the DMA interfaces are:

TxDRQ: TRANSMIT DMA REQUEST
Requests a transfer of data between memory and the 8273 for a transmit operation.

TxDACK: TRANSMIT DMA ACKNOWLEDGE
The TxDACK signal notifies the 8273 that a transmit DMA cycle has been granted. It is also used with WR to transfer data to the 8273 in non-DMA mode. Note: RD must not be asserted while TxDACK is active.

RxDRQ: RECEIVE DMA REQUEST
Requests a transfer of data between the 8273 and memory for a receive operation.

RxDACK: RECEIVE DMA ACKNOWLEDGE
The RxDACK signal notifies the 8273 that a receive DMA cycle has been granted. It is also used with RD to read data from the 8273 in non-DMA mode. Note: WR must not be asserted while RxDACK is active.

RD, WR: READ, WRITE
The RD and WR signals are used to specify the direction of the data transfer.

DMA transfers require the use of a DMA controller such as the Intel 8257. The function of the DMA controller is to provide sequential addresses and timing for the transfer, at a starting address determined by the CPU. Counting of data blocks lengths is performed by the 8273.

To request a DMA transfer the 8273 raises the appropriate DMA REQUEST. DMA ACKNOWLEDGE and READ enables DMA data onto the bus (independently of CHIP SELECT). DMA ACKNOWLEDGE and WRITE transfers DMA data to the 8273 (independent of CHIP SELECT).

It is also possible to configure the 8273 in the non-DMA data transfer mode. In this mode the CPU module must pass data to the 8273 in response to non-DMA data requests indicated by status word.

Modem Interface
The 8273 Modem interface provides both dedicated and user defined modem control functions. All the control signals are active low so that EIA RS-232C inverting drivers (MC 1488) and inverting receivers (MC 1489) may be used to interface to standard modems. For asynchronous operation, this interface supports programmable NRZI data encode/decode, a digital phase locked loop for efficient clock extraction from NRZI data, and modem control ports with automatic CTS, CD monitoring and RTS generation.

This interface also allows the 8273 to operate in PRE-FRAME SYNC mode in which the 8273 prefixes 16 transitions to a frame to synchronize idle lines before transmission of the first flag.

It should be noted that all the 8273 port operations deal with logical values, for instance, bit DO of Port A will be a one when CTS (Pin 30) is a physical zero (logical one).
PORT A — INPUT PORT

During operation, the 8273 interrogates input pins CTS (Clear to Send) and CD (Carrier Detect). CTS is used to condition the start of a transmission. If during transmission CTS is lost the 8273 generates an interrupt. During reception, if CD is lost, the 8273 generates an interrupt.

The user defined input bits correspond to the 8273 PA4, PA3 and PA2 pins. The 8273 does not interrogate or manipulate these bits.

PORT B - OUTPUT PORT

During normal operation, if the CPU sets RTS active, the 8273 will not change this pin; however, if the CPU sets RTS inactive, the 8273 will activate it before each transmission and deactivate it one byte time after transmission. While the receiver is active the flag detect pin is pulsed each time a flag sequence is detected in the receive data stream. Following an 8273 reset, all pins of Port B are set to a high, inactive level.

The user defined output bits correspond to the state of PB4—PB1 pins. The 8273 does not interrogate or manipulate these bits.

Serial Data Logic

The Serial data is synchronized by the user transmit (TxC) and receive (RxC) clocks. The leading edge of TxC generates new transmit data and the trailing edge of RxC is used to capture receive data. The NRZI encoding/decoding of the receive and transmit data is programmable.

The diagnostic features included in the Serial Data logic are programmable loop back of data and selectable clock for the receiver. In the loop-back mode, the data presented to the TxD pin is internally routed to the receive data input circuitry in place of the RxD pin, thus allowing a CPU to send a message to itself to verify operation of the 8273.

In the selectable clock diagnostic feature, when the data is looped back, the receiver may be presented incorrect sample timing by the external circuitry. The user may select to substitute the TxC pin for the RxC input on-chip so that the clock used to generate the loop back data is used to sample it. Since TxD is generated off the leading edge of TxC and RxD is sampled on the trailing edge, the selected clock allows bit synchronism.

ASYNCHRONOUS MODE INTERFACE

Although the 8273 is fully compatible with the HDLC/SDLC communication line protocols, which are primarily designed for synchronous communication, the 8273 can also be used in asynchronous applications by using this interface. The interface employs a digital phase locked loop (DPLL) for clock recovery from a receive data stream and programmable NRZI encoding and decoding of data. The use of NRZI coding with SDLC transmission guarantees that within a frame, data transitions will occur at least every five bit times—the longest sequence of ones which may be transmitted without zero-bit insertion. The DPLL should be used only when NRZI coding is used since the NRZI coding will transmit zero sequence as line transitions. The digital phase locked loop also facilitates full-duplex and half-duplex asynchronous implementation with, or without modems.
Figure 5. 8273 Block Diagram Showing Control Logic Functions

Figure 6. Transmit/Receive Timing
DIGITAL PHASE LOCKED LOOP

In asynchronous applications, the clock is derived from the receiver data stream by the use of the digital phase locked loop (DPLL). The DPLL requires a clock input at 32 times the required baud rate. The receive data (RxD) is sampled with this 32X CLK and the 8273 DPLL supplies a sample pulse nominally centered on the RxD bit cells. The DPLL has a built-in "stiffness" which reduces sensitivity to line noise and bit distortion. This is accomplished by making phase error adjustments in discrete increments. Since the nominal pulse is made to occur at 32 counts of the 32X CLK, these counts are subtracted or added to the nominal, depending upon which quadrant of the four error quadrants the data edge occurs in. For example if an RxD edge is detected in quadrant A1, it is apparent that the DPLL sample "A" was placed too close to the trailing edge of the data cell; sample "B" will then be placed at

\[ T = (T_{\text{nominal}} - 2 \text{ counts}) = 30 \text{ counts of the } 32X \text{ CLK} \]

to move the sample pulse "B" toward the nominal center of the next bit cell. A data edge occurring in quadrant B1 would cause a smaller adjustment of phase with \( T = 31 \text{ counts of the } 32X \text{ CLK} \). Using this technique the DPLL pulse will converge to nominal bit center within 12 data bit times, worst case, with constant incoming RxD edges.

A method of attaining bit synchronism following a line idle is to use PRE-FRAME SYNC mode of transmission.

Figure 7. DPLL Sample Timing
SDLC LOOP

The DPLL simplifies the SDLC loop station implementation. In this application, each secondary station on a loop data link is a repeater set in one-bit delay mode. The signals sent out on the loop by the loop controller (primary station) are relayed from station to station then, back to the controller. Any secondary station finding its address in the A field captures the frame for action at that station. All received frames are relayed to the next station on the loop.

Loop stations are required to derive bit timing from the incoming NRZI data stream. The DPLL generates sample Rx clock timing for reception and uses the same clock to implement Tx clock timing.

Figure 8. SDLC Loop Application
PRINCIPLES OF OPERATION

The 8273 is an intelligent peripheral controller which relieves the CPU of many of the rote tasks associated with constructing and receiving frames. It is fully compatible with the MCS-80/85™ system bus. As a peripheral device, it accepts commands from a CPU, executes these commands and provides an Interrupt and Result back to the CPU at the end of the execution. The communication with the CPU is done by activation of CS, RD, WR, pins while the A1, A0 select the appropriate registers on the chip as described in the Hardware Description Section.

The 8273 operation is composed of the following sequence of events:

The Command Place

During the command phase, the software writes a command to the command register. The command bytes provide a general description of the type of operation requested. Many commands require more detailed information about the command. In such a case up to four parameters are written into the parameter register. The flowchart of the command phase indicates that a command may not be issued if the Status Register indicates that the device is busy. Similarly if a parameter is issued when the Parameter Buffer shows full, incorrect operation will occur.

The 8273 is a duplex device and both transmitter and receiver may each be executing a command or passing results at any given time. For this reason separate interrupt pins are provided. However, the command register must be used for one command sequence at a time.

STATUS REGISTER

The status register contains the status of the 8273 activity. The description is as follows.

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>D7</td>
<td>CBSY (Command Busy)</td>
</tr>
<tr>
<td>D6</td>
<td>CBF (Command Buffer Full)</td>
</tr>
<tr>
<td>D5</td>
<td>CPBF (Command Parameter Buffer Full)</td>
</tr>
<tr>
<td>D4</td>
<td>CRBF (Receive Buffer Full)</td>
</tr>
<tr>
<td>D3</td>
<td>RxNT (Receive Data Available)</td>
</tr>
<tr>
<td>D2</td>
<td>TxNT (Transmit Data Available)</td>
</tr>
<tr>
<td>D1</td>
<td>RxIRA (Receive Interrupt Request)</td>
</tr>
<tr>
<td>D0</td>
<td>TxIRA (Transmit Interrupt Request)</td>
</tr>
</tbody>
</table>

Bit 7 CBSY (Command Busy)

Indicates in-progress command, set for CPU poll when Command Register is full, reset upon command phase completion. It is improper to write a command when CBSY is set; it results in incorrect operation.

![Figure 9. Command Phase Flowchart](image)

Bit 6 CBF (Command Buffer Full)

Indicates that the command register is full, it is reset when the 8273 accepts the command byte but does not imply that execution has begun.

Bit 5 CPBF (Command Parameter Buffer Full)

CPBF is set when the parameter buffer is full, and is reset by the 8273 when it accepts the parameter. The CPU may poll CPBF to determine when additional parameters may be written.
Bit 4 CRBF (Command Result Buffer Full)
Indicates that an executed command immediate result is present in the Result Register. It is set by 8273 and reset when CPU reads the result.

Bit 3 RxINT (Receiver Interrupt)
RxINT indicates that the receiver requires CPU attention. It is identical to RxINT (pin 11) and is set by the 8273 either upon good/bad completion of a specified command or by Non-DMA data transfer. It is reset only after the CPU has read the result byte or has received a data byte from the 8273 in a Non-DMA data transfer.

Bit 2 TxlNT (Transmitter Interrupt)
The TxlNT indicates that the transmitter requires CPU attention. It is identical to TxlNT (pin 2). It is set by 8273 either upon good/bad completion of a specified command or by Non-DMA data transfer. It is reset only after the CPU has read the result byte or has transferred transmit data byte to the 8273 in a Non-DMA transfer.

Bit 1 RxIRA (Receiver Interrupt Result Available)
The RxIRA is set by the 8273 when an interrupt result byte is placed in the RxINT register. It is reset after the CPU has read the RxINT register.

THE EXECUTION PHASE
Upon accepting the last parameter, the 8273 enters into the Execution Phase. The execution phase may consist of a DMA or other activity, and may or may not require CPU intervention. The CPU intervention is eliminated in this phase if the system utilizes DMA for the data transfers, otherwise, for non-DMA data transfers, the CPU is interrupted by the 8273 via TxlNT and RxINT pins, for each data byte request.

THE RESULT PHASE
During the result phase, the 8273 notifies the CPU of the execution outcome of a command. This phase is initiated by:
1. The successful completion of an operation
2. An error detected during an operation.

To facilitate quick network software decisions, two types of execution results are provided:
1. An Immediate Result
2. A Non-Immediate Result

---

Figure 10. Rx Interrupt Result Byte Format

- D7: 11 1 All 8 bits received
- D6: 00 0 D0 received
- D5: 10 0 D1–D0 received
- D4: 01 0 D2–D1 received
- D3: 11 0 D3–D2 received
- D2: 00 1 D4–D3 received
- D1: 10 1 D5–D4 received
- D0: 01 1 D6–D5 received

**Partial Byte Received**

<table>
<thead>
<tr>
<th>Receiver Interrupt Result Code</th>
<th>Result Status After INT</th>
</tr>
</thead>
<tbody>
<tr>
<td>A1 match or general receive</td>
<td>Active</td>
</tr>
<tr>
<td>A2 match</td>
<td>Active</td>
</tr>
<tr>
<td>CRC error</td>
<td>Active</td>
</tr>
<tr>
<td>Abort detected</td>
<td>Active</td>
</tr>
<tr>
<td>Idle detect</td>
<td>Disabled</td>
</tr>
<tr>
<td>EOP detected</td>
<td>Disabled</td>
</tr>
<tr>
<td>Frame less than 32 bits</td>
<td>Active</td>
</tr>
<tr>
<td>DMA overrun detected</td>
<td>Disabled</td>
</tr>
<tr>
<td>Memory buffer overflow</td>
<td>Disabled</td>
</tr>
<tr>
<td>Carrier detect failure</td>
<td>Disabled</td>
</tr>
<tr>
<td>Receive interrupt overrun</td>
<td>Disabled</td>
</tr>
</tbody>
</table>

2-93
Immediate result is provided by the 8273 for commands such as Read Port A and Read Port B which have information (CTS, CD, RTS, etc.) that the network software needs to make quick operational decisions.

A command which cannot provide an immediate result will generate an interrupt to signal the beginning of the Result phase. The immediate results are provided in the Result Register; all non-immediate results are available upon device interrupt, through Tx Interrupt Result Register Txl/R or Rx Interrupt Result Register Rxl/R. The result may consist of a one-byte interrupt code indicating the condition for the interrupt and, if required, one or more bytes which detail the condition.

**Tx and Rx Interrupt Result Registers**

The Result Registers have a result code, the three high order bits D7–D5 of which are set to zero for all but the receive command. This command result contains a count that indicates the number of bits received in the last byte. If a partial byte is received, the high order bits of the last data byte are indeterminate.

All results indicated in the command summary must be read during the result phase.
Figure 12. Result Phase Flowchart—Interrupt Results
DETAILLED COMMAND DESCRIPTION

General

The 8273 HDLC/SDLC controller supports a comprehensive set of high level commands which allows the 8273 to be readily used in full-duplex, half-duplex, synchronous, asynchronous and SDLC loop configuration, with or without modems. These frame-level commands minimize CPU and software overhead. The 8273 has address and control byte buffers which allow the receive and transmit commands to be used in buffered or non-buffered modes.

In buffered transmit mode, the 8273 transmits a flag automatically, reads the Address and Control buffer registers and transmits the fields, then via DMA, it fetches the information field. The 8273 has address and control byte buffers which allow the receive and transmit commands to be used in buffered or non-buffered modes.

In non-buffered transmit mode, the 8273 transmits the beginning flag automatically, then fetches and transmits the Address, Control and Information fields from the memory, appends the FCS character and an end flag. In the non-buffered receive mode the entire contents of a frame are sent to memory with the exception of the flags and FCS.

HDLC Implementation

HDLC Address and Control field are extendable. The extension is selected by setting the low order bit of the field to be extended to a one, a zero in the low order bit indicates the last byte of the respective field.

Since Address/Control field extension is normally done with software to maximize extension flexibility, the 8273 does not create or operate upon contents of the extended HDLC Address/Control fields. Extended fields are transparently passed by the 8273 to user as either interrupt results or data transfer requests. Software must assemble the fields for transmission and interrogate them upon reception.

However, the user can take advantage of the powerful 8273 commands to minimize CPU/Software overhead and simplify buffer management in handling extended fields. For instance buffered mode can be used to separate the first two bytes, then interrogate the others from buffer. Buffered mode is perfect for a two byte address field.

The 8273 when programmed, recognizes protocol characters unique to HDLC such as Abort, which is a string of seven or more ones (01111111). Since Abort character is the same as the GA (EOP) character used in SDLC Loop applications., Loop Transmit and Receive commands are not recommended to be used in HDLC. HDLC does not support Loop mode.

Initialization Set/Reset Commands

These commands are used to manipulate data within the 8273 registers. The Set commands have a single parameter which is a mask that corresponds to the bits to be set. (They perform a logical-OR of the specified register with the mask provided as a parameter). The Register commands have a single parameter which is a mask that has a zero in the bit positions that are to be reset. (They perform a logical-AND of the specified register with the mask).

SET ONE-BIT DELAY (CMD CODE A4)

<table>
<thead>
<tr>
<th>A_1</th>
<th>A_0</th>
<th>D_7</th>
<th>D_6</th>
<th>D_5</th>
<th>D_4</th>
<th>D_3</th>
<th>D_2</th>
<th>D_1</th>
<th>D_0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

PAR: 0 1 0 0 0 0 0 0 0 0
When one bit delay is set, 8273 retransmits the received data stream one bit delayed. This mode is entered at a receiver character boundary, and should only be used by Loop Stations.

**RESET ONE-BIT DELAY (CMD CODE 64)**

<table>
<thead>
<tr>
<th>A1</th>
<th>A0</th>
<th>D7</th>
<th>D6</th>
<th>D5</th>
<th>D4</th>
<th>D3</th>
<th>D2</th>
<th>D1</th>
<th>D0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

The 8273 stops the one bit delayed retransmission mode.

**SET DATA TRANSFER MODE (CMD CODE 97)**

<table>
<thead>
<tr>
<th>A1</th>
<th>A0</th>
<th>D7</th>
<th>D6</th>
<th>D5</th>
<th>D4</th>
<th>D3</th>
<th>D2</th>
<th>D1</th>
<th>D0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

When the data transfer mode is set, the 8273 will interrupt when data bytes are required for transmission or are available from a receive. If a transmit interrupt occurs and the status indicates that there is no Transmit Result (TxIRA = 0), the interrupt is a transmit data request. If a receive interrupt occurs and the status indicates that there is no receive result (RxIRA = 0), the interrupt is a receive data request.

**RESET DATA TRANSFER MODE (CMD CODE 57)**

<table>
<thead>
<tr>
<th>A1</th>
<th>A0</th>
<th>D7</th>
<th>D6</th>
<th>D5</th>
<th>D4</th>
<th>D3</th>
<th>D2</th>
<th>D1</th>
<th>D0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

If the Data Transfer Mode is reset, the 8273 data transfers are performed through the DMA requests without interrupting the CPU.

**SET OPERATING MODE (CMD CODE 91)**

<table>
<thead>
<tr>
<th>A1</th>
<th>A0</th>
<th>D7</th>
<th>D6</th>
<th>D5</th>
<th>D4</th>
<th>D3</th>
<th>D2</th>
<th>D1</th>
<th>D0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

NOTE: In buffered mode, if a supervisory frame (no Information) Transmit command is sent in response to an early Transmit Interrupt, the 8273 will repeatedly transmit the same supervisory frame with one flag in between, until a non-supervisory transmit is issued.

Early transmitter interrupt can be used in buffered mode by waiting for a transmit complete interrupt instead of early Transmit Interrupt before issuing a transmit frame command for a supervisory frame. See Figure 14.
If this bit is zero, the interrupt will be generated only after the final flag has been transmitted.

(D2) BUFFERED MODE

If the buffered mode bit is set to a one, the first two bytes (normally the address (A) and control (C) fields) of a frame are buffered by the 8273. If this bit is a zero the address and control fields are passed to and from memory.

(D1) PREFRAME SYNC MODE

If this bit is set to a one the 8273 will transmit two characters before the first flag of a frame.

To guarantee sixteen line transitions, the 8273 sends two bytes of data (00)\textsubscript{H} if NRZI is set or data (55)\textsubscript{H} if NRZI is not set.

(D0) FLAG STREAM MODE

If this bit is set to a one, the following table outlines the operation of the transmitter.

<table>
<thead>
<tr>
<th>Transmitter State</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>Idle</td>
<td>Send Flags Immediately.</td>
</tr>
<tr>
<td>Transmit or Transmit</td>
<td>Send Flags After the</td>
</tr>
<tr>
<td>Transparent Active</td>
<td>Transmission Complete</td>
</tr>
<tr>
<td>Loop Transmit Active</td>
<td>Ignore Command.</td>
</tr>
<tr>
<td>1 Bit Delay Active</td>
<td>Ignore Command.</td>
</tr>
</tbody>
</table>

If this bit is reset to zero the following table outlines the operation of the transmitter.

<table>
<thead>
<tr>
<th>Transmitter State</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>IDLE</td>
<td>Sends Idles on Next Character boundary.</td>
</tr>
<tr>
<td>Transmit or Transmit</td>
<td>Send Idles after the</td>
</tr>
<tr>
<td>Transparent Active</td>
<td>Transmission is Complete.</td>
</tr>
<tr>
<td>Loop Transmit Active</td>
<td>Ignore Command.</td>
</tr>
<tr>
<td>1 Bit Delay Active</td>
<td>Ignore Command.</td>
</tr>
</tbody>
</table>

SET SERIAL I/O MODE (CMD CODE A0)

<table>
<thead>
<tr>
<th>CMD:</th>
<th>A\textsubscript{1}</th>
<th>A\textsubscript{0}</th>
<th>D\textsubscript{7}</th>
<th>D\textsubscript{6}</th>
<th>D\textsubscript{5}</th>
<th>D\textsubscript{4}</th>
<th>D\textsubscript{3}</th>
<th>D\textsubscript{2}</th>
<th>D\textsubscript{1}</th>
<th>D\textsubscript{0}</th>
</tr>
</thead>
<tbody>
<tr>
<td>PAR:</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

RESET SERIAL I/O MODE (CMD CODE 60)

This command allows bits set in CMD code A0 to be reset by placing zeros in the appropriate positions.

<table>
<thead>
<tr>
<th>CMD:</th>
<th>A\textsubscript{1}</th>
<th>A\textsubscript{0}</th>
<th>D\textsubscript{7}</th>
<th>D\textsubscript{6}</th>
<th>D\textsubscript{5}</th>
<th>D\textsubscript{4}</th>
<th>D\textsubscript{3}</th>
<th>D\textsubscript{2}</th>
<th>D\textsubscript{1}</th>
<th>D\textsubscript{0}</th>
</tr>
</thead>
<tbody>
<tr>
<td>PAR:</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

(D2) LOOP BACK

If this bit is set to a one, the transmit data is internally routed to the receive data circuitry.
(D1) TxC → RxC

If this bit is set to a one, the transmit clock is internally routed to the receive clock circuitry. It is normally used with the loop back bit (D2).

(D0) NRZI MODE

If this bit is set to a one, NRZI encoding and decoding of transmit and receive data is provided. If this bit is a zero, the transmit and receive data is treated as a normal positive logic bit stream.

NRZI encoding specifies that a zero causes a change in the polarity of the transmitted signal and a one causes no polarity change. NRZI is used in all asynchronous operations. Refer to IBM document GA27-3093 for details.

Reset Device Command

<table>
<thead>
<tr>
<th>A1</th>
<th>A0</th>
<th>D7</th>
<th>D6</th>
<th>D5</th>
<th>D4</th>
<th>D3</th>
<th>D2</th>
<th>D1</th>
<th>D0</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>TMR:</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

An 8273 reset command is executed by outputting a (01)H followed by (00)H to the reset register (TMR). See 8273 AC timing characteristics for Reset pulse specifications.

The reset command emulates the action of the reset pin.

1) The modem control signals are forced high (inactive level).
2) The 8273 status register flags are cleared.
3) Any commands in progress are terminated immediately.
4) The 8273 enters an idle state until the next command is issued.
5) The Serial I/O and Operating Mode registers are set to zero and DMA data register transfer mode is selected.
6) The device assumes a non-loop SDLC terminal role.

Receive Commands

The 8273 supports three receive commands: General Receive, Selective Receive, and Selective Loop Receive.

GENERAL RECEIVE (CMD CODE C0)

General receive is a receive mode in which frames are received regardless of the contents of the address field.

<table>
<thead>
<tr>
<th>A1</th>
<th>A0</th>
<th>D7</th>
<th>D6</th>
<th>D5</th>
<th>D4</th>
<th>D3</th>
<th>D2</th>
<th>D1</th>
<th>D0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

PAR: 0 1

1. If buffered mode is specified, the R0, R1 receive frame length (result) is the number of data bytes received.
2. If non-buffered mode is specified, the R0, R1 receive frame length (result) is the number of data bytes received plus two (the count includes the address and control bytes).
3. The frame check sequence (FCS) is not transferred to memory.
4. Frames with less than 32 bits between flags are ignored (no interrupt generated) if the buffered mode is specified.
5. In the non-buffered mode an interrupt is generated when a less than 32 bit frame is received, since data transfer requests have occurred.
6. The 8273 receive is always disabled when an Idle is received after a valid frame. The CPU module must issue a receive command to re-enable the receiver.
7. The intervening ABORT character between a final flag and an IDLE does not generate an interrupt.
8. If an ABORT Character is not preceded by a flag and is followed by an IDLE, an interrupt will be generated for the ABORT followed by an IDLE interrupt one character time later. The reception of an ABORT will disable the receiver.

NOTES:

1. If buffered mode is specified, the R0, R1 receive frame length (result) is the number of data bytes received.
2. If non-buffered mode is specified, the R0, R1 receive frame length (result) is the number of data bytes received plus two (the count includes the address and control bytes).
3. The frame check sequence (FCS) is not transferred to memory.
4. Frames with less than 32 bits between flags are ignored (no interrupt generated) if the buffered mode is specified.
5. In the non-buffered mode an interrupt is generated when a less than 32 bit frame is received, since data transfer requests have occurred.
6. The 8273 receive is always disabled when an Idle is received after a valid frame. The CPU module must issue a receive command to re-enable the receiver.
7. The intervening ABORT character between a final flag and an IDLE does not generate an interrupt.
8. If an ABORT Character is not preceded by a flag and is followed by an IDLE, an interrupt will be generated for the ABORT followed by an IDLE interrupt one character time later. The reception of an ABORT will disable the receiver.
SELECTIVE RECEIVE (CMD CODE C1)

<table>
<thead>
<tr>
<th>A1</th>
<th>A0</th>
<th>D7</th>
<th>D6</th>
<th>D5</th>
<th>D4</th>
<th>D3</th>
<th>D2</th>
<th>D1</th>
<th>D0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

PAR:

| LEAST SIGNIFICANT BYTE OF THE RECEIVE BUFFER LENGTH (B0) |
| 0  | 1  |

| MOST SIGNIFICANT BYTE OF RECEIVE BUFFER LENGTH (B1) |
| 0  | 1  |

| RECEIVE FRAME ADDRESS MATCH FIELD ONE (A1) |
| 0  | 1  |

| RECEIVE FRAME ADDRESS MATCH FIELD TWO (A2) |
| 0  | 1  |

Selective receive is a receive mode in which frames are ignored unless the address field matches any one of two address fields given to the 8273 as parameters.

When selective receive is used in HDLC the 8273 looks at the first character, if extended, software must then decide if the message is for this unit.

SELECTIVE LOOP RECEIVE (CMD CODE C2)

<table>
<thead>
<tr>
<th>A1</th>
<th>A0</th>
<th>D7</th>
<th>D6</th>
<th>D5</th>
<th>D4</th>
<th>D3</th>
<th>D2</th>
<th>D1</th>
<th>D0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

PAR:

| LEAST SIGNIFICANT BYTE OF THE RECEIVE BUFFER LENGTH (B0) |
| 0  | 0  |

| MOST SIGNIFICANT BYTE OF RECEIVE BUFFER LENGTH (B1) |
| 0  | 1  |

| RECEIVE FRAME ADDRESS MATCH FIELD ONE (A1) |
| 0  | 1  |

| RECEIVE FRAME ADDRESS MATCH FIELD TWO (A2) |
| 0  | 1  |

Selective loop receive operates like selective receive except that the transmitter is placed in flag stream mode automatically after detecting an EOP (01111111) following a valid received frame. The one bit delay mode is also reset at the end of a selective loop receive.

RECEIVE DISABLE (CMD CODE 5)

<table>
<thead>
<tr>
<th>A1</th>
<th>A0</th>
<th>D7</th>
<th>D6</th>
<th>D5</th>
<th>D4</th>
<th>D3</th>
<th>D2</th>
<th>D1</th>
<th>D0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

PAR: NONE

Transmit Commands

The 8273 supports three transmit commands: Transmit Frame, Loop Transmit, Transmit Transparent.

TRANSMIT FRAME (CMD CODE C8)

<table>
<thead>
<tr>
<th>A1</th>
<th>A0</th>
<th>D7</th>
<th>D6</th>
<th>D5</th>
<th>D4</th>
<th>D3</th>
<th>D2</th>
<th>D1</th>
<th>D0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

PAR:

| LEAST SIGNIFICANT BYTE OF FRAME LENGTH (L0) |
| 0  | 1  |

| MOST SIGNIFICANT BYTE OF FRAME LENGTH (L1) |
| 0  | 1  |

| ADDRESS FIELD OF TRANSMIT FRAME (A) |
| 0  | 1  |

| CONTROL FIELD OF TRANSMIT FRAME (C) |
| 0  | 1  |

Transmits one frame including: initial flag, frame check sequence, and the final flag.

If the buffered mode is specified, the L0, L1, frame length provides as a parameter is the length of the information field and the address and control fields must be input.

In unbuffered mode the frame length provided must be the length of the information field plus two and the address and control fields must be the first two bytes of data. Thus only the frame length bytes are required as parameters.
**LOOP TRANSMIT (CMD CODE CA)**

<table>
<thead>
<tr>
<th>A₁</th>
<th>A₀</th>
<th>D₇</th>
<th>D₆</th>
<th>D₅</th>
<th>D₄</th>
<th>D₃</th>
<th>D₂</th>
<th>D₁</th>
<th>D₀</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

**CMD:**

**PAR:**

- 0 1 LEAST SIGNIFICANT BYTE OF FRAME LENGTH (L₀)
- 0 1 MOST SIGNIFICANT BYTE OF FRAME LENGTH (L₁)
- 0 1 ADDRESS FIELD OF TRANSMIT FRAME (A)
- 0 1 CONTROL FIELD OF TRANSMIT FRAME (C)

Transmits one frame in the same manner as the transmit frame command except:

1) If the flag stream mode is not active transmission will begin after a received EOP has been converted to a flag.
2) If the flag stream mode is active transmission will begin at the next flag boundary for buffered mode or at the third flag boundary for non-buffered mode.
3) At the end of a loop transmit the one-bit delay mode is entered and the flag stream mode is reset.

**TRANSMIT TRANSPARENT (CMD CODED C9)**

<table>
<thead>
<tr>
<th>A₁</th>
<th>A₀</th>
<th>D₇</th>
<th>D₆</th>
<th>D₅</th>
<th>D₄</th>
<th>D₃</th>
<th>D₂</th>
<th>D₁</th>
<th>D₀</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

**CMD:**

**PAR:**

- 0 1 LEAST SIGNIFICANT BYTE OF FRAME LENGTH (L₀)
- 0 1 MOST SIGNIFICANT BYTE OF FRAME LENGTH (L₁)

The 8273 will transmit a block of raw data without protocol, i.e., no zero bit insertion, flags, or frame check sequences.

**Abort Transmit Commands**

An abort command is supported for each type of transmit command. The abort commands are ignored if a transmit command is not in progress.

**ABORT TRANSMIT FRAME (CMD CODE CC)**

<table>
<thead>
<tr>
<th>A₁</th>
<th>A₀</th>
<th>D₇</th>
<th>D₆</th>
<th>D₅</th>
<th>D₄</th>
<th>D₃</th>
<th>D₂</th>
<th>D₁</th>
<th>D₀</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

**CMD:**

**PAR:** NONE

After an abort character (eight contiguous ones) is transmitted, the transmitter reverts to sending flags or idles as a function of the flag stream mode specified.

**ABORT LOOP TRANSMIT (CMD CODE CE)**

<table>
<thead>
<tr>
<th>A₁</th>
<th>A₀</th>
<th>D₇</th>
<th>D₆</th>
<th>D₅</th>
<th>D₄</th>
<th>D₃</th>
<th>D₂</th>
<th>D₁</th>
<th>D₀</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

**CMD:**

**PAR:** NONE

After a flag is transmitted the transmitter reverts to one bit delay mode.

**ABORT TRANSMIT TRANSPARENT (CMD CODE CD)**

<table>
<thead>
<tr>
<th>A₁</th>
<th>A₀</th>
<th>D₇</th>
<th>D₆</th>
<th>D₅</th>
<th>D₄</th>
<th>D₃</th>
<th>D₂</th>
<th>D₁</th>
<th>D₀</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

**CMD:**

**PAR:** NONE

The transmitter reverts to sending flags or idles as a function of the flag stream mode specified.

**Modem Control Commands**

The modem control commands are used to manipulate the modem control ports.

When read Port A or Port B commands are executed the result of the command is returned in the result register. The Bit Set Port B command requires a parameter that is a mask that corresponds to the bits to be set. The Bit Reset Port B command requires a mask that has a zero in the bit positions that are to be reset.

**READ PORT A (CMD CODE 22)**

<table>
<thead>
<tr>
<th>A₁</th>
<th>A₀</th>
<th>D₇</th>
<th>D₆</th>
<th>D₅</th>
<th>D₄</th>
<th>D₃</th>
<th>D₂</th>
<th>D₁</th>
<th>D₀</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

**CMD:**

**PAR:** NONE

**READ PORT B (CMD CODE 23)**

<table>
<thead>
<tr>
<th>A₁</th>
<th>A₀</th>
<th>D₇</th>
<th>D₆</th>
<th>D₅</th>
<th>D₄</th>
<th>D₃</th>
<th>D₂</th>
<th>D₁</th>
<th>D₀</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

**CMD:**

**PAR:** NONE
SET PORT B BITS (CMD CODE A3)

This command allows user defined Port B pins to be set.

(D0) REQUEST TO SEND

This is a dedicated 8273 modem control signal, and reflects the same logical state of RTS pin.

RESET PORT B BITS (CMD CODE 63)

This command allows Port B user defined bits to be reset.

(D8) FLAG DETECT

This bit can be used to set the flag detect pin. However, it will be reset when the next flag is detected.

(D4-D1) USER DEFINED OUTPUTS

These bits correspond to the state of the PB4-PB1 output pins.

8273 Command Summary

<table>
<thead>
<tr>
<th>Command Description</th>
<th>Command HEX</th>
<th>Parameter</th>
<th>Results</th>
<th>Result Port</th>
<th>Completion Interrupt</th>
</tr>
</thead>
<tbody>
<tr>
<td>Set One Bit Delay</td>
<td>A4</td>
<td>Set Mask</td>
<td>None</td>
<td>—</td>
<td>No</td>
</tr>
<tr>
<td>Reset One Bit Delay</td>
<td>64</td>
<td>Reset Mask</td>
<td>None</td>
<td>—</td>
<td>No</td>
</tr>
<tr>
<td>Set Data Transfer Mode</td>
<td>97</td>
<td>Set Mask</td>
<td>None</td>
<td>—</td>
<td>No</td>
</tr>
<tr>
<td>Reset Data Transfer Mode</td>
<td>57</td>
<td>Reset Mask</td>
<td>None</td>
<td>—</td>
<td>No</td>
</tr>
<tr>
<td>Set Operating Mode</td>
<td>91</td>
<td>Set Mask</td>
<td>None</td>
<td>—</td>
<td>No</td>
</tr>
<tr>
<td>Reset Operating Mode</td>
<td>51</td>
<td>Reset Mask</td>
<td>None</td>
<td>—</td>
<td>No</td>
</tr>
<tr>
<td>Set Serial I/O Mode</td>
<td>A0</td>
<td>Set Mask</td>
<td>None</td>
<td>—</td>
<td>No</td>
</tr>
<tr>
<td>Reset Serial I/O Mode</td>
<td>60</td>
<td>Reset Mask</td>
<td>None</td>
<td>—</td>
<td>No</td>
</tr>
<tr>
<td>General Receive</td>
<td>C0</td>
<td>B0, B1</td>
<td>RIC,R0,R1,(A,C)(2)</td>
<td>RXI/R</td>
<td>Yes</td>
</tr>
<tr>
<td>Selective Receive</td>
<td>C1</td>
<td>B0,B1,A1,A2</td>
<td>RIC,R0,R1,(A,C)(2)</td>
<td>RXI/R</td>
<td>Yes</td>
</tr>
<tr>
<td>Selective Loop Receive</td>
<td>C2</td>
<td>B0,B1,A1,A2</td>
<td>RIC,R0,R1,(A,C)(2)</td>
<td>RXI/R</td>
<td>Yes</td>
</tr>
<tr>
<td>Receive Disable</td>
<td>C5</td>
<td>None</td>
<td>None</td>
<td>—</td>
<td>No</td>
</tr>
<tr>
<td>Transmit Frame</td>
<td>C8</td>
<td>L0,L1,(A,C)(1)</td>
<td>TIC</td>
<td>TXI/R</td>
<td>Yes</td>
</tr>
<tr>
<td>Loop Transmit</td>
<td>CA</td>
<td>L0,L1,(A,C)(1)</td>
<td>TIC</td>
<td>TXI/R</td>
<td>Yes</td>
</tr>
<tr>
<td>Transmit Transparent</td>
<td>C9</td>
<td>L0,L1</td>
<td>TIC</td>
<td>TXI/R</td>
<td>Yes</td>
</tr>
<tr>
<td>Abort Transmit Frame</td>
<td>CC</td>
<td>None</td>
<td>TIC</td>
<td>TXI/R</td>
<td>Yes</td>
</tr>
</tbody>
</table>
8273 Command Summary (Continued)

<table>
<thead>
<tr>
<th>Command Description</th>
<th>Command HEX</th>
<th>Parameter</th>
<th>Results</th>
<th>Result Port</th>
<th>Completion Interrupt</th>
</tr>
</thead>
<tbody>
<tr>
<td>Abort Loop Transmit</td>
<td>CE</td>
<td>None</td>
<td>TIC</td>
<td>TXI/R</td>
<td>Yes</td>
</tr>
<tr>
<td>Abort Transmit Transparent</td>
<td>CD</td>
<td>None</td>
<td>TIC</td>
<td>TXI/R</td>
<td>Yes</td>
</tr>
<tr>
<td>Read Port A</td>
<td>22</td>
<td>None</td>
<td>Port Value</td>
<td>Result</td>
<td>No</td>
</tr>
<tr>
<td>Read Port B</td>
<td>23</td>
<td>None</td>
<td>Port Value</td>
<td>Result</td>
<td>No</td>
</tr>
<tr>
<td>Set Port B Bit</td>
<td>A3</td>
<td>Set Mask</td>
<td>None</td>
<td>—</td>
<td>No</td>
</tr>
<tr>
<td>Reset Port B Bit</td>
<td>63</td>
<td>Reset Mask</td>
<td>None</td>
<td>—</td>
<td>No</td>
</tr>
</tbody>
</table>

NOTES:
1. Issued only when in buffered mode.
2. Read as results only in buffered mode.

8273 Command Summary Key
- B0—Least significant byte of the receiver buffer length.
- B1—Most significant byte of the receive buffer length.
- L0—Least significant byte of the Tx frame length.
- L1—Most significant byte of the Tx frame length.
- A1—Receive frame address match field one.
- A2—Receive frame address match field two.
- A—Address field of received frame. If non-buffered mode is specified, this result is not provided.
- C—Control field of received frame. If non-buffered mode is specified this result is not provided.
- RXI/R—Receive interrupt result register.
- TXI/R—Transmit interrupt result register.
- R0—Least significant byte of the length of the frame received.
- R1—Most significant byte of the length of the frame received.
- RIC—Receiver interrupt result code.
- TIC—Transmitter interrupt result code.

![Diagram of Frame Reception]

**Figure 15. Typical Frame Reception**

NOTE:
In order to ensure proper operation to the maximum baud rate, Receive commands or Read/Write Port commands should be written only when either the transmitter or the receiver is inactive. In full duplex systems, it is recommended that these commands be issued after servicing a transmitter interrupt but before a new transmit command is issued. When operating in full Duplex (active transmitter or receiver) with commands, the maximum data rate decreases to 49K Baud.
Figure 16a. Typical Frame Transmission, Buffered Mode

Figure 16b. Typical Frame Transmission, Non-Buffered Mode
Figure 17. 8273 System Diagram

WAVEFORMS

COMMAND PHASE

Table 2. Command Phase Timing (Full Duplex)

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Timing Parameter</th>
<th>Buffered</th>
<th>Non-Buffered</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Min</td>
<td>Max</td>
</tr>
<tr>
<td>T1</td>
<td>Between Command &amp; First Parameter</td>
<td>13</td>
<td>756</td>
</tr>
<tr>
<td></td>
<td></td>
<td>13</td>
<td>857</td>
</tr>
<tr>
<td>T2</td>
<td>Between Consecutive Parameters</td>
<td>10</td>
<td>604</td>
</tr>
<tr>
<td></td>
<td></td>
<td>10</td>
<td>705</td>
</tr>
<tr>
<td>T3</td>
<td>Command Parameter Buffer Full Bit Reset after Parameter Loaded</td>
<td>10</td>
<td>604</td>
</tr>
<tr>
<td></td>
<td></td>
<td>10</td>
<td>705</td>
</tr>
<tr>
<td>T4</td>
<td>Command Busy Bit Reset after Last Parameter</td>
<td>128</td>
<td>702</td>
</tr>
<tr>
<td></td>
<td></td>
<td>128</td>
<td>803</td>
</tr>
<tr>
<td>T5</td>
<td>CPBF Bit Reset after Last Parameter</td>
<td>10</td>
<td>604</td>
</tr>
<tr>
<td></td>
<td></td>
<td>10</td>
<td>705</td>
</tr>
</tbody>
</table>
WAVEFORMS (Continued)

RECEIVER INTERRUPT

Table 3. Receiver Interrupt Result Timing

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Timing Parameter (Clock Cycles)</th>
<th>Buffered</th>
<th>Non-Buffered</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Min</td>
<td>Max</td>
<td></td>
</tr>
<tr>
<td>T1</td>
<td>RxIRA Bit Set after RIC Read</td>
<td>18</td>
<td>29</td>
<td>tcy</td>
</tr>
<tr>
<td>T2</td>
<td>RxINT Goes Away after Last Int. Result Read</td>
<td>16</td>
<td>27</td>
<td>tcy</td>
</tr>
</tbody>
</table>

TRANSMIT INTERRUPT

Table 4. Transmit Interrupt Result

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Timing (Clock Cycle)</th>
<th>Buffered</th>
<th>Non-Buffered</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Min</td>
<td>Max</td>
<td></td>
</tr>
<tr>
<td>T1</td>
<td>TxINT Inactive after Int. Results Read</td>
<td>13</td>
<td>353</td>
<td>tcy</td>
</tr>
</tbody>
</table>
ABSOLUTE MAXIMUM RATINGS*

Ambient Temperature Under Bias .......-0°C to 70°C
Storage Temperature .................-65°C to +150°C
Voltage on Any Pin With Respect to Ground ............-0.5V to +7V
Power Dissipation ......................1 Watt

*Notice: Stresses above those listed under “Absolute Maximum Ratings” may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions above those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

D.C. CHARACTERISTICS 8273 (TA = 0°C to 70°C, VIN = +5.0V ±5%)

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Parameter</th>
<th>Min</th>
<th>Max</th>
<th>Unit</th>
<th>Test Conditions</th>
</tr>
</thead>
<tbody>
<tr>
<td>VIL</td>
<td>Input Low Voltage</td>
<td>-0.5</td>
<td>0.8</td>
<td>V</td>
<td>VCC + 0.5</td>
</tr>
</tbody>
</table>
| VIH    | Input High Voltage      | 2.0 | VCC + 0.5| V    | IOL = 2.0 mA for Data Bus Pins
| VOL    | Output Low Voltage      | 0.45| V       |      | IOL = 1.0 mA for Output Port Pins
| VOH    | Output High Voltage     | 2.4 | V       |      | IOL = 1.6 mA for All Other Pins
| IIL    | Input Load Current      | ±10 | μA      |      | I = -200 μA for Data Bus Pins
| IOL    | Output Leakage Current  | ±10 | μA      |      | IOUT = VCC to 0V
| ICC    | VCC Supply Current      | 180 | mA      |      | VCC = +5.0V ±5%                                |

CAPACITANCE 8273 (TA = 25°C, VIN = GND = 0V)

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Parameter</th>
<th>Min</th>
<th>Typ</th>
<th>Max</th>
<th>Unit</th>
<th>Test Conditions</th>
</tr>
</thead>
<tbody>
<tr>
<td>CIN</td>
<td>Input Capacitance</td>
<td>10</td>
<td>pF</td>
<td></td>
<td>t = 1 MHz</td>
<td></td>
</tr>
<tr>
<td>C/I/O</td>
<td>I/O Capacitance</td>
<td>20</td>
<td>pF</td>
<td></td>
<td>Unmeasured Pins Returned to GND</td>
<td></td>
</tr>
</tbody>
</table>

A.C. CHARACTERISTICS (TA = 0°C to 70°C, VIN = +5.0V ±5%)

CLOCK TIMING (8273)

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Parameter</th>
<th>Min</th>
<th>Typ</th>
<th>Max</th>
<th>Unit</th>
<th>Test Conditions</th>
</tr>
</thead>
<tbody>
<tr>
<td>tCY</td>
<td>Clock</td>
<td>250</td>
<td></td>
<td>1000</td>
<td>ns</td>
<td>64K Baud Max Operating Rate</td>
</tr>
<tr>
<td>tCL</td>
<td>Clock Low</td>
<td>120</td>
<td></td>
<td></td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>tCH</td>
<td>Clock High</td>
<td>120</td>
<td></td>
<td></td>
<td>ns</td>
<td></td>
</tr>
</tbody>
</table>
## A.C. CHARACTERISTICS 8273 \( (T_A = 0°C \text{ to } 70°C, V_{CC} = +5.0V \pm 5\%) \)

### READ CYCLE

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Parameter</th>
<th>Min</th>
<th>Max</th>
<th>Unit</th>
<th>Test Conditions</th>
</tr>
</thead>
<tbody>
<tr>
<td>(t_{AC})</td>
<td>Select Setup to (RD)</td>
<td>0</td>
<td></td>
<td>ns</td>
<td>(Note 2)</td>
</tr>
<tr>
<td>(t_{CA})</td>
<td>Select Hold from (RD)</td>
<td>0</td>
<td></td>
<td>ns</td>
<td>(Note 2)</td>
</tr>
<tr>
<td>(t_{RR})</td>
<td>(RD) Pulse Width</td>
<td>250</td>
<td></td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>(t_{RD})</td>
<td>Data Delay from Address</td>
<td>300</td>
<td></td>
<td>ns</td>
<td>(Note 2)</td>
</tr>
<tr>
<td>(t_{RD})</td>
<td>Data Delay from (RD)</td>
<td>200</td>
<td></td>
<td>ns</td>
<td>(C_L = 150\ pF), (Note 2)</td>
</tr>
<tr>
<td>(t_{DF})</td>
<td>Output Float Delay</td>
<td>20</td>
<td>100</td>
<td>ns</td>
<td>(C_L = 20\ pF) for Minimum; 150 pF for Maximum</td>
</tr>
<tr>
<td>(t_{DC})</td>
<td>DACK Setup to (RD)</td>
<td>25</td>
<td></td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>(t_{CD})</td>
<td>DACK Hold from (RD)</td>
<td>25</td>
<td></td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>(t_{KD})</td>
<td>Data Delay from DACK</td>
<td>300</td>
<td></td>
<td>ns</td>
<td></td>
</tr>
</tbody>
</table>

### WRITE CYCLE

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Parameter</th>
<th>Min</th>
<th>Max</th>
<th>Unit</th>
<th>Test Conditions</th>
</tr>
</thead>
<tbody>
<tr>
<td>(t_{AC})</td>
<td>Select Setup to (WR)</td>
<td>0</td>
<td></td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>(t_{CA})</td>
<td>Select Hold from (WR)</td>
<td>0</td>
<td></td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>(t_{WW})</td>
<td>(WR) Pulse Width</td>
<td>250</td>
<td></td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>(t_{DW})</td>
<td>Data Setup to (WR)</td>
<td>150</td>
<td></td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>(t_{WD})</td>
<td>Data Hold from (WR)</td>
<td>0</td>
<td></td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>(t_{DC})</td>
<td>DACK Setup to (WR)</td>
<td>25</td>
<td></td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>(t_{CD})</td>
<td>DACK Hold from (WR)</td>
<td>25</td>
<td></td>
<td>ns</td>
<td></td>
</tr>
</tbody>
</table>

### DMA

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Parameter</th>
<th>Min</th>
<th>Max</th>
<th>Unit</th>
<th>Test Conditions</th>
</tr>
</thead>
<tbody>
<tr>
<td>(t_{CQ})</td>
<td>Request Hold from (WR) or (RD) (for Non-Burst Mode)</td>
<td></td>
<td>200</td>
<td>ns</td>
<td></td>
</tr>
</tbody>
</table>

### OTHER TIMING

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Parameter</th>
<th>Min</th>
<th>Max</th>
<th>Unit</th>
<th>Test Conditions</th>
</tr>
</thead>
<tbody>
<tr>
<td>(t_{RSTW})</td>
<td>Reset Pulse Width</td>
<td>10</td>
<td>(t_{CY})</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(t_r)</td>
<td>Input Signal Rise Time</td>
<td>20</td>
<td></td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>(t_f)</td>
<td>Input Signal Fall Time</td>
<td>20</td>
<td></td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>(t_{RSTS})</td>
<td>Reset to First IOWR</td>
<td>2</td>
<td></td>
<td>(t_{CY})</td>
<td></td>
</tr>
<tr>
<td>(t_{CY32})</td>
<td>32X Clock Cycle Time</td>
<td>(13.02 \times t_{CY})</td>
<td></td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>(t_{CL32})</td>
<td>32X Clock Low Time</td>
<td>(4 \times t_{CY})</td>
<td></td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>(t_{CH32})</td>
<td>32X Clock High Time</td>
<td>(4 \times t_{CY})</td>
<td></td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>(t_{DPLL})</td>
<td>DPLL Output Low</td>
<td>(1 \times t_{CY}) - 50</td>
<td></td>
<td>ns</td>
<td></td>
</tr>
</tbody>
</table>
A.C. CHARACTERISTICS 8273 \( (T_A = 0°C \text{ to } 70°C, V_{CC} = +5.0V \pm 5\%) \) (Continued)

OTHER TIMING (Continued)

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Parameter</th>
<th>Min</th>
<th>Max</th>
<th>Unit</th>
<th>Test Conditions</th>
</tr>
</thead>
<tbody>
<tr>
<td>( t_{DCL} )</td>
<td>Data Clock Low</td>
<td>( 1 \times t_{CY} - 50)</td>
<td>ns</td>
<td></td>
<td></td>
</tr>
<tr>
<td>( t_{DCH} )</td>
<td>Data Clock High</td>
<td>( 2 \times t_{CY} )</td>
<td>ns</td>
<td></td>
<td></td>
</tr>
<tr>
<td>( t_{DCY} )</td>
<td>Data Clock</td>
<td>( 62.5 \times t_{CY} )</td>
<td>ns</td>
<td>(Note 3)</td>
<td></td>
</tr>
<tr>
<td>( t_{TD} )</td>
<td>Transmit Data Delay</td>
<td>200</td>
<td>ns</td>
<td></td>
<td></td>
</tr>
<tr>
<td>( t_{DS} )</td>
<td>Data Setup Time</td>
<td>200</td>
<td>ns</td>
<td></td>
<td></td>
</tr>
<tr>
<td>( t_{DH} )</td>
<td>Data Hold Time</td>
<td>100</td>
<td>ns</td>
<td></td>
<td></td>
</tr>
<tr>
<td>( t_{FLD} )</td>
<td>FLAG DET Output Low</td>
<td>( 8 \times t_{CY} \pm 50)</td>
<td>ns</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

NOTES:
1. All timing measurements are made at the reference voltages unless otherwise specified: Input “1” at 2.0V, “0” at 0.8V; Output “1” at 2.0V, “0” at 0.8V.
2. \( t_{AD}, t_{RD}, t_{AC}, \) and \( t_{CA} \) are not concurrent specs.
3. If receive commands or Read/Write Port commands are issued while both the transmitter and receiver are active, this specification will be \( 81.5 \ T_CY \) min.

A.C. TESTING INPUT, OUTPUT WAVEFORM

A.C. Testing: Inputs are driven at 2.4V for a logic “1” and 0.45V for a logic “0”. Timing measurements are made at 2.0V for a logic “1” and 0.8V for a logic “0”.

A.C. TESTING LOAD CIRCUIT

DEVICE UNDER TEST

\( C_L = 150 \ pF \)

\( C_L \) includes Jig Capacitance

WAVEFORMS

READ

DACK
\( A_D, A_T, CS \)
RD
DATA BUS
WAVEFORMS (Continued)

WRITE

DATA BUS

DMA

TRANSMIT

2-110
WAVEFORMS (Continued)

RECEIVE

![Waveform Diagram](image)

DPLL OUTPUT

![Waveform Diagram](image)

FLAG DETECT OUTPUT

![Waveform Diagram](image)
The Intel 8274 Multi-Protocol Series Controller (MPSC) is designed to interface High Speed Communications Lines using Asynchronous, IBM Bisync, and SDLC/HDLC protocol to Intel microcomputer systems. It can be interfaced with Intel's MCS-48, -85, -51; iAPX-86, -88, -186 and -188 families, the 8237 DMA Controller, or the 8089 1/0 Processor in polled, interrupt driven, or OMA driven modes of operation.

The MPSC is a 40 pin device fabricated using Intel's High Performance HMOS Technology.
Table 1. Pin Description

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Pin No.</th>
<th>Type</th>
<th>Name and Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLK</td>
<td>1</td>
<td>I</td>
<td>CLOCK: System clock, TTL compatible.</td>
</tr>
<tr>
<td>RESET</td>
<td>2</td>
<td>I</td>
<td>RESET: A low signal on this pin will force the MPSC to an idle state. TxD_A and TxD_B are forced high. The modem interface output signals are forced high. The MPSC will remain idle until the control registers are initialized. Reset must be true for one complete CLK cycle.</td>
</tr>
<tr>
<td>CD_A</td>
<td>3</td>
<td>I</td>
<td>CARRIER DETECT (CHANNEL A): This interface signal is supplied by the modem to indicate that a data carrier signal has been detected and that a valid data signal is present on the RxD_A line. If the auto enable control is set the 8274 will not enable the serial receiver until CD_A has been activated.</td>
</tr>
<tr>
<td>RxC_B</td>
<td>4</td>
<td>I</td>
<td>RECEIVE CLOCK (CHANNEL B): The serial data are shifted into the Receive Data input (RxD_B) on the rising edge of the Receive Clock.</td>
</tr>
<tr>
<td>CD_B</td>
<td>5</td>
<td>I</td>
<td>CARRIER DETECT (CHANNEL B): This interface signal is supplied by the modem to indicate that a data carrier signal has been detected and that a valid data signal is present on the RxD_B line. If the auto enable control is set the 8274 will not enable the serial receiver until CD_B has been activated.</td>
</tr>
<tr>
<td>CTS_B</td>
<td>6</td>
<td>I</td>
<td>CLEAR TO SEND (CHANNEL B): This interface signal is supplied by the modem in response to an active RTS signal. CTS indicates that the data terminal/computer equipment is permitted to transmit data. In addition, if the auto enable control is set, the 8274 will not transmit data bytes until CTS has been activated.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>BIT 5 OF WR3</td>
</tr>
<tr>
<td>TxC_B</td>
<td>7</td>
<td>I</td>
<td>TRANSMIT CLOCK (CHANNEL B): The serial data are shifted out from the Transmit Data output (TxD_B) on the falling edge of the Transmit Clock.</td>
</tr>
<tr>
<td>TxD_B</td>
<td>8</td>
<td>O</td>
<td>TRANSMIT DATA (CHANNEL B): This pin transmits serial data to the communications channel (Channel B).</td>
</tr>
<tr>
<td>RxD_B</td>
<td>9</td>
<td>I</td>
<td>RECEIVE DATA (CHANNEL B): This pin receives serial data from the communications channel (Channel B).</td>
</tr>
<tr>
<td>SYNDET_B</td>
<td>10</td>
<td>I/O</td>
<td>SYNCHRONOUS DETECTION (CHANNEL B): This pin is used in byte synchronous mode as either an internal sync detect (output) or as a means to force external synchronization (input). In SDLC mode, this pin is an output indicating Flag detection. In asynchronous mode it is a general purpose input (Channel B).</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>REQUEST TO SEND (CHANNEL B): General purpose output, generally used to signal that Channel B is ready to send data. When the RTS bit is reset in asynchronous mode, the signal does not go inactive (High) until the transmitter is empty. SYNDET_B or RTS_B selection is done by WR2; D7. (Channel A).</td>
</tr>
</tbody>
</table>
Table 1. Pin Description (Continued)

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Pin No.</th>
<th>Type</th>
<th>Name and Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>RDY&lt;sub&gt;B&lt;/sub&gt;/TxDRQA</td>
<td>11</td>
<td>O</td>
<td>READY (CHANNEL B)/TRANSmitter DMA REQUEST (CHANNEL A): In mode 0 this pin is RDY&lt;sub&gt;B&lt;/sub&gt; and is used to synchronize data transfers between the processor and the MPSC (Channel B). In modes 1 and 2 this pin is TxDRQA and is used by the Channel A transmitter to request a DMA transfer.</td>
</tr>
<tr>
<td>DB7</td>
<td>12</td>
<td>I/O</td>
<td>DATA BUS: The Data Bus lines are bidirectional three state lines which interface with the system's Data Bus.</td>
</tr>
<tr>
<td>DB6</td>
<td>13</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DB5</td>
<td>14</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DB4</td>
<td>15</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DB3</td>
<td>16</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DB2</td>
<td>17</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DB1</td>
<td>18</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DB0</td>
<td>19</td>
<td></td>
<td></td>
</tr>
<tr>
<td>GND</td>
<td>20</td>
<td></td>
<td>GROUND.</td>
</tr>
<tr>
<td>V&lt;sub&gt;CC&lt;/sub&gt;</td>
<td>40</td>
<td></td>
<td>POWER: +5V Supply</td>
</tr>
<tr>
<td>CTS&lt;sub&gt;A&lt;/sub&gt;</td>
<td>39</td>
<td>I</td>
<td>CLEAR TO SEND (CHANNEL A): This interface signal is supplied by the Modem in response to an active RTS signal. CTS indicates that the data terminal/computer equipment is permitted to transmit data. In addition, if the auto enable control is set, the 8274 will not transmit data bytes until CTS has been activated.</td>
</tr>
<tr>
<td>RTS&lt;sub&gt;A&lt;/sub&gt;</td>
<td>38</td>
<td>O</td>
<td>REQUEST TO SEND (CHANNEL A): General purpose output commonly used to signal that Channel A is ready to send data. When the RTS bit is reset in asynchronous mode, the signal does not go inactive (High) until the transmitter is empty.</td>
</tr>
<tr>
<td>TxD&lt;sub&gt;A&lt;/sub&gt;</td>
<td>37</td>
<td>O</td>
<td>TRANSMIT DATA (CHANNEL A): This pin transmits serial data to the communications channel (Channel A).</td>
</tr>
<tr>
<td>TxC&lt;sub&gt;A&lt;/sub&gt;</td>
<td>36</td>
<td>I</td>
<td>TRANSMIT CLOCK (CHANNEL A): The serial data are shifted out from the Transmit Data output (TxD&lt;sub&gt;A&lt;/sub&gt;) on the falling edge of the Transmit Clock.</td>
</tr>
<tr>
<td>RxC&lt;sub&gt;A&lt;/sub&gt;</td>
<td>35</td>
<td>I</td>
<td>RECEIVE CLOCK (CHANNEL A): The serial data are shifted into the Receive Data input (Rx&lt;sub&gt;D&lt;/sub&gt;A) on the rising edge of the Receive Clock.</td>
</tr>
<tr>
<td>Rx&lt;sub&gt;D&lt;/sub&gt;A</td>
<td>34</td>
<td>I</td>
<td>RECEIVE DATA (CHANNEL A): This pin receives serial data from the communications channel (Channel A).</td>
</tr>
<tr>
<td>SYNDET&lt;sub&gt;A&lt;/sub&gt;</td>
<td>33</td>
<td>I/O</td>
<td>SYNCHRONOUS DETECTION (CHANNEL A): This pin is used in byte synchronous mode as either an internal sync detect (output) or as a means to force external synchronization (input). In SDLC mode, this pin is an output indicating flag detection. In asynchronous mode it is a general purpose input (Channel A).</td>
</tr>
</tbody>
</table>
Table 1. Pin Description (Continued)

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Pin No.</th>
<th>Type</th>
<th>Name and Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>RDY_A/</td>
<td>32</td>
<td>O</td>
<td>READY: In mode 0 this pin is RDY_A and is used to synchronize data transfers between the processor and the MPSC (Channel A). In modes 1 and 2 this pin is RxDRQA and is used by the channel A receiver to request a DMA transfer.</td>
</tr>
<tr>
<td>RxDRQA</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D_TR_A</td>
<td>31</td>
<td>O</td>
<td>DATA TERMINAL READY (CHANNEL A): General purpose output.</td>
</tr>
<tr>
<td>IP_I/</td>
<td>30</td>
<td>O</td>
<td>INTERRUPT PRIORITY OUT/TRANSMITTER DMA REQUEST (CHANNEL B): In modes 0 and 1, this pin is Interrupt Priority Out. It is used to establish a hardware interrupt priority scheme with IP_I. It is low only if IP_I is low and the controlling processor is not servicing an interrupt from this MPSC. In mode 2 it is TxDRQB and is used to request a DMA cycle for a transmit operation (Channel B).</td>
</tr>
<tr>
<td>TxDRQB</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>IP_I/</td>
<td>29</td>
<td>I/O</td>
<td>INTERRUPT PRIORITY IN/RECEIVER DMA REQUEST (CHANNEL B): In modes 0 and 1, IP_I is Interrupt Priority In. A low on IP_I means that no higher priority device is being serviced by the controlling processor’s interrupt service routine. In mode 2 this pin is RxDRQB and is used to request a DMA cycle for a receive operation (Channel B).</td>
</tr>
<tr>
<td>RxDRQB</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>INT</td>
<td>28</td>
<td>O</td>
<td>INTERRUPT: The interrupt signal indicates that the highest priority internal interrupt requires service (open collector). Priority can be resolved via an external interrupt controller or a daisy-chain scheme.</td>
</tr>
<tr>
<td>INTA</td>
<td>27</td>
<td>I</td>
<td>INTERRUPT ACKNOWLEDGE: This Interrupt Acknowledge signal allows the highest priority interrupting device to generate an interrupt vector. This pin must be pulled high (inactive) in non-vector mode.</td>
</tr>
<tr>
<td>D_TR_B</td>
<td>26</td>
<td>O</td>
<td>DATA TERMINAL READY (CHANNEL B): This is a general purpose output.</td>
</tr>
<tr>
<td>A_0</td>
<td>25</td>
<td>I</td>
<td>ADDRESS: This line selects Channel A or B during data or command transfers. A low selects Channel A.</td>
</tr>
<tr>
<td>A_1</td>
<td>24</td>
<td>I</td>
<td>ADDRESS: This line selects between data or command information transfer. A low means data.</td>
</tr>
<tr>
<td>CS</td>
<td>23</td>
<td>I</td>
<td>CHIP SELECT: This signal selects the MSPC and enables reading from or writing into registers.</td>
</tr>
<tr>
<td>RD</td>
<td>22</td>
<td>I</td>
<td>READ: Read controls a data byte or status byte transfer from the MPSC to the CPU.</td>
</tr>
<tr>
<td>WR</td>
<td>21</td>
<td>I</td>
<td>WRITE: Write controls transfer of data or commands to the MPSC.</td>
</tr>
</tbody>
</table>
RESET

When the 8274 RESET line is activated, both MPSC channels enter the idle state. The serial output lines are forced to the marking state (high) and the modem interface signals (RTS, DTR) are forced high. In addition, the pointers registers are set to zero.

GENERAL DESCRIPTION

The Intel 8274 Multi-Protocol Serial Controller is a microcomputer peripheral device which supports Asynchronous, Byte Synchronous (Monosync, IBM Bisync), and Bit Synchronous (ISO's HDLC, IBM's SDLIC) protocols. This controller's flexible architecture allows easy implementation of many variations of these three protocols with low software and hardware overhead.

The Multi-Protocol Serial controller (MPSC) implements two independent serial receiver/transmitter channels.

The MPSC supports several microprocessor interface options: Polled, Wait, Interrupt driven and DMA driven. The MPSC is designed to support INTEL's MCS-85 and IAPX 86, 88, 186, 188 families.

FUNCTIONAL DESCRIPTION

Additional information on Asynchronous and Synchronous Communications with the 8274 is available respectively in the Applications Notes AP 134 and AP 145.

Command, parameter, and status information is stored in 21 registers within the MPSC (8 writable registers for each channel, 2 readable registers for Channel A and 3 readable registers for Channel B).

In the following discussion, the writable registers will be referred to as WRO through WR7 and the readable registers will be referred to as RRO through RR2.

This section of the data sheet describes how the Asynchronous and Synchronous protocols are implemented in the MPSC. It describes general considerations, transmit operation, and receive operation for Asynchronous, Byte Synchronous, and Bit Synchronous protocols.

ASYNCHRONOUS OPERATIONS

Transmitter/Receiver Initialization

(See Detailed Command Description Section for complete information)

In order to operate in asynchronous mode, each MPSC channel must be initialized with the following information:

1. Transmit/Receive Clock Rate. This parameter is specified by bits 6 and 7 of WR4. The clock rate may be set to 1, 16, 32, or 64 times the data-link bit rate. If the X1 clock mode is selected, the bit synchronization must be accomplished externally.

2. Number of Stop Bits. This parameter is specified by bits 2 and 3 of WR4. The number of stop bits may be set to 1, 1½, or 2.

3. Parity Selection. Parity may be set for odd, even, or no parity by bits 0 and 1 of WR4.

4. Receiver Character Length. This parameter sets the length of received characters to 5, 6, 7, or 8 bits. This parameter is specified by bits 6 and 7 of WR3.

5. Receiver Enable. The serial-channel receiver operation may be enabled or disabled by setting or clearing bit 0 of WR3.

6. Transmitter Character Length. This parameter sets the length of transmitted characters to 5, 6, 7, or 8 bits. This parameter is specified by bits 5 and 6 of WR5. Characters of less than 5 bits in length may be transmitted by setting the transmitted length to five bits (set bits 5 and 6 of WR5 to 0).

The MPSC then determines the actual number of bits to be transmitted from the character data byte. The bits to be transmitted must be right justified in the data byte, the next three bits must be set to 0 and all remaining bits must be set to 1. The following table illustrates the data formats for transmission of 1 to 5 bits of data.

<table>
<thead>
<tr>
<th>Byte Written</th>
<th>Number of Bits Transmitted</th>
</tr>
</thead>
<tbody>
<tr>
<td>D7 D6 D5 D4 D3 D2 D1 D0</td>
<td>(Character Length)</td>
</tr>
<tr>
<td>1 1 1 0 0 0 0 c</td>
<td>1</td>
</tr>
<tr>
<td>1 1 1 0 0 0 c c</td>
<td>2</td>
</tr>
<tr>
<td>1 1 0 0 0 c c c</td>
<td>3</td>
</tr>
<tr>
<td>1 0 0 0 c c c c</td>
<td>4</td>
</tr>
<tr>
<td>0 0 0 c c c c c</td>
<td>5</td>
</tr>
</tbody>
</table>

7. Transmitter Enable. The serial channel transmitter operation may be enabled or disabled by setting or clearing bit 3 of WR5.

For data transmission via a modem or RS-232-C interface, the following information must also be specified:

1. The Request To Send (RTS) (WR5; D1) and Data Terminal Ready (DTR) (WR5; D7) bits must be set along with the Transmit Enable bit (WR5; D9).

2. Auto Enable may be set to allow the MPSC to automatically enable the channel transmitter when the clear-to-send signal is active and to automatically enable the receiver when the carrier-detect signal is active. However, the Transmit Enable bit (WR3; D3) and Receive Enable bit (WR3; D1) must be set in order to use the Auto Enable mode. Auto Enable is controlled by bit 5 of WR3.

When loading Initialization parameters into the MPSC, WR4 information must be written before the WR1, WR3, WR5 parameters commands.

During initialization, it is desirable to guarantee that the external/status latches reflect the latest interface information. Since up to two state changes are internally stored by the MPSC, at least two Reset External/Status Interrupt commands must be issued. This procedure is most easily accomplished by simply issuing this reset command whenever the pointer register is set during initialization.

An MPSC initialization procedure (MPSC$RX$INIT) for asynchronous communication is listed in Intel Application Note AP 134.

**TRANSMIT**

The transmit function begins when the Transmit Enable bit (WR5; D3) is set. The MPSC automatically adds the start bit, the programmed parity bit (odd, even or no parity) and the programmed number of stop bits (1, 1.5 or 2 bits) to the data character being transmitted. 1.5 stop bits option must be used with X16, X32 or X64 options.

Data (TxO) output on the TRANSMIT interface, the following information must be specified:

**Receive**

The receive function begins when the Receive Enable (WR3; D0) bit is set. If the Auto Enable (WR3; D5) option is selected, then Carrier Detect (CD) must also be low. A valid start bit is detected if a low persists for at least ½ bit time on the Receive Data (RxD) input.

The data is sampled at mid-bit time, on the rising edge of RxC, until the entire character is assembled. The receiver inserts 1's when a character is less than 8 bits. If parity (WR4; D0) is enabled and the character is less than 8 bits the parity bit is not stripped from the character.

**Error Reporting**

The receiver also stores error status for each of the 3 data characters in the data buffer. Three error conditions may be encountered during data reception in the asynchronous mode:

1. **Parity.** If parity bits are computed and transmitted with each character and the MPSC is set to check parity (bit 0 in WR4 is set), a parity error will occur whenever the number of "1" bits within the character (including the parity bit) does not match the odd/even setting of the parity check flag (bit 1 in WR4). When a parity error is detected, the parity error flag (RR1; D4) is set and remains set until it is reset by the Error Reset command (WR0; D5, D4, D3).

2. **Framing.** A framing error will occur if a stop bit is not detected immediately following the parity bit (if parity checking is enabled) or immediately following the most-significant data bit (if parity checking is not enabled). When a Framing Error is detected, the Framing Error bit (RR1; D6) is set and remains set until reset by the Error Reset Command (WR0; D5, D4, D3). The detection of a Framing Error adds an additional ½ bit time to the character time so the Framing Error is not interpreted as a new start bit.

3. **Overrun.** If the CPU fails to read a data character while more than three characters have been received, the Receive Overrun bit (RR1; D5) is set. When this occurs, the fourth character assembled replaces the third character in the receive buffers. Only the overwritten character is flagged with the Receive Overrun bit. The Receive Overrun bit (RR1; D5) is reset by the Error Reset command (WR0; D5, D4, D3).
External/Status Latches

The MPSC continuously monitors the state of five external/status conditions:

1. CTS—clear-to-send input pin.
2. CD—carrier-detect input pin.
3. SYNDET—sync-detect input pin. This pin may be used as a general-purpose input in the asynchronous communication mode.
4. BREAK—a break condition (series of space bits on the receiver input pin).
5. TxUNDERRUN/EOM—Transmitter Underrun/End of Message.

A change of state in any of these monitored conditions will cause the associated status bit in RR0 to be latched (and optionally cause an interrupt).

If the External/Status Interrupt bit (WR1; D0) is enabled, Break Detect (RR0; D7) and Carrier Detect (RR0; D3) will cause an interrupt. Reset External/Status interrupts (WR0; D5, D4, D3) will clear Break Detect and Carrier Detect bits if they are set.

Command, parameter, and status information is stored in 21 registers within the MPSC (8 writable registers for each channel, 2 readable registers for Channel A and 3 readable registers for Channel B). They are all accessed via the command ports.

An internal pointer register selects which of the command or status registers will be read or written during a command/status access of an MPSC channel.

After reset, the contents of the pointer registers are zero. The first write to a command register causes the data to be loaded into Write Register 0 (WR0). The three least significant bits of WR0 are loaded into the Command/Status Pointer. The next read or write operation accesses the read or write register selected by the pointer. The pointer is reset after the read or write operation is completed.

---

**Figure 3. Command/Status Register Architecture (each serial channel)**
Asynchronous Mode Register Setup

<table>
<thead>
<tr>
<th>D7</th>
<th>D6</th>
<th>D5</th>
<th>D4</th>
<th>D3</th>
<th>D2</th>
<th>D1</th>
<th>D0</th>
</tr>
</thead>
<tbody>
<tr>
<td>WR3</td>
<td>00</td>
<td>Rx 5 b/char</td>
<td>AUTO ENABLE</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>Rx</td>
</tr>
<tr>
<td></td>
<td>01</td>
<td>Rx 7 b/char</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>ENABLE</td>
</tr>
<tr>
<td></td>
<td>10</td>
<td>Tx 6 b/char</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>11</td>
<td>Rx 8 b/char</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

| WR4| 00 | X1 Clock  | 0         | 01 | 1 STOP BIT | EVEN/ | PARITY Enable |
|    | 01 | X16 Clock |           | 10 | 1½ STOP BITS| ODD  |          |
|    | 10 | X32 Clock |           | 11 | 2 STOP BITS |       |          |
|    | 11 | X64 Clock |           |    |            |       |          |

| WR5| DTR| Tx ≤ 5 b/char| SEND BREAK| Tx 7 b/char| 0 | RTS | 0 |
|    |    |              | Tx ENABLE |              |   |

SYNCHRONOUS OPERATION—MONOSYNC, BISYNC

General

The MPSC must be initialized with the following parameters: odd or even parity (WR4; D1, D0), X1 clock mode (WR4; D7, D6), 8- or 16-bit sync character (WR4; D5, D4), CRC polynomial (WR5; D2), Transmitter Enable (WR5; D3), interrupt modes (WR1, WR2), transmit character length (WR5; D6, D5) and receive character length (WR3; D7, D6). WR4 parameters must be written before WR1, WR3, WR5, WR6 and WR7.

The data is transmitted on the falling edge of the Transmit Clock, (TxC) and is received on the rising edge of Receive Clock (RxC). The X1 clock is used for both transmit and receive operations for all three sync modes: Mono, Bi and External.

Transmit Set-Up—Monosync, Bisync

Transmit data is held high after channel reset, or if the transmitter is not enabled. A break may be programmed to generate a spacing line that begins as soon as the Send Break (WR5; D4) bit is set. With the transmitter fully initialized and enabled, the default condition is continuous transmission of the 8- or 16-bit sync character.

Using interrupts for data transfer requires that the Transmit Interrupt/DMA Enable bit (WR1; D1) be set. An interrupt is generated each time the transmit buffer becomes empty. The interrupt can be satisfied either by writing another character into the transmitter or by resetting the Transmitter Interrupt/DMA Pending latch with a Reset Transmitter Inter-

Synchronous Mode Register Setup—Monosync, Bisync

<table>
<thead>
<tr>
<th>D7</th>
<th>D6</th>
<th>D5</th>
<th>D4</th>
<th>D3</th>
<th>D2</th>
<th>D1</th>
<th>D0</th>
</tr>
</thead>
<tbody>
<tr>
<td>WR3</td>
<td>00</td>
<td>Rx 5 b/char</td>
<td>AUTO ENABLE</td>
<td>ENTER HUNT MODE</td>
<td>Rx CRC ENABLE</td>
<td>0</td>
<td>SYNC ENABLE</td>
</tr>
<tr>
<td></td>
<td>01</td>
<td>Rx 7 b/char</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>LOAD INHIBIT</td>
</tr>
<tr>
<td></td>
<td>10</td>
<td>Tx 6 b/char</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Rx</td>
</tr>
<tr>
<td></td>
<td>11</td>
<td>Rx 8 b/char</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>ENABLE</td>
</tr>
</tbody>
</table>

| WR4| 0  | 0   | 0     | 00  | 8 bit Sync | EVEN/ODD PARITY Enable |
|    |    | 0   | 01    | 16 bit Sync |            |         |
|    |    | 11  | Ext Sync |    |            |         |

| WR5| DTR| 00  | Tx ≤ 5 b/char| SEND BREAK| Tx 7 b/char| 01 | RTS |
|    |    | 01  | Tx 7 b/char  |           |              |    |
|    |    | 10  | Tx 6 b/char  |           |              |    |
|    |    | 11  | Tx 8 b/char  |           |              |    |

|     | SEND BREAK | Tx 7 b/char| 01 | RTS |
|     | Tx ENABLE  |            |    |

(SELECTS CRC-16)
rupt/DMA Pending Command (WR0; D5, D4, D3). If nothing more is written into the transmitter, there can be no further Transmit Buffer Empty interrupt, but this situation does cause a Transmit Underrun condition (RRO; D6).

Data Transfers using the RDY signal are for software controlled data transfers such as block moves. RDY tells the CPU that the MPSC is not ready to accept/provide data and that the CPU must extend the output/input cycle. DMA data transfers use the TxDREQ A/B signals which indicate that the transmit buffer is empty, and that the MPSC is ready to accept the next data character. If the data character is not loaded into the MPSC by the time the transmit shift register is empty, the MPSC enters the Transmit Underrun condition.

The MPSC has two programmable options for solving the transmit underrun condition: it can insert sync characters, or it can send the CRC characters generated so far, followed by sync characters. Following a chip or channel reset, the Transmit Underrun/EOM status bit (RRO; D6) is set in a set condition allowing the insertion of sync characters when there is no data to send. The CRC is not calculated on these automatically inserted sync characters. When the CPU detects the end message, a Reset Transmit Underrun/EOM command can be issued. This allows CRC to be sent when the transmitter has no data to send.

In the case of sync insertion, an interrupt is generated only after the first automatically inserted sync character has been loaded in the Transmit Shift Register. The status register indicates the Transmit Underrun/EOM bit and the Transmit Buffer Empty bit are set.

In the case of CRC insertion, the Transmit Underrun/EOM bit is set and the Transmit Buffer Empty bit is reset while CRC is being sent. When CRC has been completed sent, the Transmit Buffer Empty status bit is set and an interrupt is generated to indicate to the CPU that another message can begin (this interrupt occurs because CRC has been sent and sync has been loaded into the Tx Shift Register). If no more messages are to be sent, the program can terminate transmission by resetting RTS, and disabling the transmitter (WR5; D3).

**Bisync CRC Generation.** Setting the Transmit CRC enable bit (WR5; D0) indicates CRC accumulation when the program sends the first data character to the MPSC. Although the MPSC automatically transmits up to two sync characters (16 bit sync), it is wise to send a few more sync characters ahead of the message (before enabling Transmit CRC) to ensure synchronization at the receiving end.

The Transmit CRC Enable bit can be changed on the fly any time in the message to include or exclude a particular data character from CRC accumulation. The Transmit CRC Enable bit should be in the desired state when the data character is loaded into the transmit shift register. To ensure this bit in the proper state, the Transmit CRC Enable bit must be issued before sending the data character to the MPSC.

**Transmit Transparent Mode.** Transparent mode (Bisync protocol) operation is made possible by the ability to change Transmit CRC Enable on the fly and by the additional capability of inserting 16 bit sync characters. Exclusion of DLE characters from CRC calculation can be achieved by disabling CRC calculation immediately preceding the DLE character transfer to the MPSC.

In the transmit mode, the transmitter always sends the programmed number of sync bits (8 or 16) (WR4; D5, D4). When in the Monosync mode, the transmitter sends from WR6 and the receiver compares against WR7. One or two CRC polynomials, CRC 16 or SDLC, may be used with synchronous modes. In the transmit initialization process, the CRC generator is initialized by setting the Reset Transmit CRC Generator command (WR0; D7, D6).

The External/Status interrupt (WR1; D0) mode can be used to monitor the status of the CTS input as well as the Transmit Underrun/EOM latch. Optionally, the Auto Enable (WR3; D5) feature can be used to enable the transmitter when CTS is active. The first data transfer to the MPSC can begin when the External/Status interrupt (CTS (RRO; D5) status bit set) occurs following the Transmit Enable command (WR5; D3).

**Receive**

After a channel reset, the receiver is in the Hunt phase, during which the MPSC looks for character synchronization. The Hunt begins only when the receiver is enabled and data transfer begins only when character synchronization has been achieved. If character synchronization is lost, the hunt phase can be re-entered by writing the Enter Hunt Phase (WR3; D4) bit. The assembly of received data continues until the MPSC is reset or until the receiver is disabled (by command or by CD while in the Auto Enables mode) or until the CPU sets the Enter Hunt Phase bit. Under program control, all the leading sync characters of the message can be inhibited from loading the receive buffers by setting the Sync Character Load Inhibit (WR3; D1) bit. After character synchronization is achieved the assembled characters are transferred to the receive data FIFO. After
receiving the first data character, the Sync Character Load inhibit bit should be reset to zero so that all characters are received, including the sync characters. This is important because the received CRC may look like a sync character and not get received.

Data may be transferred with or without interrupts. Transferring data without interrupts is used for a purely polled operation or for off-line conditions. There are two interrupt modes available for data transfer: Interrupt on First Character Only and Interrupt on Every Character.

Interrupt on First Character Only mode is normally used to start a polling loop, a block transfer sequence using RDI to synchronize the CPU to the incoming data rate, or a DMA transfer using the RxDRQ signal. The MPSC interrupts on the first character and thereafter only interrupts after a Special Receive Condition is detected. This mode can be reinitialized using the Enable Interrupt On Next Receive Character (WR0; D5, D4, D3) command which allows the next character received to generate an interrupt. Parity Errors do not cause interrupts, but End of Frame (SDLC operation) and Receive Overrun do cause interrupts in this mode. If the external status interrupts (WR1; D0) are enabled an interrupt may be generated any time the CD changes state.

Interrupt On Every Character mode generates an interrupt whenever a character enters the receive buffer. Errors and Special Receive Conditions generate a special vector if the Status Affects Vector (WR1B; D2) is selected. Also the Parity Error may be programmed (WR1; D4, D3) to not generate the special vector while in the Interrupt On Every Character mode.

The Special Receive Condition interrupt can only occur while in the Receive Interrupt On First Character Only or the Interrupt On Every Receive Character modes. The Special Receive Condition interrupt is caused by the Receive Overrun (WR1; D5) error condition. The error status reflects an error in the current word in the receive buffer, in addition to any Parity or Overrun errors since the last Error Reset (WR0; D5, D4, D3). The Receive Overrun and Parity error status bits are latched and can only be reset by the Error Reset (WR0; D5, D4, D3) command.

The CRC check result may be obtained by checking for CRC bit (RR1; D6). This bit gives the valid CRC result 16 bit times after the second CRC byte has been read from the MPSC. After reading the second CRC byte, the user software must read two more characters (may be sync characters) before checking for CRC result in RR1. Also for proper CRC computation by the receiver, the user software must reset the Receive CRC Checker (WR0; D7, D6) after receiving the first valid data character. The receive CRC Enable bit (WR3; D3) may also be enabled at this time.

SYNCHRONOUS OPERATION—SDLC

General

Like the other synchronous operations the SDLC mode must be initialized with the following parameters: SDLC mode (WR4; D5, D4), SDLC polynomial (WR5; D2), Request to Send, Data Terminal Ready, transmit character length (WR5; D6, D5), interrupt modes (WR1; WR2), Transmit Enable (WR5; D3), Receive Enable (WR3; D0), Auto Enable (WR3; D5) and External/Status Interrupt (WR1; D0). WR4 parameters must be written before WR1, WR3, WR5, WR6 and WR7.

The Interrupt modes for SDLC operation are similar to those discussed previously in the synchronous operations section.

Synchronous Mode Register Setup—SDLC/HDLC

<table>
<thead>
<tr>
<th>D7</th>
<th>D6</th>
<th>D5</th>
<th>D4</th>
<th>D3</th>
<th>D2</th>
<th>D1</th>
<th>D0</th>
</tr>
</thead>
<tbody>
<tr>
<td>WR3</td>
<td>00</td>
<td>Rx 5 b/char</td>
<td>AUTO</td>
<td>ENTER</td>
<td>Rx</td>
<td>ADDRESS</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>01</td>
<td>7 b/char</td>
<td>ENABLES</td>
<td>HUNT</td>
<td>CRC</td>
<td>SEARCH</td>
<td></td>
</tr>
<tr>
<td></td>
<td>10</td>
<td>Rx 6 b/char</td>
<td></td>
<td>MODE</td>
<td>ENABLE</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>11</td>
<td>Rx 8 b/char</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>WR4</td>
<td>0</td>
<td></td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>WR5</td>
<td>DTR</td>
<td></td>
<td>SEND</td>
<td>Tx</td>
<td>0</td>
<td>RTS</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>00 Tx ≤ 5 b/char</td>
<td>BREAK</td>
<td>ENABLE</td>
<td>SELECTS</td>
<td>Tx</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>01 Tx 7 b/char</td>
<td></td>
<td></td>
<td>SDLC/HDLC</td>
<td>CRC</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>10 Tx 6 b/char</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>11 Tx 8 b/char</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Transmit

After a channel reset, the MPSC begins sending SDLC flags.

Following the flags in an SDLC operation the 8-bit address field, control field and information field may be sent to the MPSC by the microprocessor. The MPSC transmits the Frame Check Sequence using the Transmit Underrun feature. The MPSC automatically inserts a zero after every sequence of 5 consecutive 1's except when transmitting Flags or Aborts.

SDLC—like protocols do not have provision for fill characters within a message. The MPSC therefore automatically terminates an SDLC frame when the transmit data buffer and output shift register have no more bits to send. It does this by sending the two bytes of CRC and then one or more flags. This allows very high-speed transmissions under OMA or CPU control without requiring the CPU to respond quickly to the end-of-message situation.

After a reset, the Transmit Underrun/EOM status bit is in the set state and prevents the insertion of CRC characters during the time there is no data to send. Flag characters are sent. The MPSC begins to send the frame when data is written into the transmit buffer. Between the time the first data byte is written, and the end of the message, the Reset Transmit Underrun/EOM (WR0; D7, D6) command must be issued. The Transmit Underrun/EOM status bit (RR0; D6) is in the reset state at the end of the message which automatically sends the CRC characters.

The MPSC can be programmed to receive all frames or it can be programmed to the Address Search Mode. In the Address Search Mode, only frames with addresses that match the value in WR6 or the global address (0FFH) are received by the MPSC. Extended address recognition must be done by the microprocessor software.

The control and information fields are received as data.

SDLC/HDLC CRC calculation does not have an 8-bit delay, since all characters are included in the calculation, unlike Byte Synchronous Protocols.

Reception of an abort sequence (7 or more 1's) will cause the Break/Abort bit (RR0; D7) to be set and will cause an External/Status interrupt, if enabled. After the Reset External/Status Interrupts Command has been issued, a second interrupt will occur at the end of the abort sequence.

Receive

After initialization, the MPSC enters the Hunt phase, and remains in the Hunt phase until the first Flag is received. The MPSC never again enters the Hunt phase unless the microprocessor writes the Enter Hunt command. The MPSC will also detect flags separated by a single zero. For example, the bit pattern 01111110111110 will be detected as two flags.

MPSC

Detailed Command/Status Description

GENERAL

The MPSC supports an extremely flexible set of serial and system interface modes.

The system interface to the CPU consists of 8 ports or buffers:

<table>
<thead>
<tr>
<th>CS</th>
<th>A1</th>
<th>A0</th>
<th>Read Operation</th>
<th>Write Operation</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>Ch. A Data Read</td>
<td>Ch. A Data Write</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>Ch. A Status Read</td>
<td>Ch. A Command/Parameter</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>Ch. B Data Read</td>
<td>Ch. B Data Write</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>Ch. B Status Read</td>
<td>Ch. B Command/Parameter</td>
</tr>
<tr>
<td>1</td>
<td>X</td>
<td>X</td>
<td>High Impedance</td>
<td>High Impedance</td>
</tr>
</tbody>
</table>

Data buffers are addressed by A1 = 0, and Command ports are addressed by A1 = 1.

COMMAND/STATUS DESCRIPTION

The following command and status bytes are used during initialization and execution phases of operation. All Command/Status operations on the two channels are identical, and independent, except where noted.
Detailed Register Description

Write Register 0 (WR0):

<table>
<thead>
<tr>
<th>Command/Status Pointer</th>
<th>Register Pointer</th>
</tr>
</thead>
<tbody>
<tr>
<td>D7 D6 D5 D4 D3 D2 D1 D0</td>
<td>COMMAND/STATUS POINTERS</td>
</tr>
</tbody>
</table>

WR0

D2, D1, D0—Command/Status Register Pointer bits determine which write-register the next byte is to be written into, or which read-register the next byte is to be read from. After reset, the first byte written into either channel goes into WR0. Following a read or write to any register (except WR0) the pointer will point to WR0.

D5, D4, D3—Command bits determine which of the basic seven commands are to be performed.

Command 0 Null—has no effect.
Command 1 Send Abort—causes the generation of eight to thirteen 1's when in the SDLC mode.

Command 2 Reset External/Status Interrupts—resets the latched status bits of RR0 and re-enables them, allowing interrupts to occur again.

Command 3 Channel Reset—resets the Latched Status bits of RR0, the interrupt prioritization logic and all control registers for the channel. Four extra system clock cycles should be allowed for MPSC reset time before any additional commands or controls are written into the channel.

Command 4 Enable Interrupt on Next Receive Character—if the Interrupt on First Receive Character mode is selected, this command reactivates that mode after each complete message is received to prepare the MPSC for the next message.

Command 5 Reset Transmitter Interrupt/DMA Pending—if The Transmit Interrupt/DMA Enable mode is selected, the MPSC automatically interrupts or requests DMA data transfer when the transmit buffer becomes empty. When there are no more characters to be sent, issuing this command prevents further transmitter interrupts or DMA requests until the next character has been completely sent.

Command 6 Error Reset—error latches, Parity and Overrun errors in RR1 are reset.

Command 7 End of Interrupt—resets the interrupt-in-service latch of the highest-priority internal device under service.

D7, D6 CRC Reset Code.
00 Null—has no effect.
01 Reset Receive CRC Checker—resets the CRC checker to 0's. If in SDLC mode the CRC checker is initialized to all 1's.
10 Reset Transmit CRC Generator—resets the CRC generator to 0's. If in SDLC mode the CRC generator is initialized to all 1's.
11 Reset Tx Underrun/End of Message Latch.
Write Register 1 (WR1):

<table>
<thead>
<tr>
<th>D7</th>
<th>D6</th>
<th>D5</th>
<th>D4</th>
<th>D3</th>
<th>D2</th>
<th>D1</th>
<th>D0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

External/Status Interrupt Enable—allows interrupt to occur as the result of transitions on the CD, CTS or SYNDET inputs. Also allows interrupts as the result of a Break/Abort detection and termination, or at the beginning of CRC, or sync character transmission when the Transmit Underrun/EOM latch becomes set.

Transmitter Interrupt/DMA Enable—allows the MPSC to interrupt or request a DMA transfer when the transmitter buffer becomes empty.

Status Affects vector—(WR1, D2 active in channel B only.) If this bit is not set, then the fixed vector, programmed in WR2, is returned from an interrupt acknowledge sequence. If the bit is set then the vector returned from an interrupt acknowledge is variable as shown in the Interrupt Vector Table.

D4, D3 Receive Interrupt Mode.
0 0 Receive Interrupts/DMA Disabled.
0 1 Receive Interrupt on First Character Only or Special Condition.
1 0 Interrupt on All Receive Characters or Special Condition (Parity Error is a Special Receive Condition).
1 1 Interrupt on All Receive Characters or Special Condition (Parity Error is not a Special Receive Condition).

Wait on Receive/Transmit—when the following conditions are met the RDY pin is activated, otherwise it is held in the High-Z state. (Conditions: Interrupt Enabled Mode, Wait Enabled, CS = 0, A0 = 0/1, and A1 = 0). The RDY pin is pulled low when the transmitter buffer is full or the receiver buffer is empty and it is driven High when the transmitter buffer is empty or the receiver buffer is full. The RDYA and RDYB may be wired OR connected since only one signal is active at any one time while the other is in the High Z state.

Wait Enable—enables the wait function.

WR2 Channel A Only

Channel A and Channel B both use interrupts.
Channel A uses DMA, Channel B uses interrupts.
Channel A and Channel B both use DMA.

Priority—this bit specifies the relative priorities of the internal MPSC interrupt/DMA sources.

(Highest) RxA, TxA, RxB, TxB, ExTA, ExTB (Lowest).

Interrupt Code—specifies the behavior of the MPSC when it receives an interrupt acknowledge sequence from the CPU. (See Interrupt Vector Mode Table.)
0 X X Non-vectored interrupts—intended for use with external DMA CONTROLLER. The Data Bus remains in a high impedance state during INTA sequences.

1 0 0 8085 Vector Mode 1—intended for use as the primary MPSC in a daisy chained priority structure. (See System Interface section).

1 0 1 8085 Vector Mode 2—intended for use as any secondary MPSC in a daisy chained priority structure. (See System Interface section).

1 1 0 8086/88 Vector Mode—intended for use as either a primary or secondary in a daisy chained priority structure. (See System Interface section).

Must be zero.

zero Pin 10 = RTSB

one Pin 10 = SYNDRET_B

Write Register 2 (WR2): Channel A Only

NOTE:
*External Status Interrupt only if EXT Interrupt Enable (WR1; D0) is set.
The following table describes the MPSC’s response to an interrupt acknowledge sequence:

<table>
<thead>
<tr>
<th>DS</th>
<th>D4</th>
<th>D3</th>
<th>IP</th>
<th>MODE</th>
<th>INTA</th>
<th>Data Bus</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>Non-vectored</td>
<td>Any INTA</td>
<td>D7</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1st INTA</td>
<td>D0</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>85 Mode 1</td>
<td>2nd INTA</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>85 Mode 1</td>
<td>3rd INTA</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>85 Mode 1</td>
<td>1st INTA</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>1</td>
<td>85 Mode 1</td>
<td>2nd INTA</td>
<td></td>
<td>High Impedance</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1</td>
<td>85 Mode 1</td>
<td>3rd INTA</td>
<td></td>
<td>High Impedance</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>86 Mode</td>
<td>1st INTA</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>86 Mode</td>
<td>2nd INTA</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>86 Mode</td>
<td>3rd INTA</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>85 Mode</td>
<td>1st INTA</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>85 Mode</td>
<td>2nd INTA</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>85 Mode</td>
<td>3rd INTA</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>85 Mode</td>
<td>1st INTA</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>85 Mode</td>
<td>2nd INTA</td>
<td></td>
</tr>
</tbody>
</table>

**NOTE:**
*These bits are variable if the "status affects vector" mode has been programmed, (WR1B, D2).

### Interrupt/DMA Mode, Pin Functions, and Priority

<table>
<thead>
<tr>
<th>Ch. A</th>
<th>WR2</th>
<th>Int/DMA Mode</th>
<th>Pin Functions</th>
<th>Priority</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>D2</td>
<td>D1</td>
<td>D0</td>
<td>CH. A</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>INT</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>INT</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>DMA</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>DMA</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>DMA</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>DMA</td>
</tr>
</tbody>
</table>

**NOTE:**
1. Special Receive Condition
Interrupt Vector Mode Table

<table>
<thead>
<tr>
<th>8085 Modes</th>
<th>V4</th>
<th>V3</th>
<th>V2</th>
<th>V1</th>
<th>V0</th>
<th>Channel</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>8086/88 Mode</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Tx Buffer Empty</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
<td>B</td>
<td>Ext/Status Change</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>0</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td>Rx Char. Available</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>1</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td>Special Rx Condition (Note 1)</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
<td>A</td>
<td>Tx Buffer Empty</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>0</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td>Ext/Status Change</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>1</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td>Rx Char. Available</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td>Special Rx Condition (Note 1)</td>
</tr>
</tbody>
</table>

**NOTE:**
1. Special Receive Condition = Parity Error, Rx Overrun Error, Framing Error, End of Frame (SDLC).

Write Register 2 (WR2): Channel B

**WR2 CHANNEL B**
D7–D0 Interrupt vector—This register contains the value of the interrupt vector placed on the data bus during interrupt acknowledge sequences.

Write Register 3 (WR3):

0 0 Rx 5 BITS/CHAR
0 1 Rx 7 BITS/CHAR
1 0 Rx 6 BITS/CHAR
1 1 Rx 8 BITS/CHAR
Receiver Enable—A one enables the receiver to begin. This bit should be set only after the receiver has been initialized.

Sync Character Load Inhibit—A one prevents the receiver from loading sync characters into the receive buffers. In SDLC, this bit must be zero.

Address Search Mode—If the SDLC mode has been selected, the MPSC will receive all frames unless this bit is a 1. If this bit is a 1, the MPSC will receive only frames with address (0FFH) or the value loaded into WR6. This bit must be zero in non-SDLC modes.

Receive CRC Enable—A one in this bit enables (or re-enables) CRC calculation. CRC calculation starts with the last character placed in the Receiver FIFO. A zero in this bit disables, but does not reset, the Receiver CRC generator.

Enter Hunt Phase—After initialization, the MPSC automatically enters the Hunt mode. If synchronization is lost, the Hunt phase can be re-entered by writing a one to this bit.

Auto Enable—A one written to this bit causes CD to be automatic enable signal for the receiver and CTS to be an automatic enable signal for the transmitter. A zero written to this bit limits the effect of CD and CTS signals to setting/resetting their corresponding bits in the status register (RRI).

Receive Character length

0 0  Receive 5 Data bits/character
0 1  Receive 7 Data bits/character
1 0  Receive 8 Data bits/character
1 1  Receive 8 Data bits/character

Parity—A one in this bit causes a parity bit to be added to the programmed number of data bits per character for both the transmitted and received character. If the MPSC is programmed to receive 8 bits per character, the parity bit is not transferred to the microprocessor. With other receiver character lengths, the parity bit is transferred to the microprocessor.

Even/Odd Parity—If parity is enabled, a one in this bit causes the MPSC to transmit and expect even parity, and a zero causes it to send and expect odd parity.

Stop bits/sync mode

Selects synchronous modes

Async mode, 1 stop bit/character

Async mode, 1½ stop bits/character

Async mode, 2 stop bits/character

Sync mode select

8-bit sync character

16-bit sync character

SDLC mode (Flag sync)

External sync mode

Clock Mode—Selects the clock/data rate multiplier for both the receiver and the transmitter. 1x mode must be selected for synchronous modes. If the 1x mode is selected, bit synchronization must be done externally.
0 0 1 0 1 1

0 0 Clock rate = Data rate × 1
0 1 Clock rate = Data rate × 16
1 0 Clock rate = Data rate × 32
1 1 Clock rate = Data rate × 64

Write Register 5 (WR5):

<table>
<thead>
<tr>
<th>D7</th>
<th>D6</th>
<th>D5</th>
<th>D4</th>
<th>D3</th>
<th>D2</th>
<th>D1</th>
<th>D0</th>
</tr>
</thead>
<tbody>
<tr>
<td>LSB</td>
<td>MSB</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D7</td>
<td>D6</td>
<td>D5</td>
<td>D4</td>
<td>D3</td>
<td>D2</td>
<td>D1</td>
<td>D0</td>
</tr>
</tbody>
</table>

- **D7** Transmit CRC Enable-A one in this bit enables the transmitter CRC generator. The CRC calculation is done when a character is moved from the transmit buffer into the shift register. A zero in this bit disables CRC calculations. If this bit is not set when a transmitter underrun occurs, the CRC will not be sent.
- **D6, D5** Transmit Character length
  - 0 0 Transmit 1–5 bits/character
  - 0 1 Transmit 7 bits/character
  - 1 0 Transmit 6 bits/character
  - 1 1 Transmit 8 bits/character

Five or less mode allows transmission of one to five bits per character. The microprocessor must format the data in the following way:

<table>
<thead>
<tr>
<th>D7</th>
<th>D6</th>
<th>D5</th>
<th>D4</th>
<th>D3</th>
<th>D2</th>
<th>D1</th>
<th>D0</th>
</tr>
</thead>
</table>
| 1 1 1 1 0 0 0 B0 | Sends one data bit
| 1 1 1 1 0 0 B1 B0 | Sends two data bits
| 1 1 0 0 0 B2 B1 B0 | Sends three data bits
| 1 0 0 0 B3 B2 B1 B0 | Sends four data bits
| 0 0 0 B4 B3 B2 B1 B0 | Sends five data bits

- **D1** Request to Send-A one in this bit forces the RTS pin active (low) and zero in this bit forces the RTS pin inactive (high).

- **D2** CRC Select-A one in this bit selects the CRC-16 polynomial (X^16 + X^15 + X^2 + 1) and a zero in this bit selects the CCITT-CRC polynomial (X^16 + X^12 + X^5 + 1). In SDLC mode, CCITT-CRC must be selected.

- **D3** Transmitter Enable-A zero in this bit forces a marking state on the transmitter output. If this bit is set to zero during data or sync character transmission, the marking state is entered after the character has been sent. If this bit is set to zero during transmission of a CRC character, sync or flag bits are substituted for the remainder of the CRC bits.

- **D4** Send Break-A one in this bit forces the transmit data low. A zero in this bit allows normal transmitter operation.

- **D6, D5** Transmit Character length
  - 0 0 Transmit 1–5 bits/character
  - 0 1 Transmit 7 bits/character
  - 1 0 Transmit 6 bits/character
  - 1 1 Transmit 8 bits/character

- **D7** Data Terminal Ready-When set, this bit forces the DTR pin active (low). When reset, this bit forces the DTR pin inactive (high).
Write Register 6 (WR6):

WR6
D7–D0
Sync/Address—This register contains the transmit sync character in Monosync mode, the low order 8 sync bits in Bisync mode, or the Address byte in SDLC mode.

WR7
D7–D0
Sync/Flag—This register contains the receive sync character in Monosync mode, the high order 8 sync bits in Bisync mode, or the Flag character (01111110) in SDLC mode. WR7 is not used in External Sync mode.

Write Register 7 (WR7):

RR0
D0
Receive Character Available—This bit is set when the receive FIFO contains data and is reset when the FIFO is empty.

D1
Interrupt In-Service*—If an Internal Interrupt is pending, this bit is set at the falling edge of the second INTA pulse of an INTA cycle. In non-vectoring mode, this bit is set at the falling edge of RD after pointer 2 is specified. This bit is reset when an EOI command is issued and there are no other interrupts in-service at that time.

D2
Transmit Buffer Empty—This bit is set whenever the transmit buffer is

*This bit is only valid when IPI is active low and is always zero in Channel B.
empty except when CRC characters are being sent in a synchronous mode. This bit is reset when the transmit buffer is loaded. This bit is set after an MPSC reset.

**D3**

Carrier Detect—This bit contains the state of the CD pin at the time of the last change of any of the External/Status bits (CD, CTS, Sync/Hunt, Break/Abort, or Tx Underrun/EOM). Any change of state of the CD pin causes the CD bit to be latched and causes an External/Status interrupt. This bit indicates current state of the CD pin immediately following a Reset External/Status Interrupt command.

**D4**

Sync/Hunt—In asynchronous modes, the operation of this bit is similar to the CD status bit, except that Sync/Hunt shows the state of the SYNDET input. Any High-to-Low transition on the SYNDET pin sets this bit, and causes an External/Status interrupt (if enabled). The Reset External/Status Interrupt command is issued to clear the interrupt. A Low-to-High transition clears this bit and sets the External/Status interrupt. When the External/Status interrupt is set by the change in state of any other input or condition, this bit shows the inverted state of the SYNDET pin at time of the change. This bit must be read immediately following a Reset External/Status Interrupt command to read the current state of the SYNDET input.

In the External Sync mode, the Sync/Hunt bit operates in a fashion similar to the Asynchronous mode, except the Enter Hunt Mode control bit enables the external sync detection logic. When the External Sync Mode and Enter Hunt Mode bits are set (for example, when the receiver is enabled following a reset), the SYNDET input must be held High by the external logic until external character synchronization is achieved. A High at the SYNDET input holds the Sync/Hunt status in the reset condition.

When external synchronization is achieved, SYNDET must be driven Low on the second rising edge of RxC after the rising edge of RxC on which the last bit of the sync character was received. In other words, after the sync pattern is detected, the external logic must wait for two full Receive Clock cycles to activate the SYNDET input. Once SYNDET is forced Low, it is good practice to keep it Low until the CPU informs the external sync logic that synchronization has been lost or a new message is about to start. The High-to-Low transition of the SYNDET output sets the Sync/Hunt bit, which sets the External/Status interrupt. The CPU must clear the interrupt by issuing the Reset External/Status Interrupt Command.

When the SYNDET input goes High again, another External/Status interrupt is generated that must also be cleared. The Enter Hunt Mode control bit is set whenever character synchronization is lost or the end of message is detected. In this case, the MPSC again looks for a High-to-Low transition on the SYNDET input and the operation repeats as explained previously. This implies the CPU should also inform the external logic that character synchronization has been lost and that the MPSC is waiting for SYNDET to become active.

In the Monosync and Bisync Receive modes, the Sync/Hunt status bit is initially set to 1 by the Enter Hunt Mode bit. The Sync/Hunt bit is reset when the MPSC establishes character synchronization. The High-to-Low transition of the Sync/Hunt bit causes an External/Status interrupt that must be cleared by the CPU issuing the Reset External/Status Interrupt command. This enables the MPSC to detect the next transition of other External/Status bits.

When the CPU detects the end of message or that character synchronization is lost, it sets the Enter Hunt Mode control bit, which sets the Sync/Hunt bit to 1. The Low-to-High transition of the Sync/Hunt bit sets the External/Status Interrupt, which must also be cleared by the Reset External/Status Interrupt Command. Note that the SYNDET pin acts as an output in this mode, and goes low every time a sync pattern is detected in the data stream.
In the SDLC mode, the Sync/Hunt bit is initially set by the Enter Hunt mode bit, or when the receiver is disabled. In any case, it is reset to 0 when the opening flag of the first frame is detected by the MPSC. The External/Status interrupt is also generated, and should be handled as discussed previously.

Unlike the Monosync and Bisync modes, once the Sync/Hunt bit is reset in the SDLC mode, it does not need to be set when the end of message is detected. The MPSC automatically maintains synchronization. The only way the Sync/Hunt bit can be set again is by the Enter Hunt Mode bit, or by disabling the receiver.

Clear to Send—This bit contains the inverted state of the CTS pin at the time of the last change of any of the External/Status bits (CD, CTS, Sync/Hunt, Break/Abort, or Tx Underrun/EOM). Any change of state of the CTS pin causes the CTS bit to be latched and causes an External/Status interrupt. This bit indicates the inverse of the current state of the CTS pin immediately following a Reset External/Status Interrupt command.

Transmitter Underrun/End of Message—This bit is set when a reset (internal or external). The only command that can reset this bit is the Reset Transmit Underrun/EOM Latch command (WR0, D9 and D7). When the Transmit Underrun condition occurs, this bit is set, which causes the External/Status interrupt which must be reset by issuing a Reset External/Status command (WR0; command 2).

Break/Abort—in the Asynchronous Receive mode, this bit is set when a Break sequence (null character plus framing error) is detected in the data stream. The External/Status interrupt, if enabled, is set when break is detected. The interrupt service routine must issue the Reset External/Status Interrupt command (WR0, Command 2) to the break detection logic so the Break sequence termination can be recognized.

The Break/Abort bit is reset when the termination of the Break sequence is detected in the incoming data stream. The termination of the Break sequence also causes the External/Status interrupt to be set. The Reset External/Status Interrupt command must be issued to enable the break detection logic to look for the next Break sequence. A single extraneous null character is present in the receiver after the termination of a break; it should be read and discarded.

In the SDLC Receive mode, this status bit is set by the detection of an Abort sequence (seven or more 1's). The External/Status interrupt is handled the same way as in the case of a Break. The Break/Abort bit is not used in the Synchronous Receive mode.

All Sent—This bit is set when all characters have been sent, in asynchronous modes. It is reset when characters are in the transmitter, in asynchronous modes. In synchronous modes, this bit is always set.

Residue Codes—Bit synchronous protocols allow I-fields that are not an integral number of characters. Since transfers from the MPSC to the CPU are character oriented, the residue codes provide the capability of receiving leftover bits. Residue bits are right justified in the last data byte received or first CRC byte.

Parity Error—if parity is enabled, this bit is set for received characters whose parity does not match the programmed sense (Even/Odd). This bit is latched. Once an error occurs, it remains set until the Error Reset command is written.

Receive Overrun Error—This bit indicates that the receive FIFO has been overloaded by the receiver. The last character in the FIFO is overwritten and flagged with this error. Once the overwritten character is read, this error condition is latched until reset by the Error Reset command. If the MPSC is in the status affects vector mode, the overrun causes a special Receive Condition Vector.

CRC/Framing Error—in async modes, a one in this bit indicates a receive framing error. In synchronous modes, a one in this bit indicates that the calculated CRC value does not match the last two bytes received. It can be reset by issuing an Error Reset command.
## SDLC Residue Code Table (1 Field Bits in 2 Previous Bytes)

<table>
<thead>
<tr>
<th>RR1</th>
<th>8 bits/char</th>
<th>7 bits/char</th>
<th>6 bits/char</th>
<th>5 bits/char</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>First CRC Byte</td>
<td>Last Data Byte</td>
<td>First CRC Byte</td>
<td>Last Data Byte</td>
</tr>
<tr>
<td>D3</td>
<td>D2</td>
<td>D1</td>
<td></td>
<td>D3</td>
</tr>
<tr>
<td>1 0 0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0 1 0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1 1 0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0 0 1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1 0 1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0 1 1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1 1 1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0 0 0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

### Read Register 1 (RR1): (Special Receive Condition Mode)

**MSB**

- D7
- D6
- D5
- D4
- D3
- D2
- D1
- D0

**LSB**

- ALL SENT

**FIRST CRC BYTE**

- 0 0 0
- 0 0 1
- 0 1 0
- 0 1 1
- 1 0 0
- 1 0 1
- 1 1 0
- 1 1 1

**LAST DATA BYTE**

- 0
- 1
- 2
- 3
- 4
- 5
- 6
- 7

**RESIDUE DATA 8 BITS/CHAR. MODE**

- 8

**PARITY ERROR**

**Rx OVERRUN ERROR**

**CRC/FRAMING ERROR**

**END OF FRAME (SDLC/HDLC MODE)**

170102-14
End of Frame—This bit is valid only in SDLC mode. A one indicates that a valid ending flag has been received. This bit is reset either by an Error Reset command or upon reception of the first character of the next frame.

**SYSTEM INTERFACE**

**General**

The MPSC to Microprocessor System interface can be configured in many flexible ways. The basic interface types are polled, wait, interrupt driven, or direct memory access driven.

Polled operation is accomplished by repetitively reading the status of the MPSC, and making decisions based on that status. The MPSC can be polled at any time.

Wait operation allows slightly faster data throughput for the MPSC by manipulating the Ready input to the microprocessor. Block Read or Write Operations to the MPSC are started at will by the microprocessor and the MPSC deactivates its RDY signal if it is not yet ready to transmit the new byte, or if reception of new byte is not completed.

Interrupt driven operation is accomplished via an internal or external interrupt controller. When the MPSC requires service, it sends an interrupt request signal to the microprocessor, which responds with an interrupt acknowledge signal. When the internal or external interrupt controller receives the acknowledge, it vectors the microprocessor to a service routine, in which the transaction occurs.

DMA operation is accomplished via an external DMA controller. When the MPSC needs a data transfer, it requests a DMA cycle from the DMA controller. The DMA controller then takes control of the bus and simultaneously does a read from the MPSC and a write to memory or vice-versa.

The following section describes the many configurations of these basic types of system interface techniques for both serial channels.

**POLLED OPERATION**

In the polled mode, the CPU must monitor the desired conditions within the MPSC by reading the appropriate bits in the read registers. All data available, status, and error conditions are represented by the appropriate bits in read registers 0 and 1 for channels A and B.

There are two ways in which the software task of monitoring the status of the MPSC has been reduced. One is the "ORing" of all conditions into the Interrupt Pending bit (RR0; D1 channel A only). This bit is set when the MPSC requires service, allowing the CPU to monitor one bit instead of four status registers. The other is available when the "status-affects-vector" mode is selected. By reading RR2 Channel B, the CPU can read a vector who’s value will indicate that one or more of group of conditions has occurred, narrowing the field of possible conditions. See WR2 and RR2 in the Detailed Command Description section.

**WAIT OPERATION**

Wait Operation is intended to facilitate data transmission or reception using block move operations. If a block of data is to be transmitted, for example, the CPU can execute a String I/O instruction to the MPSC. After writing the first byte, the CPU will attempt to write a second byte immediately as is the case of block move. The MPSC forces the RDY signal low which inserts wait states in the CPU’s write cycle until the transmit buffer is ready to accept a new byte. At that time, the RDY signal is high allowing the CPU to finish the write cycle. The CPU then attempts the third write and the process is repeated.

Similar operation can programmed for the receiver. During initialization, wait on transmit (WR1; D5 = 0)
Software Flow, Polled Operation

![Flow Chart]

**NOTES:**
1. RRO; D0 is reset automatically when the data is read.
2. RRO; D2 is reset automatically when the data is written.

or wait on receive (WR1; D5 = 1) can be selected. The wait operation can be enabled/disabled by setting/resetting the Wait Enable Bit (WR1; D7).

**NOTE:**
CAUTION: ANY CONDITION THAT CAN CAUSE THE TRANSMITTER TO STOP (E.G., CTS GOES INACTIVE) OR THE RECEIVER TO STOP (E.G., RX DATA STOPS) WILL CAUSE THE MPSC TO HANG THE CPU UP IN WAIT STATES UNTIL RESET. EXTREME CARE SHOULD BE TAKEN WHEN USING THIS FEATURE.

Interrupt Driven Operation

The MPSC can be programmed into several interrupt modes: Non-Vectored, 8085 vectored, and 8088/86 vectored. In both vectored modes, multiple MPSC's can be daisy-chained.

In the vectored mode, the MPSC responds to an interrupt acknowledge sequence by placing a call instruction (8085 mode) and interrupt vector (8085 and 8088/86 mode) on the data bus.

The MPSC can be programmed to cause an interrupt due to up to 14 conditions in each channel. The status of these interrupt conditions is contained in Read Registers 0 and 1. These 14 conditions are all directed to cause 3 different types of internal interrupt request for each channel: receive/interrupts, transmit interrupts and external/status interrupts (if enabled).

This results in up to 6 internal interrupt request signals. The priority of those signals can be programmed to one of two fixed modes:

<table>
<thead>
<tr>
<th>Highest Priority</th>
<th>Lowest Priority</th>
</tr>
</thead>
<tbody>
<tr>
<td>RxA RxB TxA TxB ExTA ExTB</td>
<td>RxA TxA RxB TxB ExTA ExTB</td>
</tr>
</tbody>
</table>

The interrupt priority resolution works differently for vectored and non-vectored modes.

Hardware Configuration, Polled Operation

![Hardware Diagram]
## Interrupt Condition Grouping

<table>
<thead>
<tr>
<th>Condition</th>
<th>Mode</th>
<th>Internal Interrupt Request</th>
</tr>
</thead>
<tbody>
<tr>
<td>Receive Character</td>
<td>Special Receive Condition Interrupt</td>
<td>Interrupt on all receive characters</td>
</tr>
<tr>
<td>Parity Error</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Receive Overflow Error</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Framing Error</td>
<td></td>
<td></td>
</tr>
<tr>
<td>End of Frame (SDLC Only)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>First Data Character</td>
<td></td>
<td></td>
</tr>
<tr>
<td>First Non-Sync Character (Sync Modes)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Valid Address Byte (SDLC Only)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CD Transition</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CTS Transition</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sync Transition</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TX Underrun/Abend</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Break/Abort Detect</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Transmit Buffer Empty</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### PRIORITY RESOLUTION: VECTORED MODE

Any interrupt condition can be accepted internally to the MPSC at any time, unless the MPSC's internal INTA signal is active, unless a higher priority interrupt is currently accepted, or if IPF is inactive (high). The MPSC's internal INTA is set on the leading (falling) edge of the first External INTA pulse and reset on the trailing (rising) edge of the second External INTA pulse. After an interrupt is accepted internally, and External INT request is generated and the IPF goes inactive. IPF and IPF are used for daisy-chaining MPSC's together.
The MPSC's internal INTA is set on the leading (falling) edge of the first external INTA pulse, and reset on the trailing (rising) edge of the second external INTA pulse. After an interrupt is accepted internally, and external INT request is generated and IPO goes inactive (high). IPO and IPI are used for daisy-chaining MPSC's together.

Each of the six interrupt sources has an associated In-Service latch. After priority has been resolved, the highest priority In-Service latch is set. After the In-Service latch is set, the INT pin goes inactive (high).

**NOTE:**
If the External INT pin is active and the IPI signal is pulled inactive high, the INT signal will also go inactive. IPI qualifies the External INT Signal.
Lower priority interrupts are not accepted internally while the In-Service latch is set. However, higher priority interrupts are accepted internally and a new external INT request is generated. If the CPU responds with a new INTA sequence, the MPSC will respond as before, suspending the lower priority interrupt.

After the interrupt is serviced, the End-of-Interrupt (EOI) command should be written to the MPSC. This command will cause an internal pulse that is used to reset the In-Service Latch which allows service for lower priority interrupts in the daisy-chain to resume, provided a new INTA sequence does not start for a higher priority interrupt (higher than the highest under service). If there is no interrupt pending internally, the I/O follows IPI.
Non-Vectored Interrupt Timing

PRIORITY RESOLUTION: NON-VECTORED MODE

In non-vectored mode, the MPSC does not respond to interrupt acknowledge sequences. The INTA input (pin 27) must be pulled high for proper operation. The MPSC should be programmed to the Status-Affects-Vector mode, and the CPU should read RR2 (Ch. B) in its service routine to determine which interrupt requires service.

In this case, the internal pointer being set to RR2 provides the same function as the internal INTA signal in the vectored mode. It inhibits acceptance of any additional internal interrupts and its leading edge starts the interrupt priority resolution circuit. The interrupt priority resolution is ended by the leading edge of the read signal used by the CPU to retrieve the modified vector. The leading edge of read sets the In-Service latch and forces the external INT output inactive (high). The internal pointer is reset to zero after the trailing edge of the read pulse.

NOTE:
That if RR2 is specified but not read, no internal interrupts, regardless of priority, are accepted.
DAISY CHAINING MPSC

In the vectored interrupt mode, multiple MPSC's can be daisy-chained on the same INT, INTA signals. These signals, in conjunction with the IPI and IPO allow a daisy-chain-like interrupt resolution scheme. This scheme can be configured for either 8085 or 8086/88 based system.

In either mode, the same hardware configuration is called for. The INT request lines are wire-OR'ed together at the input of a TTL inverter which drives the INT pin of the CPU. The INTA signal from the CPU drives all of the daisy-chained MPSC's.

The MPSC drives IPO (Interrupt Priority Output) inactive (high) if IPI (Interrupt Priority Input) is inactive (high), or if the MPSC has an interrupt pending.

The IPO of the highest priority MPSC is connected to the IPI of the next highest priority MPSC, and so on.

If IPI is active (low), the MPSC knows that all higher priority MPSC's have no interrupts pending. The IPI pin of the highest priority MPSC is strapped active (low) to ensure that it always has priority over the rest.

MPSC's Daisy-chained on an 8088/86 CPU should be programmed to the 8088/86 Interrupt mode (WR2; D4, D3 Ch. A). MPSC's Daisy-chained on an 8085 CPU should be programmed to 8085 interrupt mode 1 if it is the highest priority MPSC. In this mode, the highest priority MPSC issues the CALL instruction during the first INTA cycle, and the interrupting MPSC provides the interrupt vector during the following INTA cycles. Lower priority MPSC's should be programmed to 8085 interrupt mode 2.

MPSC's used alone in 8085 systems should be programmed to 8085 mode 1 interrupt operation.
DMA Acknowledge Circuit

Each MPSC can be programmed to utilize up to four DMA channels: Transmit Channel A, Receive Channel A, Transmit Channel B, Receive Channel B. Each DMA Channel has an associated DMA Request line. Acknowledgement of a DMA cycle is done via normal data read or write cycles. This is accomplished by encoding the DACK signals to generate A₀, A₁, and CS, and multiplexing them with the normal A₀, A₁, and CS signals.

PERMUTATIONS

Channels A and B can be used with different system interface modes. In all cases it is possible to poll the MPSC. The following table shows the possible permutations of interrupt, wait, and DMA modes for channels A and B. Bits D₁, D₀ of WR2 Ch. A determine these permutations.

<table>
<thead>
<tr>
<th>Permutation</th>
<th>Channel A</th>
<th>Channel B</th>
</tr>
</thead>
<tbody>
<tr>
<td>WR2 Ch. A</td>
<td>D₁</td>
<td>D₀</td>
</tr>
<tr>
<td>0 0</td>
<td>Wait</td>
<td>Wait</td>
</tr>
<tr>
<td>0 1</td>
<td>DMA Polled</td>
<td>Interrupt Poll</td>
</tr>
<tr>
<td>1 0</td>
<td>DMA Polled</td>
<td>DMA Polled</td>
</tr>
</tbody>
</table>

NOTE:
D₁, D₀ = 1, 1 is illegal.
The circuit was not designed based on a worst-case timing analysis. Specific implementations should include this timing analysis.
PROGRAMMING HINTS

This section will describe some useful programming hints which may be useful in program development.

Asynchronous Operation

At the end of transmission, the CPU must issue "Reset Transmit Interrupt/DMA Pending" command in WR0 to reset the last transmit empty request which was not satisfied. Failing to do so will result in the MPSC locking up in a transmit empty state forever.

Non-Vectored Mode

In non-vectored mode, the Interrupt Acknowledge pin (INTA) on the MPSC must be tied high through a pull-up resistor. Failing to do so will result in unpredictable response from the 8274.

HDLC/SDLC Mode

When receiving data in SDLC mode, the CRC bytes must be read by the CPU (or DMA controller) just like any other data field. Failing to do so will result in receiver buffer overflow. The CRC bytes are not to be used for CRC verification. Residue bits may be contained in the first CRC byte. Also, the End of Frame Interrupt indicates that the entire frame has been received. At this point, the CRC result (RR1: D6) and residue code (RR1: D3, D2, D1) may be checked.

Status Register RR2

RR2 contains the vector which gets modified to indicate the source of interrupt (See the section titled MPSC Modes of Operation). However, the state of the vector does not change if no new interrupts are generated. The contents of RR2 are only changed when a new interrupt is generated. In order to get the correct information, RR2 must be read only after an interrupt is generated, otherwise it will indicate the previous state.

Initialization Sequence

The MPSC initialization routine must issue a channel Reset Command at the beginning. WR4 should be defined before other registers. At the end of the initialization sequence, Reset External/Status and Error Reset commands should be issued to clear any spurious interrupts which may have been caused at power up.

Transmit Under-run/EOM Latch

In SDLC/HDLC, bisync and monosync mode, the transmit under-run/EOM must be reset to enable the CRC check bytes to be appended to the transmit frame or transmit message. The transmit under-run/EOM latch can be reset only after the first character is loaded into the transmit buffer. When the transmitter under-runs at the end of the frame, CRC check bytes are appended to the frame/message. The transmit under-run/EOM latch can be reset at any time during the transmission after the first character. However, it should be reset before the transmitter under-runs otherwise, both bytes of the CRC may not be appended to the frame/message. In the receive mode in bisync operation, the CPU must read the CRC bytes and two more SYNC characters before checking for valid CRC result in RR1.

Sync Character Load Inhibit

In bisync/monosync mode only, it is possible to prevent loading sync characters into the receive buffers by setting the sync character load inhibit bit (WR3; D1 = 1). Caution must be exercised in using this option. It may be possible to get a CRC character in the received message which may match the sync character and not get transferred to the receive buffer. However, sync character load inhibit should be enabled during all pre-frame sync characters so the software routine does not have to read them from the MPSC.

In SDLC/HDLC mode, sync character load inhibit bit must be reset to zero for proper operation.

EOI Command

EOI command can only be issued through channel A irrespective of which channel had generated the interrupt.

Priority in DMA Mode

There is no priority in DMA mode between the following four signals: TxDReq(CH), RxDRq(CH), TxDReq(CHB), RxDRq(CHB). The priority between these four signals must be resolved by the DMA controller. At any given time, all four DMA channels from the 8274 are capable of going active.
**ABSOLUTE MAXIMUM RATINGS**

Ambient Temperature
- Under Bias: 0°C to +70°C
- Storage Temperature
  - Ceramic Package: -65°C to +150°C
  - Plastic Package: -40°C to +125°C

Voltage on Any Pin with Respect to Ground: -0.5V to +7.0V

*Notice: Stresses above those listed under “Absolute Maximum Ratings” may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions above those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.*

**D.C. CHARACTERISTICS** \( T_A = 0°C \text{ to } +70°C; \ V_{CC} = +5V \pm 10\% \)

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Parameter</th>
<th>Min</th>
<th>Max</th>
<th>Units</th>
<th>Test Conditions</th>
</tr>
</thead>
<tbody>
<tr>
<td>( V_{IL} )</td>
<td>Input Low Voltage</td>
<td>-0.5</td>
<td>+0.8</td>
<td>V</td>
<td></td>
</tr>
<tr>
<td>( V_{IH} )</td>
<td>Input High Voltage</td>
<td>+2.0</td>
<td>( V_{CC} + 0.5 )</td>
<td>V</td>
<td></td>
</tr>
<tr>
<td>( V_{OL} )</td>
<td>Output Low Voltage</td>
<td></td>
<td>+0.45</td>
<td>V</td>
<td>( I_{OL} = 2.0 \ mA )</td>
</tr>
<tr>
<td>( V_{OH} )</td>
<td>Output High Voltage</td>
<td></td>
<td>+2.4</td>
<td>V</td>
<td>( I_{OH} = 200 \ \mu A )</td>
</tr>
<tr>
<td>( I_{IL} )</td>
<td>Input Leakage Current</td>
<td>±10</td>
<td>( \mu A )</td>
<td>( V_{IN} = V_{CC} \text{ to } 0V )</td>
<td></td>
</tr>
<tr>
<td>( I_{OL} )</td>
<td>Output Leakage Current</td>
<td>±10</td>
<td>( \mu A )</td>
<td>( V_{OUT} = V_{CC} \text{ to } 0.45V )</td>
<td></td>
</tr>
<tr>
<td>( I_{CC} )</td>
<td>( V_{CC} ) Supply Current</td>
<td></td>
<td>200</td>
<td>mA</td>
<td></td>
</tr>
</tbody>
</table>

**CAPACITANCE** \( T_A = 25°C; \ V_{CC} = \text{GND} = 0V \)

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Parameter</th>
<th>Min</th>
<th>Max</th>
<th>Units</th>
<th>Test Conditions</th>
</tr>
</thead>
<tbody>
<tr>
<td>( C_{IN} )</td>
<td>Input Capacitance</td>
<td>10</td>
<td>pF</td>
<td></td>
<td>( f_c = 1 \ MHz )</td>
</tr>
<tr>
<td>( C_{OUT} )</td>
<td>Output Capacitance</td>
<td>15</td>
<td>pF</td>
<td></td>
<td>Unmeasured pins returned to GND</td>
</tr>
<tr>
<td>( C_{I/O} )</td>
<td>Input/Output Capacitance</td>
<td>20</td>
<td>pF</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### A.C. CHARACTERISTICS

**$T_A = 0^\circ C$ to $+70^\circ C$; $V_{CC} = +5V \pm 10\%$**

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Parameter</th>
<th>Min</th>
<th>Max</th>
<th>Units</th>
<th>Test Conditions</th>
</tr>
</thead>
<tbody>
<tr>
<td>$t_{CY}$</td>
<td>CLK Period</td>
<td>250</td>
<td>4000</td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>$t_{CL}$</td>
<td>CLK Low Time</td>
<td>105</td>
<td>2000</td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>$t_{CH}$</td>
<td>CLK High Time</td>
<td>105</td>
<td>2000</td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>$t_r$</td>
<td>CLK Rise Time</td>
<td>0</td>
<td>30</td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>$t_f$</td>
<td>CLK Fall Time</td>
<td>0</td>
<td>30</td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>$t_{AR}$</td>
<td>A0, A1 Setup to RD ↓</td>
<td></td>
<td></td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>$t_{AD}$</td>
<td>A0, A1 to Data Output Delay</td>
<td>200</td>
<td></td>
<td>ns</td>
<td>$C_L = 150 \text{ pF}$</td>
</tr>
<tr>
<td>$t_{RA}$</td>
<td>A0, A1 Hold after RD ↑</td>
<td>0</td>
<td></td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>$t_{RD}$</td>
<td>RD ↓ to Data Output Delay</td>
<td>200</td>
<td></td>
<td>ns</td>
<td>$C_L = 150 \text{ pF}$</td>
</tr>
<tr>
<td>$t_{RR}$</td>
<td>RD Pulse Width</td>
<td>250</td>
<td></td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>$t_{DF}$</td>
<td>Output Float Delay</td>
<td>120</td>
<td></td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>$t_{AW}$</td>
<td>CS, A0, A1 Setup to WR ↓</td>
<td>0</td>
<td></td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>$t_{WA}$</td>
<td>CS, A0, A1 Hold after WR ↑</td>
<td>0</td>
<td></td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>$t_{WW}$</td>
<td>WR Pulse Width</td>
<td>250</td>
<td></td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>$t_{DW}$</td>
<td>Data Setup to WR ↑</td>
<td>150</td>
<td></td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>$t_{WD}$</td>
<td>Data Hold after WR ↑</td>
<td>0</td>
<td></td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>$t_{PI}$</td>
<td>IPI Setup to INTA ↓</td>
<td>0</td>
<td></td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>$t_{IP}$</td>
<td>IPI Hold after INTA ↑</td>
<td>10</td>
<td></td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>$t_{II}$</td>
<td>INTA Pulse Width</td>
<td>250</td>
<td></td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>$t_{PIO}$</td>
<td>IPI ↓ to IPO Delay</td>
<td>100</td>
<td></td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>$t_{ID}$</td>
<td>INTA ↓ to Data Output Delay</td>
<td>200</td>
<td></td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>$t_{CO}$</td>
<td>RD or WR to DRQ ↓</td>
<td>150</td>
<td></td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>$t_{RV}$</td>
<td>Recovery Time Between Controls</td>
<td>300</td>
<td></td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>$t_{CW}$</td>
<td>CS, A0, A1 to RDY A or RDY B Delay</td>
<td>140</td>
<td></td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>$t_{DCY}$</td>
<td>Data Clock Cycle</td>
<td>4.5</td>
<td>tcy</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$t_{DCL}$</td>
<td>Data Clock Low Time</td>
<td>180</td>
<td></td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>$t_{DCH}$</td>
<td>Data Clock High Time</td>
<td>180</td>
<td></td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>$t_{TD}$</td>
<td>TxC to TxD Delay (x1 Mode)</td>
<td>300</td>
<td></td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>$t_{DS}$</td>
<td>RxD Setup to RxC ↑</td>
<td>0</td>
<td></td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>$t_{DH}$</td>
<td>RxD Hold after RxC ↑</td>
<td>140</td>
<td></td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>$t_{ITD}$</td>
<td>TxC to INT Delay</td>
<td>4</td>
<td>6</td>
<td>tcy</td>
<td></td>
</tr>
<tr>
<td>$t_{IRD}$</td>
<td>RxC to INT Delay</td>
<td>7</td>
<td>10</td>
<td>tcy</td>
<td></td>
</tr>
<tr>
<td>$t_{PL}$</td>
<td>CTS, CD, SYNDET Low Time</td>
<td>200</td>
<td></td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>$t_{PH}$</td>
<td>CTS, CD, SYNDET High Time</td>
<td>200</td>
<td></td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>$t_{IPD}$</td>
<td>External INT from CTS, CD, SYNDET</td>
<td>500</td>
<td></td>
<td>ns</td>
<td></td>
</tr>
</tbody>
</table>
A.C. TESTING INPUT/OUTPUT WAVEFORM

INPUT/OUTPUT

2.4 V

0.45 V

TEST POINTS

2.0 V

0.8 V

170102-27

A.C. Testing: Inputs are driven at 2.4V for a Logic "1" and 0.45V for a Logic "0". Timing measurements are made at 2.0V for a Logic "1" and 0.8V for a Logic "0".

A.C. TESTING LOAD CIRCUIT

DEVICE UNDER TEST

C_L = 150 pF

C_L includes Jig Capacitance

170102-28

WAVEFORMS

CLOCK CYCLE

CLK

t_Cy

t_CH

t_f

t_CL

170102-29

READ CYCLE

CS, A0, A1

RD

DB0-DB7

t_AR

t_RA

t_RA

t_RD

t_DF

t_AD

170102-30
WAVEFORMS (Continued)

WRITE CYCLE

[Diagram showing waveforms for WRITE cycle]

DMA CYCLE

[Diagram showing waveforms for DMA cycle]

READ/WRITE CYCLE (SOFTWARE POLLED MODE)

[Diagram showing waveforms for READ/WRITE cycle]
NOTES:
1. INTA signal as RD signal.
2. IPI signal acts as CS signal.
WAVEFORMS (Continued)

TRANSMIT DATA CYCLE

RECEIVE DATA CYCLE

OTHER TIMING
82530/82530-6
SERIAL COMMUNICATIONS CONTROLLER (SCC)

- Two Independent Full Duplex Serial Channels
- On Chip Crystal Oscillator, Baud-Rate Generator and Digital Phase Locked Loop for Each Channel
- Programmable for NRZ, NRZI or FM Data Encoding/Decoding
- Diagnostic Local Loopback and Automatic Echo for Fault Detection and Isolation
- System Clock Rates:
  - 4 MHz for 82530
  - 6 MHz for 82530-6
- Max Bit Rate (6 MHz)
  - Externally Clocked: 1.5 Mbps
  - Self-Clocked:
    - 375 Kbps FM CODING
    - 187 Kbps NRZI CODING
    - 93 Kbps Asynchronous
- Interfaces with Any INTEL CPU, DMA or I/O Processor
- Available in Express Version
- Asynchronous Modes
  - 5–8 bit Character; Odd, Even or No Parity; 1, 1.5 or 2 Stop Bits
  - Independent Transmit and Receive Clocks: 1X, 16X, 32X or 64X Programmable Sampling Rate
  - Error Detection: Framing, Overrun and Parity
  - Break Detection and Generation
- Bit Synchronous Modes
  - SDLC Loop/Non-Loop Operation
  - CRC-16 or CCITT Generation Detection
  - Abort Generation and Detection
  - I-field Residue Handling
  - CCITT X.25 Compatible
- Byte Synchronous Modes
  - Internal or External Character Synchronization (1 or 2 Characters)
  - Automatic CRC Generation and Checking (CRC 16 or CCITT)
  - IBM Bisync Compatible

The INTEL 82530 Serial Communications Controller (SCC) is a dual-channel, multi-protocol data communications peripheral. It is designed to interface high speed communications lines using Asynchronous, Byte synchronous and Bit synchronous protocols to INTEL's microprocessors based systems. It can be interfaced with Intel's MCS51/96, iAPX86/88/186 and 188 in polled, interrupt driven or DMA driven modes of operation.

The SCC is a 40-pin device manufactured using INTEL's high-performance HMOS* II technology.

* HMOS is a patented process of Intel Corporation.
Figure 1. 82530 Internal Block Diagram
Figure 2. Pin Configurations

The following section describes the pin functions of the SCC. Figure 2 details the pin assignments.

### Table 1. Pin Description

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Pin No.</th>
<th>Type</th>
<th>Name and Function</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>DIP</strong></td>
<td><strong>PLCC</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>DB0</strong></td>
<td>40 1</td>
<td>I/O</td>
<td>DATA BUS: The Data Bus lines are bi-directional three-state lines which interface with the system’s Data Bus. These lines carry data and commands to and from the SCC.</td>
</tr>
<tr>
<td><strong>DB1</strong></td>
<td>1 2</td>
<td>I/O</td>
<td></td>
</tr>
<tr>
<td><strong>DB2</strong></td>
<td>39 44</td>
<td>I/O</td>
<td></td>
</tr>
<tr>
<td><strong>DB3</strong></td>
<td>2 3</td>
<td>I/O</td>
<td></td>
</tr>
<tr>
<td><strong>DB4</strong></td>
<td>38 43</td>
<td>I/O</td>
<td></td>
</tr>
<tr>
<td><strong>DB5</strong></td>
<td>3 4</td>
<td>I/O</td>
<td></td>
</tr>
<tr>
<td><strong>DB6</strong></td>
<td>37 42</td>
<td>I/O</td>
<td></td>
</tr>
<tr>
<td><strong>DB7</strong></td>
<td>4 5</td>
<td>I/O</td>
<td></td>
</tr>
<tr>
<td><strong>INT</strong></td>
<td>5 6</td>
<td>O</td>
<td>INTERRUPT REQUEST: The interrupt signal is activated when the SCC requests an interrupt. It is an open drain output.</td>
</tr>
<tr>
<td><strong>IEO</strong></td>
<td>6 7</td>
<td>O</td>
<td>INTERRUPT ENABLE OUT: IEO is High only if IEI is High and the CPU is not servicing an SCC interrupt or the SCC is not requesting an interrupt (Interrupt Acknowledge cycle only). IEO is connected to the next lower priority device’s IEI input and thus inhibits interrupts from lower priority devices.</td>
</tr>
<tr>
<td><strong>IEI</strong></td>
<td>7 8</td>
<td>I</td>
<td>INTERRUPT ENABLE IN: IEI is used with IEO to form an interrupt daisy chain when there is more than one interrupt-driven device. A High IEI indicates that no other higher priority device has an interrupt under service or is requesting an interrupt.</td>
</tr>
<tr>
<td>Symbol</td>
<td>Pin No.</td>
<td>Type</td>
<td>Name and Function</td>
</tr>
<tr>
<td>--------</td>
<td>---------</td>
<td>------</td>
<td>-------------------</td>
</tr>
<tr>
<td>INTA</td>
<td>8 9</td>
<td>I</td>
<td>INTERRUPT ACKNOWLEDGE: This signal indicates an active Interrupt Acknowledge cycle. During this cycle, the SCC interrupt daisy chain settles. When RD becomes active, the SCC places an interrupt vector on the data bus (if IEI is High). INTA is latched by the rising edge of CLK.</td>
</tr>
<tr>
<td>VCC</td>
<td>9 10</td>
<td></td>
<td>POWER: +5V Power supply.</td>
</tr>
<tr>
<td>RDY&lt;sub&gt;A&lt;/sub&gt;/REQ&lt;sub&gt;A&lt;/sub&gt;</td>
<td>10 11</td>
<td>O</td>
<td>READY/REQUEST: (output, open-drain when programmed for a Ready function, driven High or Low when programmed for a Request function). These dual-purpose outputs may be programmed as Request lines for a DMA controller or as Ready lines to synchronize the CPU to the SCC data rate. The reset state is Ready.</td>
</tr>
<tr>
<td>RDY&lt;sub&gt;B&lt;/sub&gt;/REQ&lt;sub&gt;B&lt;/sub&gt;</td>
<td>30 34</td>
<td>O</td>
<td>SYNCHRONIZATION: These pins can act either as inputs, outputs or part of the crystal oscillator circuit. In the Asynchronous receive mode (crystal oscillator option not selected), these pins are inputs similar to CTS and CD. In this mode, transitions on these lines affect the state of the Synchronous/Hunt status bits in Read Register 0 (Figure 9) but have no other function. In External Synchronization mode with the crystal oscillator not selected, these lines also act as inputs. In this mode, SYNC must be driven LOW two receive clock cycles after the last bit in the synchronous character is received. Character assembly begins on the rising edge of the receive clock immediately preceding the activation of SYNC. In the Internal Synchronization mode (Monosync and Bisync) with the crystal oscillator not selected, these pins act as outputs and are active only during the part of the receive clock cycle in which synchronous characters are recognized. The synchronous condition is not latched, so these outputs are active each time a synchronization pattern is recognized (regardless of characters boundaries). In SDLC mode, these pins act as outputs and are valid on receipt of a flag.</td>
</tr>
<tr>
<td>SYNC&lt;sub&gt;A&lt;/sub&gt;</td>
<td>11 12</td>
<td>I/O</td>
<td>RECEIVE/TRANSMIT CLOCKS: These pins can be programmed in several different modes of operation. In each channel, RTxC may supply the receive clock, the transmit clock, the clock for the baud rate generator, or the clock for the Digital Phase Locked Loop. These pins can be programmed for use with the respective SYNC pins as a crystal oscillator. The receive clock may be 1, 16, 32, or 64 times the data rate in Asynchronous modes.</td>
</tr>
<tr>
<td>SYNC&lt;sub&gt;B&lt;/sub&gt;</td>
<td>29 33</td>
<td>I/O</td>
<td>RECEIVE DATA: These lines receive serial data at standard TTL levels.</td>
</tr>
<tr>
<td>RTxC&lt;sub&gt;A&lt;/sub&gt;</td>
<td>12 13</td>
<td>I</td>
<td>TRANSMIT/RECEIVE CLOCKS: These pins can be programmed in several different modes of operation. TRxA may supply the receive clock or the transmit clock in the input mode or supply the output of the Digital Phase Locked Loop, the crystal oscillator, the baud rate generator, or the transmit clock in the output mode.</td>
</tr>
<tr>
<td>RTxC&lt;sub&gt;B&lt;/sub&gt;</td>
<td>28 32</td>
<td>I</td>
<td>TRANSMIT DATA: These output signals transmit serial data at standard TTL levels.</td>
</tr>
<tr>
<td>RxD&lt;sub&gt;A&lt;/sub&gt;</td>
<td>13 14</td>
<td>I</td>
<td>DATA TERMINAL READY/REQUEST: These outputs follow the state programmed into the DTR bit. They can also be used as general purpose outputs or as Request lines for a DMA controller.</td>
</tr>
<tr>
<td>RxD&lt;sub&gt;B&lt;/sub&gt;</td>
<td>27 31</td>
<td>O</td>
<td></td>
</tr>
<tr>
<td>TRxA</td>
<td>14 15</td>
<td>I/O</td>
<td></td>
</tr>
<tr>
<td>TRxB</td>
<td>26 30</td>
<td>I/O</td>
<td></td>
</tr>
<tr>
<td>TXA</td>
<td>15 16</td>
<td>O</td>
<td></td>
</tr>
<tr>
<td>TXB</td>
<td>25 29</td>
<td>O</td>
<td></td>
</tr>
</tbody>
</table>
Table 1. Pin Description (Continued)

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Pin No.</th>
<th>Type</th>
<th>Name and Function</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>DIP</td>
<td>PLCC</td>
<td></td>
</tr>
<tr>
<td>RTS_A</td>
<td>17</td>
<td>20</td>
<td>O</td>
</tr>
<tr>
<td>RTS_B</td>
<td>23</td>
<td>26</td>
<td>O</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>REQUEST TO SEND: When the Request to Send (RTS) bit in Write Register 5 is set (Figure 10), the RTS signal goes Low. When the RTS bit is reset in the Asynchronous mode and Auto Enable is on, the signal goes High after the transmitter is empty. In Synchronous mode or in Asynchronous mode with Auto Enable off, the RTS pin strictly follows the state of the RTS bit. Both pins can be used as general-purpose outputs.</td>
</tr>
<tr>
<td>CTS_A</td>
<td>18</td>
<td>21</td>
<td>I</td>
</tr>
<tr>
<td>CTS_B</td>
<td>22</td>
<td>25</td>
<td>I</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>CLEAR TO SEND: If these pins are programmed as Auto Enables, a Low on the inputs enables the respective transmitters. If not programmed as Auto Enables, they may be used as general-purpose inputs. Both inputs are Schmitt-trigger buffered to accommodate slow rise-time inputs. The SCC detects pulses on these inputs and can interrupt the CPU on both logic level transitions.</td>
</tr>
<tr>
<td>CDA_A</td>
<td>19</td>
<td>22</td>
<td>I</td>
</tr>
<tr>
<td>CDA_B</td>
<td>21</td>
<td>24</td>
<td>I</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>CARRIER DETECT: These pins function as receiver enables if they are programmed for Auto Enables; otherwise they may be used as general-purpose input pins. Both pins are Schmitt-trigger buffered to accommodate slow rise time signals. The SCC detects pulses on these pins and can interrupt the CPU on both logic level transitions.</td>
</tr>
<tr>
<td>CLK</td>
<td>20</td>
<td>23</td>
<td>I</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>CLOCK: This is the system SCC clock used to synchronize internal signals.</td>
</tr>
<tr>
<td>GND</td>
<td>31</td>
<td>35</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>GROUND</td>
</tr>
<tr>
<td>D/C</td>
<td>32</td>
<td>37</td>
<td>I</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>DATA/COMMAND SELECT: This signal defines the type of information transferred to or from the SCC. A High means data is transferred; a Low indicates a command.</td>
</tr>
<tr>
<td>CS</td>
<td>33</td>
<td>38</td>
<td>I</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>CHIP SELECT: This signal selects the SCC for a read or write operation.</td>
</tr>
<tr>
<td>A/B</td>
<td>34</td>
<td>39</td>
<td>I</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>CHANNEL A/CHANNEL B SELECT: This signal selects the channel in which the read or write operation occurs.</td>
</tr>
<tr>
<td>WR</td>
<td>35</td>
<td>40</td>
<td>I</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>WRITE: When the SCC is selected this signal indicates a write operation. The coincidence of RD and WR is interpreted as a reset.</td>
</tr>
<tr>
<td>RD</td>
<td>36</td>
<td>41</td>
<td>I</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>READ: This signal indicates a read operation and when the SCC is selected, enables the SCC's bus drivers. During the Interrupt Acknowledge cycle, this signal gates the interrupt vector onto the bus if the SCC is the highest priority device requesting an interrupt.</td>
</tr>
</tbody>
</table>
GENERAL DESCRIPTION

The Intel 82350 Serial Communications Controller (SCC) is a dual-channel, multi-protocol data communications peripheral. The SCC functions as a serial-to-parallel, parallel-to-serial convertor/controller. The SCC can be software-configured to satisfy a wide range of serial communications applications. The device contains sophisticated internal functions including on-chip baud rate generators, digital phase locked loops, various data encoding and decoding schemes, and crystal oscillators that reduce the need for external logic.

In addition, diagnostic capabilities—automatic echo and local loopback—allow the user to detect and isolate a failure in the network. They greatly improve the reliability and fault isolation of the system.

The SCC handles Asynchronous formats, Synchronous byte-oriented protocols such as IBM Bisync, and Synchronous bit-oriented protocols such as HDLC and IBM SDLC. This versatile device supports virtually any serial data transfer application (Terminal, Personal Computer, Peripherals, Industrial Controller, Telecommunication system, etc.).

The 82530 can generate and check CRC codes in any Synchronous mode and can be programmed to check data integrity in various modes. The SCC also has facilities for modem control in both channels. In applications where these controls are not needed, the modem control can be used for general purpose I/O.

The Intel 82530 is designed to support Intel's MCS51/96, iAPX 86/88 and iAPX 186/188 families.

ARCHITECTURE

The 82530 internal structure includes two full-duplex channels, two baud rate generators, internal control and interrupt logic, and a bus interface to a non-multiplexed CPU bus. Associated with each channel are a number of read and write registers for mode control and status information, as well as logic necessary to interface modems or other external devices.

The logic for both channels provides formats, synchronization, and validation for data transferred to and from the channel interface. The modem control inputs are monitored by the control logic under program control. All of the modem control signals are general-purpose in nature and can optionally be used for functions other than modem control.

The register set for each channel includes ten control (write) registers, two synchronous character (write) registers, and four status (read) registers. In addition, each baud rate generator has two (read/write) registers for holding the time constant that determines the baud rate. Finally, associated with the interrupt logic is a write register for the interrupt vector accessible through either channel, a write-only Master Interrupt Control register and three read registers; one containing the vector with status information (Channel B only), one containing the vector without status (A only), and one containing the Interrupt Pending bits (A only).

The registers for each channel are designated as follows:

WR0–WR15—Write Registers 0 through 15.
RR0–RR3, RR10, RR12, RR13, RR15—Read Registers 0 through 3, 10, 12, 13, 15.

Table 2 lists the functions assigned to each read or write register. The SCC contains only one WR2 and WR9, but they can be accessed by either channel. All other registers are paired (one for each channel).

DATA PATH

The transmit and receive data path illustrated in Figure 3 is identical for both channels. The receiver has three 8-bit buffer registers in a FIFO arrangement, in addition to the 8-bit receive shift register. This scheme creates additional time for the CPU to service an interrupt at the beginning of a block of high-speed data. Incoming data is routed through one of several paths (data or CRG) depending on the selected mode (the character length in asynchronous modes also determines the data path).

The transmitter has an 8-bit transmit data buffer register loaded from the internal data bus and a 20-bit transmit shift register that can be loaded either from the sync-character registers or from the transmit data register. Depending on the operational mode, outgoing data is routed through one of four main paths before it is transmitted from the Transmit Data output (TxD).
Table 2. Read and Write Register Functions

<table>
<thead>
<tr>
<th>READ REGISTER FUNCTIONS</th>
<th>WRITE REGISTER FUNCTIONS</th>
</tr>
</thead>
<tbody>
<tr>
<td>RR0 Transmit/Receive buffer status and External status</td>
<td>WR0 CRC initialize, initialization commands for the various modes, shift right/shift left command</td>
</tr>
<tr>
<td>RR1 Special Receive Condition status</td>
<td>WR1 Transmit/Receive interrupt and data transfer mode definition</td>
</tr>
<tr>
<td>RR2 Modified interrupt vector (Channel B only) Unmodified interrupt (Channel A only)</td>
<td>WR2 Interrupt vector (accessed through either channel)</td>
</tr>
<tr>
<td>RR3 Interrupt Pending bits (Channel A only)</td>
<td>WR3 Receive parameters and control</td>
</tr>
<tr>
<td>RR8 Receive buffer</td>
<td>WR4 Transmit/Receive miscellaneous parameters and modes</td>
</tr>
<tr>
<td>RR10 Miscellaneous status</td>
<td>WR5 Transmit parameters and controls</td>
</tr>
<tr>
<td>RR12 Lower byte of baud rate generator time constant</td>
<td>WR6 Sync characters or SDLC address field</td>
</tr>
<tr>
<td>RR13 Upper byte of baud rate generator time constant</td>
<td>WR7 Sync character or SDLC flag</td>
</tr>
<tr>
<td>RR15 External/Status interrupt information</td>
<td>WR8 Transmit buffer</td>
</tr>
<tr>
<td></td>
<td>WR9 Master interrupt control and reset (accessed through either channel)</td>
</tr>
<tr>
<td></td>
<td>WR10 Miscellaneous transmitter/receiver control bits</td>
</tr>
<tr>
<td></td>
<td>WR11 Clock Mode control</td>
</tr>
<tr>
<td></td>
<td>WR12 Lower Byte of baud rate generator time constant</td>
</tr>
<tr>
<td></td>
<td>WR13 Upper Byte of baud rate generator time constant</td>
</tr>
<tr>
<td></td>
<td>WR14 Miscellaneous control bits</td>
</tr>
<tr>
<td></td>
<td>WR15 External/Status interrupt control</td>
</tr>
</tbody>
</table>
FUNCTIONAL DESCRIPTION

The functional capabilities of the SCC can be described from two different points of view: as a data communications device, it transmits and receives data in a wide variety of data communications protocols; as a microprocessor peripheral, it interacts with the CPU and provides vectored interrupts and handshaking signals.

DATA COMMUNICATIONS CAPABILITIES

The SCC provides two independent full-duplex channels programmable for use in any common asynchronous or synchronous data-communications protocol. Figure 4 and the following description briefly detail these protocols.

Asynchronous Modes

Transmission and reception can be accomplished independently on each channel with five to eight bits per character, plus optional even or odd parity. The transmitter can supply one, one-and-a-half or two stop bits per character and can provide a break output at any time. The receiver break-detection logic interrupts the CPU both at the start and at the end of a received break. Reception is protected from spikes by a transient spike-rejection mechanism that checks the signal one-half a bit time after a Low level is detected on the receive data input (RxD A or RxD B). If the Low does not persist (as in the case of a transient), the character assembly process does not start.

Framing errors and overrun errors are detected and buffered together with the partial character on which they occur. Vectored interrupts allow fast servicing of error conditions using dedicated routines. Furthermore, a built-in checking process avoids the interpretation of a framing error as a new start bit: a framing error results in the addition of one-half a bit time to the point at which the search for the next start bit begins.

The SCC does not require symmetric transmit and receive clock signals—a feature allowing use of the wide variety of clock sources. The transmitter and receiver can handle data at a rate of 1, 1/16, 1/32 or 1/64 of the clock rate supplied to the receive and transmit clock inputs. In the asynchronous modes, a data rate equal to the clock rate, 1x mode, requires external synchronization. In asynchronous modes, the SYNC pin may be programmed as an input used for functions such as monitoring a ring indicator.

![Figure 4. SCC Protocols](image-url)
Synchronous Modes

The SCC supports both byte-oriented and bit-oriented synchronous communication. Synchronous-byte-oriented protocols can be handled in several modes allowing character synchronization with a 6-bit or 8-bit synchronous character (Monosync), any 12-bit or 16-bit synchronous pattern (Bisync), or with an external synchronous signal. Leading synchronous characters can be removed without interrupting the CPU.

5- or 7-bit synchronous characters are detected with 8- or 16-bit patterns in the SCC by overlapping the larger pattern across multiple incoming synchronous characters as shown in Figure 5.

CRC checking for Synchronous byte-oriented mode is delayed by one character time so that the CPU may disable CRC checking on specific characters. This permits the implementation of protocols such as IBM Bisync.

Both CRC-16 \((X^{16} + X^{15} + X^2 + 1)\) and CCITT \((X^{16} + X^{12} + X^5 + 1)\) error checking polynomials are supported. Either polynomial may be selected in all synchronous modes. Users may preset the CRC generator and checker to all 1s or all 0s. The SCC also provides a feature that automatically transmits CRC data when no other data is available for transmission. This allows for high-speed transmissions under DMA control, with no need for CPU intervention at the end of a message. When there is no data or CRC to send in synchronous modes, the transmitter inserts 6-, 8-, or 16-bit synchronous characters, regardless of the programmed character length.

The SCC supports synchronous bit-oriented protocols, such as SDLC and HDLC, by performing automatic flag sending, zero insertion, and CRC generation. A special command can be used to abort a frame in transmission. At the end of a message, the SCC automatically transmits the CRC and trailing flag when the transmitter underruns. The transmitter may also be programmed to send an idle line consisting of continuous flag characters or a steady marking condition.

If a transmit underrun occurs in the middle of a message, an external status interrupt warns the CPU of this status change so that an abort may be issued. The SCC may also be programmed to send an abort itself in case of an underrun, relieving the CPU of this task. One to eight bits per character can be sent allowing reception of a message with no prior information about the character structure in the information field of a frame.

The receiver automatically acquires synchronization on the leading flag of a frame in SDLC or HDLC mode and provides a synchronization signal on the SYNC pin (an interrupt can also be programmed). The receiver can be programmed to search for frames addressed by a single byte (or four bits within a byte) of a user-selected address or to a global broadcast address. In this mode, frames not matching either the user-selected or broadcast address are ignored. The number of address bytes can be extended under software control. For receiving data, an interrupt on the first received character, or an interrupt on every character, or on special condition only (end-of-frame) can be selected. The receiver automatically deletes all 0s inserted by the transmitter during character assembly. CRC is also calculated and is automatically checked to validate frame transmission. At the end of transmission, the status of a received frame is available in the status registers. In SDLC mode, the SCC must be programmed to use the SDLC CRC polynomial, but the generator and checker may be preset to all 1s or all 0s. The CRC is inverted before transmission and the receiver checks against the bit pattern 00011101010001111.

NRZ, NRZI or FM coding may be used in any 1X mode. The parity options available in asynchronous modes are available in synchronous modes.

The SCC can be conveniently used under DMA control to provide high-speed reception or transmission.
In reception, for example, the SCC can interrupt the CPU when the first character of a message is received. The CPU then enables the DMA to transfer the message to memory. The SCC then issues an end-of-frame interrupt and the CPU can check the status of the received message. Thus, the CPU is freed for other service while the message is being received. The CPU may also enable the DMA first and have the SCC interrupt only on end-of-frame. This procedure allows all data to be transferred via DMA.

**SDLC LOOP MODE**

The SCC supports SDLC Loop mode in addition to normal SDLC. In a loop topology, there is a primary controller station that manages the message traffic flow and any number of secondary stations. In Loop mode, the SCC performs the functions of a secondary station while an SCC operating in regular SDLC mode can act as a controller (Figure 6).

**Figure 6. An SDLC Loop**

A secondary station in an SDLC Loop is always listening to the messages being sent around the loop, and in fact must pass these messages to the rest of the loop by retransmitting them with a one-bit-time delay. The secondary station can place its own message on the loop only at specific times. The controller signals that secondary stations may transmit messages by sending a special character, called an EOP (End of Poll), around the loop. The EOP character is the bit pattern 11111110. Because of zero insertion during messages, this bit pattern is unique and easily recognized.

When a secondary station has a message to transmit and recognizes an EOP on the line, it changes the last binary one of the EOP to a zero before transmission. This has the effect of turning the EOP into a flag sequence. The secondary station now places its message on the loop and terminates the message with an EOP. Any secondary stations further down the loop with messages to transmit can then append their messages to the message of the first secondary station by the same process. Any secondary stations without messages to send merely echo the incoming messages and are prohibited from placing messages on the loop (except upon recognizing an EOP).

SDLC Loop mode is a programmable option in the SCC. NRZ, NRZI, and FM coding may all be used in SDLC Loop mode.

**BAUD RATE GENERATORS**

Each channel in the SCC contains a programmable baud rate generator. Each generator consists of two 8-bit time constant registers that form a 16-bit time constant, a 16-bit down counter, and a flip-flop on the output producing a square wave. On startup, the flip-flop on the output is set in a High state, the value in the time constant register is loaded into the counter, and the counter starts counting down. The output of the baud rate generator toggles upon reaching zero, the value in the time constant register is loaded into the counter, and the process is repeated. The time constant may be changed at any time, but the new value does not take effect until the next load of the counter.

The output of the baud rate generator may be used as either the transmit clock, the receive clock, or both. It can also drive the digital phase-locked loop (see next section).

If the receive clock or transmit clock is not programmed to come from the TRxC pin, the output of the baud rate generator may be echoed out via the TRxC pin.

The following formula relates the time constant to the baud rate. (The baud rate is in bits/second, the BR clock frequency is in Hz, and clock mode is 1, 16, 32, or 64.)

\[
\text{time constant} = \frac{\text{BR clock frequency}}{2 \times \text{baud rate} \times \text{clock mode}} - 2
\]
The SCC contains a digital phase locked-loop (DPLL) to recover clock information from a data-stream with NRZI or FM encoding. The DPLL is driven by a clock that is nominally 32 (NRZI) or 16 (FM) times the data rate. The DPLL uses this clock, along with the datastream, to construct a clock for the data. This clock may then be used as the SCC receive clock, the transmit clock, or both.

For NRZI coding, the DPLL counts the 32X clock to create nominal bit times. As the 32X clock is counted, the DPLL is searching the incoming datastream for edges (either 1/0 or 0/1). Whenever an edge is detected, the DPLL makes a count adjustment (during the next counting cycle), producing a terminal count closer to the center of the bit cell.

For FM encoding, the DPLL still counts from 1 to 31, but with a cycle corresponding to two bit times. When the DPLL is locked, the clock edges in the datastream should occur between counts 15 and 16 and between counts 31 and 0. The DPLL looks for edges only during a time centered on the \(15/16\) counting transition.

The 32X clock for the DPLL can be programmed to come from either the RTxC input or the output of the baud rate generator. The DPLL output may be programmed to be echoed out of the SCC via the TRxC pin (if this pin is not being used as an input).

### DATA ENCODING

The SCC may be programmed to encode and decode the serial data in four different ways (Figure 7). In NRZ encoding, a 1 is represented by a High level and a 0 is represented by a Low level. In NRZI encoding, as 1 is represented by no change in level and a 0 is represented by a change in level. In FM1 (more properly, bi-phase mark) a transition occurs at the beginning of every bit cell. A 1 is represented by an additional transition at the center of the bit cell and a 0 is represented by no additional transition at the center of the bit cell. In FM0 (bi-phase space), a transition occurs at the beginning of every bit cell. A 0 is represented by an additional transition at the center of the bit cell, and a 1 is represented by no additional transition at the center of the bit cell. In addition to these four methods, the SCC can be used to decode Manchester (bi-phase level) data by using the DPLL in the FM mode and programming the receiver for NRZ data. Manchester encoding always produces a transition at the center of the bit cell. If the transition is 0/1 the bit is a 0. If the transition is 1/0 the bit is a 1.

### DIGITAL PHASE LOCKED LOOP

<table>
<thead>
<tr>
<th>Rate (BAUD)</th>
<th>Time Constant (decimal notation)</th>
<th>Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>19200</td>
<td>102</td>
<td>—</td>
</tr>
<tr>
<td>9600</td>
<td>206</td>
<td>—</td>
</tr>
<tr>
<td>7200</td>
<td>275</td>
<td>0.12%</td>
</tr>
<tr>
<td>4800</td>
<td>414</td>
<td>—</td>
</tr>
<tr>
<td>3600</td>
<td>553</td>
<td>0.06%</td>
</tr>
<tr>
<td>2400</td>
<td>830</td>
<td>—</td>
</tr>
<tr>
<td>2000</td>
<td>996</td>
<td>0.04%</td>
</tr>
<tr>
<td>1800</td>
<td>1107</td>
<td>0.03%</td>
</tr>
<tr>
<td>1200</td>
<td>1662</td>
<td>—</td>
</tr>
<tr>
<td>600</td>
<td>3326</td>
<td>—</td>
</tr>
<tr>
<td>300</td>
<td>6654</td>
<td>—</td>
</tr>
<tr>
<td>150</td>
<td>13310</td>
<td>—</td>
</tr>
<tr>
<td>134.5</td>
<td>14844</td>
<td>0.0007%</td>
</tr>
<tr>
<td>110</td>
<td>18151</td>
<td>0.0015%</td>
</tr>
<tr>
<td>75</td>
<td>26622</td>
<td>—</td>
</tr>
<tr>
<td>50</td>
<td>39334</td>
<td>—</td>
</tr>
</tbody>
</table>

### AUTO ECHO AND LOCAL LOOPBACK

The SCC is capable of automatically echoing everything it receives. This feature is useful mainly in asynchronous modes, but works in synchronous and SDLC modes as well. In Auto Echo mode TxD is RxD. Auto Echo mode can be used with NRZI or FM encoding with no additional delay, because the datastream is not decoded before retransmission. In Auto Echo mode, the CTS input is ignored as a transmitter enable (although transitions on this input can still cause interrupts if programmed to do so). In this mode, the transmitter is actually bypassed and the programmer is responsible for disabling transmitter interrupts and READY/REQUEST on transmit.

The SCC is also capable of local loopback. In this mode, TxD is RxD just as in Auto Echo mode. However, in Local Loopback mode, the internal transmit data is tied to the internal receive data and RxD is ignored (except to be echoed out via TxD). CTS and CD inputs are also ignored as transmit and receive enables. However, transitions on these inputs can still cause interrupts. Local Loopback works in asynchronous, synchronous and SDLC modes with NRZ, NRZI or FM coding of the data stream.

### SERIAL BIT RATE

To run the 82530 (4 MHz/6 MHz) at 1/1.5 Mbps the receive and transmit clocks must be externally generated and synchronized to the system clock. If the serial clocks (RTxC and TRxC) and the system clock (CLK) are asynchronous, the maximum bit rate is 880 Kbps/1.3 Mbps. For self-clocked operation, i.e using the on chip DPLL, the maximum bit rate is 125/187 Kbps if NRZI coding is used and 250/375 Kbps if FM coding is used.
Figure 7. Data Encoding Methods

Table 4. Maximum Bit Rates

<table>
<thead>
<tr>
<th>Mode</th>
<th>System Clock</th>
<th>System Clock/Serial Clock</th>
<th>Serial Bit Rate</th>
<th>Conditions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Serial clocks generated externally</td>
<td>4 MHz</td>
<td>4</td>
<td>1 Mbps</td>
<td>Serial clocks synchronized with system clock. Refer to parameter #3 and 10 in general timings.</td>
</tr>
<tr>
<td></td>
<td>6 MHz</td>
<td>4</td>
<td>1.5 Mbps</td>
<td>Serial clocks synchronized with system clock. Refer to parameter #3 and #10 in general timings.</td>
</tr>
<tr>
<td></td>
<td>4 MHz</td>
<td>4.5</td>
<td>880 Kbps</td>
<td>Serial clocks and system clock asynchronous.</td>
</tr>
<tr>
<td></td>
<td>6 MHz</td>
<td>4.5</td>
<td>1.3 Mbps</td>
<td>Serial clocks and system clock asynchronous.</td>
</tr>
<tr>
<td>Self-clocked operation</td>
<td>4 MHz</td>
<td>32</td>
<td>125 Kbps</td>
<td></td>
</tr>
<tr>
<td>NRZI</td>
<td>6 MHz</td>
<td>32</td>
<td>187 Kbps</td>
<td></td>
</tr>
<tr>
<td>FM</td>
<td>4 MHz</td>
<td>16</td>
<td>250 Kbps</td>
<td></td>
</tr>
<tr>
<td></td>
<td>6 MHz</td>
<td>16</td>
<td>375 Kbps</td>
<td></td>
</tr>
<tr>
<td>ASYNC</td>
<td>4 MHz</td>
<td>16</td>
<td>62.5 Kbps</td>
<td></td>
</tr>
<tr>
<td></td>
<td>6 MHz</td>
<td>16</td>
<td>93.75 Kbps</td>
<td></td>
</tr>
</tbody>
</table>

I/O INTERFACE CAPABILITIES

The SCC offers the choice of Polling, Interrupt (vectored or nonvectored) and Block Transfer modes to transfer data, status, and control information to and from the CPU. The Block Transfer mode can be implemented under CPU or DMA control.

POLLING

All interrupts are disabled. Three status registers in the SCC are automatically updated whenever any function is performed. For example, end-of-frame in SDLC mode sets a bit in one of these status registers. The idea behind polling is for the CPU to periodically read a status register until the register contents indicate the need for data to be transferred. Only one register needs to be read; depending on its contents, the CPU either writes data, reads data, or continues. Two bits in the register indicate the need for data transfer. An alternative is a poll of the Interrupt Pending register to determine the source of an interrupt. The status for both channels resides in one register.
INTERRUPTS

When a SCC responds to an Interrupt Acknowledge signal (INTA) from the CPU, an interrupt vector may be placed on the data bus. This vector is written in WR2 and may be read in RR2A or RR2B (Figures 9 and 10).

To speed interrupt response time, the SCC can modify three bits in this vector to indicate status. If the vector is read in Channel A, status is never included; if it is read in Channel B, status is always included.

Each of the six sources of interrupts in the SCC (Transmit, Receive and External/Status interrupts in both channels) has three bits associated with the interrupt source: Interrupt Pending (IP), Interrupt Under Service (IUS), and Interrupt Enable (IE). Operation of the IE bits is straightforward. If the IE bit is set for a given interrupt source, then that source can request interrupts. The exception is when the MIE (Master Interrupt Enable) bit in WR9 is reset and no interrupts may be requested. The IE bits are write-only.

The other two bits are related to the interrupt priority chain (Figure 8). As a peripheral, the SCC may request an interrupt only when no higher-priority device is requesting one, e.g., when IEI is High. If the device in question requests an interrupt, it pulls down INT. The CPU then responds with INTA, and the interrupting device places the vector on the data bus.

In the SCC, the IP bit signals a need for interrupt servicing. When an IP bit is 1 and the IEI input is High, the INT output is pulled Low, requesting an interrupt. In the SCC, if the IE bit is not set by enabling interrupts, then the IP for that source can never be set. The IP bits are readable in RR3A.

The IUS bits signal that an interrupt request is being serviced. If an IUS is set, all interrupt sources of lower priority in the SCC and external to the SCC are prevented from requesting interrupts. The internal interrupt sources are inhibited by the state of the internal daisy chain, while lower priority devices are inhibited by the IEO output of the SCC being pulled Low and propagated to subsequent peripherals. An IUS bit is set during an Interrupt Acknowledge cycle if there are no higher priority devices requesting interrupts.

There are three types of interrupts: Transmit, Receive and External/Status interrupts. Each interrupt type is enabled under program control with Channel A having higher priority than Channel B, and with Receiver, Transmit and External/Status interrupts prioritized in that order within each channel. When the Transmit interrupt is enabled, the CPU is interrupted when the transmit buffer becomes empty. (This implies that the transmitter must have had a data character written into it so that it can become empty.) When enabled, the receiver can interrupt the CPU in one of three ways:

- Interrupt on First Receive Character or Special Receive condition.
- Interrupt on all Receive Characters or Special Receive condition.
- Interrupt on Special Receive condition only.

Interrupt-on-First-Character or Special-Condition and Interrupt-on-Special-Condition-Only are typically used with the Block Transfer mode. A Special-Receive-Condition is one of the following: receiver overrun, framing error in Asynchronous mode, End-of-Frame in SDLC mode and, optionally, a parity error. The Special-Receive-Condition interrupt is different from an ordinary receive character available interrupt only in the status placed in the vector.

![Figure 8. Daisy Chaining SCC's](image-url)
during the Interrupt-Acknowledge cycle. In Interrupt on First Receive Character, an interrupt can occur from Special Receive conditions any time after the first receive character interrupt.

The main function of the External/Status interrupt is to monitor the signal transitions of the CTS, CD, and SYNC pins; however, an External/Status interrupt is also caused by a Transmit Underrun condition, or a zero count in the baud rate generator, or by the detection of a Break (asynchronous mode), Abort (SDLC mode) or EOP (SDLC Loop mode) sequence in the data stream. The interrupt caused by the Abort or EOP has a special feature allowing the SCC to interrupt when the Abort or EOP sequence is detected or terminated. This feature facilitates the proper termination of the current message, correct initialization of the next message, and the accurate timing of the Abort condition in external logic in SDLC mode. In SDLC Loop mode this feature allows secondary stations to recognize the wishes of the primary station to regain control of the loop during a poll sequence.

CPU/DMA BLOCK TRANSFER

The SCC provides a Block Transfer mode to accommodate CPU block transfer functions and DMA controllers. The Block Transfer mode uses the READY/REQUEST output in conjunction with the READY/REQUEST bits in WR1. The READY/REQUEST output can be defined under software control as a READY line in the CPU Block Transfer mode (WR1; D6 = 0) or as a request line in the DMA Block Transfer mode (WR1; D6 = 1). To a DMA controller, the SCC REQUEST output indicates that the SCC is ready to transfer data to or from memory. To the CPU, the READY line indicates that the SCC is not ready to transfer data, thereby requesting that the CPU extend the I/O cycle. The DTR/REQUEST line allows full-duplex operation under DMA control.

PROGRAMMING

Each channel has fifteen Write registers that are individually programmed from the system bus to configure the functional personality of each channel. Each channel also has eight Read registers from which the system can read Status, Baud rate, or Interrupt information.

Only the four data registers (Read, Write for channels A and B) are directly selected by a High on the D/C input and the appropriate levels on the RD, WR and A/B pins. All other registers are addressed indirectly by the content of Write Register 0 in conjunction with a Low on the D/C input and the appropriate levels on the RD, WR and A/B pins. If bit 3 in WW0 is 1 and bits 4 and 5 are 0 then bits 0, 1, 2 address the higher registers 8 through 15. If bits 3, 4, 5 contain a different code, bits 0, 1, 2 address the lower registers 0 through 7 as shown on Table 5.

Writing to or reading from any register except RR0, WR0 and the Data Registers thus involves two operations.

<table>
<thead>
<tr>
<th>D/C “Point High” Code in WR0</th>
<th>D2 in WR0</th>
<th>D1 in WR0</th>
<th>D0</th>
<th>Write Register</th>
<th>Read Register</th>
</tr>
</thead>
<tbody>
<tr>
<td>High</td>
<td>Either Way</td>
<td>X</td>
<td>X</td>
<td>Data</td>
<td>Data</td>
</tr>
<tr>
<td>Low</td>
<td>Not True</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Low</td>
<td>Not True</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Low</td>
<td>Not True</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>Low</td>
<td>Not True</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>Low</td>
<td>Not True</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>4</td>
</tr>
<tr>
<td>Low</td>
<td>Not True</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>5</td>
</tr>
<tr>
<td>Low</td>
<td>Not True</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>6</td>
</tr>
<tr>
<td>Low</td>
<td>Not True</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>7</td>
</tr>
<tr>
<td>Low</td>
<td>True</td>
<td>0</td>
<td>0</td>
<td>Data</td>
<td>Data</td>
</tr>
<tr>
<td>Low</td>
<td>True</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>9</td>
</tr>
<tr>
<td>Low</td>
<td>True</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>10</td>
</tr>
<tr>
<td>Low</td>
<td>True</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>11</td>
</tr>
<tr>
<td>Low</td>
<td>True</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>12</td>
</tr>
<tr>
<td>Low</td>
<td>True</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>13</td>
</tr>
<tr>
<td>Low</td>
<td>True</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>14</td>
</tr>
<tr>
<td>Low</td>
<td>True</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>15</td>
</tr>
</tbody>
</table>
First write the appropriate code into WRO, then follow this by a write or read operation on the register thus specified. Bits 0 through 4 in WW0 are automatically cleared after this operation, so that WW0 then points to WR0 or RR0 again.

Channel A/Channel B selection is made by the A/B input (High = A, Low = B)

The system program first issues a series of commands to initialize the basic mode of operation. This is followed by other commands to qualify conditions within the selected mode. For example, the Asynchronous mode, character length, clock rate, number of stop bits, even or odd parity might be set first. Then the interrupt mode would be set, and finally, receiver or transmitter enable.

READ REGISTERS

The SCC contains eight read registers (actually nine, counting the receive buffer (RR8) in each channel). Four of these may be read to obtain status information (RR0, RR1, RR10, and RR15). Two registers (RR12 and RR13) may be read to determine the baud rate generator time constant. RR2 contains either the unmodified interrupt vector (Channel A) or the vector modified by status information (Channel B). RR3 contains the Interrupt Pending (IP) bits (Channel A). Figure 9 shows the formats for each read register.

The status bits of RR0 and RR1 are carefully grouped to simplify status monitoring: e.g. when the interrupt vector indicates a Special Receive Condition interrupt, all the appropriate error bits can be read from a single register (RR1).

WRITE REGISTERS

The SCC contains 15 write registers (16 counting WR8, the transmit buffer) in each channel. These write registers are programmed separately to configure the functional "personality" of the channels. In addition, there are two registers (WR2 and WR9) shared by the two channels that may be accessed through either of them. WR2 contains the interrupt vector for both channels, while WR9 contains the interrupt control bits. Figure 10 shows the format of each write register.

![Figure 9. Read Register Bit Functions](230834-9)

![Figure 9. Read Register Bit Functions](230834-10)
Figure 9. Read Register Bit Functions (Continued)
Figure 9. Read Register Bit Functions (Continued)
Figure 10. Write Register Bit Functions
Figure 10. Write Register Bit Functions (Continued)
Figure 10. Write Register Bit Functions (Continued)
**Figure 10. Write Register Bit Functions (Continued)**

2-171
Figure 10. Write Register Bit Functions (Continued)
82530 TIMING

The SCC generates internal control signals from WR and RD that are related to CLK. Since CLK has no phase relationship with WR and RD, the circuitry generating these internal control signals must provide time for metastable conditions to disappear. This gives rise to a recovery time related to CLK. The recovery time applies only between bus transactions involving the SCC. The recovery time required for proper operation is specified from the rising edge of WR or RD in the first transaction involving the SCC to the falling edge of WR or RD in the second transaction involving the SCC. This time, \( T_{REC} \) must be at least 6 CLK cycles plus 130 ns, for the 82530-6.

Read Cycle Timing

Figure 11 illustrates Read cycle timing. Addresses on A/B and D/C and the status on INTA must remain stable throughout the cycle. If CS falls after RD falls or if it rises before RD rises, the effective RD is shortened.

![Figure 11. Read Cycle Timing](image)
Write Cycle Timing

Figure 12 illustrates Write cycle timing. Addresses on A/B and D/C and the status on INTA must remain stable throughout the cycle. If CS falls after WR falls or if it rises before WR rises, the effective WR is shortened.

Interrupt Acknowledge Cycle Timing

Figure 13 illustrates Interrupt Acknowledge cycle timing. Between the time INTA goes Low and the falling edge of RD, the internal and external IEI/IEO daisy chains settle. If there is an interrupt pending in the SCC and IEI is High when RD falls, the Acknowledge cycle is intended for the SCC. In this case, the SCC may be programmed to respond to RD Low by placing its interrupt vector on D0-D7 and it then sets the appropriate Interrupt-Under-Service internally.
ABSOLUTE MAXIMUM RATINGS*

Case Temperature
  Under Bias ...................... 0°C to +70°C

Storage Temperature
  Ceramic Package ............ −65°C to +150°C
  Plastic Package ............ −40°C to +125°C

Voltage on Any Pin with
  Respect to Ground ............ −0.5V to +7.0V

*Notice: Stresses above those listed under “Absolute Maximum Ratings” may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions above those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

D.C. CHARACTERISTICS  \( T_C = 0°C \) to 70°C; \( V_{CC} = +5V \pm 5\%

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Parameter</th>
<th>Min</th>
<th>Max</th>
<th>Units</th>
<th>Test Conditions</th>
</tr>
</thead>
<tbody>
<tr>
<td>VIL</td>
<td>Input Low Voltage</td>
<td>−0.3</td>
<td>+0.8</td>
<td>V</td>
<td></td>
</tr>
<tr>
<td>VIH</td>
<td>Input High Voltage</td>
<td>+2.4</td>
<td>( V_{CC} + 0.3 )</td>
<td>V</td>
<td></td>
</tr>
<tr>
<td>VOL</td>
<td>Output Low Voltage</td>
<td></td>
<td>+0.45</td>
<td>V</td>
<td>( I_{OL} = 2.0 \text{ mA} )</td>
</tr>
<tr>
<td>VOH</td>
<td>Output High Voltage</td>
<td></td>
<td>+2.4</td>
<td>V</td>
<td>( I_{OH} = -250 \mu A )</td>
</tr>
<tr>
<td>IIL</td>
<td>Input Leakage Current</td>
<td>±10</td>
<td></td>
<td>\mu A</td>
<td>0.4V to 2.4V</td>
</tr>
<tr>
<td>IOL</td>
<td>Output Leakage Current</td>
<td>±10</td>
<td></td>
<td>\mu A</td>
<td>0.4V to 2.4V</td>
</tr>
<tr>
<td>ICC</td>
<td>( V_{CC} ) Supply Current</td>
<td>250</td>
<td></td>
<td>mA</td>
<td></td>
</tr>
</tbody>
</table>

CAPACITANCE  \( T_C = 25°C \); \( V_{CC} = \text{GND} = 0V \)

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Parameter</th>
<th>Min</th>
<th>Max</th>
<th>Units</th>
<th>Test Conditions</th>
</tr>
</thead>
<tbody>
<tr>
<td>CIN</td>
<td>Input Capacitance</td>
<td>10</td>
<td></td>
<td>pF</td>
<td>( f_c = 1 \text{ MHz} )</td>
</tr>
<tr>
<td>Cout</td>
<td>Output Capacitance</td>
<td>15</td>
<td></td>
<td>pF</td>
<td>Unmeasured pins returned to GND</td>
</tr>
<tr>
<td>C_{I/O}</td>
<td>Input/Output Capacitance</td>
<td>20</td>
<td></td>
<td>pF</td>
<td></td>
</tr>
</tbody>
</table>
## A.C CHARACTERISTICS

$T_C = 0^\circ C \text{ to } +70^\circ C; V_{CC} = +5V \pm 5\%$

### READ AND WRITE TIMING

<table>
<thead>
<tr>
<th>Number</th>
<th>Symbol</th>
<th>Parameter</th>
<th>82530 (4 MHz)</th>
<th>82530-6 (6 MHz)</th>
<th>Units</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>tCL</td>
<td>CLK Low Time</td>
<td>105</td>
<td>70</td>
<td>ns</td>
</tr>
<tr>
<td>2</td>
<td>tCH</td>
<td>CLK High Time</td>
<td>105</td>
<td>70</td>
<td>ns</td>
</tr>
<tr>
<td>3</td>
<td>tf</td>
<td>CLK Fall Time</td>
<td>20</td>
<td>10</td>
<td>ns</td>
</tr>
<tr>
<td>4</td>
<td>tr</td>
<td>CLK Rise Time</td>
<td>20</td>
<td>15</td>
<td>ns</td>
</tr>
<tr>
<td>5</td>
<td>tCY</td>
<td>CLK Cycle Time</td>
<td>250</td>
<td>165</td>
<td>ns</td>
</tr>
<tr>
<td>6</td>
<td>tAW</td>
<td>Address to WR ↓ Setup Time</td>
<td>80</td>
<td>0</td>
<td>ns</td>
</tr>
<tr>
<td>7</td>
<td>tWA</td>
<td>Address to WR ↑ Hold Time</td>
<td>0</td>
<td>0</td>
<td>ns</td>
</tr>
<tr>
<td>8</td>
<td>tAR</td>
<td>Address to RD ↓ Setup Time</td>
<td>80</td>
<td>0</td>
<td>ns</td>
</tr>
<tr>
<td>9</td>
<td>tRA</td>
<td>Address to RD ↑ Hold Time</td>
<td>0</td>
<td>0</td>
<td>ns</td>
</tr>
<tr>
<td>10</td>
<td>tIC</td>
<td>INTA to CLK ↑ Setup Time</td>
<td>5</td>
<td>5</td>
<td>ns</td>
</tr>
<tr>
<td>11</td>
<td>tIW</td>
<td>INTA to WR ↓ Setup Time (Note 1)</td>
<td>200</td>
<td>55</td>
<td>ns</td>
</tr>
<tr>
<td>12</td>
<td>tWI</td>
<td>INTA to WR ↑ Hold Time</td>
<td>0</td>
<td>0</td>
<td>ns</td>
</tr>
<tr>
<td>13</td>
<td>tIR</td>
<td>INTA to RD ↓ Setup Time (Note 1)</td>
<td>200</td>
<td>55</td>
<td>ns</td>
</tr>
<tr>
<td>14</td>
<td>tRI</td>
<td>INTA to RD ↑ Hold Time</td>
<td>0</td>
<td>0</td>
<td>ns</td>
</tr>
<tr>
<td>15</td>
<td>tCI</td>
<td>INTA to CLK ↑ Hold Time</td>
<td>100</td>
<td>100</td>
<td>ns</td>
</tr>
<tr>
<td>16</td>
<td>tCLW</td>
<td>CS Low to WR ↓ Setup Time</td>
<td>0</td>
<td>0</td>
<td>ns</td>
</tr>
<tr>
<td>17</td>
<td>tWCS</td>
<td>CS to WR ↑ Hold Time</td>
<td>0</td>
<td>0</td>
<td>ns</td>
</tr>
<tr>
<td>18</td>
<td>tCHW</td>
<td>CS High to WR ↓ Setup Time</td>
<td>100</td>
<td>5</td>
<td>ns</td>
</tr>
<tr>
<td>19</td>
<td>tCLR</td>
<td>CS Low to RD ↓ Setup Time (Note 1)</td>
<td>0</td>
<td>0</td>
<td>ns</td>
</tr>
<tr>
<td>20</td>
<td>tRCS</td>
<td>CS to RD ↑ Hold Time (Note 1)</td>
<td>0</td>
<td>0</td>
<td>ns</td>
</tr>
<tr>
<td>21</td>
<td>tCHR</td>
<td>CS High to RD ↓ Setup Time (Note 1)</td>
<td>100</td>
<td>5</td>
<td>ns</td>
</tr>
<tr>
<td>22</td>
<td>tRR</td>
<td>RD Low Time (Note 1)</td>
<td>390</td>
<td>150</td>
<td>ns</td>
</tr>
<tr>
<td>23</td>
<td>Null</td>
<td>Parameter Deleted</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>24</td>
<td>tRDI</td>
<td>RD ↑ to Data Not Valid Delay</td>
<td>0</td>
<td>0</td>
<td>ns</td>
</tr>
<tr>
<td>25</td>
<td>tRDV</td>
<td>RD ↓ to Data Valid Delay</td>
<td>250</td>
<td>105</td>
<td>ns</td>
</tr>
<tr>
<td>26</td>
<td>tDF</td>
<td>RD ↑ to Output Float Delay (Note 2)</td>
<td>70</td>
<td>45</td>
<td>ns</td>
</tr>
</tbody>
</table>

**NOTES:**

1. Parameter does not apply to Interrupt Acknowledge transactions.
2. Float delay is defined as the time required for a $+0.5V$ change in the output with a maximum D.C. load and minimum A.C. load.
A.C. TESTING INPUT, OUTPUT WAVEFORM

A.C. Testing: Inputs are driven at 2.4V for a Logic "1" and 0.45V for a Logic "0". Timing measurements are made at 2.0V for a Logic "1" and 0.8V for a Logic "0".

A.C. TESTING LOAD CIRCUIT

DEVICE UNDER TEST

$C_L = 150 \text{ pF}$

$C_L$ Includes Jig Capacitance

Figure 14. Read and Write Timing
### INTERRUPT ACKNOWLEDGE TIMING, RESET TIMING, CYCLE TIMING

<table>
<thead>
<tr>
<th>Number</th>
<th>Symbol</th>
<th>Parameter</th>
<th>82530 (4 MHz)</th>
<th>82530-6 (6 MHz)</th>
<th>Units</th>
</tr>
</thead>
<tbody>
<tr>
<td>27</td>
<td>tAD</td>
<td>Address Required Valid to Read Data</td>
<td>Min 590</td>
<td>Min 325</td>
<td>ns</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Valid Delay</td>
<td>Max 390</td>
<td>Max 60</td>
<td></td>
</tr>
<tr>
<td>28</td>
<td>TWW</td>
<td>WR Low Time</td>
<td>0</td>
<td>0</td>
<td>ns</td>
</tr>
<tr>
<td>29</td>
<td>tDW</td>
<td>Data to WR \downarrow Setup Time</td>
<td>0</td>
<td>0</td>
<td>ns</td>
</tr>
<tr>
<td>30</td>
<td>tWD</td>
<td>Data to WR \uparrow Hold Time</td>
<td>0</td>
<td>0</td>
<td>ns</td>
</tr>
<tr>
<td>31</td>
<td>tWRV</td>
<td>WR \downarrow to Ready Valid Delay (Note 4)</td>
<td>240</td>
<td>200</td>
<td>ns</td>
</tr>
<tr>
<td>32</td>
<td>tRRV</td>
<td>RD \downarrow to Ready Valid Delay (Note 4)</td>
<td>240</td>
<td>200</td>
<td>ns</td>
</tr>
<tr>
<td>33</td>
<td>tWRI</td>
<td>WR \downarrow to READY/REQ Not Valid Delay</td>
<td>240</td>
<td>200</td>
<td>ns</td>
</tr>
<tr>
<td>34</td>
<td>tRRI</td>
<td>RD \downarrow to READY/REQ Not Valid Delay</td>
<td>240</td>
<td>200</td>
<td>ns</td>
</tr>
<tr>
<td>35</td>
<td>tDWR</td>
<td>WR \uparrow to DTR/REQ Not Valid Delay</td>
<td>5 tCY + 300</td>
<td>5 tCY + 250</td>
<td>ns</td>
</tr>
<tr>
<td>36</td>
<td>tDRD</td>
<td>RD \uparrow to DTR/REQ Not Valid Delay</td>
<td>5 tCY + 300</td>
<td>5 tCY + 250</td>
<td>ns</td>
</tr>
<tr>
<td>37</td>
<td>tIID</td>
<td>INTA to RD \downarrow (Acknowledge) Delay</td>
<td>250</td>
<td>250</td>
<td>ns</td>
</tr>
<tr>
<td>38</td>
<td>tII</td>
<td>RD (Acknowledge) Low Time</td>
<td>285</td>
<td>125</td>
<td>ns</td>
</tr>
<tr>
<td>39</td>
<td>tIDV</td>
<td>RD \downarrow (Acknowledge) to Read Data</td>
<td>190</td>
<td>100</td>
<td>ns</td>
</tr>
<tr>
<td>40</td>
<td>tEI</td>
<td>IEI to RD \downarrow (Acknowledge) Setup Time</td>
<td>120</td>
<td>100</td>
<td>ns</td>
</tr>
<tr>
<td>41</td>
<td>tIE</td>
<td>IEI to RD \uparrow (Acknowledge) Hold Time</td>
<td>0</td>
<td>0</td>
<td>ns</td>
</tr>
<tr>
<td>42</td>
<td>tEIEO</td>
<td>IEI to IEO Delay Time</td>
<td>120</td>
<td>100</td>
<td>ns</td>
</tr>
<tr>
<td>43</td>
<td>tCEQ</td>
<td>CLK \uparrow to IEO Delay</td>
<td>250</td>
<td>250</td>
<td>ns</td>
</tr>
<tr>
<td>44</td>
<td>tRII</td>
<td>RD \downarrow to INT Inactive Delay (Note 4)</td>
<td>500</td>
<td>500</td>
<td>ns</td>
</tr>
<tr>
<td>45</td>
<td>tRW</td>
<td>RD \uparrow to WR \downarrow Delay for No Reset</td>
<td>30</td>
<td>15</td>
<td>ns</td>
</tr>
<tr>
<td>46</td>
<td>tWR</td>
<td>WR \uparrow to RD \downarrow Delay for No Reset</td>
<td>30</td>
<td>30</td>
<td>ns</td>
</tr>
<tr>
<td>47</td>
<td>tRES</td>
<td>WR and RD Coincident Low for Reset</td>
<td>250</td>
<td>250</td>
<td>ns</td>
</tr>
<tr>
<td>48</td>
<td>tREC</td>
<td>Valid Access Recovery Time</td>
<td>6 tCY + 200</td>
<td>6 tCY + 130</td>
<td>ns</td>
</tr>
</tbody>
</table>

**NOTES:**

3. Parameter applies only between transactions involving the SCC.
4. Open-drain output, measured with open-drain test load.
5. Parameter is system dependent. For any SCC in the daisy chain, tIID must be greater than the sum of tCEQ for the highest priority device in the daisy chain, tEI for the SCC and tEIEO for each device separating them in the daisy chain.
Figure 15. Interrupt Acknowledge Timing

Figure 16. Reset Timing

Figure 17. Cycle Timing
## GENERAL TIMING

<table>
<thead>
<tr>
<th>Number</th>
<th>Symbol</th>
<th>Parameter</th>
<th>82530 (4 MHz)</th>
<th>82530-6 (6 MHz)</th>
<th>Units</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>Min</td>
<td>Max</td>
<td>Min</td>
</tr>
<tr>
<td>1</td>
<td>tRCC</td>
<td>RxC ↑ to CLK ↑ Setup Time (Notes 1, 4)</td>
<td>100</td>
<td>100</td>
<td>-200</td>
</tr>
<tr>
<td>2</td>
<td>tRRC</td>
<td>RxD to RxC ↑ Hold Time (X1 Mode) (Note 1)</td>
<td>0</td>
<td>0</td>
<td>3 tCY</td>
</tr>
<tr>
<td>3</td>
<td>tRCR</td>
<td>RxD to RxC ↑ Hold Time (X1 Mode) (Note 1)</td>
<td>150</td>
<td>150</td>
<td>3 tCY</td>
</tr>
<tr>
<td>4</td>
<td>tDRC</td>
<td>RxD to RxC ↓ Setup Time (X1 Mode) (Notes 1, 5)</td>
<td>0</td>
<td>0</td>
<td>+200</td>
</tr>
<tr>
<td>5</td>
<td>tRCD</td>
<td>RxD to RxC ↓ Hold Time (X1 Mode) (Notes 1, 5)</td>
<td>150</td>
<td>150</td>
<td>ns</td>
</tr>
<tr>
<td>6</td>
<td>tSRC</td>
<td>SYNC to RxC ↑ Setup Time (Note 1)</td>
<td>-200</td>
<td>-200</td>
<td>ns</td>
</tr>
<tr>
<td>7</td>
<td>tRCS</td>
<td>SYNC to RxC ↑ Hold Time (Note 1)</td>
<td>3 tCY</td>
<td>+200</td>
<td>3 tCY</td>
</tr>
<tr>
<td>8</td>
<td>tTCC</td>
<td>TxC ↓ to CLK ↑ Setup Time (Notes 2, 4)</td>
<td>100</td>
<td>100</td>
<td>ns</td>
</tr>
<tr>
<td>9</td>
<td>tTCT</td>
<td>TxC ↓ to TxD Delay (X1 Mode) (Note 2)</td>
<td>300</td>
<td>300</td>
<td>ns</td>
</tr>
<tr>
<td>10</td>
<td>tTCD</td>
<td>TxC ↑ to TxD Delay (X1 Mode) (Notes 2, 5)</td>
<td>300</td>
<td>300</td>
<td>ns</td>
</tr>
<tr>
<td>11</td>
<td>tTDT</td>
<td>TxD to TRxC Delay (Send Clock Echo)</td>
<td>200</td>
<td>200</td>
<td>ns</td>
</tr>
<tr>
<td>12</td>
<td>tDCH</td>
<td>RTxC High Time</td>
<td>180</td>
<td>150</td>
<td>ns</td>
</tr>
<tr>
<td>13</td>
<td>tDCL</td>
<td>RTxC Low Time</td>
<td>180</td>
<td>150</td>
<td>ns</td>
</tr>
<tr>
<td>14</td>
<td>tDCY</td>
<td>RTxC Cycle Time</td>
<td>4 tCY</td>
<td>4 tCY</td>
<td>ns</td>
</tr>
<tr>
<td>15</td>
<td>tCLCL</td>
<td>Crystal Oscillator Period (Note 3)</td>
<td>250</td>
<td>1000</td>
<td>165</td>
</tr>
<tr>
<td>16</td>
<td>tRCH</td>
<td>TRxC High Time</td>
<td>180</td>
<td>150</td>
<td>ns</td>
</tr>
<tr>
<td>17</td>
<td>tRCL</td>
<td>TRxC Low Time</td>
<td>180</td>
<td>150</td>
<td>ns</td>
</tr>
<tr>
<td>18</td>
<td>tRCY</td>
<td>TRxC Cycle Time (Note 6)</td>
<td>4 tCY</td>
<td>4 tCY</td>
<td>ns</td>
</tr>
<tr>
<td>19</td>
<td>tCC</td>
<td>CD or CTS Pulse Width</td>
<td>200</td>
<td>200</td>
<td>ns</td>
</tr>
<tr>
<td>20</td>
<td>tSS</td>
<td>SYNC Pulse Width</td>
<td>200</td>
<td>200</td>
<td>ns</td>
</tr>
<tr>
<td>21</td>
<td>tWRT</td>
<td>WR to RTS Valid Delay</td>
<td>6 tCY</td>
<td>6 tCY</td>
<td>ns</td>
</tr>
<tr>
<td>22</td>
<td>tWDT</td>
<td>WR to DTR Valid Delay</td>
<td>5 tCY</td>
<td>5 tCY</td>
<td>ns</td>
</tr>
</tbody>
</table>

**NOTES:**

1. RxC is RTxC or TRxC, whichever is supplying the receive clock.
2. TxC is TRxC or RTxC, whichever is supplying the transmit clock.
3. Both RTxC and SYNC have 30 pF capacitors to ground connected to them.
4. Parameter applies only if the data rate is one-fourth the system clock (CLK) rate. In all other cases, no phase relationship between RxC and CLK or TxC and CLK is required.
5. Parameter applies only to FM encoding/decoding.
6. Only applies to transmitter and receiver. For DPll and Baud Rate Generator Timings, the requirements are identical to system clock, CLK, specifications.
Figure 18. General Timing
Designing With the 82510 Asynchronous Serial Controller

FAISAL IMDAD-HAQUE
APPLICATIONS ENGINEER
1.0 INTRODUCTION

The emergence of asynchronous communications as the most widely used protocol (it commands the largest installed base of nodes, exceeding HDLC/SDLC, the second most popular protocol, by a factor of 10 to 1) for point to point serial links has led to the need for an asynchronous communications component with high integration to reduce component count and decrease the cost of a serial port. The trend towards higher data rates and multiple job, multiple user systems has underscored the need for an intelligent serial controller to improve system throughput and decrease the CPU load normally associated with asynchronous serial communications.

The 82510 CMOS Asynchronous Serial Controller is designed to improve asynchronous communications throughput and reduce system cost by integrating functions and simplifying the system interface. Two independent FIFOs, and Control Character Recognition (CCR), provide data buffering and increase software efficiency. Two Baud Rate Generators/Timers, an On-Chip Crystal Oscillator and seven Programmable I/O pins provide a high degree of integration and reduce system component count. This application note will demonstrate the use of these features in an Asynchronous Communications Environment.

1.1 Goal

The goal of this application note is to demonstrate the use of the major 82510 features in an asynchronous communications environment and to depict basic hardware and software design techniques for the 82510. It will discuss interfaces using both polling and interrupt techniques, as well as the impact of FIFOs using either scheme. An application example covering the application of Error Free File Transfer is also provided.

1.2 Scope

The application note describes the operation of the 82510 ASC in a normal (non 8051 9-bit) asynchronous communications mode. The majority of the discussion is focused towards the systems aspects of the Controller. The use of the 82510 in a multidrop or 8051 9-bit asynchronous environment is not covered. This application note assumes that the reader is familiar with the 82510 in terms of pin description, register architecture and interrupt structure. It is also assumed that the reader is familiar with the information provided in the 82510 Data Sheet.

The initial sections of the application note provide an overview of the 82510 and its major functional blocks.

This is followed by a discussion of the hardware design and system interface considerations in sections three and four. The fifth section provides some software techniques for transmitting and receiving data as well as the use of timers. Section seven briefly discusses the file transfer application based on the XMODEM protocol and includes the software listings.

2.0 82510 DESCRIPTION

2.1 Overview

The 82510 can be divided into seven functional blocks (See Figure 1): Bus Interface Unit (BIU), Timing Unit, Modem Interface Module, Tx FIFO, Rx FIFO, Tx Machine and Rx Machine. All blocks, except BIU can generate a block interrupt request to the 82510 interrupt logic. In the case of the Rx Machine, Timing Unit and Modem Interface Module, multiple sources (errors and status events) within the block cause the block interrupt request to become active. All of the blocks have registers associated with them. The registers, allow configuration, provide status information about events/errors, and may also be used to send commands to each block.

2.2 Bus Interface Unit (BIU)

The Bus Interface Unit (BIU) interfaces the 82510 functional blocks to the system or CPU bus. It provides read and write access to the 82510 registers and controls the generation of interrupts to the external world. The interrupt logic resolves contention between block interrupt requests, on a priority basis. The BIU also has the Hardware Reset circuitry, which is driven by the RESET pin. The reset signal clears all internal Flip Flops, and Registers and puts them in a predefined state. All activities on the Bus interface, including register accesses by the CPU, are synchronized to an internal (82510) system clock, supplied via the CLK pin.

2.3 Receive Machine (RxM)

The Rx Machine (RxM) converts the serial data to parallel and writes it to the Rx FIFO, along with the appropriate flags (available in the Receive Flags Register). The Rx Machine can be configured for control character recognition, data sampling and DPLL operation. The software can check for noise, control character, break, address or parity and framing errors by reading the status or character flags. Optionally, the Receive Status bits (in RST), when enabled, can generate interrupt requests. The Rx Machine block Interrupt request is reflected in the General Status Register and is set when an enabled interrupt request within the Rx Ma-
chine (i.e. RST bits) becomes active. The Rx Machine has eight registers associated with it:

Receive Data (RXD)—Receive Data Character
Receive Flags (RXF)—Receive Character Flags
Receive Status (RST)—Receive Events and Receive Errors
Receive Interrupt Enable (RIE)—Enables Interrupts on corresponding bits in RST
Receive Mode (RMD)—Receive Machine Configuration
Receive Command (RCM)—Receiver Command Register
Line Control (LCR)—16450 Register, Character Attribute Configuration
Line Status (LSR)—16450 Status Register, Tx and Rx status

2.4 Transmit Machine (TxM)

The Tx Machine reads characters from the Tx FIFO and transmits them serially over the TXD line. The Tx Machine can also transmit additional character attributes (9th bit of Data, Address Marker, Software Parity) available from the Transmit Flags, if configured in the appropriate mode. The Tx Machine Idle interrupt request is reflected in the GSR and LSR registers to indicate that the Transmitter is either Empty or Disabled. The Tx Machine has six registers associated with it:

Line Control (LCR)—16450 Register, Character Attribute Configuration
Line Status (LSR)—16450 Status Register, Tx and Rx status
Transmit Mode (TMD)—Tx Machine Configuration
Transmit Command (TCM)—Transmit Command Register
Transmit Flags (TXF)—Transmit Character Flags
Transmit Data (TXD)—Transmit Data Character

2.5 Modem Interface Module

The Modem Interface module is responsible for the modem interface and general purpose I/O pins. It will generate Interrupts (if enabled) upon transitions in the modem input pins (DCD,CTS,RI, and DSR). The modem output pins can be controlled by the CPU, also the RTS pin can be used to provide flow control, in the automatic transmission mode. It is the source of the Modem Interrupt bit in GSR. This bit is set whenever there is a state change in the DCD, RI, DSR or CTS inputs (reflected in Modem Status Register) and the corresponding enable bits are set. The function and direction of the multifunction pins can be reprogrammed and is available as a configuration option. Multifunction pins, when configured as outputs, can be controlled by the CPU through the Modem Control Register. The Modem module has four registers associated with it:

Modem Status Register (MSR)—State transitions on modem input pins, and State of the modem input pins
Modem Control (MCR)—Control state of Modem Output pins
Modem Interrupt Enable (MIE)—Enable Interrupt on State transitions in modem input pins
I/O Pin Mode (PMD)—Functions and Directions of Multifunction pins

2.6 Timing Unit

The Timing Unit is responsible for the generation of the System Clock, using either its Crystal Oscillator or an externally generated clock, and generation of the Tx and Rx clocks from either the On-Chip Baud Rate Generators or the SCLK pin. It is also responsible for generating Timer Expired interrupts when the BRGs/Timers are configured for use as Timers. There are ten registers associated with the Timing Unit, four of these are used in the Timer mode only.

Timer Status (TMST)—Timer A and/or Timer B expired
Timer Interrupt Enable (TMIE)—Enables Interrupts upon Expiration of Timers A or B in TMST
Timer Control (TMCR)—Start and Disable Timers
Clock Configure (CLCF)—Select source and mode for Tx and Rx clocks
BRG B Configuration (BBCF)—Mode and Clock source of BRG B
BRG B LSB of Divisor (BBL)—Least Significant Byte of BRG B Divisor/Count

BRG A MSB of Divisor (BBH)—Most Significant Byte of BRG B Divisor/Count

BRG A Configuration (BACF)—Mode and Clock source of BRG A

BRG A LSB of Divisor (BAL)—Least Significant Byte of BRG A Divisor/Count

BRG A MSB of Divisor (BAH)—Most Significant Byte of BRG A Divisor/Count

2.7 FIFOs, Rx and Tx

The Dual FIFOs (transmit and receive), serve as buffers for the 82510. They buffer the transmitter and Receiver from the CPU. Each of the FIFOs has a programmable threshold. The threshold is the FIFO level which will generate an interrupt. The threshold is used to optimize the CPU throughput and provide increased interrupt to service latency for higher baud rates. It can be configured through the FIFO Mode Register. Each FIFO character has flags associated with it (TxF and RxF). As each character is read from the Rx FIFO its flags are put into the RxF register. Before a write to TXD (if character configuration requires) the character flags are written to the TXF register. The two FIFOs are totally independent of each other and each FIFO can generate an interrupt request which indicates that the configured threshold has been met.

3.0 HARDWARE DESIGN

3.1 System Interface

The 82510 has a standard I/O peripheral interface, it has a demultiplexed Bus, which consists of a bidirectional eight bit Data Bus, and three Address lines. Interrupt, Read, Write, Chip Select and Reset pins complete the system interface. The three address lines along with the Bank register are used to select a particular register.

3.1.1 REGISTER ACCESS

The 82510 registers are logically divided into four banks. Only one bank can be accessed at any one time. Each register bank occupies eight I/O addresses. To select a register, the correct Bank must first be selected by writing to the GIR/Bank register (the GIR/Bank register I/O address is two (A0 = 0, A1 = 1, A2 = 0). Then one of the eight I/O space addresses is selected by outputting a value (between zero and seven) to the 82510 address pins A0–A2.
### BANK ZERO 8250—COMPATIBLE BANK

<table>
<thead>
<tr>
<th>Register</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0 Address</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>TxD</td>
<td>Tx Data bit 7</td>
<td>Tx Data bit 6</td>
<td>Tx Data bit 5</td>
<td>Tx Data bit 4</td>
<td>Tx Data bit 3</td>
<td>Tx Data bit 2</td>
<td>Tx Data bit 1</td>
<td>Tx Data bit 0</td>
<td>0 —</td>
</tr>
<tr>
<td>RxD</td>
<td>Rx Data bit 7</td>
<td>Rx Data bit 6</td>
<td>Rx Data bit 5</td>
<td>Rx Data bit 4</td>
<td>Rx Data bit 3</td>
<td>Rx Data bit 2</td>
<td>Rx Data bit 1</td>
<td>Rx Data bit 0</td>
<td>0 —</td>
</tr>
<tr>
<td>BAL</td>
<td>BRGA LSB Divide Count (DLAB = 1)</td>
<td>0</td>
<td>02H</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BAH</td>
<td>BRGA MSB Divide Count (DLAB = 1)</td>
<td>1</td>
<td>00H</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Register</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0 Address</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>GER</td>
<td>0</td>
<td>0</td>
<td>Timer Interrupt Enable</td>
<td>Tx Machine Interrupt Enable</td>
<td>Modern Interrupt Enable</td>
<td>Rx Machine Interrupt Enable</td>
<td>Tx FIFO Interrupt Enable</td>
<td>Rx FIFO Interrupt Enable</td>
<td>1 00H</td>
</tr>
<tr>
<td>GIR/BANK</td>
<td>0</td>
<td>BANK Pointer bit 1</td>
<td>BANK Pointer bit 0</td>
<td>0</td>
<td>Active Block Int bit 2</td>
<td>Active Block Int bit 1</td>
<td>Active Block Int bit 0</td>
<td>Interrupt Pending</td>
<td>2 01H</td>
</tr>
<tr>
<td>LCR</td>
<td>DLAB</td>
<td>Divisor</td>
<td>Set Break</td>
<td>Parity Mode bit 2</td>
<td>Parity Mode bit 1</td>
<td>Parity Mode bit 0</td>
<td>Stop bit Length bit 0</td>
<td>Character Length bit 1</td>
<td>Character Length bit 0</td>
</tr>
<tr>
<td>MCR</td>
<td>0</td>
<td>0</td>
<td>OUT 0 Complement</td>
<td>Loopback Control bit</td>
<td>OUT 2 Complement</td>
<td>OUT 1 Complement</td>
<td>RTS Complement</td>
<td>DTR Complement</td>
<td>4 00H</td>
</tr>
<tr>
<td>LSR</td>
<td>0</td>
<td>TxM Idle</td>
<td>Tx FIFO Interrupt</td>
<td>Break Detected</td>
<td>Framing Error</td>
<td>Parity Error</td>
<td>Overrun Error</td>
<td>Rx FIFO Int Reg</td>
<td>5 60H</td>
</tr>
<tr>
<td>MSR</td>
<td>DCD Input Inverted</td>
<td>RI Input Inverted</td>
<td>DSR Input Inverted</td>
<td>CTS Input Inverted</td>
<td>State Change in DCD</td>
<td>State Change in RI</td>
<td>State Change in DSR</td>
<td>State Change in CTS</td>
<td>6 00H</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Register</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0 Address</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACR0</td>
<td>Address or Control Character Zero</td>
<td>7</td>
<td>00H</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### BANK ONE—GENERAL WORK BANK

<table>
<thead>
<tr>
<th>Register</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0 Address</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>TxD</td>
<td>Tx Data bit 7</td>
<td>Tx Data bit 6</td>
<td>Tx Data bit 5</td>
<td>Tx Data bit 4</td>
<td>Tx Data bit 3</td>
<td>Tx Data bit 2</td>
<td>Tx Data bit 1</td>
<td>Tx Data bit 0</td>
<td>0 —</td>
</tr>
<tr>
<td>RxD</td>
<td>Rx Data bit 7</td>
<td>Rx Data bit 6</td>
<td>Rx Data bit 5</td>
<td>Rx Data bit 4</td>
<td>Rx Data bit 3</td>
<td>Rx Data bit 2</td>
<td>Rx Data bit 1</td>
<td>Rx Data bit 0</td>
<td>0 —</td>
</tr>
<tr>
<td>RxF</td>
<td>—</td>
<td>Rx Char OK</td>
<td>Rx Char Noisy</td>
<td>Rx Char Parity Error</td>
<td>Address or Control Character</td>
<td>Break Flag</td>
<td>Rx Char Framing Error</td>
<td>Ninth Data bit of Rx Char</td>
<td>1 —</td>
</tr>
<tr>
<td>TxF</td>
<td>Address Marker bit</td>
<td>Software Parity bit</td>
<td>Ninth bit of Data Char</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1 —</td>
</tr>
<tr>
<td>GIR/BANK</td>
<td>0</td>
<td>BANK Pointer bit 1</td>
<td>BANK Pointer bit 0</td>
<td>0</td>
<td>Active Block Int bit 2</td>
<td>Active Block Int bit 1</td>
<td>Active Block Int bit 0</td>
<td>Interrupt Pending</td>
<td>2 01H</td>
</tr>
<tr>
<td>TMST</td>
<td>—</td>
<td>—</td>
<td>Gate B State</td>
<td>Gate A State</td>
<td>—</td>
<td>—</td>
<td>Timer B Expired</td>
<td>Timer A Expired</td>
<td>3 30H</td>
</tr>
<tr>
<td>TMCR</td>
<td>0</td>
<td>0</td>
<td>Trigger Gate B</td>
<td>Trigger Gate A</td>
<td>0</td>
<td>0</td>
<td>Start Timer B</td>
<td>Start Timer A</td>
<td>3 —</td>
</tr>
<tr>
<td>MCR</td>
<td>0</td>
<td>0</td>
<td>OUT 0 Complement</td>
<td>Loopback Control bit</td>
<td>OUT 2 Complement</td>
<td>OUT 1 Complement</td>
<td>RTS Complement</td>
<td>DTR Complement</td>
<td>4 00H</td>
</tr>
</tbody>
</table>

Figure 2. 82510 Register Map
## BANK ONE—GENERAL WORK BANK (Continued)

<table>
<thead>
<tr>
<th>Register</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
<th>Address</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>FLR</td>
<td>—</td>
<td>—</td>
<td>Rx FIFO Level</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>Tx FIFO Level</td>
<td>4</td>
<td>00H</td>
<td></td>
</tr>
<tr>
<td>RST</td>
<td>Address/Control Character Received</td>
<td>—</td>
<td>Break Terminated</td>
<td>—</td>
<td>Break Detected</td>
<td>—</td>
<td>Framing Error</td>
<td>—</td>
<td>Parity Error</td>
<td>Overrun Error</td>
</tr>
<tr>
<td>RCM</td>
<td>Rx Enable</td>
<td>Rx Disable</td>
<td>Flush RxM</td>
<td>Flush Rx FIFO</td>
<td>Lock Rx FIFO</td>
<td>Open Rx FIFO</td>
<td>0</td>
<td>0</td>
<td>—</td>
<td></td>
</tr>
<tr>
<td>MSR</td>
<td>DCD Complement</td>
<td>RI Input Inverted</td>
<td>DSR Input Inverted</td>
<td>CTS Input Inverted</td>
<td>State Change in DCD</td>
<td>State Change in RI</td>
<td>State Change in DSR</td>
<td>State Change inCTS</td>
<td>6</td>
<td>00H</td>
</tr>
<tr>
<td>TCM</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td></td>
</tr>
<tr>
<td>GSR</td>
<td>—</td>
<td>—</td>
<td>Timer Interrupt</td>
<td>TxM Interrupt</td>
<td>Modem Interrupt</td>
<td>RxDM Interrupt</td>
<td>Tx FIFO Interrupt</td>
<td>Rx FIFO Interrupt</td>
<td>7</td>
<td>12H</td>
</tr>
<tr>
<td>ICM</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>Software Reset</td>
<td>Manual Int Acknowledge Command</td>
<td>Status Clear</td>
<td>Power Down Mode</td>
<td>0</td>
<td>7</td>
</tr>
</tbody>
</table>

## BANK TWO—GENERAL CONFIGURATION

<table>
<thead>
<tr>
<th>Register</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
<th>Address</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>FMD</td>
<td>0</td>
<td>0</td>
<td>Rx FIFO Threshold</td>
<td>0</td>
<td>0</td>
<td>Tx FIFO Threshold</td>
<td>1</td>
<td>00H</td>
<td></td>
<td></td>
</tr>
<tr>
<td>GIR/BANK</td>
<td>0</td>
<td>—</td>
<td>BANK Pointer bit 1</td>
<td>1</td>
<td>BANK Pointer bit 0</td>
<td>0</td>
<td>Active Block Int bit 2</td>
<td>Active Block Int bit 1</td>
<td>Active Block Int bit 0</td>
<td>Interrupt Pending</td>
</tr>
<tr>
<td>TMD</td>
<td>Error Echo Disable</td>
<td>Control Character Echo Disable</td>
<td>9-bit Character Length</td>
<td>Transmit Mode</td>
<td>Software Parity Mode</td>
<td>Stop Bit Length</td>
<td>3</td>
<td>00H</td>
<td></td>
<td></td>
</tr>
<tr>
<td>IMD</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>Interrupt Acknowledge Mode</td>
<td>Rx FIFO Depth</td>
<td>Loopback or Echo Mode of Operation</td>
<td>4</td>
<td>0CH</td>
<td></td>
</tr>
<tr>
<td>ACR1</td>
<td>—</td>
<td>—</td>
<td>Address or Control Character</td>
<td>1</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td></td>
</tr>
<tr>
<td>RIE</td>
<td>Address/Control Character Recognition Interrupt Enable</td>
<td>Address/Control Character Match Interrupt Enable</td>
<td>Break Terminate Interrupt Enable</td>
<td>Break Detect Interrupt Enable</td>
<td>Framing Error Interrupt Enable</td>
<td>Parity Error Interrupt Enable</td>
<td>Overrun Error Interrupt Enable</td>
<td>0</td>
<td>6</td>
<td>1EH</td>
</tr>
<tr>
<td>RMD</td>
<td>Address/Control Character Mode</td>
<td>Disable DPLL</td>
<td>Sampling Window Mode</td>
<td>Start bit Sampling Mode</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>7</td>
<td>00H</td>
<td></td>
</tr>
</tbody>
</table>

## BANK THREE—MODEM CONFIGURATION

<table>
<thead>
<tr>
<th>Register</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
<th>Address</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLCF</td>
<td>Rx Clock Mode</td>
<td>Rx Clock Source</td>
<td>Tx Clock Mode</td>
<td>Tx Clock Source</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>00H</td>
<td></td>
</tr>
<tr>
<td>BACF</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>BRGA</td>
<td>Mode</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>BBL</td>
<td>BRGB LSB Divide Count (DLAB = 1)</td>
<td>0</td>
<td>05H</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BBH</td>
<td>BRGB MSB Divide Count (DLAB = 1)</td>
<td>1</td>
<td>00H</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

*Figure 2. 82510 Register Map (Continued)*

2-187
### BANK THREE—MODEM CONFIGURATION (Continued)

<table>
<thead>
<tr>
<th>Register</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
<th>Address</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>GIR/BANK</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0</td>
<td></td>
<td></td>
<td>2</td>
<td>01H</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0</td>
<td></td>
<td></td>
<td>0</td>
<td></td>
<td></td>
<td>BRGB</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Mode</td>
<td></td>
</tr>
<tr>
<td>BBCF</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>3</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>84</td>
</tr>
<tr>
<td>PMD</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MIE</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td>5</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0FH</td>
</tr>
<tr>
<td>TMIE</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>6</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>00H</td>
</tr>
</tbody>
</table>

**Figure 2. 82510 Register Map (Continued)**

#### 3.1.2 READ AND WRITE CYCLES

Like most other I/O based peripherals the Read and Write pins are used to access data in the 82510. Each read or write cycle has specified setup and hold times in order for the data to be transferred correctly to/from the 82510. The critical timings for the read cycle are:

1. Address Valid to Read Active (Tavl)
2. Command Access Time to Data Valid (Trldv)
3. Command Active Width (Trlrh)

The less critical parameters are:

4. Address Hold to Read Inactive (Trhax)
5. Data Out Float Delay after Read Inactive (Trhdz)

The critical timings for the write cycle are:

1. Address Valid to Write Low (Tavl)
2. Write Active Time (Twlwh)
3. Data Valid to Write Inactive (Tdvwh)

The less critical parameters are:

4. Address and Chip Select Hold Time After Write Inactive (Twhax)
5. Data Hold Time After Write Inactive (Twhdx)

These timings determine the number of wait states required for the 82510 and the CPU interface. The interfaces for some popular microprocessors are discussed in the following sections.

#### 3.1.3 80186 INTERFACE

The exact interface is shown in Figure 3. The schematic shows the 80186 interface to the 82510 on a local bus. Although the Data Bus is buffered, it is possible to directly connect the 82510 to the 80186 data bus; because the Data Float Delay after read inactive is 40 ns for the 82510, which is well under the 85 ns requirement of the 80186. The timing equations for the interface are given below.

**Read Cycle:**

Address to Read Low = Tclcl − Tcav_{max} + Tcl_{min} − Latch Delay_{max}

Read Access Time = 2Tccl − Tcl_{max} − Tdvcl − Transceiver Delay_{max}

Read Active Time = Trlrh = 2Tccl − 46

**Write Cycle:**

Address Valid to Write Active = Tclcl + Tcvct_{min} − Tcav_{max} − Latch Prop. Delay_{max}

Write Active Time = Twlwh = 2Tccl − 40

Data Valid to Write Inactive = 2Tccl − Tcldv_{max} − Transceiver Delay_{max} + Tcvct_{min}
Figure 3. 82510 Interface to 80186
The user can transfer data to the 82510, using the DMA capabilities of the 80186, by using the RTS pin, in automatic modem control mode, as a DMA request line. The RTS pin, in automatic mode, will go inactive as soon as the Tx FIFO and the Tx shift register are empty. It will become active once a data character is written to the TXD register.

In most 80186 DMA transfers the user has to make sure that the DMA request line goes inactive at least two clock cycles from the end of the DMA deposit cycle. In this case, the extra DMA cycle is not a problem, because the Tx FIFO will buffer the data to prevent an overrun (Since the Tx FIFO can buffer up to four characters, the RTS pin only needs to go inactive two clocks before the end of the deposit phase of the fourth DMA). Typically RTS will go inactive five (82510) system clocks after the rising edge of write.

3.1.4 80286 INTERFACE

The 80286 interface is shown in Figure 4. The 82510 is on the local bus, and is using the control signals from the 82288 Bus Controller. The Data Enable (OE) is qualified by the 82510 Chip Select, to avoid Data Bus contention between the 82510 and the CPU. The timing equations for the Read and Write Cycles are given below.

Read Cycle:

Address Valid to Read Active = T1 (CLK period) + T29min (CLK to cmd active) - T16max (ALE active delay) - Latch Prop. Delaymax

Read Access to Data Valid = 2T1 (CLK period) - T29max (CLK to cmd active) - T8 (Read Data Setup Time) - Transceiver Delaymax

Read Active Time = 2T1 (CLK period) - T29max (CLK to cmd active) + T30min (CLK to cmd inactive)

Write Cycle:

Address to Write Low = T1 (CLK period) + T29min (CLK to cmd active) - T16max (ALE active delay) - Latch Delaymax

Write Active Time = 2 T1 (CLK period) - T29max (CLK to cmd active) + T30min (CLK to cmd inactive)

Data to Write High = 3 T1 - T14min (Write Data Valid Delay) + T30min (CLK to cmd inactive) - Xcvr. Delaymax

Using an 8 MHz 80286 with the 82510 at 18.432 MHz (divide by two—9.216 MHz) requires two wait states. The critical timings are the read cycle timings—Read Access Time and Read Active Width. Inserting two wait states means that the access times for the relevant parameters will be increased by 250 ns.

NOTE:
The address decoding scheme of the 80286 interface is different from the IBM PC/PC AT I/O addresses for the serial ports, therefore the interface shown in Figure 4 cannot be used in PC/PC AT oriented designs.

3.1.5 80386 INTERFACE

The 80386 interface to the 82510 is given in Figure 5. The example uses the Basic I/O interface given in the 80386 Hardware Reference Manual section 8.3. The only differences are in the specific address lines used for chip select generation, and the additional wait states in the wait state generation logic. The address lines A3, A4 and A5 are used to select one of the eight register address spaces in the 82510, therefore, A6 and A7, rather than A4 and A5, are used in the I/O decoder. This causes a granularity of four in the 82510's I/O address space, i.e., the addresses of two consecutive registers in the 82510 differ by four.

The 82510 requires one additional wait state (as currently specified), the design assumes that the PAL equations are modified for that purpose. The user may also externally generate the wait states and connect to the "other ready logic" input ORed with the RDY pin of PAL 2. The two read timings Read Active width and Read Access time to Data Valid each require one additional wait state in order to meet the 82510 timing requirements. The timings are given below. (82510 times are at 9.216 MHz)

Read Cycle:

Read Access to Data Valid = 253.25 ns

82510 Trldv = 308

additional time reqd. = 308-253.25

= 54.75 ns

Read Active Width = 269.25

82510 Trirh = 308

additional time reqd. = 308-269.25

= 38.75 ns

Address Valid to Read Active = 132.75 ns

82510 TAVRL = 7 ns

Since each additional wait state adds 62.5 ns at 16 MHz, the 82510 requires one additional wait state.
Figure 5. 80386 Interface to the 82510
The required recovery time between successive commands is 123 ns for the 82510, this is well within the 331.75 ns provided by the Basic I/O interface.

Write Cycle:
- Addven to Write Low = 132.75 ns
- 82510 $T_{AVWL} = 7$ ns
- Write Active Time = 300.5 ns
- 82510 $T_{WLWH} = 231$ ns
- Data to Write High = 289.5 ns
- 82510 $T_{OVWH} = 90$ ns

NOTE:
The interface shown in Figure 5 uses a different address decoding scheme than that used for the IBM PC/PC AT families, for the serial ports. Therefore, the interface in Figure 5 can not be used in PC/PC AT compatible designs.

3.2 Reset
The 82510 can be reset either through hardware (Reset pin) or Software (reset command via Internal Command Register-ICM). Either reset would cause the 82510 to return to its default wake up mode. In this mode the register contents are reset to their default values and the device is in the 16450 compatible configuration. The Reset pulse must be held active for at least eight system clocks, the system clock should be running during reset active time.

3.2.1 DEFAULT MODES FOR 16450 COMPATIBILITY

Upon reset the 82510 will return to its Default Wake Up mode. The default register bank is bank zero. The registers in bank zero are identical to the 16450 register set, and provide complete software compatibility with the 16450* in the IBM PC environment. The registers in the other banks have default values, which configure the 82510 for 16450 emulation. The recommended system clock (for PC compatibility) is 18.432 MHz, this allows the baud rate generation to be done in a manner compatible with the PC software. Also since in the PC family the interrupt request pin of the UART is gated by the OUT2 pin, The OUT2 pin must be available in the 16450 compatibility mode, consequently the user is restricted to an external clock source when using the 82510 in the IBM PC compatible mode. The default pin out is given in Figure 6 and the configuration is given in Table 1. The default register values are given in the 82510 register map shown in Figure 2 in section 3.1.1.

Table 1. 82510 Default Configuration

<table>
<thead>
<tr>
<th>INTERRUPTS</th>
<th>Auto Acknowledge</th>
<th>All Interrupts Disabled</th>
</tr>
</thead>
<tbody>
<tr>
<td>RECEIVE</td>
<td>Stand Ctrl. Char. Recogn. disabled</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Digital Phase Locked Loop (DPLL) disabled</td>
<td></td>
</tr>
<tr>
<td></td>
<td>3/16 Sampling</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Majority Vote Start bit</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Non μlan (Normal) mode</td>
<td></td>
</tr>
<tr>
<td></td>
<td>BkD, FE, OE, PE Int. enabled</td>
<td></td>
</tr>
<tr>
<td>FIFO</td>
<td>Rx FIFO Depth = 1</td>
<td>Tx FIFO Threshold = 0</td>
</tr>
<tr>
<td>AUTO ECHO</td>
<td>Disabled</td>
<td></td>
</tr>
<tr>
<td>LOOP BACK</td>
<td>Configured</td>
<td></td>
</tr>
<tr>
<td></td>
<td>for Local Loopback</td>
<td></td>
</tr>
<tr>
<td>CLOCK OPTIONS</td>
<td>Baud Rate = 57.6K</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Rx Clock = 16 x</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Rx Clock Source = BRG B</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Tx Clock = 16 x</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Tx Clock Source = BRG B</td>
<td></td>
</tr>
<tr>
<td></td>
<td>BRG A Mode = BRG</td>
<td></td>
</tr>
<tr>
<td></td>
<td>BRG A Source = Sys. Clock</td>
<td></td>
</tr>
<tr>
<td></td>
<td>BRG B Mode = BRG</td>
<td></td>
</tr>
<tr>
<td></td>
<td>BRG B Source = BRG A Output</td>
<td></td>
</tr>
<tr>
<td>TRANSMIT</td>
<td>Manual Control of RTS</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1 Stop Bit</td>
<td></td>
</tr>
<tr>
<td></td>
<td>No Parity</td>
<td></td>
</tr>
<tr>
<td></td>
<td>5 Bit Character</td>
<td></td>
</tr>
</tbody>
</table>

*16450 is the PC AT version of the INS 8250A.
3.3 System Clock Options

The term “System Clock” refers to the clock which provides timings for most of the 82510 circuitry. The 82510 has two modes of system clock usage. It can generate its system clock from its On-Chip Crystal Oscillator and an external crystal, or it can use an externally generated clock, input to the device through the CLK pin. The selection of the system clock option is done during reset. The default system clock source is an externally generated clock, which can be reconfigured by a strapping option on the RTS pin. During Reset, the RTS pin is an input; it is internally pulled high, if it is externally driven low, then the 82510 expects to use the Crystal Oscillator for system clock generation; otherwise it is set up for using an external clock source. This can be done by using an open collector inverter to RTS, the input of the inverter is the Reset signal. The 82510 has a pull up resistor in the RTS circuitry so no external pull up is needed. In the crystal oscillator mode the CLK/X1 pin is automatically configured to X1, and the OUT2/X2 pin is configured to X2. In the External Clock mode, the CLK/X1 is configured to CLK and the OUT2/X2 is configured to OUT2.

If the Crystal Oscillator is being used to supply the system clock, then the clock frequency is always divided by two before being fed into the rest of the 82510 circuitry. If, however an external clock source is being used to supply the system clock, then the user has two options:

1. Use the System Clock after division by two, e.g. if a 8 MHz clock is being fed into the CLK pin, then the actual frequency of the 82510 system clock will be 4 MHz (default).

2. Disable Division by two and use the direct undivided clock, e.g. if an 8 MHz clock is being fed into the CLK pin, then the actual frequency of the 82510 system clock is also 8 MHz.

The divide by two option is the default mode of operation in the External Clock mode of the 82510. A strapping option can be used to disable the Divide By Two operation (For Crystal Oscillator Mode Divide By Two must always be active). During Reset, the DTR pin is an input; it is internally pulled high, if it is externally driven low then the Divide By Two operation is disabled. The strapping option is identical to the one used on RTS for selection of the System Clock source.

The 82510 system clock must be chosen with care since it influences the wait state performance, Baud Rate Generation (if being used as source frequency for the BRGs), the power consumption, and the Timer counting period. The power consumption of the 82510 is dependent upon the system clock frequency. If using the system clock as a source for the Baud Rate Generator(s), then the system clock frequency must be a baud rate multiple in order to minimize frequency deviation. For standard baud rates a multiple of 1.8432 MHz can be used, in fact the 18.432 MHz maximum frequency was chosen with this particular criteria in mind.
Figure 9. Timing Flow of the 82510
3.3.1 POWER DOWN MODE

The 82510 has a "power down" mode to reduce power consumption when the device is not in use. The 82510 powers down when the power down command is issued via the Internal Command Register (ICM). There are two modes of power down, Power Down Sleep and Power Down Idle.

3.3.1.1 Sleep Mode

This is the mode when even the system clock of the 82510 is shut down. The system clock source of the 82510 can either be the Crystal Oscillator or an external clock source. If the Crystal Oscillator is being used and the power down command is issued, then the 82510 will automatically enter the Sleep mode. If an external clock is being used, then the user must disable the external clock in addition to issuing the Power Down command, to enter the Sleep mode. The benefit of this mode is the increased savings in power consumption (typical power consumption in the Sleep mode is in the range of hundreds of microAmps. However, upon wake up, if using a crystal oscillator, the user must reprogram the device. The data is preserved if the external clock is disabled after the power down command, and enabled prior to exiting the power down mode. To exit this mode the user can either issue a Hardware reset, or read the FIFO Level Register (FLR) and then issue a software reset (if using a Crystal Oscillator). In either case the contents of the 82510 registers are not preserved and the device must be reprogrammed prior to operation.

NOTE:
If the Crystal Oscillator is being used then the user must allow about 1 ms for the oscillator to wake up before issuing the software reset.

3.3.1.2 Idle Mode

The 82510 is said to be in the Idle mode when the Power Down command is issued and the system clock is still running (i.e. the system clock is generated externally and not disabled by the user). In this mode the contents of all registers and memory cells are preserved, however, the power consumption in this mode is greater than in the Sleep mode. Reading FLR will take the 82510 out of this mode.

NOTE:
The data read from FLR when exiting Power Down is incorrect and must be ignored.

4.0 INTERRUPT BEHAVIOR

4.1 FIFO Usage

The 82510 has two independent four bytes transmit and receive FIFOs. Each FIFO can generate an interrupt request, when the FIFO level meets the Threshold requirements. The FIFOs can have a considerable impact on the performance of an asynchronous communications system. For systems using high baud rates they can provide increased interrupt-to-service latency reducing the chances of an overrun occurring. In systems constrained for CPU time, the FIFOs can increase the CPU Bandwidth by reducing the number of interrupt requests generated during asynchronous communications. It can reduce the interrupt load on the CPU by up to 75%. By choosing the FIFO thresholds which reflect the system bandwidth or service latency requirements, the user can achieve data rates and system throughput, unattainable with traditional UARTs.

Table 2. The Power Down Modes

<table>
<thead>
<tr>
<th>Mode</th>
<th>Clock Source</th>
<th>Exit Procedure</th>
<th>Power Consumption</th>
<th>Data Preservation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sleep</td>
<td>Crystal Oscill.</td>
<td>H/W Reset or Read FLR and Issue S/W Reset</td>
<td>100–900 µA</td>
<td>Not Preserved</td>
</tr>
<tr>
<td></td>
<td>Automatically Disabled</td>
<td></td>
<td></td>
<td>Must be Reprogrammed</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>External Clock</td>
<td>Enable External Clock, Read FLR and Issue S/W Reset</td>
<td>100–900 µA</td>
<td>Not Preserved</td>
</tr>
<tr>
<td></td>
<td>Must be Disabled by User</td>
<td>H/W Reset</td>
<td></td>
<td>Must be Reprogrammed</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Read FLR</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Idle</td>
<td>External Clock</td>
<td>H/W Reset</td>
<td>1–3 mA</td>
<td>All Data Preserved</td>
</tr>
<tr>
<td></td>
<td>Running</td>
<td>Read FLR</td>
<td></td>
<td>Does Not Need to be Reprogrammed</td>
</tr>
</tbody>
</table>

2-196
4.1.1 INTERRUPT-TO-SERVICE LATENCY

The interrupt-to-service latency is the time delay from the generation of an interrupt request, to when the interrupt source in the 82510 is actually serviced. Its primary application is in the reception of data. In traditional UARTs the CPU must read the current character in the Receive Buffer before it is overrun by the next incoming character. The Rx FIFO in the 82510 can buffer up to four characters, allowing an interrupt-to-service latency of up to four character transmission times. The character transmission time is the time period required to transmit one full character at the given Baud Rate. It is dependent upon the baud rate and is given by equation (1):

\[
\text{Character Transmission Time} = \frac{\text{Num. of Bits per Character Frame}}{\text{Baud Rate}}
\]

The Transmit and Receive FIFO thresholds should be selected with consideration to two factors the Baud rate, and the (CPU Bandwidth allocated for Asynchronous Channels is dependent upon the number of channels supported since it does not include the overhead of supporting other peripherals) number of Asynchronous Serial ports being supported by the CPU. In order to avoid overrun, the interrupt-to-service delay must be less than the time it takes to fill the 82510 Rx FIFO. The relationship is given by equation (2):

\[
\text{Int_to_Service latency} < \text{Buffer size} \times \frac{10}{\text{Baud rate}}
\]

Going back to equation (2):

\[
\text{Int_to_Service latency} = \frac{\# \text{ of Channels} \times (\# \text{ of int. sources per channel}) \times \text{Time required to service interrupt}}{4 \times 2 \times \text{Time required to service interrupt}}
\]

The Time required to service interrupt has been calculated to be 100 μs for a slightly optimized service routine. RMX86 interrupt service time is given as 250 μs and for other operating systems it should be slightly higher.

Example

Calculate the maximum baud rate that can be supported by a 6 MHz PC AT to support four Full Duplex Asynchronous channels using

a) The 82510 with four byte FIFO.

b) The 82510 with one byte FIFO.

Assumptions:

- CPU dedicated to Asynchronous communications.
- UART Interrupts limited to Transmission and Reception only.
- Interrupt Routines are optimized for fast throughput.
- 10 bits per character frame.

4.2 Interrupt Handling

The 82510 has 16 different sources of interrupt, each of these sources, when set and enabled, will cause their respective block interrupt requests to go active. The block interrupt request, if enabled, will set the 82510’s INT pin high, and will be reflected as a pending interrupt in the General Interrupt Register (GIR) if no other higher priority block is requesting service. If a higher priority block interrupt is also active at the same time, then the General Interrupt Register will reflect the higher priority request as the source of the 82510 interrupt. The lower priority interrupt will issue a new edge on the interrupt pin only after the higher priority interrupt is acknowledged and if no other priority block requests are present. Both the block interrupts and the individual sources within the blocks are maskable. The block interrupts are enabled through the General Enable Register (GER) which prevents masked bits in the General Status Register (GSR) from being decoded into the General Interrupt Register. This does not prevent the block request from being set in the General Status Register, it only prevents the masked GSR bits from being decoded into the General Interrupt Register, and thus generating any interrupts. The individual sources within the block are masked out via the corresponding interrupt enable register associated with the specific block (Rx Machine, Timing Unit and the Modem I/O module each have an Interrupt Enable register).
Figure 9. 82510's Interrupt Scheme
4.2.1 THE INTERRUPT SCHEME

The 82510 interrupt logic consists of the following elements:

4.2.1.1 Interrupt Sources Within Blocks

Three of the 82510 functional blocks (Rx Machine, Timer, Modem I/O) have more than one possible source of interrupts, for instance the Rx Machine has seven different sources of interrupts—standard control character recognition (Std. CCR), control character Match (special CCR), Break Detect, Break Terminated, Overrun Error, Parity Error, and Framing Error. The multiple sources are represented as Status bits in the Status registers of each of these blocks. When enabled the Status bits cause the block request to set in the General Status Register. There is no difference in the behavior of the INT pin or the block status bits in GSR, for multiple sources within a block becoming active simultaneously. The corresponding block status bit in GSR is set when one or more interrupt sources within the block become active. When the status register for the block is read all the active interrupt sources within the block are reset. Each source within the three blocks can be masked through its respective enable register.

4.2.1.2 General Status Register (GSR)

This register holds the status of the six 82510 blocks (all except Bus Interface Unit). Each bit when set indicates that the particular block is requesting interrupt service, and if enabled via the General Enable Register, will cause an interrupt.

4.2.1.3 General Enable Register (GER)

This register is used to enable/disable the corresponding bits in the General Status Register. It can be programmed by the CPU at any time.

<table>
<thead>
<tr>
<th>Block</th>
<th>Priority</th>
<th>GIR CODE 3 2 1 (Bits)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Timers</td>
<td>5 (highest)</td>
<td>1 0 1</td>
</tr>
<tr>
<td>Tx Machine</td>
<td>4</td>
<td>1 0 0</td>
</tr>
<tr>
<td>Rx Machine</td>
<td>3</td>
<td>0 1 1</td>
</tr>
<tr>
<td>Rx FIFO</td>
<td>2</td>
<td>0 1 0</td>
</tr>
<tr>
<td>Tx FIFO</td>
<td>1</td>
<td>0 0 1</td>
</tr>
<tr>
<td>Modem I/O</td>
<td>0 (lowest)</td>
<td>0 0 0</td>
</tr>
</tbody>
</table>

4.2.1.4 Priority Resolver and General Interrupt Register

If more than one enabled Interrupt request from GSR is active, then the priority resolver is used to resolve contention. The priority resolver finds the highest priority pending and enabled interrupt in GSR and decodes it into the General Interrupt Register (bits 3 to 1). The General Interrupt Register can be read at any time.

NOTE:
GIR is updated continuously, so while the user may be serving one interrupt source, a new interrupt with higher priority may update GIR and replace the older one.

4.2.2 INTERRUPT ACKNOWLEDGE MODES

The 82510 has two modes of interrupt acknowledge—Manual acknowledge and Automatic acknowledge. In Manual Acknowledge mode, the user has to issue an explicit Acknowledge Command via the Internal Command Register (ICM) in order to cause the INT pin to go low. In Automatic Acknowledge mode the INT pin will go low as soon as an active or pending interrupt request is serviced by the CPU. An operation is considered to be a service operation if it causes the source of the interrupt (within the 82510) to become inactive (the specific status bit is reset). The service procedures for each source vary, see section 4.2.3.2 for details.

4.2.2.1 Automatic Acknowledgement

In the automatic acknowledge mode, a service operation by the CPU will be considered as an automatic acknowledgement of the interrupt. This will force the INT pin low for two clock cycles, after that the INT pin is updated i.e. if there is an active enabled source pending then the INT pin is set high again (reflected in GIR). This mode is useful in an edge triggered Interrupt system. Servicing any enabled and active GSR bit will cause Auto Acknowledge to occur (independently of the source currently decoded in the GIR register). This can be used to rearrange priorities of the 82510 block requests.
GSR 5
TIMER

GSR 3 (MODEM)

GSR 1 (TX FIFO)

GIR

VECTOR = 10

0

VECTOR = 02

VECTOR = 0

GIR = 1

INT

8259A

GIR

USER

OPERATIONS

READ GIR

ISSUE EOI

SERVE MODEM

ISSUE EOI

READ GIR

ISSUE EOI

READ GIR

SERVE MODEM

ISSUE EOI

8259A → Edge Triggered
Non Auto EOI
82510 Automatic Acknowledge

231928-10

Figure 10. Automatic Acknowledge Mode Operation

NOTE:
Vector refers to GIR bit (3-0)
82510: Manual Ack. Mode
8259A: Edge Triggered Non AEOI

231928-11

Figure 11. Manual Acknowledge Mode Operation
4.2.2.2 Manual Mode of Acknowledgement

The Manual Acknowledgement Mode requires that, unlike the automatic mode where a service operation is considered as an automatic acknowledge, an explicit acknowledge command be issued to the 82510 to cause INT to go inactive. In this mode the CPU has complete control over the timing of the Interrupts. Before exiting the service routine, the CPU can check the GIR register to see if other interrupts are pending and can service those interrupts in the same invocation, avoiding the overhead of another interrupt as in the Automatic mode. Of course the user has the option of issuing the acknowledge command immediately after the service, which would be similar in behavior to the automatic mode. If the manual acknowledge command is given before the active source has been serviced and no higher priority request is pending, then the same source will immediately generate a new interrupt. Therefore, the software must make sure that the Manual Acknowledge command is issued after the interrupt source has been serviced by the CPU (see section 4.2.3.2. for more details on interrupt service procedures for each source).
4.2.3 GENERAL INTERRUPT HANDLER

In general an interrupt handler for the 82510 must first identify the interrupt source within the 82510, transfer control to the appropriate service routine and then service the active source. The active source can be identified from two registers—General Interrupt Register, or General Status Register. The GIR register identifies the highest priority active block interrupt request. The GSR register identifies all active (pending or in service) Block Interrupt Requests. The typical operation of the 82510 interrupt handler is given in Figure 12. The two major issues of concern are the source identification and Control Transfer to the appropriate service routine.

Since the 82510 registers are divided into banks, and the interrupt handler may change register banks during service, it is best to save the bank being used by the main program and then do the interrupt processing. Upon completion of service, the original bank value is restored to the GIR/Bank register.

4.2.3.1 Source Identification

The 82510 has 16 interrupt sources, and the CPU must identify the source before performing any service. Although the procedure varies, the typical method would be to identify the block requesting service by reading

![Diagram](231928-13)  
*Figure 13. Bypassing the 82510 Fixed Interrupt Priority*
GIR bits 3-1. If the source is either Tx Machine, Tx FIFO, or Rx FIFO, no further identification is needed, the user can transfer control to the service routine (in most cases, only one Timer will be used, therefore the Timer Routine can also be directly invoked). All modem I/O interrupts can be handled via one routine as all the modem interrupt sources are supplementary to the modem handshaking function. The Rx Machine, however, has two different types of interrupt sources, event indications (CCR/Address recognition CCR/Address Match, Break Detect, Break Terminate, and Overrun Error), and error indications (Parity Error, Framing Error, these error indications do not refer to any particular character, they just indicate that the specific error was detected during reception). For most applications, the error indicators can be masked off, and only the event driven interrupts enabled. The error indicators can be read from the Receive Flags prior to reading a character from the FIFO. This interrupt scheme can be used, because the Receive character error indicators are available in the Receive Flags, and can be checked by the Receive routine before reading the character from the Rx FIFO.

Since all active status bits (except Rx FIFO interrupt in LSR and RST) are reset when the corresponding block status register is read, the interrupt routine must check for all possible active sources within the block, and service each active source before exiting the interrupt handler.

The 82510 interrupt contention is resolved on a fixed priority basis. In some applications the fixed priority may not be suitable for the user. For these cases the user can bypass the 82510's priority resolution by using the General Status Register (rather than GIR) to determine the block interrupt sources requesting service. Each source is checked in order of user priority and serviced when identified (There will be no problem with using this algorithm in auto acknowledge mode because the INT pin will go low as soon as a pending and enabled interrupt request goes low). The user will be trading some service latency time for additional source identification time, this algorithm’s efficiency will improve as the number of block sources to verify is reduced. See Figure 13 for the algorithm.

### 4.2.3.2 Interrupt Service

A service operation is an operation performed by the CPU, which causes the source of the 82510 interrupt to go inactive (it will reset the particular status bit causing the interrupt). An interrupt request within the 82510 will not reset until the interrupt source has been serviced. Each source can be serviced in two or three different ways; one general way is to disable the particular status bit causing the interrupt, via the corresponding block enable register. Setting the appropriate bit of the enable register to zero will mask off the corresponding bit in the status register, thus causing the INT pin to go inactive. The same effect can be achieved by masking off the particular block interrupt request in GSR via the General Enable Register. Another method, which is applicable to all sources, is to issue the Status Clear command from the Internal Command Register. The detailed service requirements for each source are given below:

<table>
<thead>
<tr>
<th>Interrupt Source</th>
<th>Status Bits &amp; Registers</th>
<th>Interrupt Masking</th>
<th>Specific Service</th>
<th>General Service</th>
</tr>
</thead>
<tbody>
<tr>
<td>Timers</td>
<td>TMST (1-0)</td>
<td>TMIE (1-0)</td>
<td>Read TMST</td>
<td>Issue Status Clear (StC)</td>
</tr>
<tr>
<td></td>
<td>GSR (5)</td>
<td>GER (5)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Tx Machine</td>
<td>GSR (4)</td>
<td>GER (4)</td>
<td>Write Character to Tx FIFO</td>
<td>Issue StC</td>
</tr>
<tr>
<td></td>
<td>LSR (6)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Rx Machine</td>
<td>LSR (4-1)</td>
<td>RIE (7-1)</td>
<td>Read RST or LSR Write 0 to bit in RST/LSR</td>
<td>Issue StC</td>
</tr>
<tr>
<td></td>
<td>RST (7-1)</td>
<td>GER (2)</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>GSR (2)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Rx FIFO</td>
<td>RST/LSR (0)</td>
<td>GER (0)</td>
<td>Write 0 to LSR/RST Bit zero, Read Character(s)</td>
<td>Issue StC</td>
</tr>
<tr>
<td></td>
<td>GSR (0)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Tx FIFO</td>
<td>LSR (5)</td>
<td>GER (1)</td>
<td>Write to FIFO Read GIR</td>
<td>Issue StC</td>
</tr>
<tr>
<td></td>
<td>GSR (1)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Modem</td>
<td>MSR (3-0)</td>
<td>MIE (3-0)</td>
<td>Read MSR write 0 into the appropriate bits of MSR (3-0)</td>
<td>Issue StC</td>
</tr>
<tr>
<td></td>
<td>GSR (3)</td>
<td>GER (3)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**NOTE:**
The procedures listed in Table 4 will cause the INT pin to go low only if the 82510 is in the automatic acknowledge mode. Otherwise, only the internal source(s) are decoded, the INT pin will go low only when the Manual Acknowledge command is issued.
4.3 Polling

The 82510 can be used in a polling mode by using the *General Status Register* to determine the status of the various 82510 blocks, this is useful when the software must manage all the blocks at once. If the software is dedicated to performing one function at a time, then the specific status registers for the block can be used, e.g. if the software is only going to be Transmitting, it can monitor the Tx FIFO level by polling the *FIFO Level Register*, and write data whenever the Tx FIFO level decreases. Reception of data can be done in the same manner.

5.0 SOFTWARE CONSIDERATIONS

5.1 Configuration

The 82510 must be configured for the appropriate modes before it can be used to transmit or receive data. Configuration is done via read and write registers, each functional block (except for BIU) has a configuration register. Typically the configuration is done once after start up, however, the FIFO thresholds and the interrupt masks can be reconfigured dynamically. If the 82510 configuration is not known at start up it is best to bring the device to a known state by issuing a software reset command (ICM register, bank one). At this point all block interrupts are masked out in GER and all configuration and status registers have default values. The bank register is pointing to bank zero. The 82510 can now be configured as follows:

1. If BRG A is being issued as a baud rate generator then load the baud rate count into BAL and BAH registers.
2. Configure the character attributes in LCR register (Parity, Stop Bit Length, and Character Length).

(Note if interrupts are being used, steps 1 and 2 can also be done at the end, since the user will have to return to bank zero to set the interrupt masks in GER)

3. Load ACR0 register with the appropriate Control or Address character (if using the Control Character Match or Address Match capability of the 82510).
4. Switch to Bank two.

(In this Bank the configuration can be done in any order)
5. Configure the Receive and Transmit FIFO thresholds if using different thresholds than the default).

6. Configure the Transmit Mode Register for the Stop Bit length, modem control, and if using echo or 9 bit length or software parity, configure the appropriate bits of the register. The default mode of the modem control is Manual, if using the FIFO then the automatic mode would be most useful).

7. Configure the Rx FIFO depth, interrupt acknowledge mode, 8-bit or normal mode and echo modes in IMD register.

8. Load ACR1 if necessary

9. Enable Rx Machine Interrupts as necessary via RIE.

10. Configure RMD for CCR, DPLL operation, Sampling Window, and start bit.

11. Switch to Bank 3.

12. Configure CLCF register for Tx and Rx clocks and or Sources

13. Configure BACF register for BRG A mode and source.

14. Load BBL and BBH if RGB is being used (as either a BRG or a Timer).

15. Configure BBCF register if necessary.

16. If reconfiguration of the modem pin is necessary then program the PMD register.

17. Enable any modem interrupt sources, if required, via MIE register.

18. Enable Timer interrupts, if necessary, via TMIE.

19. If using interrupts then
   i) Switch to Bank zero.

   Disable Interrupts at CPU (either by masking the request at the interrupt controller or executing the CLI instruction).

   ii) Enable the appropriate 82510 Block interrupts by setting bits in the GER register. (CPU interrupts can now be reenabled, but it is recommended to switch banks before enabling the CPU interrupts).

   NOTE:

   At this stage it is best to leave the TxM and Tx FIFO interrupt disabled. See section 6.3 Transmit Operation for details)

20. Switch to Bank One. Load Transmit Flags if using 9-bit characters, or 8051 9-bit mode or software parity. If using interrupts CPU interrupts can now be enabled.

Bank One is used for general operation, the 82510 can now be used to transmit or receive characters.
Figure 14. Configuration Flow Chart
Figure 14. Configuration Flow Chart (Continued)
The transmitter has two status flags. Tx Machine Idle and Tx FIFO interrupt request, each of these conditions may cause an interrupt, if enabled. The Transmit Idle condition indicates that the Tx Machine is either empty or disabled. The Tx FIFO interrupt bit is set only when the level of the Tx FIFO is less than or equal to the threshold. These interrupts should remain disabled until data is available for transmission. Because outside of disabling the corresponding GSR status bits, the only way to service Tx Idle is by writing data to the Transmitter. Otherwise, the Tx Machine interrupt may occur when no data is available for transmission, and as a result will keep the INT pin active, preventing the 82510 from generating any further interrupts (unless the Transmit Interrupt routine automatically disables the Tx Machine Idle and Tx FIFO interrupt requests in GSR). The threshold of the Tx FIFO is programmable from three to zero, at a threshold of three the Tx FIFO will generate an interrupt after a character has been transmitted. While at a threshold of zero the interrupt will be generated only when the Tx FIFO is empty. For most applications a threshold of zero can be used. If the threshold is dynamically configured, i.e. it is being modified during operation, then the Tx FIFO level must be checked before writing data to the transmitter.

5.2 Transmit Operation

5.2.1 GENERAL OPERATION

To transmit a character the CPU must write it to the TXD register, this character along with the flags from the Tx Flags register is loaded to the top of the TX FIFO. If the Tx Machine is empty, then the character is loaded into the shift register, where it is serially transmitted out via the TXD pin (the flags are not transmitted unless the 82510's configuration requires their transmission e.g. if software parity is selected then the S/W parity bit is transmitted as the parity bit of the character). The CPU may write more than one character into the FIFO, it can write four characters in a burst (five if the Tx Machine is empty) or it can check the FIFO level before each write, to avoid an overrun condition to the transmitter. In the case of the latter, the software overhead of checking the FIFO level must be less than the time required to transmit a character, otherwise the transmit routine may not exit until another exit condition has been met.

e.g. at 288,000 bps for an 8-bit char no parity

It takes 34.7 μs to transmit one character.

If the time, from the write to TXD to the reading of the Transmit FIFO level, is greater than 34.7 μs then the Tx FIFO level will never reach higher than zero, and the FIFO will always appear to be empty. Therefore, if the transmit routine is checking for a higher level in the FIFO it may not be able to return until some other exit condition—such as no more data available—is met. This can be a problem in the interrupt handler, where the service routine is required to be efficient and fast.

The software can wait until the Tx FIFO threshold has been met or if the Transmitter is empty. Since the Tx Machine interrupt is high priority (second highest priority, with Timer being the highest), the interrupt line will not be released to other lower priority, pending 82510 sources until the Tx Machine interrupt has been serviced. If no data is available for transmission, then the only way to acknowledge the interrupt is by disabling it in the General Enable Register. Thus the Tx Machine interrupt should not be enabled until there is data available for transmission. The Tx Machine interrupt should be disabled after transmission is completed.

5.2.1.2 Transmission By Polling

Transmission on a polling basis can be done by using the General Status Register and/or the FIFO Level Register. The software can wait until the Tx FIFO and/or the Tx Machine Idle bits are set in the General Status Register, and then do a set number of writes to the TXD register. This method is useful when the software is trying to manage other functions such as modem control, timer management and data reception, simultaneously with transmission.

If management of other functions is not needed while transmitting, then continuous transmission can be done by monitoring the Tx FIFO level. A new character is written to TXD as soon as the FIFO level drops by one level.

5.2.1.1 Transmit Interrupt Handler

The Transmit Interrupt Handler will be invoked when either the Tx FIFO threshold has been met or if the Transmitter is empty. Since the Tx Machine interrupt is high priority (second highest priority, with Timer being the highest), the interrupt line will not be released to other lower priority, pending 82510 sources until the Tx Machine interrupt has been serviced. If no data is available for transmission, then the only way to acknowledge the interrupt is by disabling it in the General Enable Register. Thus the Tx Machine interrupt should not be enabled until there is data available for transmission. The Tx Machine interrupt should be disabled after transmission is completed.
NOTE:
TxM Idle and Tx FIFO Empty interrupts are enabled by the Main Program, when data transmission is required.

Figure 16. 16 Tx Interrupt Handler Flow Chart
Figure 17. Using GSR for Polling
Figure 18. Data Transmission by Monitoring FIFO Level
Figure 19. Break Transmission Using Tx FIFO to Measure Break Length
5.2.1.3 Break Transmission

The 82510 will transmit a break when bit six of the Line Control Register is set high. This will cause the TXD pin to be held at Mark for one or more character time. The Tx FIFO can be used to program a variable length break, see Figure 19 for details. If the break command is issued in the midst of character transmission the TXD pin will go low, but the transmitter will not be disabled. The characters from the Tx FIFO will be shifted out on to the Tx Machine and lost. To prevent the erroneous transmission of data, The CPU must make sure the Transmitter is empty or disabled before issuing the Send Break command.

5.3 Data Reception

The receiver provides the 82510 with three types of information:

a) Data characters received

b) Rx Flags for each data character

c) Status information on events within the Rx Machine.

The Rx FIFO interrupt request goes active when the Rx FIFO level is greater than the threshold, if the interrupt for this bit is enabled then it will generate an interrupt to the CPU. This is a request for the CPU to read characters from the 82510. Each character on the Rx FIFO has flags associated with it, all of these flags are generated by the Rx Machine during reception of the character. These flags provide information on the integrity of the character, e.g. whether the character was received OK, or if there were any errors. The receiver status is provided via the Receive Status Register (RST), which provides information on events occurring within the Rx Machine, since the last time RST was read. The information may or may not apply to the current character being read from the RDX register. The CPU may read one or more characters from the Rx FIFO. After each read, if the FIFO contains more than a single character, a new character is loaded into the RDX register and the flags for that character are placed into the RXF register. The software can check for the Rx character OK bit in the flags to make sure that the character was received without any problems.

5.3.1 RECEIVE INTERRUPT HANDLER

The Receiver will generate two types of interrupts, Rx FIFO interrupt and Rx Machine Interrupt. The Rx FIFO interrupt requires that the CPU read data characters from the Rx FIFO. If the Rx Machine interrupts are disabled then the CPU should also check for errors in the character before moving it to a valid buffer. The interrupts generated by the Rx Machine can be divided into two categories—occurrence of errors during reception of data (parity error, framing error, overrun error), or the occurrence of certain events (Control/Address character received, Break detected, Break Terminated). For typical applications, the error status of each received character can be checked via the Receive Flags, and the events can be handled via interrupts.

5.3.2 RECEIVING DATA BY POLLING

To receive data through polling, the 82510 can use the General Status or the Receive Status Registers to check for the Rx FIFO request. If the Receive routine does not generate time outs or modem pin transitions, then the data can also be received by monitoring the Rx FIFO level in the FIFO Level Register. The implementation using GSR would be useful in applications where the software routine must monitor the timer for time outs or the modem pins for change in status. The example polling routine illustrates the use of the FIFO Level Register in receiving data. It waits for the Rx FIFO request before beginning data reception. The procedure Rx_Data_Poll will receive the number of characters requested in Char_count and place them in the Receive buffer.
Figure 20. Rx FIFO Interrupt Handler
#define base 0x3f8; /* base address of 82510 */
#define buff_size 128;

Rx_Data_Poll (Char_count, Rxbuffer)
int Char_count; /* Total # of bytes to be received */
char *Rxbuffer [buffsize];

int count = 0;
int status, IvI, Rok;

While (((status = (Inp(base+7) & 0x05)) == 0x01) /* If Rx FIFO Req in GSR set */
    /* Assume in bank one */)
    /* If Rx FIFO is not empty */
    While ((((IvI = ((Inp (base+4) & 0x70)/0x10)0&((count < (Char_count))
    /* If Character Received OK */
    if (((Rok = (Inp (base+1) & 0x60)) == 0x40)
        Rxbuffer [count] = Inp (base);
        ++count;
}

Figure 21. Example Polling Routine

5.4.3 CONTROL CHARACTER HANDLING

The 82510 has two modes of control character recognition. It can recognize either standard ASCII or standard EBCDIC control characters, or it can recognize a match with two user programmed control (or Address Characters in MCS-51 9-bit mode, for Automatic Wake up) characters. Each mode generates an interrupt through the Receive Status Register. The Receive Flags also indicate whether the character being read is a control character. The usage of CCR depends on the maximum number of possible control characters that can be received at any one time. Applications such as Terminal Drivers, which have no more than two control characters outstanding, such as XON and Ctrl-C, or XOFF and Ctrl-C, can use just the Control Character Matching capabilities of the 82510.

5.3.3 BREAK RECESSION

The 82510 has two status indications of break reception, Break Detect indicates that a break has been detected on the RXD pin. Break Terminated indicates that the Break previously detected on the RXD line has terminated and normal Data reception can resume. Each of these status bits can generate an interrupt request through the Rx Machine Interrupt request. Normal consequence of break is to abort the data reception or to introduce a line idle delay in the middle of data reception. In the case of the former, the Break Detect interrupt can be used to reset the 82510 Receive Machine and the Rx routine flags; in the case of the latter, the break terminated interrupt can be used to filter out the break characters and resume normal reception. Each break character is identified by a break flag in the Rx Flags Register (the CCR flag, Framing error, and CCR Match flag also may become active when a break character is received) and is loaded onto the Rx FIFO as a NULL character. If break continues even after the Rx FIFO is full, then an overrun error will occur but no further break characters will be loaded on to the Rx FIFO. The user can also measure the length of the break character stream by using the Timer.
Figure 22. Handling Control Character Interrupts
Figure 23. Using Control Character Match in Terminal Ports
5.3.4 DATA INTEGRITY

To improve the reliability of the incoming data the 82510 provides a digital filter, a Digital Phase Locked Loop, and multiple sampling windows (which provide a noise indication bit).

5.3.4.1 Digital Filter

The Digital Filter is used to filter spikes in the input data. The Rx Machine uses a 2 of 3 filter. The output is determined by the majority of samples. If at least two of the three samples are "1" then the output will be a "1". Spikes of one sample duration will be filtered but spikes of two or more samples duration will not be filtered.

5.3.4.2 Digital Phase Locked Loop

The Digital Phase Locked Loop (DPLL) is used by the Rx Machine to synchronize to the incoming data, and adjust for any jitter in the incoming data.

The 82510 DPLL operates on the assumption that a transition in the incoming data indicates the beginning of a new bit cell. A valid asynchronous character frame will contain one or more transitions depending upon the data. If upon occurrence of the transition, the DPLL phase expectation is different from the sampled phase, then there is jitter in the incoming data. The DPLL will compensate for the phase shift by adjusting its phase expectations, until the expected phase and the sampled phase are locked in. The user can enable or disable the DPLL through the Receive Mode Register (RMD).

5.3.4.3 Sampling Windows

The sampling windows are used to generate the data bit, by repeated sampling of the RXD line. The bit polarity decision is based upon a majority vote of the samples. If a majority of the samples are "1" then the bit is a "1". If all samples are not in agreement then the Noisy Character bit in the RXF register is set. The sampling windows are programmable for either 3 of 16 or 7 of 16. The 3/16 mode improves the jitter tolerance of the medium. While the 7/16 window improves the impulse noise tolerance of the channel.

The sampling windows also provide a Noisy character bit in the RXF register. This bit indicates that the current character being read had some noise in one or more of its bits (all the samples were not in agreement). This bit can be used along with the Parity and Framing error bits to provide an indication of noise on the channel. For example, if the Noisy Character bit and the Parity or the Framing errors occur simultaneously, then the noise is probably sufficient to merit a complete check of the communications channel. The noisy bit can also be used to determine when the cable is too long or the baud rate is too high. The user would keep a tally of the noisy characters, and if more than a certain number of characters were received with noise indications, then either the baud rate should be lowered or the distance between the two nodes should be reduced.

5.4 Timer Usage

The 82510 has two baud rate generators, each of these can be configured to operate as Timers. Typical applications use BRG A as a BRG and BRG B as a Timer. Since both the Transmitter and the Receiver may need to generate time outs, it is best to use the Timer as a Time Base to decrement ticks (upon a Timer Expired Interrupt) from (software implemented) Tx and/or Rx counters. The Timer can also be used to time out the Rx FIFO and read characters that otherwise may not have been able to exceed the Rx FIFO threshold.

5.4.1 USE AS A TIME BASE

The transmitter and the receiver routines use a software variable which acts as a counter. The variable is loaded with the required number of ticks that are needed for the Time Out period. Once started the Timer generates an interrupt each time it expires, the interrupt handler then decrements the counters. Once loaded the software monitors the counters until their value reaches zero, this would indicate to the software that the required time period has elapsed. The Time Base value should be selected with regards to the CPU interrupt load. The CPU load will increase substantially when the Timer is used as a Time Base, therefore using the Timer in this mode at very high baud rates may cause character overruns. A time base of 5 or 1 ms is probably the most useful. An additional benefit of the Time Base is that it can support more than two counters if required.
BRG-B is used as Timer.
BRG-A is used as BRG.
TB Ex bit in TMST Enabled.

Tx_Timer_Count contains count for Transmitter.
Rx_Timer_Count contains count for Receiver.

5.4.2 USE FOR RX FIFO TIME OUT

In the 82510, Rx FIFO interrupts will occur only after the FIFO level has exceeded the threshold. Due to this mechanism and the nonuniform arrival rate of characters in asynchronous communications, there is a chance that characters will be "trapped" in the Rx FIFO for an extended period of time.

For example, assume the 82510 is a serial port on a system and is connected to a terminal. The user is entering a command line. The Rx FIFO Threshold = 3, and at the end only two bytes are received. Since the FIFO threshold has not been exceeded, the Rx FIFO interrupt is not generated. No other characters are received for 30 minutes, if the characters (in the Rx FIFO) are a line feed and carriage return, respectively, the CPU may be waiting for the CR to process the characters it has received. Consequently the characters will not be processed for 30 minutes.

In order to avoid such situations, a Rx FIFO Time Out mechanism can be implemented by using the 82510 Timer. The time out indicates that a certain amount of time has elapsed since the last read operation was performed. It causes the CPU to check the Rx FIFO and read any characters that are present.

In applications where the character reception occurs in a spurious manner (the exact number of characters cannot be guaranteed), the Rx FIFO Time Out is the only way to prevent characters from being trapped. The time out period is measured from the last read operation, every read operation resets the Rx FIFO Timer. To synchronize with the beginning of the data reception, initially the Rx FIFO threshold is set to zero. After the first character has been received, the threshold is adjusted to the desired value. When a Rx FIFO time out occurs and no data is available, the threshold is reset to zero. In error free data transmission, the beginning of data transmission is signaled by the reception of a control character, such as SOH or STX, the Rx FIFO time out mechanism should be triggered to the reception of these control characters.
RX FIFO INTERRUPT ROUTINE

RX FIFO THRESHOLD = 0

YES

CHANGE RX FIFO THRESHOLD

NO

READ CHARACTER UNTIL FIFO EMPTY

(RE)LOAD RX FIFO TIME OUT COUNT

EXIT

MAIN RX ROUTINE

RX FIFO TIME OUT (COUNT = 0)

YES

RX FIFO EMPTY

NO

RESET RX FIFO THRESHOLD TO ZERO

READ CHARACTER UNTIL RX FIFO EMPTY

(RE)LOAD RX FIFO TIME OUT COUNT

EXIT

Figure 25. Rx FIFO Time Out Flow Chart
6.0 82510 IMPLEMENTATION OF XMODEM

The 82510 XMODEM implementation is a file transfer program for the 82510 based on the XMODEM protocol. The software runs on the PC AT on a especially designed adapter board (the adapter board design is shown in Figure 33). The software uses most of the 82510 features including the baud rate generator, Timer, Control Character Recognition and FIFOs. The software uses an interrupt driven implementation, written in both assembly and C languages.

6.1 XMODEM Protocol

XMODEM is a popular error free data transfer protocol for asynchronous communications. Data is transferred in fixed length 128 byte packets, each packet has a checksum for error checking. The packets are delineated by control characters, which act as flags between the Receiver and the Transmitter. There are four control characters, SOH, EOT, ACK, and NAK. SOH indicates the Start of a Packet, EOT indicates the End Of Transmission; ACK and NAK are positive or negative acknowledgements of the packet respectively. The packet structure and protocol flow of XMODEM is provided in the figures given below.

6.2 Software

Interrupts are used to transmit and receive data. The software is implemented as two independent finite state machines—Transmit State Machine and Receive State Machine. Each state machine is triggered by external events such as user commands and data or Control Character reception. The state machines communicate with the 82510 interrupt service routines through software flags. The overall structure of the main routine is given in Figure 31. The major modules of the software are given in the hierarchy Chart, Figure 34, which lists the different modules in order.

The interface between the main program and the interrupt service routine is done through global flags. The interrupt handler services four sources—Transmit, Timer, Receive, and Control Characters. Each of the interrupt sources communicates with each of the state machines through the global flags. The state machines keep track of their individual states through state variables. The interface between the individual states within a state machine is done through state flags. The state machine diagrams are given in Figure 29 and Figure 30.
Figure 28. Protocol Flow for Receive Side of XMODEM
Figure 29. Transmit State Machine
Figure 30. Rx State Machine
6.2.1 TRANSMISSION OF DATA

The Transmit interrupts are disabled until data transmission is required, this prevents unnecessary Transmit interrupts. The Transmit interrupt is enabled when a packet has been assembled or if a Control Character is required to be transmitted. Upon invocation the Transmit interrupt service routine reads characters from the packet buffer and writes it to the Tx FIFO. Since it does not require the use of the Transmit Flags, no information is written to the TXF register.

6.2.2 RECEPTION OF DATA

Data reception begins only after a Start of Header (SOH) control character is received. This control character puts the receiver in a data reception mode. After receiving the SOH, the CCR interrupt is disabled (since all data being received now is transparent and can not be interpreted as a control character). After 132 characters are received, the CCR interrupt is reenabled and the corresponding ACK or NAK sent to the Transmitting system. The receiver has a time out feature, which causes it to check the Rx FIFO for any remaining characters. End of Transmission is indicated by an EOT control character, which causes the file to be closed and the Receiver to go into the Idle state.
#include "C\ftpdef"
#include "C\licefntti.h"
#include "C\licefntlib.h"
#include "C\licefntdio.h"

#include "C\fcntl.h"
#include "C\stdio.h"

/*.
**.
-Sepember 1986
.*/

int eof = false;  // end of file flag */
int rnk = 0;
int hrflg, nrfig, rter_pkt_null, f;    // Packet number updated by receiver
int pkt, rwtocnt;                         // Time out counter for receiver
int key = 0;                                // % of SOH characters received
int safcnt = 0;                             // % of Rx FIFO Interrupts
int ccrnt = 0;                              // % of Ctl-Char Interrupts
int tx_state = tx_idle;                    // Transmitter State Variable
int rx_state = rs_idle;                    // Receiver State Variable
int tx_cad = inactive;                     // Indicates a Valid Tx Command was issued
int rx_cad = inactive;                     // Indicates a valid Rx Command was issued

/* File to be Transmitted */
char tx_file_name[40] = " 
/* File to be Received */
char rs_file_name[40] = " 

int send_cce_req = inactive;  // Flag - Request to Tx Ctrl-Char */
int intvec = 0;                // contains the GIR vector */
int i;                        // Tx Buffer */
char tdata[132], rdata[132], rsbuf[128], rxbuf[128];  // Rx Buffer */
char rdata[131], rs_f_buf[32000];  // Rx File Stored in this buffer */

struct packet {               // Pointer to the next character in the buffer */
char head, pack_num, pack_cmpl, buffer[128], chsum, 
)

struct packet respacket, tpack.
82510 XMODEM Implementation (Continued)
PAGE 3 MAIN PROGRAM ftp c 82510 XMODEM

104 /**************************************************************************/
105 /** MAIN ROUTINE ***/
106 /**************************************************************************/
107 main ()
108 {
109 int q,txfl,rxfl,
110 int rval, r = 0,
111 int cmd = 0,
112 int wlan_status = 0,
113 int ecode = 0,
114 FILE *fp,
115 FILE *rep,
116 int twert,
117 int wact,
118 int retx_cnt = 0,  // Retransmit count */
119 int locnt = 0,  // Time Out Count */
120 int te_seces, re_seces,
121 int t, s, lpcnt = 0,
122
123 /*******************
124 */ Clear Screen */
125 */ Sign On Message */
126 */ Initialize 82510 and Variables */
127 */ Print Menu */
128 */ Enable Interrupts in 8259A */
129 */ Keep Track of # of Loops */
130
131 /*******************
132 while (quit=false)
133 /*******************
134 /** display protocol parameters */
135 /*******************
136
137 /*******************
138 /* main while loop */
139 /*******************
140 while (quit=false)
141 {
142
143 /*******************
144 */ while (quit=false) */
145 /*******************
146
147. ++ lpcnt;
148 mv_curs (6,30),
149 printf ("loop # = %u",lpcnt),
150 mv_curs (4,50),
151 printf ("tx int cnt = %u",txintcnt),
152 mv_curs (5,50),
153 printf ("cr int cnt = %u",crintcnt);
154 mv_curs (4,1),
155 printf ("interrupt vector = %u\m", intvec),
156 q = imp (tpa+4),
157 txfl = q & 0x07,
158 mv_curs (5,1),
159 printf ("TX FIFO = %u\",txfl),
160 q = imp (tpa+4),
161 rffl = q & 0x08;
162 mv_curs (4,1),
163 printf ("RX FIFO = %u\m",rffl/16),
164 mv_curs (4,50),
165 printf ("SOH count = %d\",sohcnt),
166 printf ("CR『",cr);
process the Command */
188 if ((key_ebhbit())) 0)
189 quit = process_cmd();
190
191 else
192 {
193
194 //*************Process Tx STATE MACHINE************
195 //*** Process Tx STATE MACHINE
196 //*** revision 0
197 //***
198 //*************Process Rx STATE MACHINE************
199 switch (tx_state) {
200 case tx_idle:
201
202
203 //*************TRANSMITTER IDLE STATE************
204 //*** TRANSMITTER IDLE STATE
205 //***
206 //*** Checks for a Send Ctrl-Char
207 //*** Checks for the Transmit Command
208 //*** Checks for a Send Ctrl-Char
209 //***
210 //***
211 //*************TRANSMITTER IDLE STATE************
212
213
214
215
216
217 if ((send_ccr_req == active) && (!tx_req))
218 {
219
220
221
222
223
224
225   send_ccr_reginactive,
226 }
/* If the Transmit Command is issued then Wait for a NAK */
if (tx_cmd == active) {
    tx_cmd = inactive;
    get_crr_rq = active,
    tx_time_cnt = 200*60, /* 40 sec. Time Out */
    tx_state = wait_NAK;
}
break;
case wait_NAK: /* Waiting for a NAK character to begin Tx */

/*TRANSMITTER WAITING FOR A NAK TO BEGIN */

switch (wm_status) {

case time_out: /* If Time Out then Abort Transmission */
    tx_state = tx_idle;
    beep();
    printf("Time OUT!!! receiver not ready");
    cii (tx_r, tx_c);
    open_wind (tx_r, tx_c,"NONE");
    break;

case waiting: /* If no Time Out and no NAK received then do nothing */
    break;

case rx_NAK: /* If NAK received then Open file and advance to Transmit Packet State */
    fp = fopen (ts_file_name,"rb");
    if (fp == NULL) {
        beep();
        printf("ERROR!!! file does not exist");
        cii (tx_r,tx_c);
        open_wind (tx_r,tx_c,"none");
        tx_state = tx_idle;
    }
else {
    tx_state = tx_rdy;
    txflag = mkpkt; /* First task for Tx
    is to Prepare Packet */
    wm_status = 0; /* Reset Wait_NAK Flag */
    }
break;
}
break; /* end wait nak */
case tx_rdy
    // TRANSMITTER READY TO TRANSMIT
    // three stages of transmission
    // prepare packet
    // Int Handler Transmitting
    // or retransmit request
    //
    /* Any Control Character To Transmit */
    if (ISSEND_CCReq == active) && (tx_req=0)
        tx_req=ctl_chr,
    /* Which Stage of transmission */
    switch (txflag)
        case mkpkt
            /* Prepare Packet */
            if (tx_req=0)
                
                amppkt (pkt_send,pf);
                cpy2buf (I);
                tx_req=pkt;
                // Request Int. Handler
to Tx data in buffer
                txflag =txmtg;
                // Start Transmission
                txinds=0;
                tx_temb (I);
                // Enable TM and Tx FIFO
                Interrupts
            
        }
        break,
        case txmtg
            if (tx_req==0)
                // Interrupt Handler Resets
                this flag to 0, when 132
                bytes are transmitted
            
            
            txinds=0;
            prmsg ("packet transmitted");
            get_ccreq=active,
            // Wait for ACK or NAK
            tx_time_cnt = 100*10,
            // 10 sec Time Out
            tx_state = wait_CC;
            // Wait for cc Character
            txflag =mkpkt;
            tx_r_dirc (I);
            // Disble TM and Tx FIFO
            Interrupts
            
        }
        else
            /* Tx_req not reset then
            The Retransmit request is
            issued by the Wait _CC
            state */
            break,
            case retx
                /* The Retransmit request is
                issued by the Wait _CC
                state */
                output(tba+tca),txm,
                // enable tx, flush tx fifo
                & tm */
                tx_req=pkt;
                // transmit Packet,pkt in
                buffer */
                txflag =txmtg;
                // Next task - Retransmit */
                tx_temb (I);
                // Enable TM and Tx FIFO
                Interrupts */
                
                break;
                /* End tx rdy case */
            
            break;
    
    break;
82510 XMODEM Implementation (Continued)
else
  /* If Retransmit Count Not exceeded then go back to
     Transmit stage - task is retransmit */
  {  
    txfig = trab,  
    ++ rets_cnt,  
    tx_state = tx_rdy,  
  }
  break,
  
case rs_ACK:  /* ACK Received*/
  prmsg("ACK received"),  
  rets_cnt=0,  
  tocnt = 0,  
  ++pkts_sent,  
  printf("pkts_sent = %lu", pkts_sent);  
  if (eof ==false)  /* If more data to transmit 
     then return to mkpkt stage and tx new pkt */
    
  {  
    txfig = mkpkt,  
    tx_state = tx_rdy,  
  }
  else
    
  {  
    prmsg("sending EOT"),  /* if end of file , then
      send EOT */
    ccr_to_ts = EOT,  
    ts_req =ctl chr,  
    ts_i_evb (),  
    while (ts_req != 0), /* wait for Int Handler 
      to reset flag */
    ts_i_dis (),  
    get_ccr_rq =active, /* wait for Ack */
    while (get_ccr_rq ==active),  
    prmsg("EOT acknowledgement received"),  
    if (ccr_to_get == ACK) /* ACK read . Close File */
      
    {  
      a = fclose (fp),  
      abort_ts(),  
      prmsg("file transmission complete"),  
      
    }  
    ts_state = tx_idle, /* Return to Idle */
  }
  break,
  }  /* end wait_cc case */

/* end switch tx state */

82510 XMODEM Implementation (Continued)
82510 XMODEM Implementation (Continued)
PAGE 16  MAIN PROGRAM  ftp c  82510 XMODEM

507 case time_out  /* if time out & not in the midst of 
508 packet reception then send NAK */
509 
510  if ( (exp_pkt_num != 0) & (tx_bytes_cnt == 0) )
511  { 
512    prmsg ("tx time out !!!!! sending NAK");
513    if (send_ccr_req == inactive)
514        ccr_to_is_NAK;
515    send_ccr_req = active;
516  }
517  }
518  rx_time_cnt = 800+10;
519  break;
520  
521 case EOT  /* if End of Test rcvd,  
522            and data rcvd then  
523            send ACK and save all  
524            packets received in  
525            file */
526   ++ out_cnt,
527   open_wnd (25,50, "End of Test");
528   if (exp_pkt_num != 0)  
529        
530    if (send_ccr_req == inactive) /* Send ACK */
531      send_ccr_req = active,
532      ccr_to_is_ACK;
533   rx_state = rx_idle;  // Receiver Returns to 
534   // create file */
535   rsfp = open (rx_file_name, "ab");
536   rsfp = fwrite (framebuf, 128, exp_pkt_num-1, rsfp);
537   if (fwrite (0)
538      
539    prmsg ("Write file error.");
540    printf ("error = %d", fwrite error (rsfp));
541   }
542   rsfp = fclose (rsfp);
543   if (rsfp =0)
544    prmsg ("file received");
545   else
546    prmsg ("Error in closing file ");
547   
548  break;
549  
550  break;
551  
552  /* PA*/
553  case rx_pkt  /* Packet reception */
554  
555  
556  
557  
558  /* PA*/
559  /* Packet reception */
560  
561  
562  
563  
564  
565  
566  82510 XMODEM Implementation (Continued)
AP-401

82510 XMODEM Implementation (Continued)
PAGE 11

MAIN PROGRAM ftp c 82510 XMODEM

627.

buf_copy (exp_pkt_num),

628.

++ exp_pkt_num,

629.

else

630.

prmmsg ("old packet retransmitted"),

631.

}

632.

}

633.

else

634.

sh_pkt_param (), /* If error then show

635.

packet & checksum and

636.

packet complement */

637.

rx_state = rx_ready,

638.

mskint4 (), /* Enable Cli-Char int*/

639.

set_bank (00);

640.

outp ((bpa+1),(inp(bpa+1):ccien)),

641.

set_bank (01);

642.

embint4 (),

643.

send_crr_req = active;, /* Send ACK */

644.

crr_to_is = ACK,

645.

646.

647.

648.

}

649.

break;

650.

651.

} /* end switch rx state */

652.

653.

654.

655. */ end else */

656.

657. */ end while quit */

658.

reset (0),

659.

/* reset 82510 */

660.

outp ((bpa+1),00),

661.

/* disable 82510 interrupts */

662.

663.

cmd = 0x10,

664.

/* enable 8359A interrupt */

665.

v=inp (0x21),

666.

/* 00010000 */

667.

668.

cmd = (v ! cmd),

669.

670.

671. */ end main */

672.

82510 XMODEM Implementation (Continued)
82510 XMODEM Implementation (Continued)
PAGE 14  MAIN PROGRAM  ftp c 82510 XMODEM

718.
719. init ()
720. /********************************************************/
721. /** Initialise Software and Configures */
722. /** the 82510. Also sets up the interrupt */
723. /** Handler */
724. /** */
725. /** */
726. /***********************************************************/
727. }
728. }
729. }
730. }
731. initpack ()
732. }
733. }
734. }
735. }
736. }
737. }
738. }
739. }
740. }
741. }
742. }
743. }
744. }
745. }
746. }
747. }
748. }
749. }
750. }
751. }
752. }
753. }
754. }
755. }
756. }
757. }
758. }
759. }
760. }
761. }
762. }
763. }
764. }
765. }
766. }
767. }
768. }
769. }
770. }
771. }
772. }
773. }
774. }
775. }
776. }
777. }

82510 XMODEM Implementation (Continued)
82510 XMODEM Implementation (Continued)
82510 XMODEM Implementation (Continued)

```
838 set_dlab (bank)
839 /**************************************************************************/
840 /**
841 /** Set DLAB bit to allow access to **/
842 /** Divisor Registers **/
843 /**
844 /**************************************************************************/
845
846 int bank,
847 {
848    int inval,
849    set_bank (00),
850    inval = inp(bpa +3);
851    inval +inval = 0x80;       /* set dlab in LCR*/
852    outp ((bpa+3),inval);
853    set_bank (bank);
854 }
855
856 reset_dlab (bank)
857 /**************************************************************************/
858 /**
859 /** Reset DLAB bit of LCR **/
860 /**
861 /**************************************************************************/
862
863 int bank,
864 {
865    int inval;
866    set_bank (00),
867    inval = inp(bpa +3);
868    inval = (inval & 0x7f);   /* dlab = 0 in LCR*/
869    outp ((bpa+3),inval);
870    set_bank (bank);
871 }
```
PAGE 17  MAIN PROGRAM  ftp c 82510 XMODEM

872  /*********************************************************************/
873  /***/
874  /***/
875  /***/
876  /***/
877  /***/
878  /***/
879  /***/
880  /***/
881  /***/
882  /***/
883  /***/
884  /***/
885  /*********************************************************************/
886
887  isr_310 ()
888  {
889  int  source ,
890  int  cmd_b ,
891  int  st_b ,
892  int  i ,
893  int  flg ,
894  int  gival ,
895  int  rxrvi ,
896  int  txr_ch ,
897  int  tx_e_ch ,
898  int  gival = inp (bpa+2) ,
899  int  source = getvec () ,
900  intvec = source ,
901  switch (source) {
902  case timer
903         /*********************************************************************/
904  /***/
905  /***/
906  /***/
907  /*********************************************************************/
908  /***/
909  /***/
910  /***/
911  /***/
912  /***/
913  /*********************************************************************/
914  st_b = inp (bpa+3) ,
915  if (tx_time_cnt > 0)  /* Decrement Transmit Counter */
916      tx_time_cnt = tx_time_cnt - 1 ,
917  if (rx_time_cnt > 0)  /* Decrement Receive Counter */
918      rx_time_cnt = rx_time_cnt - 1 ,
919  cmd_b = 0x32 ,
920  outp ( (bpa+3) , cmd_b ) ,  /* restart timer */
921  outp ( (bpa+7) , 0x08 ) ,  /* manual ack */
922  break ,
923

82510 XMODEM Implementation (Continued)
case txm
  case xf:
  /*******************************************************************************/
  /**** TRANSMITTER SERVICE ROUTINE /****
  /**** transmits four characters /****
  /**** and resets tx_req flag when /****
  /**** whole packet transmitted /****
  /******************************************************************************/
  if (tx_req) 0) /* If data to send */
    if (tx_req == pkt) /* request to send Packet */
      for (i = 0; i < 4, i++)
        tx_char = txdata [i + tx_ind],
        outp (bpa, tx_char),
      tx_ind ++4,
      tx byte_cnt ++4,
    if (tx_ind > 13) /* if 132 char sent then */
      tx_req = 0, /* reset Tx request */
  }
else
  if (tx_req == ctl_chr) /* if ctrl char transmission */
    requested , then transmit the
    char in ccr_to_tx */
        outp (bpa, ccr_to_tx),
      tx_req = 0,
  }
  else
    /* if no data to transmit */
  /* then disable tx interrupts */
  set_bank (00),
  outp ((bpa+1), (inp(bpa) &twidb)),
  set_bank (01),
  }
  outp ((bpa+7), 0x08), /* issue manual acknowledge */
    break,

82510 XMODEM Implementation (Continued)
PAGE 19   MAIN PROGRAM ftp.c 82510 XMODEM

972. case ccr
973. /********************************************************************************/  
974. / **** Control Character Service Routine ****/  
975. /**** if control char = NAK or ACK ****/  
976. /**** inform transmitter ****/  
977. /**** if SOH or EOT ****/  
978. /**** inform receiver ****/  
979. /********************************************************************************/  
980.  
981.  
982.  
983. /* control_char,  
984.  
985.  
986.  
987.  
988.  
989.  
990.  
991.  
992.  
993.  
994.  
995.  
996.  
997.  
998.  
999.  
1000.  
1001.  
1002.  
1003.  
1004.  
1005.  
1006.  
1007.  
1008.  
1009.  
1010.  
1011.  
1012.  
1013.  
1014.  
1015.  
1016.  
1017.  
1018.  
1019.  
1020.  
1021.  
1022.  
1023.  
1024.  
1025.  
1026.  
1027.  
++control,  
  flags = inp (bps +5);  /* read BSR register to service  
  Mem interrupt */  
  
  flags = inp (bps+1),  
  ctc = inp (bps);  
  if ((flags & 0xff) ==0x40) /* if no errors and ctl. char */  
  {  
    /* then process control char. */  
    /* and send to ts or rs state */  
    
    switch (ctc)  
    {  
      case NAK:  
        
        case ACK:  
          if (get_ccr_rq == active)  
          {  
            /* inform transmitter that  
            ctl. char. received */  
            ccr_to_get = ctc;  
            
            if (ctc ==SOH)  
            /* if SOH disable CCR int. */  
            {  
              set_bank (00),  
              outp (((bps+1) & ccrdb));  
              set_bank (01);  
              }  
            
            if (rs_state == rs_rdy)  
            /* if receiver waiting for  
            SOH and ready to recv  
            then inform receiver of  
            a valid ctl. char. */  
            
            if (rs_state == rs_rdy)  
            {  
              ctl_rxd_flg =active,  
              rs_ctl_chr =ctc;  
              
              break,  
              
              break,  
              
              }  
            
            outp (((bps+7) &00));  /* issue manual ack */  
            
            break,  
            
            break,  
            }  
          }  
        }  
      }  
    }  
  }
2.247

82510 XMODEM Implementation (Continued)
82510 XMODEM Implementation (Continued)
case 'I'
  if (tx_state == tx_idle) /* Transmit Command only accepted if idle */
  {
    CLMS (),
    CLL (tx_r,tx_c),
    MV_CURR (tx_r,tx_c),
    printf ("file "); /* Get name of file to Tx */
    scanf ("%s", &ts_file_name);
    CLL (tx_r,tx_c),
    open_wind (tx_r,tx_c,"transmitting"),
    open_wind (tx_r,tx_c+14,tx_file_name),
    ts_cmd = active, /* Activates flag to signal
    Transmit idle state */
  }
} else
  {
    beep (),
    prmsg ("transmission in progress").
  }
break;
  case 'T'
  CLMS (),
  CLL (rx_r,rx_c),
  MV_CURR (rx_r,rx_c),
  printf ("file "); /* Get rx file name */
  scanf ("%s", &rs_file_name),
  CLL (rx_r,rx_c),
  open_wind (rx_r,rx_c,"enabled"),
  open_wind (rx_r,rx_c+14,rx_file_name),
  rx_cmd = active, /* Activate flag to signal
  rx state machine */
  break;
  case 'R'
  rst510 (), /* reset 82510 */
  open_wind (24,30,"device reset"),
  break;
  case 't'
  rst510 (), /* reinitialize 82510 */
  init (),
  embuf ()
  beep (),
  prmsg ("$2510 reinitialized"),
  break;
  case 'T'
  excep = system ("d:\micom");
  default
    BEEP ();
    prmsg ("incorrect command, reenter"),
    break;
  if (exitflag == true) /* if exit command issued, 
  then quit program */
  return (true),
else return (false).
  /# end of command processing */
82510 XMODEM Implementation (Continued)
PAGE 14

MAIN PROGRAM ftp c 82510 XMODEM

1252
1253 check_wait ();
1254
1255 /***************************************************************************/
1256 /***************************************************************************/
1257 /***************************************************************************/
1258 /***************************************************************************/
1259 /***************************************************************************/
1260 /***************************************************************************/
1261 /***************************************************************************/
1262 /***************************************************************************/
1263 /***************************************************************************/
1264 /***************************************************************************/
1265 /***************************************************************************/
1266 /***************************************************************************/
1267 if (!ts_time_cnt) & (get_csr_rq == active))  /* if tx Timer expired
and still waiting */
1268 for ts Timer */
1269 return (time_out);
1270 else
1271 if (get_csr_rq == inactive) /* Ctrl-Char rcvd then
return status */
1272 return (status);
1273 switch (crr_to_get)
1274 { case ACK
1275 return (rs_ACK);
1276 break;
1277 case NAK
1278 return (rs_NAK);
1279 break;
1280 default:
1281 return (rs_gen);
1282 break;
1283 /
1284 /* corrupted Ctrl-char */
1285 if (!ts_time_cnt) & (get_csr_rq == active))
1286 return (waiting);
1287 else
1288 return (waiting);
1289 return (waiting);
1290 return (waiting);
1291 }
void abort_tx()
{
    // Abort transmission, reinitialize
    abort = true;
    abort_transmitter = true;
    abort_flag = true;
    abort_response = true;
    abort_message = "transmit reset";
}

void wait_rx()
{
    // Check for timeout
    if (check_time_out || check_timeout_rx)
    {
        // Return the following value
        return;
    }
    // Check for SOH - SOH received
    if (check_SOH)
    {
        // SOH received
        return;
    }
    // Check for EOT - EOT received
    if (check_EOL)
    {
        // EOT received
        return;
    }
    // Check for time out - rx timer expired
    if (check_time_out)
    {
        // Time out
        return;
    }
    // Check for waiting - waiting for event
    if (check_waiting)
    {
        // Waiting
        return;
    }
    // If no event detected, return
    return;
}

if (ctl_rad_flag == active) && (rx_time_cnt > 0))
{
    // Set CTL_rad flag to inactive
    // Return the rx character
    return (rx_ctl_chr);
}
else
{
    if (rx_time_cnt == 0)
    {
        // Return time_out
        return (time_out);
    }
    // Return (waiting)
    return;
}
82510 XMODEM Implementation (Continued)

1358 chpkt (pktnum)
1359 /***********************************************************************/
1360 /** verifies the checksum and packet **/
1361 /** number of the received packet **/
1362 /** returns a status code **/
1363 /** **/
1364 /** EOK - Packet Ok **/
1365 /** EPNUM - Error in packet number **/
1366 /** ECHKSUM - Error in Check Sum **/
1367 /** EPCKMPL - Error in packet complement **/
1368 /** **/
1369 /***********************************************************************/
1370 int pktnum,.
1371 { 
1372 int i, 
1373 int chm, 
1374 int sum = 0,
1375 char cmpi,rcmpi,cph,checkm,
1376 
1377 
1378 cph = pktnum,
1379 if (cmpi == rdata[i]) /* packet number correct */
1380 {
1381 cmpi = rdata[i],
1382 rcmpi = rdata[i];
1383 
1384 
1385 for (i=1; i<=100; i++)
1386 sum = sum + rdata[i],
1387 chm = sum % 255,
1388 checkm = chm,
1389 
1390 if (checkm == rdata[i]) /* checksum correct */
1391 return (eok),
1392 else
1393 return (echksum). 
1394 
1395 
1396 else
1397 return (epchk).
1398 
1399 else
1400 if (rdata[i]) 
1401 # old packet number received */
1402 return (eold),
1403 else
1404 return (epnum). 
1405 
1406 } 
1407 }
2.254

82510 XMODEM Implementation (Continued)
PROCEDURE BUF_CPY

if (packt_id <= 128) { for (i=0; i<128; i++)
rx_buf [index] = xbuf [i];
}
else
prmsg ("file too big, cannot save in memory");
}
82510 XMODEM Implementation (Continued)
#!/**
** Protocol Control */
** characters */

#define NAK 0x14 // Negative Ack */
#define ACK 0x06 // Positive Ack */
#define SOH 0x01 // Start of Header */
#define EOT 0x04 // End of Text */
#define CAN 0x18
#define NUL 0x00

/**** interrupt source ****/

#define timer 05 /* 82510 int vectors */
define tm 04
define ccr 03
define raf 02
define txf 01
define tlen 0x12 /* unmask TxM and Tx FIFO */
define tsidb 0x24
#define ccien 0x14 /* enable CCR interrupts */
define ccidb 0x23 /* mask CCR int */
define blinkb 0x25 /* enable, block interrupts through GER for 82510 */
define eol 0x0A /* end of interrupt */
define ip0 0x20 /* 8259A port 0 */
define ip01 0x21 /* 8259A port 1 */

82510 XMODEM Implementation (Continued)
#include "ftp_def"

PROCEDURE CLR
    clear screen

PROCEDURE VOFF
    Turns Reverse Video OFF

PROCEDURE RVON
    Reverse Video ON

82510 XMODEM Implementation (Continued)
61. OPEN_WIND (row, col, stg)
62. int row,
63. int col,
64. char stg[3];
65. /***********************************************************************
66. ///// PROEDURE OPEN_WIND
67. ///// prints a string in reverse video at the given location
68. ///// *****************************************************************
69. PROCEDURE OPEN_WIND
70. int
71. /***************************************************************************/
72. { // MV_CURS (row, col);
73. RVON ();
74. printf ("%s", stg);
75. VOFF();
76. }
77. BEEP ();
78. /***************************************************************************/
79. PROCEDURE BEEP
80. /***************************************************************************
81. produces a beep
82. /***************************************************************************/
83. PROCEDURE CLL
84. /***************************************************************************/
85. clear line at given coordinate
86. /***************************************************************************/
87. PROCEDURE CLL
88. /***************************************************************************/
89. /***************************************************************************/
90. /***************************************************************************/
91. /***************************************************************************/
92. /***************************************************************************/
93. /***************************************************************************/
94. /***************************************************************************/
95. /***************************************************************************/
96. /***************************************************************************/
97. /***************************************************************************/
98. /***************************************************************************/
99. /***************************************************************************/
100. /***************************************************************************/
101. /***************************************************************************/
102. /***************************************************************************/
103. /***************************************************************************/
104. /***************************************************************************/
105. /***************************************************************************/
106. /***************************************************************************/
107. /***************************************************************************/
108. /***************************************************************************/
109. /***************************************************************************/
110. /***************************************************************************/
111. /***************************************************************************/
112. /***************************************************************************/
113. /***************************************************************************/
114. /***************************************************************************/
115. /***************************************************************************/
116. /***************************************************************************/
117. /***************************************************************************/
118. /***************************************************************************/
119. /***************************************************************************/
120. /***************************************************************************/

82510 XMODEM Implementation (Continued)
**procedure CLMS**

```c
PROCEDURE CLMS

PROCEDURE MV_CURS

MV_CURS (x,y)
```

**procedure MV_CURS**

```c
MV_CURS (x,y)
```

82510 XMODEM Implementation (Continued)
82510 XMODEM Implementation (Continued)
High Performance Driver for 82510

DAN GAVISH and TSVIKA KURTS
SYSTEM VALIDATION
1.0 OVERVIEW

The 82510 Asynchronous Serial Controller is a CHMOS UART which provides high integration features to offload the host CPU and to reduce the system cost.

This Ap-Note presents a mechanism for reduction and optimization of interrupt handling during asynchronous communication using the 82510. The mechanism is valuable in applications where handling of interrupts degrades system performance i.e., when high baud rate is used, when multiple channels are handled or when real-time constraints exist. This implementation of the mechanism is a software driver that transmits or receives characters at 288000 bits per second.

The driver is based on the burst algorithm which uses the 82510 features (FIFOs, Timers, Control Character Recognition etc.) to reduce CPU overhead. CPU is significantly off-loaded for other tasks — about 75% of the usual load is saved.

The driver can be easily modified to run in conjunction with other 82510 features such as the MCS-51 9-bit Protocol.

This document provides a full description of the driver. The burst algorithm is presented in Section 3, the software module flow-charts and their descriptions are presented in Section 6, and the PL/M software listing is given in Appendix A.

2.0 INTRODUCTION

2.1 CPU Load Consideration

The trend towards multi-tasking systems, combined with higher baud rates and increasing the number of channels per CPU, has led to the need for decreasing the CPU bandwidth consumed by the async communications for each byte transfer. Whenever the CPU is interrupted, a certain amount of CPU time is lost in implementing the context switch. This overhead can be as high as hundreds of microseconds per interrupt, depending on the specific operating system parameters. Thus, in high baud-rate or multi-channel environments, where the interrupt frequency is very high, a substantial portion of the CPU time is taken up by this interrupt overhead. Therefore, systems usually require minimization of the number of interrupt events. In the case of an asynchronous communication channel, reduction of the number of interrupts can be achieved by servicing (i.e., transferring to/from the buffer) as many characters as possible whenever the interrupt routine is activated. This can be done by utilizing FIFOs to hold received or transmitted characters, so that the CPU is interrupted only after a certain number of characters have been received or transmitted. Using a receive FIFO may cause a potential problem: Due to the random rate of character arrival in asynchronous communications, there is a chance that characters will be "trapped" in the Rx FIFO for extended periods of time. In order to avoid such situations, a Rx FIFO time-out mechanism can be implemented using the 82510 timer. The timeout indicates that a certain amount of time has elapsed since the last read operation was performed. It causes the CPU to check the Rx FIFO and read any characters that are present. This process, however, introduces the additional overhead of the timer interrupt. This Ap-Note describes the use of the burst algorithm to avoid the timer interrupt overhead while maintaining the use of the Rx FIFO.

2.2 82510 Features Used In This Implementation

The following new 82510 features were used in this implementation:

2.2.1 FIFOs

The 82510 is equipped with 2 four-byte FIFOs, one for reception and one for transmission. While characters are being received, a Rx FIFO interrupt is generated, when the Rx FIFO occupancy increases above a programmable threshold. While characters are being transmitted, a Tx FIFO interrupt is generated, when the Tx FIFO occupancy drops below a programmable threshold. The two thresholds are software programmable, for maximum optimization to the system requirements.

2.2.2 TIMER

The 82510 is equipped with two on chip timers. Each timer can be used as a baud rate generator or as a general purpose timer. When two independent baud rates are required for transmit and receive, the two timers can be used to generate both baud rates internally. Otherwise, one timer can be used for external purposes. The timer is loaded with its initial value by a software command and it counts down using system clock pulses. When it expires, a maskable interrupt is generated.
2.2.3 CONTROL CHARACTER RECOGNITION

Depending on the application, the software usually checks the received characters to determine whether certain control characters have been received, in which case special processing is performed. This loads the CPU, as every received character should be compared to a list of control characters. With the 82510, the CPU is offloaded from this overhead. Every received character is checked by the 82510, and compared to either a standard set of control characters (ASCII or EBCDIC) or to special user defined control characters. The software does not need to check the received characters, and a special interrupt is provided when a received control character is detected by the 82510. The specific operation mode (standard set, user defined, etc.) is programmable.

2.2.4 INTERRUPT CONTROLLING MECHANISM

The twenty possible interrupt sources of the 82510 are grouped into six blocks: Timer, Tx machine, Rx machine, Rx FIFO, Tx FIFO, or Modem. Interrupt source blocks are prioritized. The interrupt management is performed by the 82510 hardware. The CPU is interrupted by a single 82510 interrupt signal. The interrupt handler is reported on the highest priority pending interrupt block (GIR) and on all the pending interrupt blocks (GSR), as well as on the specific interrupt source. Interrupts are maskable at the block level and source level. Interrupts can be automatically acknowledged (become not pending) when serviced by the software, or manually acknowledged by an explicit command.

3.0 THE BURST ALGORITHM

3.1 Background

The 82510 FIFOs are used to reduce the CPU interrupt load. When a burst of characters is transmitted or received, the CPU is interrupted only once per transmission or reception of up to four characters. FIFO thresholds are programmable; thus, when high system interrupt latency is expected, an optimal threshold may be selected for the desired trade-off between the CPU load, and the acceptable system interrupt latency. The required Rx FIFO threshold is also a function of the receive character rate. When the rate is high, a deep FIFO is required. When the rate is very low (e.g., hundreds of milliseconds between characters), a low threshold is needed, to reduce the maximum character service latency (a character is available to the application program only after it is stored in the receive buffer).

The software mechanism described here tunes the Rx FIFO threshold dynamically when the incoming character rate is variable. The algorithm uses one of the 82510 on-chip timers for time measurement, in order to automatically adapt the threshold to the character reception rate. This is done without loading the CPU with the overhead of serving excessive interrupts generated by the timer mechanism itself.

3.2 Burst Algorithm Description

The 82510 timer is initialized to the time-out value with every Rx FIFO interrupt. The time-out value is the maximum acceptable time between a character's reception and its storage in the receive buffer, but not less than five character-times. Upon reception of the next character, the timer status is examined to determine whether the character rate is high (the timer has not yet expired) or low (the timer has expired).
The algorithm is best described as a finite state machine that can be in one of three modes: HUNTING mode, SINGLE mode, or BURST mode. In HUNTING mode, after the first character received interrupts the CPU, the mode switches to SINGLE. On receiving a character in SINGLE mode (that is the second character) the timer is examined; if the character rate is very low, the mode is switched back to HUNTING. Otherwise, the rate is high enough to switch to BURST mode. In BURST mode, the Rx FIFO threshold is maximal. The machine remains in BURST mode as long as a burst of characters is being received. When the rate of character reception becomes low, the timer eventually expires generating a timer interrupt which switches the mode back to HUNTING.

Note that while a burst of characters is being received, the CPU is interrupted only once per four received characters. If the characters are received at a very low rate, an interrupt occurs for each received character. The CPU is interrupted by the timer only once, when the burst terminates. See Figure 1 for a state diagram.

For more details about the burst algorithm see paragraph 6.2.

4.0 SOFTWARE MODULE MAP

The driver contains the following software modules:

- **MAIN**
- **BURST ALGORITHM**
  - Burst Algorithm Initialization (*)
  - Rx FIFO Step (*)
  - HUNTING mode
  - SINGLE mode
  - BURST mode
  - Timer Step (*)
- **INITIALIZATIONS**
  - Wait for Modem Status
- **INTERRUPT HANDLER**
  - Rx FIFO Interrupt Service Routine
  - Tx FIFO Interrupt Service Routine
  - Status Interrupt Service Routine
  - Timer Interrupt Service Routine
  - Modem Interrupt Service Routine

(*) The burst algorithm modules are called by the initialization module and by the interrupt handler modules.
5.0 HARDWARE VEHICLE DESCRIPTION

The driver was tested at 288000 baud, on an 80186 based system, with an 8 MHz local bus running with 2 wait-states, and an 18.432 MHz 82510 clock. Two stations were involved: one transmitter station and one receiver station. Each station consisted of an iSBC186/51 with a 82510 based SBX board connected to it. See Appendix B for description of the SBX board.

This driver is, nonetheless, suitable for running in a large number of system environments.

6.0 SOFTWARE MODULE DESCRIPTIONS

6.1 MAIN

The MAIN module is a simple example of an application program that uses the driver.

The communication is done between two stations: One station is the transmitter and the other one is the receiver. After interrupts are enabled, the program waits for the Finish_Tx flag or the Finish_Rx flag (for the transmitter or receiver station, respectively) to be set. In the transmitter station, the driver is preloaded with the transmit data. In the receiver station, the received data is displayed after data reception is complete.
6.2 The Burst Algorithm Modules

6.2.1 INITIALIZE THE BURST ALGORITHM

This module is called by the initialization module.

The global variable Burst_algo is used to indicate the current burst algorithm mode.

The burst algorithm is most useful at a baud rate of 9600 or higher. At lower baud rates, where the Rx interrupt rate is very low, the burst algorithm is degenerated (Low_baud is assigned to Burst_algo). At a baud rate of 9600 or more, the burst algorithm mechanism is initialized and starts by disabling the timer interrupt.

The initial state of the burst algorithm is HUNTING mode. In this mode, it is looking for (hunting) the first character. The Rx FIFO threshold is zero, thus the first character received interrupts CPU. This interrupt starts the burst algorithm mechanism.
6.2.2 BURST ALGORITHM MECHANISM

Modules HUNTING, SINGLE, BURST are called by Rx FIFO interrupt service routine. Module BURST&TIMER is called by timer interrupt service routine.

6.2.2.1 HUNTING Mode

Hunting for the first character received is the first step in the burst algorithm. After the first character is detected, received and handled, it must be determined if reception will be at high or low rate. This is done by starting the timer. HUNTING mode ends by assigning the second step, i.e., SINGLE mode, to Burst_algo.

6.2.2.2 SINGLE Mode

When the second character is received, the burst algorithm is in SINGLE mode. Timer status is read (TMST). If the status indicates that the timer has expired, the receive character rate is low and there is no need to increase the Rx FIFO threshold. The burst algorithm returns to its first state, i.e., HUNTING mode. However, if the timer has not expired, the receive character rate is high, and the Rx FIFO threshold is set to the maximal allowable value. The timer is restarted and the timer interrupt is enabled so that, if it expires before the Rx FIFO exceeds the threshold, a timer interrupt will occur.

SINGLE mode is ended by assigning the third step, BURST mode, to BURST_algo.

---

Figure 5. The Burst Algorithm
6.2.2.3 BURST Mode

The algorithm enters BURST mode as soon as the receive character rate is evaluated as high, i.e., when two successive characters are received without a timer expiration. The FIFO is now working at full threshold and the timer is used as a timeout watch dog. BURST mode is the most time-critical path of the algorithm. Therefore, it consumes a minimum amount of real time.

The timer is restarted, in order to restart a new timeout measurement. The timer status is read to trigger automatic reset of the previous status; this is done to avoid the timer interrupt if the timer has expired during the Rx FIFO interrupt service routine execution.

6.2.2.4 Timer Interrupt and Burst Algorithm

If the character reception rate becomes low, then the time between two successive Rx FIFO interrupts increases. Hence, a reduction in the reception rate causes the timeout to expire, and a timer interrupt occurs. This drives the algorithm back to HUNTING mode. The timer interrupt is disabled and the Rx FIFO threshold is configured to zero, to issue an Rx interrupt on the first hunted character.

![Figure 6. Timer Interrupt and BURST Algorithm](image)

Table 1. BURST Algorithm Modes

<table>
<thead>
<tr>
<th>Mode</th>
<th>FIFO Threshold</th>
<th>Timer</th>
<th>Timer-Interrupt</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hunting</td>
<td>0</td>
<td>Idle</td>
<td>Disabled</td>
</tr>
<tr>
<td>Single</td>
<td>0</td>
<td>Started</td>
<td>Disabled</td>
</tr>
<tr>
<td>Burst</td>
<td>Max.</td>
<td>Restarted</td>
<td>Enabled</td>
</tr>
</tbody>
</table>

6.3 Initializations

This module initializes the driver. It is called at program start-up.

The 82510 is configured for the specific operation mode by the CONFIG_82510 submodule: A Software Reset command is issued, and then the character configuration is selected. In the receiver station ACR0 and ACR1 Registers are loaded with the End-Of-File ASCII character, so that the Control Character Recognition feature of the 82510 can be used to detect the specific file terminator. In the transmitter station, the ASCII characters XOFF and XON are loaded to ACRO and ACR1, respectively, to detect transmit-off/on requests automatically. The use of the control character recognition feature of the 82510 reduces system overhead, as the software does not need to check every received character. A special interrupt is received when the 82510 hardware detects a received control character.

Interrupt sources are enabled (note that a Tx interrupt will occur immediately). BRGA is loaded to generate the required baud rate (288000 baud in this specific implementation). Rx FIFO depth is set to 4. The Tx and Rx FIFO thresholds are initialized to 0. BRGB is selected to function as a timer, and is loaded with the timeout value (7 ms at 18.432 MHz, in this implementation). The RxC and TxC sources are selected to be BRGA.

The burst algorithm parameters are initialized by INIT_BURST. WAIT_FOR_MODEM_STATUS is called and implements a wait until the modem handshake DSR signal is set. If WAIT_FOR_MODEM_STATUS returns with a timeout error, the modem error is processed. If no error has occurred, the following parameters are initialized: Finish_Rx and Finish_Tx flags, receive and transmit buffer pointers, and the receiver flag. All status registers are cleared by issuing a STATUS CLEAR command to the ICM register.
Figure 7. Initializations
Figure 8. 82510 Configurations
6.3.1 WAIT__FOR__MODEM__STATUS

This module waits, with a timeout, for the DSR modem handshake signal to be set. DSR should be active before any communication starts (it indicates that the modem is active). The returned Modem_Handshake flag indicates normal return (true) or timeout error return (false).

![Flowchart](Figure%209.%20Wait__For__Modem__Status)
6.4 Interrupt Handler

The interrupt handler services the 82510 interrupt sources. Since this is a time-critical path, the code is optimized to minimize real time consumption.

The interrupt handler services only one interrupt source at a time. This prevents CPU resource starvation from other interrupt driven devices. Interrupts are enabled at the beginning of the interrupt handler, so that higher priority interrupt sources are not disabled by the 82510 interrupt handler.

6.4.1 INTERRUPT HANDLER STRUCTURE

The interrupt handler identifies the highest priority pending 82510 interrupt, by reading GIR. The interrupt handler was designed so that shorter paths are assigned to more real time sensitive interrupt sources. Rx FIFO interrupt is the most sensitive, Tx FIFO is the second most sensitive, and so on.

The programmable interrupt controller (8259A) is assumed to be configured to “edge triggering mode” and “non-automatic end of the interrupt” mode.

![Diagram of Interrupt Handler](image)
6.4.2 Rx FIFO INTERRUPT SERVICE ROUTINE

The Rx FIFO interrupt service routine first empties the Rx FIFO. The receive data register (RXD) is read, as many times as indicated by the FIFO occupancy register (FLR), and the characters are stored in Rx__Buf.

After emptying the Rx FIFO, the Rx FIFO interrupt service routine executes the burst algorithm (see paragraph 6.2.2). Before leaving the Rx FIFO interrupt service routine, the FIFO occupancy register is rechecked, to empty the Rx FIFO of characters that may have been received during the Rx FIFO interrupt service routine itself. This can happen if the Rx FIFO interrupt service routine has been interrupted by a higher priority interrupt.

Figure 11. Rx FIFO Interrupt Service Routine
6.4.3 Tx FIFO INTERRUPT SERVICE ROUTINE

The Tx FIFO interrupt service routine fills the Tx FIFO with transmit characters while checking for the End-Of-File terminator. According to the FIFO occupancy register (FLR), the Tx FIFO is loaded (by writing to TXD) until it is full or until the End-Of-File character is detected. The transmitted characters are taken from Tx_Buf. If an End-Of-File character is identified, then the transmission is immediately ended by disabling all 82510 interrupts and setting the Finish_Tx flag.

6.4.4 STATUS INTERRUPT SERVICE ROUTINE

The status interrupt service routine has four objectives:

- To empty the Rx FIFO.
- To stop reception if an End-Of-File character is identified by the control character recognition mechanism (in the receiver station).
- To disable or enable the Tx interrupt if a XOFF or XON character, respectively, is identified by the control character recognition mechanism (in the transmitter station).
- To handle parity, framing, or overrun errors (in the receiver station).

Figure 12. Tx FIFO Intr Service Routine
First the Rx FIFO is emptied. In the receiver station, the RST register is checked to determine whether an End-Of-File terminator has been identified by the 82510, in which case reception is stopped immediately by disabling all interrupt sources and setting the Finish_Rx flag. In the transmitter station, the received characters are checked to identify the received control character. If XOFF is identified, Tx interrupt is disabled. If XON is identified, Tx interrupt is enabled. Note that the software does not need to check for any control character during normal reception; the control characters are identified by the 82510 device.

RST is checked for parity, framing or overrun errors. If one of these errors has occurred, then the error handling routine is executed.

If status interrupt occurs while Burst_algo is assigned to BURST mode, the timer is restarted.

Note that status interrupt is enabled at both stations.

![Status Intr. Service Routine](292038-13)

Figure 13. Status Intr. Service Routine
6.4.5 TIMER INTERRUPT SERVICE ROUTINE

A timer interrupt occurs when the receive character rate becomes low. The timer interrupt service routine first empties the Rx FIFO and then switches the burst algorithm to HUNTING mode.

![Diagram of TIMER Intr Service Routine](292038-14)

Figure 14. TIMER Intr Service Routine

6.4.6 MODEM INTERRUPT SERVICE ROUTINE

Modem interrupt occurs if one of the modem lines has dropped during transmission or reception. The modem interrupt service routine reads the MSR register to acknowledge the modem interrupt. The modem error routine is then executed.

![Diagram of MODEM Intr Service Routine](292038-15)

Figure 15. MODEM Intr Service Routine
APPENDIX A
PL/M SOURCE FILE

/*******************************************************************************
* * 82510 - HIGH PERFORMANCE Driver *
* *
* This driver is optimized for Real Time Systems. It supports *
* high system performance. It is based on the "BURST algorithm"
*******************************************************************************/

HIGHPERFORMANCE: DO ;

/*******************************************************************************
* LITERALS
*******************************************************************************

DECLARE LIT LITERALLY 'LITERALLY';
DECLARE TRUE LIT 'OFFH';
DECLARE FALSE LIT 'O0H';
DECLARE BAUD_9600 LIT '003CH'; /* Character configurations */
DECLARE BAUD_19200 LIT '001EH';
DECLARE BAUD_288000 LIT '0002H';
DECLARE DLAB_0 LIT '01111111B'; /* Reset DLAB */
DECLARE DLAB_1 LIT '10000000B'; /* Set DLAB */
DECLARE CR LIT 'ODH'; /* Control characters */
DECLARE LF LIT '0AH';
DECLARE X_Off LIT '13H';
DECLARE X_On LIT '11H';
DECLARE End_Of_File LIT '1AH';
DECLARE BASE_510 LIT '080H' /* 82510 registers */
DECLARE NAS0 LIT '00000000B';
DECLARE WORK1 LIT '00100000B';
DECLARE GEN2 LIT '01000000B';
DECLARE MODM3 LIT '01100000B';
DECLARE TXD LIT 'BASE_510 + 0' /* BANK 0 - NAS */
DECLARE RXD LIT 'BASE_510 + 0';
DECLARE BAL LIT 'BASE_510 + 0';
DECLARE BAH LIT 'BASE_510 + 2';
DECLARE GER LIT 'BASE_510 + 2';
DECLARE GIR LIT 'BASE_510 + 4';
DECLARE BANK LIT 'BASE_510 + 4';
DECLARE LCR LIT 'BASE_510 + 6';
DECLARE MCR LIT 'BASE_510 + 8';
DECLARE LSR LIT 'BASE_510 +10';
DECLARE MSR LIT 'BASE_510 +12';
DECLARE ACR0 LIT 'BASE_510 +14';
DECLARE RXF LIT 'BASE_510 +2' /* BANK 1 - WORK */
DECLARE TXF LIT 'BASE_510 +2';
DECLARE TMST LIT 'BASE_510 +6';
DECLARE TMCR LIT 'BASE_510 +6';
DECLARE FLR LIT 'BASE_510 +8';
DECLARE RST LIT 'BASE_510 +10';
DECLARE RCM LIT 'BASE_510 +10';
DECLARE TCM LIT 'BASE_510 +12';
DECLARE GSR LIT 'BASE_510 +14';
DECLARE TCM LIT 'BASE_510 +14';
DECLARE PMD LIT 'BASE_510 +2' /* BANK 2 - GENERAL CONFIGURE */
DECLARE TMD LIT 'BASE_510 +6';

2-278
DECLARE IMD LIT 'BASE_510 + 8';
DECLARE ACR1 LIT 'BASE_510 +10';
DECLARE RIE LIT 'BASE_510 +12';
DECLARE RMD LIT 'BASE_510 +14';
DECLARE CLCF LIT 'BASE_510 + 0';
DECLARE BBL LIT 'BASE_510 +0';
DECLARE BACF LIT 'BASE_510 +10';
DECLARE RIE LIT 'BASE_510 +12';
DECLARE RMD LIT 'BASE_510 +14';
DECLARE TMIE LIT 'BASE_510 +16';
DECLARE OUT2 MCR LIT '0001000B';
DECLARE DTR MCR LIT '0000000B';
DECLARE DSR-MSR LIT '0010000B';
DECLARE CLRSTAT ICM LIT '0000010B';
DECLARE INTR S10 LIT '21H';
DECLARE PORT-S0130M LIT 'OE2H';
DECLARE EN 80130 LIT 'OFDH';
DECLARE EOI LIT 'OE2H';
DECLARE COMM-EOI LIT '61H';
DECLARE ENRTX GER LIT '0000111B';
DECLARE ENTX GER LIT '0000001B';
DECLARE ENTXSTAT GER LIT '0000111B';
DECLARE DISRX-GER LIT '0000001B';
DECLARE DISTX GER LIT '0000111B';
DECLARE RXTHRESH0-FMD LIT '0000000B';
DECLARE RXTHRESH3-FMD LIT '0010000B';
DECLARE RXTHRESH1-FMD LIT '0000000B';
DECLARE RXTHRESH2-FMD LIT '0000000B';
DECLARE UART')))
DECLARE TX_PTR POINTER PUBLIC ; /* Transmit buffer */
DECLARE TX_BUF BASED TX_PTR (3000) BYTE ;
DECLARE TX_TX WORD PUBLIC ;
DECLARE RX_BUF(3000) BYTE PUBLIC ; /* Receive buffer */
DECLARE IX_RX WORD PUBLIC ;
DECLARE INTR_VEC BYTE PUBLIC ;
DECLARE FIN_TX BYTE PUBLIC ; /* Finish Transmission flag */
DECLARE FIN_RX BYTE PUBLIC ; /* Finish Reception flag */
DECLARE RX_CHR BYTE PUBLIC ;
DECLARE TX_CHR BYTE PUBLIC ;
DECLARE RX_OCC BYTE PUBLIC ;
DECLARE STAT BYTE PUBLIC ;
DECLARE BAUD WORD PUBLIC ;
DECLARE TEMP BYTE PUBLIC ;
DECLARE FIN BYTE PUBLIC ;
DECLARE BURST_ALGO BYTE PUBLIC ;
DECLARE RECEIVER BYTE PUBLIC ; /* Receive station */
DECLARE BURST_ALGO BYTE PUBLIC ; /* BURST algorithm */
DECLARE MODEM_HANDSHAKE BYTE PUBLIC ;
DECLARE COUNTER WORD PUBLIC ;
DECLARE RX_ERROR BYTE PUBLIC ; /* Error occurred during */
/* reception */

/****************************************************************************
* I/O console utilities
$INCLUDE (:F1:TIOPP.PEX)
*/

/* Setup and H/W configurations
$INCLUDE (:F1:HPUTIL.PEX)
*/

DECLARE MAIN LABEL PUBLIC ;

****************************************************************************
* Procedure INITIALIZATIONS
****************************************************************************
* input: none
* output: none
* function: driver initialization: parameters, 82510
* configuration, modem status check.
* called by: Main
* calling: CONFIG_82510, INITIALIZE_BURST, WAIT_FOR_MODEM
* 
* Init the Interrupt mechanism by enable Interrupt in GER register
* At the Receive station: Enable Rx FIFO, Status and Modem Interrupts
* Disable Timer Interrupt
* At the Transmit station: Enable Tx FIFO, Status and Modem Interrupts
* 
* flowchart: figure 7 description: paragraph 6.3
****************************************************************************

INITIALIZATIONS: PROCEDURE PUBLIC ;

DISABLE ;
CALL SET_INTERRUPT(INTR_510, INTR_HANDLER) ; /* Install THE INTR HANDLER */
TX_CHR=00 ; /* Clear TX_CHR and RX_CHR */
RX_CHR=00 ;

292038-18
CALL TEXT ; /* TX_PTR is a pointer to the transmitted */
/* data */
IX_TX=0FFFH ; /* The index buffer are assigned to -1 */
IX_RX=0FFFH ;
FIN_TX=FALSE ; /* Init Finish Transmit and receive flags */
FIN_RX=FALSE ;
RX_BUF(0)=0 ; /* Reset the flag */
RX_ERROR=FALSE ;
BAUD=BAUD_288000 ; /* The Async communication Baud rate is */
/* the 82510-full scale 288000 */
CALL CONFIG_82510 ; /* Configured the 82510 : */
/* S/W reset, character length, parity, */
/* stop bit, baud rate and fifo threshol */

FLICTX-FALSE ;
FIN_RX-l"ALSE ;
RX_BUF(0)-O ;
RX_ERROR "alse ;/* Reset the flag */
BAUD-BAUD_288000 ; /* The Async communication Baud rate is */
/* the 82510-full scale 288000 */
CALL CONFIG_82510 ; /* Configured the 82510 : */
/* S/W reset, character length, parity, */
/* stop bit, baud rate and fifo threshol */

/*****************************************************************************/
/* INITIALIZE_BURST */
/*****************************************************************************/
* input: none *
* output: Burst_Algo *
* function: start Burst algorithm in Hunting mode *
* called by: INITIALIZATIONS *
* calling: none *
* *
* flowchart: figure 4 description: paragraph 6.2.1 *

*****************************************************************************/
IF BAUD<=BAUD_9600 THEN BURST_ALGO-HUNTING_MODE ; /* HUNTING mode: */
/* Rx FIFO threshold is 0 */
/* Timer interrupt is disable */
ELSE BURST_ALGO-LOW_BAUD ;
CALL WAIT_FOR_MODEM_STATUS ; /* Wait for Modem handshake line "DSR" */
TEMP = INPUT(RXD) ; /* if ACTIVE set MODEM_HANDSHAKE */
TEMP = INPUT(RXD) ;
TEMP = INPUT(RXD) ;
TEMP = INPUT(RXD) ;
TEMP = INPUT(RST) ;
END INITIALIZATIONS ;

/*****************************************************************************/
* Procedure CONFIG_82510 *
/*****************************************************************************/
* input: none *
* output: none *
* function: configure the 82510 to a specific operation *
* mode *
* called by: INITIALIZATIONS *
* calling: none *
* *
* flowchart: figure 8 description: paragraph 6.3 *

*****************************************************************************/
CONFIG_82510: PROCEDURE PUBLIC ;
/* Perform Software reset */
OUTPUT(BANK) = WORK1; /* Move to work bank */
OUTPUT(ICH) = SWRES_CMND; /* S/W reset command */

2-281
/* BANK ZERO - NAS (The default BANK) */

/* Configured the character by writing to LCR: */
/* 1 stop bit, 8 bit length, non parity */
OUTPUT(LCR)=((STOPBIT_1 + CHRL_8 + PARITY_NON) ;
OUTPUT(MCR)=(DTR_MCR OR OUT2_MCR) ;

/* Required only in IBM PC environment: */
/* set OUT2 signal to control an external */
/* 3-state buffer that drives the 82510 */
/* interrupt signal */

IF RECEIVER THEN OUTPUT(ACR0)=End_Of_File ;
/* At the Receive station EOF is */
ELSE OUTPUT(ACR0)= X_OFF ;
/* At the Transmit station "X Off" is */
/* recognized to stop transmission */
/* temporary */
/* Enable 82510 Interrupt by set GER, */
/* done at the end of INITIALIZATIONS */
/* Init the 82510 Interrupt mechanism */
DISABLE ;
IF RECEIVER THEN OUTPUT(GER)=ENRX_GER ;
ELSE OUTPUT(GER)=ENTXSTAT_GER ;
/* at the Receive station */
/* and the Transmit station */
/* Configured baud rate to 288000 */
/* by writing to BRG A (BAL and BAH) */
OUTPUT(LCR)=INPUT(LCR) OR DLAB_1; /*Set DLAB to allow access to BRG */
OUTPUT(BAL)=LOW (BAUD_288000) ;
OUTPUT(BAH)=HIGH (BAUD_288000) ;
OUTPUT(LCR)=INPUT(LCR) AND DLAB_0; /* reset DLAB */

/* BANK TWO - General configuration */
OUTPUT(BANK)=GEN2 ;

OUTPUT(IMD)=(AUTOACK_IMD OR FIFO_IMD) ;
/* Automatic interrupt acknowledge, */
/* Rx Fifo depth is four bytes */
OUTPUT(FMD)=(TXTHRESH0_FMD OR RXTHRESH0_FMD) ;
/* Rx Fifo threshold is temporally zero */
/* for HUNTING mode (BURST algorithm) */
/* Tx Fifo threshold is zero for max */
/* interrupt latency */

IF RECEIVER THEN OUTPUT(ACR1)=End_Of_File ;
/* At the Receive station EOF is */
ELSE OUTPUT(ACR1)=X_ON ;
/* At the Transmit station "X On" is */
/* recognized to continue transmission */

OUTPUT(RIE) = (ACRSTAT_RIE OR INPUT(RIE)) ;
/* Enable interrupt on programmed control*/
/* character received (ACR0/ACR1) */

/* BANK THREE - MODEM configuration */
OUTPUT(BANK)=MODM3 ;

OUTPUT(BBCF)=(TIMOD_BBCF) ; /* BRG B configured to TIMER mode */
OUTPUT(BANK) = NASO; /* Move to nas bank to set DLAB */
OUTPUT(LCR)=INPUT(LCR) OR DLAB_1 ; /* Set DLAB to allow access to BRG */
OUTPUT(BANK) = MODM3 ; /* MODEM bank */
OUTPUT(BBL) = LOW (TIME_EXP); /* Set max timeout (7ms if 18Mhz crystal)*/
OUTPUT(BBH) = HIGH(TIME_EXP);  /* to issue interrupt when time has */
OUTPUT(BANK) = NASO; /* expired. Move to NAS bank again */
OUTPUT(LCR) = INPUT(LCR) AND DLAB_0; /* Reset DLAB */
OUTPUT(BANK) = MODM3;  /* Switch to BANK THREE - MODEM */
OUTPUT(CLCF) = RTXCLK_BRGA_CLCF; /* The receive and transmit clock source */
      /* is BRG A */
OUTPUT(TMIE) = TIMBI_TMIE;  /* Enable Timer block interrupt */
      /* (still disabled in Timer bit in GER) */
      /* BANK ONE - general WORK */
OUTPUT(BANK) = WORK1;
OUTPUT(ICM) = CLRSTAT_ICM;  /* Issues a command to clear all */
      /* status registers */

/* Remain in   WORK - THE runtime bank */

END CONFIG_82510;

/***********************************************************/
/* Procedure WAIT_FOR_MODEM_STATUS */
/***********************************************************/
   input: none
   output: Modem Handshake
   function: waits with a timeout for DSR active,
             returns status flag
   called by: INITIALIZATIONS
   calling: none
   flowchart: figure 9 description: paragraph 6.3.1
***********************************************************/

WAIT_FOR_MODEM_STATUS: PROCEDURE PUBLIC;

MODEM_HANDSHAKE = FALSE;
COUNTER = WAIT_TIME;

DO WHILE (NOT MODEM_HANDSHAKE) AND ((COUNTER=COUNTER-1) > 0);
    IF (INPUT(MSR) AND DSR_MSR) <> 0 THEN MODEM_HANDSHAKE = TRUE;
END;

END WAIT_FOR_MODEM_STATUS;

/***********************************************************/
/* Procedure INTERRUPT_HANDLER */
/***********************************************************/
   input: Tx Buffer
   output: Rx Buffer, Finish.Tx, Finish.Rx
   function: service all 82510 interrupt sources:
             Rx FIFO, Tx FIFO, Status, Timer, Modem
   called by: 82510 hardware interrupt
   calling: Rx_Fifo_Intr, Tx_Fifo_Intr, Status_Intr, 
             Timer_Intr, Modem_Intr
   flowchart: figure 10 description: paragraph 6.4, 6.4.1
***********************************************************/

INTR_HANDLER: PROCEDURE INTERRUPT_INTR_510 REENTRANT PUBLIC;

ENABLE ;  /* Enable Interrupts of */
      /* HIGHER priority devices */
INTR_VEC=INPUT(GIR);  /* Get the 82510-highest priority */
      /* pending interrupt */
/**.***********************************************************************
 * Rx FIFO INTR
 *-------------------------------------------------------------------------
 * input: none
 * output: Rx_Buffer, Burst_Algo
 * function: service Rx Fifo interrupt
 * receive characters; store in receive buffer
 * called by: INTERRUPT HANDLER
 * calling: BURST_ALGO
 * flowchart: figure 11 description: paragraph 6.4.2
 *-------------------------------------------------------------------------/ 

IF INTR_VEC=RXI_GIR THEN DO;

    RX_OCC=INPUT(FLR) ; /* Rx fifo level occupancy */
    RX_OCC=SHR(RX_OCC,4) ; /* to get it's real value */
                        /* - OPTIMIZE code - */
                        /* Empty the Rx FIFO and store the */
                        /* received character in RX_BUF */
    RX_BUF(IX_RX:=IX_RX+1)=INPUT(RXD) ; /* Read the first character immediately */
                        /* to save Real Time */
    DO WHILE (RX_OCC:=RX_OCC-1) > 0 ; RX_BUF(IX_RX:=IX_RX+1)=INPUT(RXD) ; END ;

/*******************************************************************************
 * BURST_ALGORITHM
 *-------------------------------------------------------------------------
 * input: Burst_Algo
 * output: Burst_Algo
 * function: execute a step in the burst algorithm
 * after characters are received
 * called by: Rx_FIFO_INTR
 * calling: none
 * flowchart: figure 5 description: par. 6.2.2.1 to 6.2.2.3
 *-------------------------------------------------------------------------/

/*******************************************************************************
 * B U R S T _ M O D E - step 3 (full fifo threshold) *
 * Reset the Timer status
 * Restart the Timer
 *-------------------------------------------------------------------------/ 

IF BURST_ALGO = BURST_MODE THEN DO;
    TEMP = INPUT(TMST); OUTPUT(TMCR) =STARTIMB_TMCR;
END;

/*******************************************************************************
 * H U N T I N G _ M O D E - step 1 *
 * Operate the TIMER*
 * Change to step 2 SINGLE mode
 *-------------------------------------------------------------------------/

ELSE IF BURST_ALGO = HUNTING_MODE THEN DO;
    OUTPUT(TMCR)=STARTIMB_TMCR ; BURST_ALGO=SINGLE_MODE ;
END ;
/* S I N G L E M O D E - s t e p 2 */
* If TIME has expired, means the receive rate is LOW, return to HUNTING mode
* If TIME did NOT expire, means the receive rate is HIGH, set Rx FIFO threshold, restart the
* Timer and switch to BURST mode
*/

ELSE IF BURST_ALGO = SINGLE_MODE THEN DO;

IF ((INPUT(TMST) AND STARTIMB TMST) <>0) THEN
    BURST_ALGO= HUNTING_MODE ;
ELSE DO;
    OUTPUT(BANK) = GEN2;/* Switch to BANK TWO - General Config */
    OUTPUT(BANK) = NASO; /* Switch to BANK ZERO - NAS */
    OUTPUT(GER) = ENTIMRX_GER;
    /* Enable TIMER,RX and MODEM interrupts */
    OUTPUT(BANK)=WORK1; /* Switch to BANK ONE - WORK */
    BURST_ALGO = BURST_MODE;
    TEMP = INPUT(TMST); /* Reset timer status */
    OUTPUT(TMCR) = STARTIMB TMCR;
END;

END; /* End of SINGLE mode */

/* ....End of BURST algorithm */

/* Another try to empty the Rx fifo */
/* before leaving the interrupt handler */

DO WHILE (INPUT(FLR) <>0) ;
    RXBUF(IX RX:=IX RX+1)=INPUT(RXD) ;
END ; /* End of Rx fifo interrupt */

ELSE IF INTR_VEC=TXI GIR THEN DO ;
    TX_OCC=INPUT(FLR) AND MASK_TXOCC ;
    /* Tx fifo level occupancy */
    /* Fill Tx FIFO, the transmitted characters are taken from TX_buf */
    DO WHILE (TX_OCC:=TX_OCC+1)<5 ;
        OUTPUT(TXD)=TX_BUF(IX TX:=IX TX+1);
        IF TX_BUF(IX TX)=End Of File·THEN DO ;
            OUTPUT(BANK)=NASO ; /* Disable Tx interrupt, as the transit */
            OUTPUT(GER)=DISTX GER; /* delimiter character was identified */
            OUTPUT(BANK)=WORK1; /* Switch to BANK ONE - WORK */
            TX_OCC = 5 ; /* load TX_OCC to terminate external loop*/
            FIN_TX = TRUE ; /* Set Finish transmit flag */
        END ;
    END ; /* End of TXFIFO_INTR */
STATUS_INTR

Function: service Status interrupt

Receive station: EOF terminate the reception

Transmit station: X_Off Disable the transmission

X_On Enable the transmission

Called by: INTERRUPT HANDLER

Flowchart: figure 13 description: paragraph 6.4.4

STATUS_INTR

input: none

output: Finish_Rx

function: service Status interrupt

Receive station: EOF terminate the reception

Transmit station: X_Off Disable the transmission

X_On Enable the transmission

called by: INTERRUPT HANDLER

calling: none

flowchart: figure 13 description: paragraph 6.4.4

ELSE IF INTR_VEC=STATI_GIR THEN DO;

STAT=INPUT(RST); /* Get the current RST status */

RX_OCC=INPUT(FLR); /* Rx fifo level occupancy */

RX_OCC=SHR(RX_OCC,4);

DO WHILE (RX_OCC>0 AND (NOT FIN_RX));

RX_OCC=RX_OCC-1; /* First, empty Rx FIFO */

RX_CHR=INPUT(RXD);

IF RECEIVER THEN RX_BUF(IX_RX:=IX_RX+1)=RX_CHR;

ELSE DO;

IF RX_CHR = X_OFF THEN DO;

OUTPUT(BANK)=NAS0; /* Switch to BANK ZERO - NAS */

OUTPUT(GER) = INPUT(GER) AND DISTX_GER; /* Disable Transmit interrupt */

OUTPUT(BANK)=WORK1; /* Switch to BANK ONE - WORK */

END;

ELSE IF RX_CHR = X_ON THEN DO;

OUTPUT(BANK)=NAS0;

OUTPUT(GER) = INPUT(GER) OR ENTX_GER; /* Enable Transmit interrupt again */

OUTPUT(BANK)=WORK1;

END;

END;

END;

IF RECEIVER THEN DO;

IF ((STAT AND ACRSTAT_RST) <> 0) THEN DO;

OUTPUT(BANK)=NAS0; /* If End_Of_Line was recognized, */

OUTPUT(GER) = DISRTX_GER;

OUTPUT(BANK)=WORK1; /* Disable 82510-interrupts and the */

FIN_RX = TRUE; /* Reception */

END;

ELSE IF ((STAT AND ERRCHR_RST) <> 0) THEN DO;

CALL WRITE('@("** ERROR in character Status ",0));

CALL ERROR_CHAR_HANDLER;

IF BURST_ALGO=BURST_MODE THEN DO;

/* In BURST mode do: */

TEMP = INPUT(TMST); /* Reset timer status */

OUTPUT(TMCR) = STARTIMB_THCK;

END; /* Restart TIMER */

END;

END;

END; /* End of STATUS interrupt */
ELSE IF INTR_VEC=TIMI_GIR THEN DO;
  IF ((RX_OCC:=INPUT(FLR))<>0) THEN DO;
    RX_OCC=SHR(RX_OCC,4); /* Rx fifo level occupancy, shift right */
    RX_BUF(IX_RX:=IX_RX+1)=INPUT(RXD);
    DO WHILE (RX_OCC:=RX_OCC-1) > 0 ;
     RX_BUF(IX_RX:=IX_RX+1)=INPUT(RXD);
    END;
  END;

******************************************************************************
*  BURST & TIMER
******************************************************************************
* input: Burst_Alg0
* output: Burst_Alg0
* function: execute a step in the burst algorithm
* called by: TIMER_INTR
* calling: none
* flowchart: figure 6 description: paragraph 6.2.2.4
******************************************************************************
OUTPUT(BANK) = GEN2; /* Switch to BANK TWO - General Config */
OUTPUT(FMD) = TXTHRESH0_FMD OR RXTHRESH0_FMD;
  /* Rxfifo threshold=0, Txfifo threshold=0*/
OUTPUT(BANK) = NAS0; /* Switch to BANK ZERO - NAS */
OUTPUT(GER) = ENRX_GER; /* Disable Timer interrupt and */
OUTPUT(BANK) = WORK1; /* Enable RX,STAT,MODEM interrupts */
TEMP = INPUT(TMST); /* Acknowledge TIMER interrupt */
BURST_ALGO = HUNTING_MODE; /* Back to HUNTING mode */
END; /* End of TIMER interrupt */
**MODEM_INTR**

* input: none
* output: none
* function: service Modem interrupt and handle modem errors.
* Modem interrupt is occurred if No Modem was setup, or
  if DSR was dropped in the middle of the communication
* called by: INTERRUPT_HANDLER
* calling: none
* flowchart: figure 15  description: paragraph 6.4.6

```plaintext
****
STAT=INPUT(MSR);
END;

OUTPUT(PORT_EOI)=COMM_EOI;
END INTR_HANDLER;
```

**ERROR_MODEM_HANDLER**

```plaintext
ERROR_MODEM_HANDLER: PROCEDURE PUBLIC;
MODEM_HANDSHAKE = FALSE; /* Flag indicates that an Error occurred*/
            /* in Modem */
END ERROR_MODEM_HANDLER;
```

**ERROR_CHAR_HANDLER**

```plaintext
ERROR_CHAR_HANDLER: PROCEDURE PUBLIC;
RX_ERROR = TRUE; /* Flag indicates that an Error occurred*/
            /* during Reception */
OUTPUT(BANK) = NAS0; /* Switch to BANK ZERO - NAS */
OUTPUT(GER) = DISRTX_GER; /* Disable all the 82510 Interrupts */
OUTPUT(BANK) = WORK1; /* Switch to BANK ONE - WORK */
END ERROR_CHAR_HANDLER;
```
PROCEDURE LOOP

LOOP procedure is executed until Transmission/Reception Finishes
* or until the loop ends.

LOOP: PROCEDURE PUBLIC
DECLARE N WORD;
DECLARE NUM WORD;
DECLARE MAXLOOP BYTE;
MAXLOOP = 20;
NUM = 0;
DO WHILE ( (NOT FIN_TX) AND (NOT FIN_RX) AND (NUM<MAXLOOP) );
    NUM = NUM + 1;
    CALL WRITELN(('@... Background Program ...',0));
    ENABLE;
    CALL TIME(5000);
END;

IF FIN_TX THEN CALL WRITELN(('@Transmission - ENDED ',0));
IF FIN_RX THEN CALL WRITELN(('@Reception - ENDED ',0));

OUTPUT(BANK)=NASO;
OUTPUT(GER)=DISRTX_GER;
OUTPUT(BANK)=WORK1;

IF RECEIVER THEN DO;
    CALL WRITELN(('@The Received Message:',0));
    CALL DISPTEXT(@RX_BUF);
END;
ELSE IF (NOT FIN_TX) THEN /* The Transmit station */
    CALL WRITELN(('@** ERROR -THE Transmission NOT ended successfully',0));
ELSE IF (NOT FIN_RX) THEN /* The Receiving station */
    CALL WRITELN(('@** ERROR -THE Reception NOT ended successfully',0));
END LOOP;

PROCEDURE TEXT

TEXT: PROCEDURE PUBLIC;

TX_PTR=@('>',CR,LF,
'ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789abcdefghijklmnopqrstuvwxyz0123456789',
CR,LF,
'ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789abcdefghijklmnopqrstuvwxyz0123456789',
CR,LF,
'ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789abcdefghijklmnopqrstuvwxyz0123456789',
CR,LF,
'ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789abcdefghijklmnopqrstuvwxyz0123456789',
CR,LF,
'ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789abcdefghijklmnopqrstuvwxyz0123456789',
CR,LF,
'ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789abcdefghijklmnopqrstuvwxyz0123456789',
CR,LF,
'ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789abcdefghijklmnopqrstuvwxyz0123456789',
CR,LF,
'ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789abcdefghijklmnopqrstuvwxyz0123456789',
CR,LF,
'ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789abcdefghijklmnopqrstuvwxyz0123456789',
CR,LF,
'ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789abcdefghijklmnopqrstuvwxyz0123456789',
CR,LF,End_Of_File,0);

END TEXT; /* End_Of_File-terminate the Transmission*/
/* External procedures*/

* WRITELN: I/O console utility - display a string, end with CR
* MENU: I/O console utility - display a menu, enter the user selection
* DISPTEXT: I/O console utility - display the contents of the receive buffer (Rx_buf)
* INIT_HARDWARE_SETUP: Setup and Hardware configurations of the specific station

/*************************************************************************
* Procedure MAIN
*************************************************************************/

input: Finish Rx, Finish Tx
output: Receiver flag
function: get station type (Rx or Tx) from the operator;
wait till communication is completed; display;
RECEIVER STATION SHOULD BE ACTIVATED FIRST

* called by: Application
* calling: INITIALIZATIONS, LOOP
* flowchart: figure 3 description: paragraph 6.1

MAIN:

CALL INIT_HARDWARE_SETUP /* External, Setup and H/W configurations*/
FIN=FALSE ;
DO WHILE NOT(FIN) ;
SELECTION=0 ;
CALL WRITELN('@('Station: (Quit/Transmitter/Receiver)',0));
SELECTION=MENU(SELECTION,@('Station: (Quit/Transmitter/Receiver)',0)); /* Get operator selection. */
/* Receiver station should be activated */
/* prior to the transmitter station */
DO CASE SELECTION ;
FIN=TRUE ; /* 0 - quit of HIGH PERFORMANCE Driver */
DO ; /* 1 - Transmit station */
RECEIVER=FALSE ;
CALL INITIALIZATIONS ;
CALL/ LOOP ;
END ;
DO ; /* 2 - Receive station */
RECEIVER=TRUE ;
CALL INITIALIZATIONS ;
CALL LOOP ;
END ;
END ;

CALL EXIT ;

END HIGHPERFORMANCE ;

/***********************************************************************/
APPENDIX B
82510 BASED SBX SERIAL CHANNEL

This document describes the implementation of an 82510 based SBX board that provides an RS-232 interface to any iSBC board which has an SBX connector. The SBX can be useful for customers that need a fast software development vehicle while the 82510 system hardware is still in the design stage. The customer can also use the SBX for evaluation of the 82510 in a system environment.

In order to minimize the customer's software development costs, the RMX86/286 Terminal Device Driver for the 82510 has also been developed and can be run by the RMX user on his iSBC with the SBX-82510 board described herewith. The RMX86/286 drivers are available from INSITE, along with the source code and the documentation.

BOARD DESCRIPTION (See Figure B-1)

The following 82510 signals are connected directly to the SBX connector (installed on the pin side): DATA, ADDRESS, INTERRUPT, RESET, READ#, WRITE# and CS#. Wait states are generated by a shift register logic (U5, U7), clocked by the MCLK signal of the SBX interface. The number of wait states is selected by installing one of the eight jumpers to select one parallel output of the shift register. The 82510 is clocked by an 18.432 MHz Crystal (using its on-chip oscillator). A discrete transistor is used to pull down the RTS# signal during RESET to set the crystal mode (note that in a larger board, an unused open collector inverter or three-state gate can be used for this purpose). The 82510 is connected to the communication channel through RS-232 line drivers and receivers. Either a 25 pin D-Type connector (P) or a 26 pin Flat-Cable connector (F) is used to connect the board to the RS-232 channel.
Fig. B.1: Board Schematics

### WAIT State Generator

<table>
<thead>
<tr>
<th># of WAIT States</th>
<th>Jumper to CLOSE</th>
<th># of WAIT States</th>
<th>Jumper to CLOSE</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>S1</td>
<td>5</td>
<td>S5</td>
</tr>
<tr>
<td>2</td>
<td>S2</td>
<td>6</td>
<td>S6</td>
</tr>
<tr>
<td>3</td>
<td>S3</td>
<td>7</td>
<td>S7</td>
</tr>
<tr>
<td>4</td>
<td>S4</td>
<td>8</td>
<td>S8</td>
</tr>
</tbody>
</table>

Only One Jumper Should Be Closed at a Time

---

<table>
<thead>
<tr>
<th>#</th>
<th>Type</th>
<th>VCC</th>
<th>GND</th>
<th>-12</th>
<th>+12</th>
</tr>
</thead>
<tbody>
<tr>
<td>U1</td>
<td>82510</td>
<td>21</td>
<td>7</td>
<td>1</td>
<td>14</td>
</tr>
<tr>
<td>U2</td>
<td>1488</td>
<td>5, 9</td>
<td>7</td>
<td>1</td>
<td>14</td>
</tr>
<tr>
<td>U3</td>
<td>1489</td>
<td>14</td>
<td>7</td>
<td></td>
<td></td>
</tr>
<tr>
<td>U4</td>
<td>1489</td>
<td>14</td>
<td>7</td>
<td></td>
<td></td>
</tr>
<tr>
<td>U5</td>
<td>74LS164</td>
<td>1, 2, 14</td>
<td>7</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>#</th>
<th>Type</th>
<th>VCC</th>
<th>GND</th>
<th>-12</th>
<th>+12</th>
</tr>
</thead>
<tbody>
<tr>
<td>U6</td>
<td>Jumper</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>U7</td>
<td>74LS00</td>
<td>14</td>
<td>7</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Either P or F should be installed.
Using the 8273 SDLC/HDLC Protocol Controller

JOHN BEASTON
MICROCOMPUTER APPLICATIONS
INTRODUCTION

The Intel 8273 is a Data Communications Protocol Controller designed for use in systems utilizing either SDLC or HDLC (Synchronous or High-Level Data Link Control) protocols. In addition to the usual features such as full duplex operation, automatic Frame Check Sequence generation and checking, automatic zero bit insertion and deletion, and TTL compatibility found on other single component SDLC controllers, the 8273 features a frame level command structure, a digital phase locked loop, SDLC loop operation, and diagnostics.

The frame level command structure is made possible by the 8273's unique internal dual processor architecture. A high-speed bit processor handles the serial data manipulations and character recognition. A byte processor implements the frame level commands. These dual processors allow the 8273 to control the necessary byte-by-byte operation of the data channel with a minimum of external hardware. The digital phase locked loop (DPLL) provides a means of clock recovery from the received data stream on-chip. This feature, along with the frame level commands, makes SDLC loop operation extremely simple and flexible. Diagnostics in the form of both data and clock loopback are available to simplify board debug and link testing. The 8273 is a dedicated function peripheral in the MCS-80/85 Microcomputer family and as such, it interfaces to the 8080/8085 system with a minimum of external hardware.

This application note explains the 8273 as a component and shows its use in a generalized loop configuration and a typical 8085 system. The 8085 system was used to verify the SDLC operation of the 8273 on an actual IBM SDLC data communications link.

The first section of this application note presents an overview of the SDLC/HDLC protocols. It is fairly tutorial in nature and may be skipped by the more knowledgeable reader. The second section describes the 8273 from a functional standpoint with explanation of the block diagram. The software aspects of the 8273, including command examples, are discussed in the third section. The fourth and fifth sections discuss a loop SDLC configuration and the 8085 system respectively.

SDLC/HDLC OVERVIEW

SDLC is a protocol for managing the flow of information on a data communications link. In other words, SDLC can be thought of as an envelope-addressed, stamped, and containing an s.a.s.e.—in which information is transferred from location to location on a data communications link. (Please note that while SDLC is discussed specifically, all comments also apply to HDLC except where noted.) The link may be either point-to-point or multi-point, with the point-to-point configuration being either switched or nonswitched. The information flow may use either full or half duplex exchanges. With this many configurations supported, it is difficult to find a synchronous data communications application where SDLC would not be appropriate.

Aside from supporting a large number of configurations, SDLC offers the potential of a $2 \times$ increase in throughput over the presently most prevalent protocol: Bi-Sync. This performance increase is primarily due to two characteristics of SDLC: full duplex operation and the implied acknowledgement of transferred information. The performance increase due to full duplex operation is fairly obvious since, in SDLC, both stations can communicate simultaneously. Bi-Sync supports only half-duplex (two-way alternate) communication. The increase from implied acknowledgement arises from the fact that a station using SDLC may acknowledge previously received information while transmitting different information. Up to 7 messages may be outstanding before an acknowledgement is required. These messages may be acknowledged as a block rather than singly. In Bi-Sync, acknowledgements are unique messages that may not be included with messages containing information and each information message requires a separate acknowledgement. Thus the line efficiency of SDLC is superior to Bi-Sync. On a higher level, the potential of a $2 \times$ increase in performance means lower cost per unit of information transferred. Notice that the increase is not due to higher data link speeds (SDLC is actually speed independent), but simply through better line utilization.

Getting down to the more salient characteristics of SDLC; the basic unit of information on an SDLC link is that of the frame. The frame format is shown in Figure 1. Five fields comprise each frame: flag, address, control, information, and frame check sequence. The flag fields (F) form the boundary of the frame and all

<table>
<thead>
<tr>
<th>Opening Flag</th>
<th>Address Field (A)</th>
<th>Control Field (C)</th>
<th>Information Field (I)</th>
<th>Frame Check Sequence (FCS)</th>
<th>Closing Flag</th>
</tr>
</thead>
<tbody>
<tr>
<td>011111110</td>
<td>8 Bits</td>
<td>8 Bits</td>
<td>Any Length 0 to N Bits</td>
<td>16 Bits</td>
<td>011111110</td>
</tr>
</tbody>
</table>

Figure 1. SDLC Frame Format
other fields are positionally related to one of the two flags. All frames start with an opening flag and end with a closing flag. Flags are used for frame synchronization. They also may serve as time-fill characters between frames. (There are no intraframe time-fill characters in SDLC as there are in Bi-Sync.) The opening flag serves as a reference point for the address (A) and control (C) fields. The frame check sequence (FCS) is referenced from the closing flag. All flags have the binary configuration 01111110 (7EH).

SDLC is a bit-oriented protocol, that is, the receiving station must be able to recognize a flag (or any other special character) at any time, not just on an 8-bit boundary. This, of course, implies that a frame may be N-bits in length. (The vast majority of applications tend to use frames which are multiples of 8 bits long, however.)

The fact that the flag has a unique binary pattern would seem to limit the contents of the frame since a flag pattern might inadvertently occur within the frame. This would cause the receiver to think the closing flag was received, invalidating the frame. SDLC handles this situation through a technique called zero bit insertion. This technique specifies that within a frame a binary 0 be inserted by the transmitter after any succession of five contiguous binary 1s. Thus, no pattern of 01111110 is ever transmitted by chance. On the receiving end, after the opening flag is detected, the receiver removes any 0 following 5 consecutive 1s. The inserted and deleted 0s are not counted for error determination.

Before discussing the address field, an explanation of the roles of an SDLC station is in order. SDLC specifies two types of stations: primary and secondary. The primary is the control station for the data link and thus has responsibility of the overall network. There is only one predetermined primary station, all other stations on the link assume the secondary station role. In general, a secondary station speaks only when spoken to. In other words, the primary polices the secondaries for responses. In order to specify a specific secondary, each secondary is assigned a unique 8-bit address. It is this address that is used in the frame’s address field.

When the primary transmits a frame to a specific secondary, the address field contains the secondary’s address. When responding, the secondary uses its own address in the address field. The primary is never identified. This ensures that the primary knows which of many secondaries is responding since the primary may have many messages outstanding at various secondary stations. In addition to the specific secondary address, an address common to all secondaries may be used for various purposes. (An all 1s address field is usually used for this “All Parties” address.) Even though the primary may use this common address, the secondaries are expected to respond with their unique address. The address field is always the first 8 bits following the opening flag.

The 8 bits following the address field form the control field. The control field embodies the link-level control of SDLC. A detailed explanation of the commands and responses contained in this field is beyond the scope of this application note. Suffice it to say that it is in the control field that the implied acknowledgement is carried out through the use of frame sequence numbers. None of the currently available SDLC single chip controllers utilize the control field. They simply pass it to the processor for analysis. Readers wishing a more detailed explanation of the control field, or of SDLC in general, should consult the IBM documents referenced on the front page overleaf.

In some types of frames, an information field follows the control field. Frames used strictly for link management may or may not contain one. When an information field is used, it is unrestricted in both content and length. This code transparency is made possible because of the zero bit insertion mentioned earlier and the bit-oriented nature of SDLC. Even main memory core dumps may be transmitted because of this capability. This feature is unique to bit-oriented protocols. Like the control field, the information field is not interpreted by the SDLC device; it is merely transferred to and from memory to be operated on and interpreted by the processor.

The final field is the frame check sequence (FCS). The FCS is the 16 bits immediately preceding the closing flag. This 16-bit field is used for error detection through a Cyclic Redundancy Checkword (CRC). The 16-bit transmitted CRC is the complement of the remainder obtained when the A, C, and I fields are “divided” by a generating polynomial. The receiver accumulates the A, C, and I fields and also the FCS into its internal CRC register. At the closing flag, this register contains one particular number for an error-free reception. If this number is not obtained, the frame was received in error and should be discarded. Discarding the frame causes the station to not update its frame sequence numbering. This results in a retransmission after the station sends an acknowledgement from previous frames. [Unlike all other fields, the FCS is transmitted MSB (Most Significant Bit) first.] The A, C, and I fields are transmitted LSB (Least Significant Bit) first.) The details of how the FCS is generated and checked is beyond the scope of this application note and since all single component SDLC controllers handle this function automatically, it is usually sufficient to know only that an error has or has not occurred. The IBM documents contain more detailed information for those readers desiring it.

The closing flag terminates the frame. When the closing flag is received, the receiver knows that the preceding 16 bits constitute the FCS and that any bits between the control field and the FCS constitute the information field.
SDLC does not support an interframe time-fill character such as the SYN character in Bi-Sync. If an unusual condition occurs while transmitting, such as data is not available in time from memory or CTS (Clear-to-Send) is lost from the modem, the transmitter aborts the frame by sending an Abort character to notify the receiver to invalidate the frame. The Abort character consists of eight contiguous 1s sent without zero bit insertion. Intraframe time-fill consists of either flags, Abort characters, or any combination of the two.

While the Abort character protects the receiver from transmitted errors, errors introduced by the transmission medium are discovered at the receiver through the FCS check and a check for invalid frames. Invalid frames are those which are not bounded by flags or are too short, that is, less than 32 bits between flags. All invalid frames are ignored by the receiver.

Although SDLC is a synchronous protocol, it provides an optional feature that allows its use on basically asynchronous data links—NRZI (Non-Return-to-Zero-Inverted) coding. NRZI coding specifies that the signal condition does not change for transmitting a binary 1, while a binary 0 causes a change of state. Figure 2 illustrates NRZI coding compared to the normal NRZ. NRZI coding guarantees that an active line will have a transition at least every 5-bit times; long strings of zeros cause a transition every bit time, while long strings of 1s are broken up by zero bit insertion. Since asynchronous operation requires that the receiver sampling clock be derived from the received data, NRZI encoding plus zero bit insertion make the design of clock recovery circuitry easier.

Loop operation defines a new special character: the EOP (End-of-Poll) character which consists of a 0 followed by 7 contiguous, non-zero bit inserted, ones. After the loop controller transmits a message, it idles the line (sends all 1s). The final zero of the closing flag plus the first 7 1s of the idle form an EOP character. While
repeating, the secondaries monitor their incoming line for an EOP character. When an EOP is detected, the secondary checks to see if it has a message to transmit. If it does, it changes the seventh 1 to a 0 (the one bit delay allows time for this) and repeats the modified EOP (now alias flag). After this flag is transmitted, the secondary resumes its one bit delay repeater function. Notice that the final zero of the secondary's closing flag plus the repeated 1s from the controller form an EOP for the next down-loop secondary, allowing it to insert a message if it desires.

One might wonder if the secondary missed any messages from the controller while it was inserting its own message. It does not. Loop operation is basically half-duplex. The controller waits until it receives an EOP before it transmits its next message. The controller's reception of the EOP signifies that the original message has propagated around the loop followed by any messages inserted by the secondaries. Notice that secondaries cannot communicate with one another directly, all secondary-to-secondary communication takes place by way of the controller.

Loop protocol does not utilize the normal Abort character. Instead, an abort is accomplished by simply transmitting a flag character. Down loop, the receiver sees the abort as a frame which is either too short (if the abort occurred early in the frame) or one with an FCS error. Either results in a discarded frame. For more details on loop operation, please refer to the IBM documents referenced earlier.

Another protocol very similar to SDLC which the 8273 supports is HDLC (High-Level Data Link Control). There are only three basic differences between the two: HDLC offers extended address and control fields, and the HDLC Abort character is 7 contiguous 1s as opposed to SDLC's 8 contiguous 1s.

Extended addressing, beyond the 256 unique addresses possible with SDLC, is provided by using the address field's least significant bit as the extended address modifier. The receiver examines this bit to determine if the octet should be interpreted as the final address octet. As long as the bit is 0, the octet that contains it is considered an extended address. The first time the bit is a 1, the receiver interprets that octet as the final address octet. Thus the address field may be extended to any number of octets. Extended addressing is illustrated in Figure 4a.

A similar technique is used to extend the control field although the extension is limited to only one extra control octet. Figure 4b illustrates control field extension.

Those readers not yet asleep may have noticed the similarity between the SDLC loop EOP character (a 0 followed by 7 1s) and the HDLC Abort (7 1s). This possible incompatibility is neatly handled by the HDLC protocol not specifying a loop configuration.

This completes our brief discussion of the SDLC/HDLC protocols. Now let us turn to the 8273 in particular and discuss its hardware aspects through an explanation of the block diagram and generalized system schematics.

**Figure 4**

**BASIC 8273 OPERATION**

It will be helpful for the following discussions to have some idea of the basic operation of the 8273. Each operation, whether it is a frame transmission, reception or port read, etc., is comprised of three phases: the Command, Execution, and Result phases. Figure 5 shows the sequence of these phases. As an illustration of this sequence, let us look at the transmit operation.

**Figure 5. 8273 Operational Phases**

When the CPU decides it is time to transmit a frame, the Command phase is entered by the CPU issuing a Transmit Frame command to the 8273. It is not sufficient to just instruct the 8273 to transmit. The frame level command structure sometimes requires more information such as frame length and address and control field content. Once this additional information is sup-
plied, the Command phase is complete and the Execution phase is entered. It is during the Execution phase that the actual operation, in this case a frame transmission, takes place. The 8273 transmits the opening flag, A and C fields, the specified number of I field bytes, inserts the FCS, and closes with the closing flag. Once the closing flag is transmitted, the 8273 leaves the Execution phase and begins the Result phase. During the Result phase the 8273 notifies the CPU of the outcome of the command by supplying interrupt results. In this case, the results would be either that the frame is complete or that some error condition causes the transmission to be aborted. Once the CPU reads all of the results (there is only one for the Transmit Frame command), the Result phase and consequently the operation, is complete. Now that we have a general feeling for the operation of the 8273, let us discuss the 8273 in detail.

HARDWARE ASPECTS OF THE 8273

The 8273 block diagram is shown in Figure 6. It consists of two major interfaces: the CPU module interface and the modem interface. Let's discuss each interface separately.

CPU Interface

The CPU interface consists of four major blocks: Control/Read/Write logic (C/R/W), internal registers, data transfer logic, and data bus buffers.

The CPU module utilizes the C/R/W logic to issue commands to the 8273. Once the 8273 receives a command and executes it, it returns the results (good/bad completion) of the command by way of the C/R/W logic. The C/R/W logic is supported by seven registers which are addressed via the A0, A1, RD, and WR signals, in addition to CS. The A0 and A1 signals are generally derived from the two low order bits of the CPU module address bus while RD and WR are the normal I/O Read and Write signals found on the system control bus. Figure 7 shows the address of each register using the C/R/W logic. The function of each register is defined as follows:

<table>
<thead>
<tr>
<th>Address Inputs</th>
<th>Control Inputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>A1</td>
<td>A0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Figure 7. 8273 Register Selection
**Command**—8273 operations are initiated by writing the appropriate command byte into this register.

**Parameter**—Many commands require more information than found in the command itself. This additional information is provided by way of the parameter register.

**Immediate Result (Result)**—The completion information (results) for commands which execute immediately are provided in this register.

**Transmit Interrupt Result (TxI/R)**—Results of transmit operations are passed to the CPU in this register.

**Receiver Interrupt Result (RxI/R)**—Receive operation results are passed to the CPU via this register.

**Status**—The general status of the 8273 is provided in this register. The Status register supplies the handshaking necessary during various phases of the 8273 operation.

**Test Mode**—This register provides a software reset function for the 8273.

The commands, parameters, and bit definition of these registers are discussed in the following software section. Notice that there are not specific transmit or receive data registers. This feature is explained in the data transfer logic discussion.

The final elements of the C/R/W logic are the interrupt lines (RxINT and TxINT). These lines notify the CPU module that either the transmitter or the receiver requires service; i.e., results should be read from the appropriate interrupt result register or a data transfer is required. The interrupt request remains active until all the associated interrupt results have been read or the data transfer is performed. Though using the interrupt lines relieves the CPU module of the task of polling the 8273 to check if service is needed, the state of each interrupt line is reflected by a bit in the Status register and non-interrupt driven operation is possible by examining the contents of these bits periodically.

The 8273 supports two independent data interfaces through the data transfer logic; receive data and transmit data. These interfaces are programmable for either DMA or non-DMA data transfers. While the choice of the configuration is up to the system designer, it is based on the intended maximum data rate of the communications channel. Figure 8 illustrates the transfer rate of data bytes that are acquired by the 8273 based on link data rate. Full-duplex data rates above 9600 baud usually require DMA. Slower speeds may or may not require DMA depending on the task load and interrupt response time of the processor.

Figure 9 shows the 8273 in a typical DMA environment. Notice that a separate DMA controller, in this case the Intel 8257, is required. The DMA controller supplies the timing and addresses for the data transfers while the 8273 manages the requesting of transfers and the actual counting of the data block lengths. In this case, elements of the data transfer interface are:

**TxDRQ**: Transmit DMA Request—Asserted by the 8273, this line requests a DMA transfer from memory to the 8273 for transmit.

**TxDACK**: Transmit DMA Acknowledge—Returned by the 8257 in response to TxDRQ, this line notifies the 8273 that a request has been granted, and provides access to the transmitter data register.

**RxDRQ**: Receive DMA Request—Asserted by the 8273, it requests a DMA transfer from the 8273 to memory for a receive operation.

**RxDACK**: Receive DMA Acknowledge—Returned by the 8257, it notifies the 8273 that a receive DMA cycle has been granted, and provides access to the receiver data register.

**RD**: Read—Supplied by the 8257 to indicate data is to be read from the 8273 and placed in memory.

**WR**: Write—Supplied by the 8257 to indicate data is to be written to the 8273 from memory.

To request a DMA transfer the 8273 raises the appropriate DMA request line; let us assume it is a transmitter request (TxDRQ). Once the 8257 obtains control of the system bus by way of its HOLD and HLDA (hold acknowledge) lines, it notifies the 8273 that TxDRQ has been granted by returning TxDACK and WR. The TxDACK and WR signals transfer data to the 8273 for a transmit, independent of the 8273 chip select pin (CS). A similar sequence of events occurs for receiver requests. This “hard select” of data into the transmitter or out of the receiver alleviates the need for the normal transmit and receive data registers addressed by a combination of address lines, CS, and WR or RD. Competi-
tive devices that do not have this "hard select" feature require the use of an external multiplexer to supply the correct inputs for register selection during DMA. (Do not forget that the SDLC controller sees both the addresses and control signals supplied by the DMA controller during DMA cycles.) Let us look at typical frame transmit and frame receive sequences to better see how the 8273 truly manages the DMA data transfer.

Before a frame can be transmitted, the DMA controller is supplied, by the CPU, the starting address for the desired information field. The 8273 is then commanded to transmit a frame. (Just how this is done is covered later during our software discussion.) After the command, but before transmission begins, the 8273 needs a little more information (parameters). Four parameters are required for the transmit frame command: the address field byte, the control field byte, and two bytes which are the least significant and most significant bytes of the information field byte length. Once all four parameters are loaded, the 8273 makes RTS (Request-to-Send) active and waits for CTS (Clear-to-Send) to go active. Once CTS is active, the 8273 starts the frame transmission. While the 8273 is transmitting the opening flag, address field, and control field; it starts making transmitter DMA requests. These requests continue at character (byte) boundaries until the pre-loaded number of bytes of information field have been transmitted.

At this point the requests stop, the FCS and closing flag are transmitted, and the TxINT line is raised, signaling the CPU that the frame transmission is complete. Notice that after the initial command and parameter loading, absolutely no CPU intervention was required (since DMA is used for data transfers) until the entire frame was transmitted. Now let's look at a frame reception.

![Figure 8. Byte Transfer Rate vs Baud Rate](image)

![Figure 9. DMA, Interrupt-Driven System](image)
The receiver operation is very similar. Like the initial transmit sequence, the DMA controller is loaded with a starting address for a receiver data buffer and the 8273 is commanded to receive. Unlike the transmitter, there are two different receive commands: General Receive, where only frames having an address field matching one of two preprogrammed 8273 address fields are transferred to memory. Let’s assume for now that we want to general receive. After the receive command, two parameters are required before the receiver becomes active: the least significant and most significant bytes of the receiver buffer length. Once these bytes are loaded, the receiver is active and the CPU may return to other tasks. The next frame appearing at the receiver input is transferred to memory using receiver DMA requests. When the closing flag is received, the 8273 checks the FCS and raises its RxINT line. The CPU can then read the results which indicate if the frame was error-free or not. (If the received frame had been longer than the pre-loaded buffer length, the CPU would have been notified of that occurrence earlier with a receiver error interrupt. The command description section contains a complete list of error conditions.) Like the transmit example, after the initial command, the CPU is free for other tasks until a frame is completely received. These examples have illustrated the 8273’s management of both the receiver and transmitter DMA channels.

It is possible to use the DMA data transfer interface in a non-DMA interrupt-driven environment. In this case, 4 interrupt levels are used: one each for TxINT and RxINT, and one each for TxDRQ and RxDRQ. This configuration is shown in Figure 10. This configuration offers the advantages that no DMA controller is required and data requests are still separated from result (completion) requests. The disadvantages of the configuration are that 4 interrupt levels are required and that the CPU must actually supply the data transfers. This, of course, reduces the maximum data rate compared to the configuration based strictly on DMA. This system could use an Intel 8259 8-level Priority Interrupt Controller to supply a vectored CALL (subroutine) address based on requests on its inputs. The 8273 transmitter and receiver make data requests by raising the respective DRQ line. The CPU is interrupted by the 8259 and vectored to a data transfer routine. This routine either writes (for transmit) or reads (for receive) the 8273 using the respective TxDACK or RxDACK line. The DACK lines serve as “hard” chip selects into and out of the 8273. TxDACK + WR writes data into the 8273 for transmit. RxDACK + RD reads data from the 8273 for receive. The CPU is notified of operation completion and results by way of TxINT and RxINT lines. Using the 8273, and the 8259, in this way, provides a very effective, yet simple, interrupt-driven interface.

Figure 11 illustrates a system very similar to that described above. This system utilizes the 8273 in a non-DMA data transfer mode as opposed to the two DMA approaches shown in Figures 9 and 10. In the non-DMA case, data transfer requests are made on the TxINT and RxINT lines. The DRQ lines are not used. Data transfer requests are separated from result requests by a bit in the Status register. Thus, in response to an interrupt, the CPU reads the Status register and branches to either a result or a data transfer routine based on the status of one bit. As before, data transfers are made via using the DACK lines as chip selects to the transmitter and receiver data registers.
Figure 11. Non-DMA Interrupt-Driven System

Figure 12. Polled System

Figure 12 illustrates the simplest system of all. This system utilizes polling for all data transfers and results. Since the interrupt pins are reflected in bits in the Status register, the software can read the Status register periodically looking for one of these to be set. If it finds an INT bit set, the appropriate Result Available bit is examined to determine if the “interrupt” is a data transfer or completion result. If a data transfer is called for, the DACK line is used to enter or read the data from the 8273. If the interrupt is a completion result, the appropriate result register is read to determine the good/bad completion of the operation.

The actual selection of either DMA or non-DMA modes is controlled by a command issued during initialization. This command is covered in detail during the software discussion.

The final block of the CPU module interface is the Data Bus Buffer. This block supplies the tri-state, bidirectional data bus interface to allow communication to and from the 8273.

Modem Interface

As the name implies, the modem interface is the modem side of the 8273. It consists of two major blocks: the modem control block and the serial data timing block.

The modem control block provides both dedicated and user-defined modem control functions. All signals supported by this interface are active low so that EIA in-
Interfacing drivers (MC1488) and inverting receivers (MC1489) may be used to interface to standard modems.

Port A is a modem control input port. Its representation on the data bus is shown in Figure 13. Bits D₀ and D₁ have dedicated functions. D₀ reflects the logical state of the CTS (Clear-to-Send) pin. If CTS is active (low), D₀ is a 1. This signal is used to condition the start of a transmission. The 8273 waits until CTS is active before it starts transmitting a frame. While transmitting, if CTS goes inactive, the frame is aborted and the CPU is interrupted. When the CPU reads the interrupt result, a CTS failure is indicated.

D₁ reflects the logical state of the CD (Carrier Detect) pin. CD is used to condition the start of a frame reception. CD must be active in time for a frame's address field. If CD is lost (goes inactive) while receiving a frame, an interrupt is generated with a CD failure result. CD may go inactive between frames.

Bits D₂ thru D₄ reflect the logical state of the PA₂ thru PA₄ pins respectively. These inputs are user defined. The 8273 does not interrogate or manipulate these bits. Bits D₅, D₆, and D₇ are not used and each is read as a 1 for a Read Port A command.

Port B is a modem control output port. Its data bus representation is shown in Figure 14. As in Port A, the bit values represent the logical condition of the pins. D₀ and D₁ are dedicated function outputs. D₀ represents the RTS (Request-to-Send) pin. RTS is normally used to notify the modem that the 8273 wishes to transmit. This function is handled automatically by the 8273. If RTS is inactive (pin is high) when the 8273 is commanded to transmit, the 8273 makes it active and then waits for CTS before transmitting the frame. One byte time after the end of the frame, the 8273 returns RTS to its inactive state. However, if RTS was active when a transmit command is issued, the 8273 leaves it active when the frame is complete.

Bit D₁ reflects the state of the Flag Detect pin. This pin is activated whenever an active receiver sees a flag character. This function is useful to activate a timer for line activity timeout purposes.

Bits D₂ thru D₄ provide four user-defined outputs. Pins PB₁ thru PB₄ reflect the logical state of these bits. The 8273 does not interrogate or manipulate these bits. D₅ and D₇ are not used. In addition to being able to output to Port B, Port B may be read using a Read Port B command. All Modem control output pins are forced high on reset. (All commands mentioned in this section are covered in detail later.)

The final block to be covered is the serial data timing block. This block contains two sections: the serial data logic and the digital phase locked loop (DPLL).

Elements of the serial data logic section are the data pins, TxD (transmit data output) and RxD (receive data input), and the respective data clocks, TXC and RXC. The transmit and receive data is synchronized by the TXC and RXC clocks. Figure 15 shows the timing for these signals. The leading edge (negative transition)
of \( TxC \) generates new transmit data and the trailing edge (positive transition) of \( RxC \) is used to capture the receive data.

It is possible to reconfigure this section under program control to perform diagnostic functions; both data and clock loopback are available. In data loopback mode, the TxD pin is internally routed to the RxD pin. This allows simple board checkout since the CPU can send an SDLC message to itself. (Note that transmitted data will still appear on the TxD pin.)

When data loopback is utilized, the receiver may be presented incorrect sample timing (\( RxC \)) by the external circuitry. Clock loopback overcomes this problem by allowing the internal routing of \( TxC \) and \( RxC \). Thus the same clock used to transmit the data is used to receive it. Examination of Figure 15 shows that this method ensures bit synchronism. The final element of the serial data logic is the Digital Phase Locked Loop.

The DPLL provides a means of clock recovery from the received data stream. This feature allows the 8273 to interface without external synchronizing logic to low cost asynchronous modems (modems which do not supply clocks). It also makes the problem of clock timing in loop configurations trivial.

To use the DPLL, a clock at 32 times the required baud rate must be supplied to the \( 32 \times CLK \) pin. This clock provides the interval that the DPLL samples the received data. The DPLL uses the \( 32 \times \) clock and the received data to generate a pulse at the DPLL output pin. This DPLL pulse is positioned at the nominal center of the received data bit cell. Thus the DPLL output may be wired to \( RxC \) and/or \( TxC \) to supply the data timing. The exact position of the pulse is varied depending on the line noise and bit distortion of the received data. The adjustment of the DPLL position is determined according to the rules outlined in Figure 16.

Adjustments to the sample phase of DPLL with respect to the received data is made in discrete increments. Referring to Figure 16, following the occurrence of DPLL...
pulse A, the DPLL counts $32 \times \text{CLK}$ pulses and examines the received data for a data edge. Should no edge be detected in 32 pulses, the DPLL positions the next DPLL pulse (B) at 32 clock pulses from pulse A. Since no new phase information is contained in the data stream, the sample phase is assumed to be at nominal $1 \times$ baud rate. Now assume a data edge occurs after DPLL pulse B. The distance from B to the next pulse C is influenced according to which quadrant (A1, B1, B2, or A2) the data edge falls in. (Each quadrant represents $8 \times 32 \times \text{CLK}$ times.) For example, if the edge is detected in quadrant A1, it is apparent that pulse B was too close to the data edge and the time to the next pulse must be shortened. The adjustment for quadrant A1 is specified as $-2$. Thus, the next DPLL pulse, pulse C, is positioned $32 - 2$ or $30 \times 32 \times \text{CLK}$ pulses following DPLL pulse B. This adjustment moves pulse C closer to the nominal bit center of the next received data cell. A data edge occurring in quadrant B2 would have caused the adjustment to be small, namely $32 + 1$ or $33 \times 32 \times \text{CLK}$ pulses. Using this technique, the DPLL pulse converges to the nominal bit center within 12 data transitions, worse case—4-bit times adjusting through quadrant A1 or A2 and 8-bit times adjusting through B1 or B2.

When the receive data stream goes idle after 15 ones, DPLL pulses are generated at 32 pulse intervals of the $32 \times \text{CLK}$. This feature allows the DPLL pulses to be used as both transmitter and receiver clocks.

In order to guarantee sufficient transitions of the received data to enable the DPLL to lock, NRZI encoding of the data is recommended. This ensures that, within a frame, data transitions occur at least every five bit times—the longest sequence of 1s which may be transmitted with zero bit insertion. It is also recommended that frames following a line idle be transmitted with preframe sync characters which provide a minimum of 12 transitions. This ensures that the DPLL is generating DPLL pulses at the nominal bit centers in time for the opening flag. (Two 00H characters meet this requirement by supplying 16 transitions with NRZI encoding. The 8273 contains a mode which supplies such a preframe sync.)

Figure 17 illustrates 8273 clock configurations using either synchronous or asynchronous modems. Notice how the DPLL output is used for both TxC and RxC in the asynchronous case. This feature eliminates the need for external clock generation logic where low cost asynchronous modems are used and also allows direct connection of 8273s for the ultimate in low cost data links. The configuration for loop applications is discussed in a following section.

This completes our discussion of the hardware aspects of the 8273. Its software aspects are now discussed.

**SOFTWARE ASPECTS OF THE 8273**

The software aspects of the 8273 involve the communication of both commands from the CPU to the 8273 and the return of results of those commands from the 8273 to the CPU. Due to the internal processor architecture of the 8273, this CPU-8273 communication is basically a form of interprocessor communication. Such communication usually requires a form of protocol of its own. This protocol is implemented through use of handshaking supplied in the 8273 Status register. The bit definition of this register is shown in Figure 18.
CBSY: Command Busy—CBSY indicates when the 8273 is in the command phase. CBSY is set when the CPU writes a command into the Command register, starting the Command phase. It is reset when the last parameter is deposited in the Parameter register and accepted by the 8273, completing the Command phase.

CBF: Command Buffer Full—When set, this bit indicates that a byte is present in the Command register. This bit is normally not used.

CPBF: Command Parameter Buffer Full—This bit indicates that the Parameter register contains a parameter. It is set when the CPU deposits a parameter in the Parameter register. It is reset when the 8273 accepts the parameter.

CRBF: Command Result Buffer Full—This bit is set when the 8273 places a result from an immediate type command in the Result register. It is reset when the CPU reads the result from the Result register.

RxINT: Receiver Interrupt—The state of the RxINT pin is reflected by this bit. RxINT is set by the 8273 whenever the receiver needs servicing. RxINT is reset when the CPU reads the results or performs the data transfer.

TxINT: Transmitter Interrupt—This bit is identical to RxINT except action is initiated based on transmitter interrupt sources.

RxIRA: Receiver Interrupt Result Available—RxIRA is set when the 8273 places an interrupt result byte into the RxI/R register. RxIRA is reset when the CPU reads the RxI/R register.

TxIRA: Transmitter Interrupt Result Available—TxIRA is the corresponding Result Available bit for the transmitter. It is set when the 8273 places an interrupt result byte in the TxI/R register and reset when the CPU reads the register.

The significance of each of these bits will be evident shortly. Since the software requirements of each 8273 phase are essentially independent, each phase is covered separately.

**Command Phase Software**

Recalling the Command phase description in an earlier section, the CPU starts the Command phase by writing a command byte into the 8273 Command register. If further information about the command is required by the 8273, the CPU writes this information into the Parameter register. Figure 19 is a flowchart of the Command phase. Notice that the CBSY and CPBF bits of the Status register are used to handshake the command and parameter bytes. Also note that the chart shows that a command may not be issued if the Status register indicates the 8273 is busy (CBSY = 1). If a command is issued while CBSY = 1, the original command is overwritten and lost. (Remember that CBSY signifies the command phase is in progress and not the actual execution of the command.) The flowchart also includes a Parameter buffer full check. The CPU must wait until CPBF = 0 before writing a parameter to the Parameter register. If a parameter is issued while CPBF = 1, the previous parameter is overwritten and lost. An example of command output assembly language software is provided in Figure 20a. This software assumes that a command buffer exists in memory. The buffer is pointed at by the HL register. Figure 20b shows the command buffer structure.

The 8273 is a full duplex device, i.e., both the transmitter and receiver may be executing commands or passing interrupt results at any given time. (Separate Rx and Tx interrupt pins and result registers are provided for this reason.) However, there is only one Command register. Thus, the Command register must be used for only one command sequence at a time and the transmitter and receiver may never be simultaneously in a command phase. A detailed description of the commands and their parameters is presented in a following section.
:FUNCTION: COMMAND DISPATCHER
:INPUTS: HL - COMMAND BUFFER ADDRESS
:OUTPUTS: NONE
:CALLS: NONE
:DESTROYs: A,B,H,L,F/F'S
:DESCRIPTION: CMDOUT ISSUES THE COMMAND + PARAMETERS
;IN THE COMMAND BUFFER POINTED AT BY HL

CMDOUT: LXI H,CMDBUF ;POINT HL AT BUFFER
        MOV B,M ;1ST ENTRY IS PAR. COUNT
        INX H ;POINT AT COMMAND BYTE
CMD1: IN STAT73 ;READ 8273 STATUS
        RLC CMD1 ;WAIT UNTIL CBSY=0
        MOV A,M ;MOVE COMMAND BYTE TO A
        OUT COMM73 ;PUT COMMAND IN COMMAND REG
CMD2: MOV A,B ;GET PARAMETER COUNT
        ANA A ;TEST IF ZERO
        RZ ;IF 0 THEN DONE
        INX H ;NOT DONE, SO POINT AT NEXT PAR
        DCR B ;DEC PARAMETER COUNT
CMD3: IN STAT73 ;READ 8273 STATUS
        ANI CPBF ;TEST CPBF BIT
        JNZ CMD3 ;WAIT UNTIL CPBF IS 0
        MOV A,M ;GET PARAMETER FROM BUFFER
        OUT PARM73 ;OUTPUT PAR TO PARAMETER REG
        JMP CMD2 ;CHECK IF MORE PARAMETERS

Figure 20A. Command Phase Software

| +4 | PARAMETER 3 |
| +3 | PARAMETER 2 |
| +2 | PARAMETER 1 |
| +1 | COMMAND |
| CMDBUF: | PARAMETER COUNT | ← HL |

Figure 20B. Command Buffer Format

Execution Phase Software

During the Execution phase, the operation specified by the Command phase is performed. If the system utilizes DMA for data transfers, there is no CPU involvement during this phase, so no software is required. If non-DMA data transfers are used, either interrupts or polling is used to signal a data transfer request.

For interrupt-driven transfers the 8273 raises the appropriate INT pin. When responding to the interrupt,
the CPU must determine whether it is a data transfer request or an interrupt signaling that an operation is complete and results are available. The CPU determines the cause by reading the Status register and interrogating the associated IRA (Interrupt Result Available) bit (TxIRA for TxINT and RxIRA for RxINT). If the IRA = 0, the interrupt is a data transfer request. If the IRA = 1, an operation is complete and the associated Interrupt Result register must be read to determine the completion status (good/bad/etc.). A software interrupt handler implementing the above sequence is presented as part of the Result phase software.

When polling is used to determine when data transfers are required, the polling routine reads the Status register looking for one of the INT bits to be set. When a set INT bit is found, the corresponding IRA bit is examined. Like in the interrupt-driven case, if the IRA = 0, a data transfer is required. If IRA = 1, an operation is complete and the Interrupt Result register needs to be read. Again, example polling software is presented in the next section.

**Result Phase Software**

During the Result phase the 8273 notifies the CPU of the outcome of a command. The Result phase is initiated by either a successful completion of an operation or an error detected during execution. Some commands such as reading or writing the I/O ports provide immediate results, that is, there is essentially no delay from the issuing of the command and when the result is available. Other commands such as frame transmit, take time to complete so their result is not available immediately. Separate result registers are provided to distinguish these two types of commands and to avoid interrupt handling for simple results.

Immediate results are provided in the Result register. Validity of information in this register is indicated to the CPU by way of the CRBF bit in the Status register. When the CPU completes the Command phase of an immediate command, it polls the Status register waiting until CRBF = 1. When this occurs, the CPU may read the Result register to obtain the immediate result. The Result register provides only the results from immediate commands.

Example software for handling immediate results is shown in Figure 21. The routine returns with the result in the accumulator. The CPU then uses the result as is appropriate.

All non-immediate commands deal with either the transmitter or receiver. Results from these commands are provided in the TxI/R (Transmit Interrupt Result) and RxI/R (Receive Interrupt Result) registers respectively. Results in these registers are conveyed to the CPU by the TxIRA and RxIRA bits of the status register. Results of non-immediate commands consist of one byte result interrupt code indicating the condition for the interrupt and, if required, one or more bytes supplying additional information. The interrupt codes and the meaning of the additional results are covered following the detailed command description.

Non-immediate results are passed to the CPU in response to either interrupts or polling of the Status register. Figure 22 illustrates an interrupt-driven result handler. (Please note that all of the software presented in this application note is not optimized for either speed or code efficiency. They are provided as a guide and to illustrate concepts.) This handler provides for interrupt-driven data transfers as was promised in the last section. Users employing DMA-based transfers do not

```
;FUNCTION: IMDRLT
;INPUTS: NONE
;OUTPUTS: RESULT REGISTER IN A
;CALLS: NONE
;DESTROYS: A, F/F'S
;DESCRIPTION: IMDRLT IS CALLED AFTER A CMDOUT FOR AN IMMEDIATE COMMAND TO READ THE RESULT REGISTER

IMDRLT: IN STAT 73 ;READ 8273 STATUS
        ANI CRBF ;TEST IF RESULT REG READY
        JZ IMDRLT ;WAIT IF CRBF=0
        IN RESL73 ;READ RESULT REGISTER
        RET ;RETURN
```

Figure 21. Immediate Result Handler
need the lines where the IRA bit is tested for zero. (These lines are denoted by an asterisk in the comments column.) Note that the INT bit is used to determine when results have been read. All results must be read. Otherwise, the INT bit (and pin) will remain high and further interrupts may be missed. These routines place the results in a result buffer pointed at by RCRBUF and TXRBUF.

A typical result handler for systems utilizing polling is shown in Figure 23. Data transfers are also handled by this routine. This routine utilizes the routines of Figure 22 to handle the results.

At this point, the reader should have a good conceptual feel about how the 8273 operates. It is now time for the particulars of each command to be discussed.

![Figure 23. Polling Result Handler](image)

### 8273 Command Description

In this section, each command is discussed in detail. In order to shorten the notation, please refer to the command key in Table 1. The 8273 utilizes five different command types: Initialization/Configuration, Receive, Transmit, Reset, and Modem Control.

![Table 1. Command Summary Key](image)
Initialization/Configuration Commands

The Initialization/Configuration commands manipulate registers internal to the 8273 that define the various operating modes. These commands either set or reset specified bits in the registers depending on the type of command. One parameter is required. Set commands perform a logical OR operation of the parameter (mask) and the internal register. This mask contains 1s where register bits are to be set. A "0" in the mask causes no change in the corresponding register bit. Reset commands perform a logical AND operation of the parameter (mask) and the internal register, i.e., the mask is "0" to reset a register bit and a "1" to cause no change. Before presenting the commands, the register bit definitions are discussed.

Operating Mode Register (Figure 24)

D7–D6: Not Used—These bits must not be manipulated by any command; i.e., D7–D6 must be 0 for the Set command and 1 for the Reset command.

D5: HDLC Abort—When this bit is set, the 8273 will interrupt when 7 1s (HDLC Abort) are received by an active receiver. When reset, an SDLC Abort (8 1s) will cause an interrupt.

D4: EOP Interrupt—Reception of an EOP character (0 followed by 7 1s) will cause the 8273 to interrupt the CPU when this bit is set. Loop controller stations use this mode as a signal that a polling frame has completed the loop. No EOP interrupt is generated when this bit is reset.

D3: Early Tx Interrupt—This bit specifies when the transmitter should generate an end of frame interrupt. If this bit is set, an interrupt is generated when the last character data has been passed to the 8273. If the user software issues another transmit command within two byte times, the final flag interrupt does not occur and the new frame is transmitted with only one flag of separation. If this restriction is not met, more than one flag will separate the frames and a frame complete interrupt is generated after the closing flag. If the bit is reset, only the frame complete interrupt occurs. This bit, when set, allows a single flag to separate consecutive frames.

D2: Buffered Address and Control—When set, the address and control fields of received frames are buffered in the 8273 and passed to the CPU as results after a received frame interrupt (they are not transferred to memory with the information field). On transmit, the A and C fields are passed to the 8273 as parameters. This mode simplifies buffer management. When this bit is reset, the A and C fields are passed to and from memory as the first two data transfers.

D1: Preframe Sync—When set, the 8273 prefices each transmitted frame with two characters before the opening flag. These two characters provide 16 transitions to allow synchronization of the opposing receiver. To guarantee 16 transitions, the two characters are 55H–55H for non-NRZI mode (see Serial I/O Register description) or 00H–00H for NRZI mode. When reset, no preframe characters are transmitted.

D0: Flag Stream—When set, the transmitter will start sending flag characters as soon as it is idle; i.e., immediately if idle when the command is issued or after a transmission if the transmitter is active when this bit is set. When reset, the transmitter starts sending Idle characters on the next character boundary if idle already, or at the end of a transmission if active.

Serial I/O Mode Register (Figure 25)

D7–D3: Not Used—These bits must be 0 for the Set command and 1 for the Reset command.

D2: Data Loopback—When set, transmitted data (TxD) is internally routed to the receive data circuitry. When reset, TxD and RxD are independent.

D1: Clock Loopback—When set, TxC is internally routed to RxC. When reset, the clocks are independent.

D0: NRZI (Non-Return to Zero Inverted)—When set, the 8273 assumes the received data is NRZI encoded, and NRZI encodes the transmitted data. When reset, the received and transmitted data are treated as a normal positive logic bit stream.

Data Transfer Mode Register (Figure 26)

D7–D1: Not Used—These bits must be 0 for the Set command and 1 for the Reset command.
**D0:** *Interrupt Data Transfer*—When set, the 8273 will interrupt the CPU when data transfers are required (the corresponding IRA Status register bit will be 0 to signify a data transfer interrupt rather than a Result phase interrupt). When reset, 8273 data transfers are performed through DMA requests on the DRQ pins without interrupting the CPU.

**Receive Commands**

The 8273 supports three receive commands plus a receiver disable function.

**General Receive**

When commanded to General Receive, the 8273 passes all frames either to memory (DMA mode) or to the CPU (non-DMA mode) regardless of the contents of the frame’s address field. This command is used for primary and loop controller stations. Two parameters are required: B0 and B1. These parameters are the LSB and MSB of the receiver buffer size. Giving the 8273 this extra information alleviates the CPU of the burden of checking for buffer overflow. The 8273 will interrupt the CPU if the received frame attempts to overfill the allotted buffer space.

**Selective Receive**

In Selective Receive, two additional parameters besides B0 and B1 are required: A1 and A2. These parameters are two address match bytes. When commanded to Selective Receive, the 8273 passes to memory or the CPU only those frames having an address field matching either A1 or A2. This command is usually used for secondary stations with A1 being the secondary address and A2 is the “All Parties” address. If only one match byte is needed, A1 and A2 should be equal. As in General Receive, the 8273 counts the incoming data bytes and interrupts the CPU if B0, B1 is exceeded.

**Selective Loop Receive**

This command is very similar in operation to Selective Receive except that One Bit Delay mode must be set.
and that the loop is captured by placing transmitter in Flag Stream mode automatically after an EOP character is detected following a selectively received frame. The details of using the 8273 in loop configurations is discussed in a later section so please hold questions until then.

The handling of interrupt results is common among the three commands. When a frame is received without error, i.e., the FCS is correct and CD (Carrier Detect) was active throughout the frame or no attempt was made to overfill the buffer; the 8273 interrupts the CPU following the closing flag to pass the completion results. These results, in order, are the receiver interrupt result code (RIC), and the byte length of the information field of the received frame (R₀, R₁). If Buffered mode is selected, the address and control fields are passed as two additional results. If Buffered mode is not selected, the address and control fields are passed as the first two data transfers and R₀, R₁ reflect the information field length plus two.

**Receive Disable**

The receiver may also be disabled using the Receive Disable command. This command terminates any receive operation immediately. No parameters are required and no results are returned.

The details for the Receive command are shown in Figure 29. The interrupt result code key is shown in Figure 30. Some explanation of these result codes is appropriate.

The interrupt result code is the first byte passed to the CPU in the RxI/R register during the Result phase. Bits D₄-D₀ define the cause of the receiver interrupt. Since each result code has specific implications, they are discussed separately below.

<table>
<thead>
<tr>
<th>Command</th>
<th>Hex Code</th>
<th>Parameters</th>
<th>Results* Rₓ/R</th>
</tr>
</thead>
<tbody>
<tr>
<td>General Receive</td>
<td>C₀</td>
<td>B₀, B₁</td>
<td>RIC, R₀, R₁, A, C</td>
</tr>
<tr>
<td>Selective Receive</td>
<td>C₁</td>
<td>B₀, B₁, A, A₂</td>
<td>RIC, R₀, R₁, A, C</td>
</tr>
<tr>
<td>Selective Loop Receive</td>
<td>C₂</td>
<td>B₀, B₁, A, A₂</td>
<td>RIC, R₀, R₁, A, C</td>
</tr>
<tr>
<td>Disable Receiver</td>
<td>C₅</td>
<td>None</td>
<td>None</td>
</tr>
</tbody>
</table>

*NOTE: A and C are passed as results only in buffered mode.

**Figure 29. Receiver Command Summary**

<table>
<thead>
<tr>
<th>RIC D₇-D₀</th>
<th>Receiver Interrupt Result Code</th>
<th>Rx Status After INT</th>
</tr>
</thead>
<tbody>
<tr>
<td>* 00000</td>
<td>A₁ Match or General Receive</td>
<td>Active</td>
</tr>
<tr>
<td>* 00001</td>
<td>A₂ Match</td>
<td>Active</td>
</tr>
<tr>
<td>000 0011</td>
<td>CRC Error</td>
<td>Active</td>
</tr>
<tr>
<td>000 00100</td>
<td>Abort Detected</td>
<td>Active</td>
</tr>
<tr>
<td>000 00101</td>
<td>Idle Detected</td>
<td>Disabled</td>
</tr>
<tr>
<td>000 00110</td>
<td>EOP Detected</td>
<td>Disabled</td>
</tr>
<tr>
<td>000 00111</td>
<td>Frame &lt; 32 Bits</td>
<td>Active</td>
</tr>
<tr>
<td>000 01000</td>
<td>DMA Overrun</td>
<td>Disabled</td>
</tr>
<tr>
<td>000 01001</td>
<td>Memory Buffer Overflow</td>
<td>Disabled</td>
</tr>
<tr>
<td>000 01010</td>
<td>Carrier Detect Failure</td>
<td>Disabled</td>
</tr>
<tr>
<td>000 01011</td>
<td>Receiver Interrupt Overrun</td>
<td>Disabled</td>
</tr>
</tbody>
</table>

*D₇-D₅* Partial Byte Received

| 111 | All 8 Bits of Last Byte                            |
| 000 | D₀                                                  |
| 100 | D₁-D₀                                              |
| 010 | D₂-D₀                                              |
| 110 | D₃-D₀                                              |
| 001 | D₄-D₀                                              |
| 101 | D₅-D₀                                              |
| 011 | D₆-D₀                                              |

**Figure 30. Receiver Interrupt Result Codes (RIC)**
The first two result codes result from the error-free reception of a frame. If the frame is received correctly after a General Receive command, the first result is returned. If either Selective Receive command was used (normal or loop), a match with A1 generates the first result code and a match with A2 generates the second. In either case, the receiver remains active after the interrupt; however, the internal buffer size counters are not reset. That is, if the receive command indicated 100 bytes were allocated to the receive buffer (B0, B1) and an 80-byte frame was received correctly, the maximum next frame size that could be received without recom­manding the receiver (resetting B0 and B1) is 20 bytes. Thus, it is common practice to recommand the receiver after each frame reception. DMA and/or memory pointers are usually updated at this time. (Note that users who do not wish to take advantage of the 8273's buffer management features may simply use B0, B1 = 0FFH for each receive command. Then frames of 65K bytes may be received without buffer overflow errors.)

The third result code is a CRC error. This indicates that a frame was received in the correct format (flags, etc.); however, the received FCS did not check with the internally generated FCS. The frame should be discarded. The receiver remains active. (Do not forget that even though an error condition has been detected, all frame information up until that error has either been transferred to memory or passed to the CPU. This information should be invalidated. This applies to all receiver error conditions.) Note that the FCS, either transmitted or received, is never available to the CPU.

The Abort Detect result occurs whenever the receiver sees either an SDLC (8 Is) or an HDLC (7 Is), depending on the Operating Mode register. However, the intervening Abort character between a closing flag and an Idle does not generate an interrupt. If an Abort character (seen by an active receiver within a frame) is not preceded by a flag and is followed by an idle, an interrupt will be generated for the Abort, followed by an Idle interrupt one character time later. The Idle Detect result occurs whenever 15 consecutive Is are received. After the Abort Detect interrupt, the receiver remains active. After the Idle Detect interrupt, the receiver is disabled and must be recommanded before further frames may be received.

If the EOP Interrupt bit is set in the Operating Mode register, the EOP Detect result is returned whenever an EOP character is received. The receiver is disabled, so the Idle following the EOP does not generate an Idle Detect interrupt.

The minimum number of bits in a valid frame between the flags is 32. Fewer than 32 bits indicates an error. If Buffered mode is selected, such frames are ignored, i.e., no data transfers or interrupts are generated. In non-Buffered mode, a < 32-bit frame generates an interrupt with the < 32-bit frame result since data transfers may already have disturbed the 8257 or interrupt handler. The receiver remains active.

The DMA Overrun results from the DMA controller being too slow in extracting data from the 8273, i.e., the RxDACK signal is not returned before the next received byte is ready for transfer. The receiver is disabled if this error condition occurs.

The Memory Buffer Overflow result occurs when the number of received bytes exceeds the receiver buffer length supplied by the B0 and B1 parameters in the receive command. The receiver is disabled.

The Carrier Detect Failure result occurs when the CD pin goes high (inactive) during reception of a frame. The CD pin is used to qualify reception and must be active by the time the address field starts to be received. If CD is lost during the frame, a CD Failure interrupt is generated and the receiver is disabled. No interrupt is generated if CD goes inactive between frames.

If a condition occurs requiring an interrupt be generated before the CPU has finished reading the previous interrupt results, the second interrupt is generated after the current Result phase is complete (the RxINT pin and status bit go low then high). However, the interrupt result for this second interrupt will be a Receive Interrupt Overrun. The actual cause of the second interrupt is lost. One case where this may occur is at the end of a received frame where the line goes idle. The 8273 generates a received frame interrupt after the closing flag and then 15-bit times later, generates an Idle Detect interrupt. If the interrupt service routine is slow in reading the first interrupt's results, the internal RxI/R register still contains result information when the Idle Detect interrupt occurs. Rather than wiping out the previous results, the 8273 adds a Receive Interrupt Overrun result as an extra result. If the system's interrupt structure is such that the second interrupt is not acknowledged (interrupts are still disabled from the first interrupt), the Receive Interrupt Overrun result is read as an extra result, after those from the first interrupt. If the second interrupt is serviced, the Receive Interrupt Overrun is returned as a single result. (Note that the INT pins supply the necessary transitions to support a Programmable Interrupt Controller such as the Intel 8259. Each interrupt generates a positive-going edge on the appropriate INT pin and the high level is held until the interrupt is completely serviced.) In general, it is possible to have interrupts occurring at one character time intervals. Thus the interrupt handling software must have at least that much response and service time.

The occurrence of Receive Interrupt Overruns is an indication of marginal software design; the system's interrupt response and servicing time is not sufficient for the
data rates being attempted. It is advisable to configure the interrupt handling software to simply read the interrupt results, place them into a buffer, and clear the interrupt as quickly as possible. The software can then examine the buffer for new results at its leisure, and take appropriate action. This can easily be accomplished by using a result buffer flag that indicates when new results are available. The interrupt handler sets the flag and the main program resets it once the results are retrieved.

Both SDLC and HDLC allow frames which are of arbitrary length (> 32 bits). The 8273 handles this N-bit reception through the high order bits (D7-D2) of the result code. These bits code the number of valid received bits in the last received information field byte. This coding is shown in Figure 30. The high order bits of the received partial byte are indeterminate. [The address, control, and information fields are transmitted least significant bit (A0) first. The FCS is complemented and transmitted most significant bit first.]

### Transmit Commands

The 8273 transmitter is supported by three Transmit commands and three corresponding Abort commands.

#### Transmit Frame

The Transmit Frame command simply transmits a frame. Four parameters are required when Buffered mode is selected and two when it is not. In either case, the first two parameters are the least and the most significant bytes of the desired frame length (L0, L1). In Buffered mode, L0 and L1 equal the length in bytes of the desired information field, while in the non-Buffered mode, L0 and L1 must be specified at the information field length plus two. (L0 and L1 specify the number of data transfers to be performed.) In Buffered mode, the address and control fields are presented to the transmitter as the third and fourth parameters respectively. In non-Buffered mode, the A and C fields must be passed as the first two data transfers.

When the Transmit Frame command is issued, the 8273 makes RTS (Request-to-Send) active (pin low) if it was not already. It then waits until CTS (Clear-to-Send) goes active (pin low) before starting the frame. If the Preframe Sync bit in the Operating Mode register is set, the transmitter prefaces two characters (16 transitions) before the opening flag. If the Flag Stream bit is set in the Operating Mode register, the frame (including Preframe Sync if selected) is started on a flag boundary. Otherwise the frame starts on a character boundary.

At the end of the frame, the transmitter interrupts the CPU (the interrupt results are discussed shortly) and returns to either Idle or Flag Stream, depending on the Flag Stream bit of the Operating Mode register. If RTS was active before the transmit command, the 8273 does not change it. If it was inactive, the 8273 will deactivate it within one character time.

#### Loop Transmit

Loop Transmit is similar to Frame Transmit (the parameter definition is the same). But since it deals with loop configurations, One Bit Delay mode must be selected.

If the transmitter is not in Flag Stream mode when this command is issued, the transmitter waits until after a received EOP character has been converted to a flag (this is done automatically) before transmitting. (The one bit delay is, of course, suspended during transmit.) If the transmitter is already in Flag Stream mode as a result of a selectively received frame during a Selective Loop Receive command, transmission will begin at the next flag boundary for Buffered mode or at the third flag boundary for non-Buffered mode. This discrepancy is to allow time for enough data transfers to occur to fill up the internal transmit buffer. At the end of a Loop Transmit, the One Bit Delay mode is re-entered and the flag stream mode is reset. More detailed loop operation is covered later.

#### Transmit Transparent

The Transmit Transparent command enables the 8273 to transmit a block of raw data. This data is without SDLC protocol, i.e., no zero bit insertion, flags, or FCS. Thus it is possible to construct and transmit a B-Sync message for front-end processor switching or to construct and transmit an SDLC message with incorrect FCS for diagnostic purposes. Only the L0 and L1 parameters are used since there are no fields in this mode. (The 8273 does not support a Receive Transparent command.)

#### Abort Commands

Each of the above transmit commands has an associated Abort command. The Abort Frame Transmit command causes the transmitter to send eight contiguous ones (no zero bit insertion) immediately and then revert to either idle or flag streaming based on the Flag Stream bit. (The 8 1s as an Abort character is compatible with both SDLC and HDLC.)

For Loop Transmit, the Abort Loop Transmit command causes the transmitter to send one flag and then revert to one bit delay. Loop protocol depends upon FCS errors to detect aborted frames.
The Abort Transmit Transparent simply causes the transmitter to revert to either idles or flags as a function of the Flag Stream mode specified.

The Abort commands require no parameters, however, they do generate an interrupt and return a result when complete.

A summary of the Transmit commands is shown in Figure 31. Figure 32 shows the various transmit interrupt result codes. As in the receiver operation, the transmitter generates interrupts based on either good completion of an operation or an error condition to start the Result phase.

The Early Transmit Interrupt result occurs after the last data transfer to the 8273 if the Early Transmit Interrupt bit is set in the Operating Mode register. If the 8273 is commanded to transmit again within two character times, a single flag will separate the frames. (Buffered mode must be used for a single flag to separate the frames. If non-Buffered mode is selected, three flags will separate the frames.) If this time constraint is not met, another interrupt is generated and multiple flags or idles will separate the frames. The second interrupt is the normal Frame Transmit Complete interrupt. The Frame Transmit Complete result occurs at the closing flag to signify a good completion.

The DMA Underrun result is analogous to the DMA Overrun result in the receiver. Since SDLC does not support intraframe time fill, if the DMA controller or CPU does not supply the data in time, the frame must be aborted. The action taken by the transmitter on this error is automatic. It aborts the frame just as if an Abort command had been issued.

Clear-to-Send Error result is generated if CTS goes inactive during a frame transmission. The frame is aborted as above.

The Abort Complete result is self-explanatory. Please note however that no Abort Complete interrupt is generated when an automatic abort occurs. The next command type consists of only one command.

**Reset Command**

The Reset command provides a software reset function for the 8273. It is a special case and does not utilize the normal command interface. The reset facility is provided in the Test Mode register. The 8273 is reset by simply outputting a 01H followed by a 00H to the Test Mode register. Writing the 01 followed by the 00 mimics the action required by the hardware reset. Since the 8273 requires time to process the reset internally, at least 10 cycles of the φCLK clock must occur between the writing of the 01 and the 00. The action taken is the same as if a hardware reset is performed, namely:

1) The modem control outputs are forced high inactive.

### Table 1: Transmitter Command Summary

<table>
<thead>
<tr>
<th>Command</th>
<th>Hex Code</th>
<th>Parameters*</th>
<th>Results Tx/R</th>
</tr>
</thead>
<tbody>
<tr>
<td>Transmit Frame Abort</td>
<td>C8 CC</td>
<td>L0, L1, A, C None</td>
<td>TIC TIC</td>
</tr>
<tr>
<td>Loop Transmit Abort</td>
<td>CA CE</td>
<td>L0, L1, A, C None</td>
<td>TIC TIC</td>
</tr>
<tr>
<td>Transmit Transparent Abort</td>
<td>C0 CD</td>
<td>L0, L1 None</td>
<td>TIC TIC</td>
</tr>
</tbody>
</table>

*NOTE: A and C are passed as parameters in buffered mode only.

**Figure 31. Transmitter Command Summary**

### Table 2: Transmitter Interrupt Result Codes

<table>
<thead>
<tr>
<th>RIC</th>
<th>Transmitter Interrupt Result Code</th>
<th>Tx Status after INT</th>
</tr>
</thead>
<tbody>
<tr>
<td>000 01100</td>
<td>Early Tx Interrupt</td>
<td>Active</td>
</tr>
<tr>
<td>000 01101</td>
<td>Frame Tx Complete</td>
<td>Idle or Flags</td>
</tr>
<tr>
<td>000 01110</td>
<td>DMA Underrun</td>
<td>Abort</td>
</tr>
<tr>
<td>000 01111</td>
<td>Clear to Send Error</td>
<td>Abort</td>
</tr>
<tr>
<td>000 10000</td>
<td>Abort Complete</td>
<td>Idle or Flags</td>
</tr>
</tbody>
</table>

**Figure 32. Transmitter Interrupt Result Codes**
2) The 8273 Status register is cleared.
3) Any commands in progress cease.
4) The 8273 enters an idle state until the next command is issued.

**Modem Control Commands**

The modem control ports were discussed earlier in the Hardware section. The commands used to manipulate these ports are shown in Figure 33. The Read Port A and Read Port B commands are immediate. The bit definition for the returned byte is shown in Figures 13 and 14. Do not forget that the returned value represents the logical condition of the pin, i.e., pin active (low) = bit set.

The Set and Reset Port B commands are similar to the Initialization commands in that they use a mask parameter which defines the bits to be changed. Set Port B utilizes a logical OR mask and Reset Port B uses a logical AND mask. Setting a bit makes the pin active (low). Resetting the bit deactivates the pin (high).

To help clarify the numerous timing relationships that occur and their consequences, Figures 34 and 35 are provided as an illustration of several typical sequences. It is suggested that the reader go over these diagrams and re-read the appropriate part of the previous sections if necessary.

**HDLC CONSIDERATIONS**

The 8273 supports HDLC as well as SDLC. Let's discuss how the 8273 handles the three basic HDLC/SDLC differences: extended addressing, extended control, and the 7 Is Abort character.

Recalling Figure 4a, HDLC supports an address field of indefinite length. The actual amount of extension used is determined by the least significant bit of the characters immediately following the opening flag. If the LSB is 0, more address field bytes follow. If the LSB is 1, this byte is the final address field byte. Software must be used to determine this extension.

<table>
<thead>
<tr>
<th>Port</th>
<th>Command</th>
<th>Hex Code</th>
<th>Parameter</th>
<th>Reg Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>A Input</td>
<td>Read</td>
<td>22</td>
<td>None</td>
<td>Port Value</td>
</tr>
<tr>
<td>B Output</td>
<td>Read</td>
<td>23</td>
<td>None</td>
<td>Port Value</td>
</tr>
<tr>
<td></td>
<td>Set</td>
<td>A3</td>
<td>Set Mask</td>
<td>None</td>
</tr>
<tr>
<td></td>
<td>Reset</td>
<td>63</td>
<td>Reset Mask</td>
<td>None</td>
</tr>
</tbody>
</table>

Figure 33. Modem Control Command Summary

If non-Buffered mode is used, the A, C, and I fields are in memory. The software must examine the initial characters to find the extent of the address field. If Buffered mode is used, the characters corresponding to the SDLC A and C fields are transferred to the CPU as interrupt results. Buffered mode assumes the two characters following the opening flag are to be transferred as interrupt results regardless of content or meaning. (The 8273 does not know whether it is being used in an SDLC or an HDLC environment.) In SDLC, these characters are necessarily the A and C field bytes, however in HDLC, their meaning may change depending on the amount of extension used. The software must recognize this and examine the transferred results as possible address field extensions.

Frames may still be selectively received as is needed for secondary stations. The Selective Receive command is still used. This command qualifies a frame reception on the first byte following the opening flag matching either of the A1 or A2 match byte parameters. While this does not allow qualification over the complete range of HDLC addresses, it does perform a qualification on the first address byte. The remaining address field bytes, if any, are then examined via software to completely qualify the frame.

Once the extent of the address field is found, the following bytes form the control field. The same LSB test used for the address field is applied to these bytes to determine the control field extension, up to two bytes maximum. The remaining frame bytes in memory represent the information field.

The Abort character difference is handled in the Operating Mode register. If the HDLC Abort Enable bit is set, the reception of seven contiguous ones by an active receiver will generate an Abort Detect interrupt rather than eight ones. (Note that both the HDLC Abort Enable bit and the EOP Interrupt bit must not be set simultaneously.)

Now let's move on to the SDLC loop configuration discussion.
LOOP CONFIGURATION

Aside from use in the normal data link applications, the 8273 is extremely attractive in loop configuration due to the special frame-level loop commands and the Digital Phase Locked Loop. Toward this end, this section details the hardware and software considerations when using the 8273 in a loop application.

The loop configuration offers a simple, low-cost solution for systems with multiple stations within a small physical location, i.e., retail stores and banks. There are two primary reasons to consider a loop configuration. The interconnect cost is lower for a loop over a multi-point configuration since only one twisted pair or fiber optic cable is used. (The loop configuration does not support the passing of distinct clock signals from station to station.) In addition, loop stations do not need the intelligence of a multi-point station since the loop protocol is simpler. The most difficult aspects of loop station design are clock recovery and implementation of one bit delay (both are handled neatly by the 8273).

Figure 36 illustrates a typical loop configuration with one controller and two down-loop secondaries. Each station must derive its own data timing from the received data stream. Recalling our earlier discussion of the DPLL, notice that TxC and RxC clocks are provided by the DPLL output. The only clock required in the secondaries is a simple, non-synchronized clock at 32 times the desired baud rate. The controller requires both 32× and 1× clocks. (The 1× is usually implemented by dividing the 32× clock with a 5-bit divider. However, there is no synchronism requirement between these clocks so any convenient implementation may be used.)
A. Error-Free Frame Transmission

B. Diagram Showing Tx Command Queing and Early Tx Interrupt
(Single flag between frames) Buffered Mode is Assumed

C. CTS Failure (or other error) During Transmission

Figure 35. Sample Transmitter Timing Diagrams
A quick review of loop protocol is appropriate. All communication on the loop is controlled by the loop controller. When the controller wishes to allow the secondaries to transmit, it sends a polling frame (the control field contains a poll code) followed by an EOP (End-of-Poll) character. The secondaries use the EOP character to capture the loop and insert a response frame as will be discussed shortly.

The secondaries normally operate in the repeater mode, retransmitting received data with one bit time of delay. All received frames are repeated. The secondary uses the one bit time of delay to capture the loop.

When the loop is idle (no frames), the controller transmits continuous flag characters. This keeps transitions on the loop for the sake of down-loop phase locked loops. When the controller has a non-polling frame to transmit, it simply transmits the frame and continues to send flags. The non-polling frame is then repeated around the loop and the controller receives it to signify a complete traversal of the loop. At the particular secondary addressed by the frame, the data is transferred to memory while being repeated. Other secondaries simply repeat it.

If the controller wants to poll the secondaries, it transmits a polling frame followed by all 1s (no zero bit insertion). The final zero of the closing frame plus the first seven 1s form an EOP. While repeating, the secondaries monitor their incoming line for an EOP. When an EOP is received, the secondary checks if it has any response for the controller. If not, it simply continues repeating. If the secondary has a response, it changes the seventh EOP one into a zero (the one bit time of delay allows time for this) and repeats it, forming a flag for the down-loop stations. After this flag is transmitted, the secondary terminates its repeater function and inserts its response frame (with multiple preceding flags if necessary). After the closing flag of the response, the secondary re-enters its repeater function, repeating the up-loop controller 1s. Notice that the final zero of the response's closing flag plus the repeated 1s from the controller form a new EOP for the next down-loop secondary. This new EOP allows the next secondary to insert a response if it desires. This gives each secondary a chance to respond.

Back at the controller, after the polling frame has been transmitted and the continuous 1s started, the controller waits until it receives an EOP. Receiving an EOP signifies to the controller that the original frame has propagated around the loop followed by any responses inserted by the secondaries. At this point, the controller may either send flags to idle the loop or transmit the next frame. Let's assume that the loop is implemented completely with the 8273s and describe the command flows for a typical controller and secondary.

The loop controller is initialized with commands which specify that the NRZI, Preframe Sync, Flag Stream, and EOP Interrupt modes are set. Thus, the controller encodes and decodes all data using NRZI format. Preframe Sync mode specifies that all transmitted frames be prefaced with 16 line transitions. This ensures that the minimum of 12 transitions needed by the DPLL to lock after an all 1s line has occurred by the time the secondary sees a frame's opening flag. Setting the Flag Stream mode starts the transmitter sending flags which idles the loop. And the EOP Interrupt mode specifies that the controller processor will be interrupted whenever the active receiver sees an EOP, indicating the completion of a poll cycle.

When the controller wishes to transmit a non-polling frame, it simply executes a Frame Transmit command. Since the Flag Stream mode is set, no EOP is formed after the closing flag. When a polling frame is to be transmitted, a General Receive command is executed first. This enables the receiver and allows reception of all incoming frames; namely, the original polling frame plus any response frames inserted by the secondaries. After the General Receive command, the frame is transmitted with a Frame Transmit command. When the frame is complete, a transmitter interrupt is gener-
ated. The loop controller processor uses this interrupt to reset Flag Stream mode. This causes the transmitter to start sending all 1s. An EOP is formed by the last flag and the first 7 1s. This completes the loop controller transmit sequence.

At any time following the start of the polling frame transmission the loop controller receiver will start receiving frames. (The exact time difference depends, of course, on the number of down-loop secondaries due to each inserting one bit time of delay.) The first received frame is simply the original polling frame. However, any additional frames are those inserted by the secondaries. The loop controller processor knows all frames have been received when it sees an EOP Interrupt. This interrupt is generated by the 8273 since the EOP Interrupt mode was set during initialization. At this point, the transmitter may be commanded either to enter Flag Stream mode, idling the loop, or to transmit the next frame. A flowchart of this sequence is shown in Figure 37.

The secondaries are initialized with the NRZI and One Bit Delay modes set. This puts the 8273 into the repeater mode with the transmitter repeating the received data with one bit time of delay. Since a loop station cannot transmit until it sees an EOP character, any transmit command is queued until an EOP is received. Thus whenever the secondary wishes to transmit a response, a Loop Transmit command is issued. The 8273 then waits until it receives an EOP. At this point, the receiver changes the EOP into a flag, repeats it, resets One Bit Delay mode stopping the repeater function, and sets the transmitter into Flag Stream mode. This captures the loop. The transmitter now inserts its message. At the closing flag, Flag Stream mode is reset, and One Bit Delay mode is set, returning the 8273 to repeater function and forming an EOP for the next down-loop station. These actions happen automatically after a Loop Transmit command is issued.

When the secondary wants its receiver enabled, a Selective Loop Receive command is issued. The receiver then looks for a frame having a match in the Address field. Once such a frame is received, repeated, and transferred to memory, the secondary's processor is interrupted with the appropriate Match interrupt result and the 8273 continues with the repeater function until an EOP is received, at which point the loop is captured as above. The processor should use the interrupt to determine if it has a message for the controller. If it does, it simply issues a Loop Transmit command and things progress as above. If the processor has no message, the software must reset the Flag Stream mode bit in the Operating Mode register. This will inhibit the 8273 from capturing the loop at the EOP. (The match frame and the EOP may be separated in time by several frames depending on how many up-loop stations inserted messages of their own.) If the timing is such that the receiver has already captured the loop when the Flag Stream mode bit is reset, the mode is exited on a flag boundary and the frame just appears to have extra closing flags before the EOP. Notice that the 8273 handles the queuing of the transmit commands and the setting and resetting of the mode bits automatically. Figure 38 illustrates the major points of the secondary command sequence.

![Figure 37. Loop Controller Flowchart](image-url)
When an off-line secondary wishes to come on-line, it must do so in a manner which does not disturb data on the loop. Figure 39 shows a typical hardware interface. The line labeled Port could be one of the 8273 Port B outputs and is assumed to be high (I) initially. Thus up-loop data is simply passed down-loop with no delay; however, the receiver may still monitor data on the loop. To come on-line, the secondary is initialized with only the EOP Interrupt mode set. The up-loop data is then monitored until an EOP occurs. At this point, the secondary's CPU is interrupted with an EOP interrupt. This sends the CPU to set One Bit Delay mode in the 8273 and then to set Port low (active). These actions switch the secondary's one bit delay into the loop. Since after the EOP only Is are traversing the loop, no loop disturbance occurs. The secondary now waits for the next EOP, captures the loop, and inserts a "new on-line" message. This signals the controller that a new secondary exists and must be acknowledged. After the secondary receives its acknowledgement, the normal command flow is used.

It is hopefully evident from the above discussion that the 8273 offers a very simple and easy to implement solution for designing loop stations whether they are controllers or down-loop secondaries.

**APPLICATION EXAMPLE**

This section describes the hardware and software of the 8273/8085 system used to verify the 8273 implementation of SDLC on an actual IBM SDLC Link. This IBM link was gratefully volunteered by Raytheon Data Systems in Norwood, Mass. and I wish to thank them for their generous cooperation. The IBM system consisted of a 370 Mainframe, a 3705 Communications Processor, and a 3271 Terminal Controller. A Comlink II Modem supplied the modem interface and all communications took place at 4800 baud. In addition to observing correct responses, a Spectron D601B Datascope was used to verify the data exchanges. A block diagram of the system is shown in Figure 40. The actual verification was accomplished by the 8273 system receiving and responding to polls from the 3705. This method was used on both point-to-point and multi-point configurations. No attempt was made to implement any higher protocol software over that of the poll and poll responses since such software would not affect the verification of the 8273 implementation. As testimony to the ease of use of the 8273, the system worked on the first try.

An SDK-85 (System Design Kit) was used as the core 8085 system. This system provides up to 4K bytes of ROM/EPROM, 512 bytes of RAM, 76 I/O pins, plus
two timers as provided in two 8755 Combination EPROM/I/O devices and two 8155 Combination RAM/I/O/Timer devices. In addition, 5 interrupt inputs are supplied on the 8085. The address, data, and control buses are buffered by the 8212 and 8216 latches and bidirectional bus drivers. Although it was not used in this application, an 8279 Display Driver/Keyboard Encoder is included to interface the on-board display and keyboard. A block diagram of the SDK-85 is shown in Figure 41. The 8273 and associated circuitry was constructed on the ample wire-wrap area provided for the user.

The example 8237/8085 system is interrupt-driven and uses DMA for all data transfers supervised by an 8257 DMA Controller. A 2400 baud asynchronous line, implemented with an 8251A USART, provides communication between the software and the user. 8253 Programmable Interval Timer is used to supply the baud rate clocks for the 8251A and 8273. (The 8273 baud rate clocks were used only during initial system debug. In actual operation, the modem supplied these clocks via the RS-232 interface.) Two 2142 1K x 4 RAMs provided 512 bytes of transmitter and 512 bytes of receiver buffer memory. (Command and result buffers, plus miscellaneous variables are stored in the 8155s.) The RS-232 interface utilized MC1488 and MC1489 RS-232 drivers and receivers. The schematic of the system is shown in Figure 42.

One detail to note is the DMA and interrupt structure of the transmit and receive channels. In both cases, the receiver is always given the higher priority (8257 DMA channel 0 has priority over the remaining channels and the 8085 RST 7.5 interrupt input has priority over the RST 6.5 input.) Although the choice is arbitrary, this technique minimizes the chance that received data could be lost due to other processor or DMA commitments.

Also note that only one 8205 Decoder is used for both peripheral and memory Chip Select. This was done to eliminate separate memory and I/O decoders since it was known beforehand that neither address space would be completely filled.

The 4 MHz crystal and 8224 Clock Generator were used only to verify that the 8273 operates correctly at that maximum spec speed. In a normal system, the 3.072 MHz clock from the 8085 would be sufficient. (This fact was verified during initial checkout.)
Figure 4.1: SDK-85 Functional Block Diagram

- CPU
- ADDRESS DECODER
- ROM (8355)
- EPROM (8755)
- RAM/COUNTER

For Bus Expansion

- Keyboard Display
- Address Field
- Data Field

SDK-85 Keyboard Layout
- RESET
- VECT
- INTR
- C
- D
- E
- F
- SINGLE
- STEP
- GO
- H
- L
- A
- B
- SUBST
- MEM
- EXAM
- REG
- 4
- 5
- 6
- PCH
- 7
- PCL
- NEXT
- EXEC
- 0
- 1
- 2
- 3
- 4
- 5
- 6
- 7
- 8
- 9
- 10
- 11
- 12
- 13
- 14
- 15

Optional: A place has been provided on the PC board for the device but the device is not included.
The software consists of the normal monitor program supplied with the SDK-85 and a program to input commands to the 8273 and to display results. The SDK-85 monitor allows the user to read and write on-board RAM, start execution at any memory location, to single-step through a program, and to examine any of the 8085’s internal registers. The monitor drives either the on-board keyboard/LED display or a serial TTY interface. This monitor was modified slightly in order to use the 8251A with a 2400 baud CRT as opposed to the 110 baud normally used. The 8273 program implements monitor-like user interface. 8273 commands are entered by a two-character code followed by any parameters required by that command. When 8273 interrupts occur, the source of the interrupt is displayed along with any results associated with it. To gain a flavor of how the user/program interface operates, a sample output is shown in Figure 43. The 8273 program prompt character is a “-” and user inputs are underlined.

The “SO 05” implements the Set Operating Mode command with a parameter of 05H. This sets the Buffer and Flag Stream modes. “SS 01” sets the 8273 in NRZI mode using the Set Serial I/O Mode command. The next command specifies General Receiver with a receiver buffer size of 0100H bytes (BO = 00, BI = 01). The “TF” command causes the 8273 to transmit a frame containing an address field of C2H and control field of 11H. The information field is 001122. The “TF” command has a special format. The L0 and L1 parameters are computed from the number of information field bytes entered.

After the TF command is entered, the 8273 transmits the frame (assuming that the modem protocol is observed). After the closing flag, the 8273 interrupts the 8085. The 8085 reads the interrupt results and places them in a buffer. The software examines this buffer for new results and if new results exist, the source of the interrupt is displayed along with the results.

In this example, the 0DH result indicates a Frame Complete interrupt. There is only one result for a transmitter interrupt, the interrupt's trailing zero results were included to simplify programming.

The next event is a frame reception. The interrupt results are displayed in the order read from the 8273. The E0H indicates a General Receive interrupt with the last byte of the information field received on an 8-bit boundary. The 03 00 (R0, R1) results show that there are 3H bytes of information field received. The remaining two results indicate that the received frame had a C2H address field and a 34H control field. The 3 bytes of information field are displayed on the next line.

The result buffer is implemented as a 255-byte circular buffer with two pointers: CNADR and LDADR. CNADR is the console pointer. It points to the next result to be displayed. LDADR is the load pointer. It points to the next empty position in the buffer into which the interrupt handler places the next result. The same buffer is used for both transmitter and receiver results. LOOPIT examines these pointers to detect when CNADR is not equal to LDADR indicating that the buffer contains results which have not been displayed. When this occurs, the program branches to the DISPLAY routine.

DISPLAY determines the source of the undisplayed results by testing the first result. This first result is not necessarily the interrupt result code. If this result is 0CH or greater, the result is from a transmitter interrupt. Otherwise it is from a receiver source. The source of the result code is then displayed on the console along with the next four results from the buffer. If the source was a transmitter interrupt, the routine merely repoints the pointer CNADR and returns to LOOPIT. For a receiver source, the receiver data buffer is displayed in addition to the receiver interrupt results before returning to LOOPIT.
Figure 44. Main Status Poll Loop

Figure 45. DISPLY Subroutine

Figure 46. GETCMD Subroutine

Figure 47. TF Subroutine
If the result buffer pointers indicate an empty buffer, the 8251A is polled for a keyboard character. If the 8251 has a character, GETCMD is called. There the character is read and checked if legal. Illegal characters simply cause a reprompt. Legal characters indicate the start of a command input. Most commands are organized as two characters signifying the command action; i.e., GR—General Receive. The software recognizes the two character command code and takes the appropriate action. For non-Transmit type commands, the hex equivalent of the command is placed in the C register and the number of parameters associated with that command is placed in the B register. The program then branches to the COMM routine.

The COMM routine builds the command buffer by reading the required number of parameters from the keyboard and placing them at the buffer pointed at by CMDBUF. The routine at COMM2 then issues this command buffer to the 8273.

If a Transmit type command is specified, the command buffer is set up similarly to the COMM routine; however, since the information field data is entered from the keyboard, an intermediate routine, TF, is called. TF loads the transmit data buffer pointed at by TxBUF. It counts the number of data bytes entered and loads this number into the command buffer as \( L_0, L_1 \). The command is then issued to the 8273 by jumping to CMDOUT.

One command does not directly result in a command being issued to the 8273. This command, Z, operates a software flip-flop which selects whether the software will respond automatically to received polling frames. If the Poll-Response mode is selected, the prompt character is changed to a '+' character. If a frame is received which contains a prearranged poll control field, the memory location POLIN is made nonzero by the receiver interrupt handler. LOOPIT examines this location and if it is nonzero, causes a branch to the TxPOL routine. The TxPOL routine clears POLIN, sets a pointer to a special command buffer at CMDBUF1, and issues the command by way of the COMM2 entry in the COMM routine. The special command buffer contains the appropriate response frame for the poll frame received. These actions only occur when the Z command has changed the prompt to a '+'. If the prompt is normal '−', polling frames are displayed as normal frames and no response is transmitted. The Poll-Response mode was used during the IBM tests.
The final two software routines are the transmitter and receiver interrupt handlers. The transmit interrupt handler, TxI, simply saves the registers on the stack and checks if loading the result buffer will fill it. If the result buffer will overfill, the program is exited and control is passed to the SDK-85 monitor. If not, the results are read from the TxI/R register and placed in the result buffer at LDADR. The DMA pointers are then reset, the registers restored, and interrupts enabled. Execution then returns to the pre-interrupt location.

The receiver interrupt handler, RxI, is only slightly more complex. As in TxI, the registers are saved and the possibility of overfilling the result buffer is examined. If the result buffer is not full, the results are read from RxI/R and placed in the buffer. At this point the prompt character is examined to see if the Poll-Response mode is selected. If so, the control field is compared with two possible polling control fields. If there is a match, the special command buffer is loaded and the poll indicator, POLIN, is made nonzero. If no match occurred, no action is taken. Finally, the receiver DMA buffer pointers are reset, the processor status restored, and interrupts are enabled. The RET instruction returns execution to the pre-interrupt location.

This completes the discussion of the 8273/8085 system design.

CONCLUSION

This application note has covered the 8273 in some detail. The simple and low cost loop configuration was explored and an 8273/8085 system was presented as a sample design illustrating the DMA/interrupt-driven interface. It is hoped that the major features of the 8273, namely the frame-level command structure and the Digital Phase Locked Loop, have been shown to be a valuable asset in an SDLC system design.
APPENDIX A

ASM80 : F1 RAY73 SRC

ISIS-11 8080/8085 MACRO ASSEMBLER X:000 MODULE PAGE 1

LOC OBJ  SEQ SOURCE STATEMENT

1 INOPRINT MOD85 HCCOND

0000 2 TRUE EQU 00H :00 FOR RAYTHEON

0000 3 TRUE EQU 00H :00 FOR SELF-TEST

0000 4 TRUE EQU 00H :00 FOR NORMAL RESPONSE

0000 5 TRUE EQU 00H :00 FOR LOOP RESPONSE

0000 6 DEM EQU 00H :00 FOR NO DEM

0000 7 . FF FOR DEM

0000 8 . FF FOR DEM

10 . GENERAL 8273 MONITOR WITH RAYTHEON POLL MODE ADDED

11 .

17 .

18 .

19 . COMMAND SUPPORTED ARE RS - RESET SERIAL I/O MODE

20 . SS - SET SERIAL I/O MODE

21 . RO - RESET OPERATING MODE

22 . SO - SET OPERATING MODE

23 . RD - RECEIVER DISABLE

24 . GP - GENERAL RECEIVE

25 . SR - SELECTIVE RECEIVE

26 . TF - TRANSMIT FRAME

27 . AF - ABORT FRAME

28 . SP - SET PORT B

29 . RP - RESET PORT B

30 . RB - RESET ONE BIT DELAY (PAR = 7F)

31 . SB - SET ONE BIT DELAY (PAR = 80)

32 . SL - SELECTIVE LOOP RECEIVE

33 . TL - TRANSMIT LOOP

34 . Z - CHANGE MODES FLIP/FLOP

35 .

36 .

37 .

38 .

39 .

40 .

41 .

42 .

43 .

44 .

45 .

46 . BUFFERED MODE MUST BE SELECTED WHEN SELECTIVE RECEIVE IS USED

47 .

48 . COMMAND FORMAT IS 'COMMAND (2 LTRS): 'PAR A1' 'PAR A2' ETC.

49 .

50 . THE TRANSMIT FRAME COMMAND FORMAT IS 'TF' 'A' 'C' 'BUFFER CONTENTS'.

51 . NO LENGTH COUNT IS NEEDED. BUFFER CONTENTS IS ENDED WITH A CR.

52 .

53 .

54 .

55 . POLLED MODE WHEN POLLED MODE IS SELECTED (DETECTED BY A '+' PROMPT), IF

611001-45

2-329
65 ; 8273 EQUATES
66 ;

0000 67 STAT73 EQU 90H ; STATUS REGISTER
0000 68 COMM73 EQU 90H ; COMMAND REGISTER
0001 69 PARAM73 EQU 91H ; PARAMETER REGISTER
0001 70 RESL73 EQU 91H ; RESULT REGISTER
0002 71 TXIR73 EQU 92H ; TX INTERRUPT RESULT REGISTER
0002 72 RXIR73 EQU 93H ; RX INTERRUPT REGISTER
0002 73 TEST73 EQU 92H ; TEST MODE REGISTER
0002 74 CPBF EQU 20H ; PARAMETER BUFFER FULL BIT
0003 75 TXINT EQU 84H ; TX INTERRUPT BIT IN STATUS REGISTER
0003 76 RXINT EQU 88H ; RX INTERRUPT BIT IN STATUS REGISTER
0003 77 TXIRA EQU 81H ; TX INT RESULT AVAILABLE BIT
0003 78 RXIRA EQU 82H ; RX INT RESULT AVAILABLE BIT
0003 79 ;
0004 80 .8253 EQUATES 81;
0004 82 MODES3 EQU 90H ; .8253 MODE WORD REGISTER
0005 83 CNTS3 EQU 91H ; COUNTER 0 REGISTER
0005 84 CNT153 EQU 92H ; COUNTER 1 REGISTER
0005 85 CNT253 EQU 93H ; COUNTER 2 REGISTER
0006 86 COBR EQU 000CH ; CONSOLE BAUD RATE (2400)
0006 87 MCCTR EQU 36H ; MODE FOR COUNTER 0
0006 88 MCNT2 EQU 000CH ; MODE FOR COUNTER 2
0006 2B817H EQU 8273 BAUD RATE LSB ADDR
0006 98 LKBR2 EQU 2B18H ; .8273 BAUD RATE MSB ADDR
0007 91 ;
0007 92 ; BAUD RATE TABLE: BAUD RATE LKB1 LKB2
0007 93 ; ------------- ----- -----
0007 94 ; 9600 2E 00
0007 95 ; 4800 5C 00
0007 96 ; 2400 89 00
0007 97 ; 1200 12 01
0007 98 ; 600 05 02
0007 99 ; 300 09 05
0008 100 ;
0008 101 ;
0008 102 ; .8257 EQUATES 103;
0008 103 ;
0008 104 MODES7 EQU 00H ; .8257 MODE PORT
0008 105 CHAD0R EQU 00H ; CH0 (RX) ADDR REGISTER
0008 106 CHCTC EQU 00H ; CH0 TERMINAL COUNT REGISTER
0008 107 CHARI0R EQU 00H ; CH0 (TX) ADDR REGISTER
0008 108 CHCTC0 EQU 00H ; CH0 TERMINAL COUNT REGISTER
0008 109 STACK0 EQU 00H ; STATUS REGISTER
0008 110 R0BUF EQU 0200H ; RX BUFFER START ADDRESS
0008 111 T0BUF EQU 0000H ; TX BUFFER START ADDRESS
0008 112 DCOMRA EQU 06H ; DISABLE RX DMA CHANNEL, RX STILL ON
0008 113 RCTC EQU 041FH ; TERMINAL COUNT AND MODE FOR RX CHANNEL
0008 114 EMARA EQU 03H ; ENABLE BOTH TX AND RX CHANNELS-EXT. WR, TX STOP
0008 115 DTOR LA EQU 06H ; DISABLE TX DMA CHANNEL, RX STILL ON
0008 116 TXTC EQU 041FH ; TERMINAL COUNT AND MODE FOR TX CHANNEL
0008 117 ;
; 8251A EQUATES

112 stdout equ 09h ; CONTROL WORD REGISTER
112 stdout equ 09h ; STATUS REGISTER
112 stdout equ 09h ; TX DATA REGISTER
112 stdout equ 09h ; RX DATA REGISTER
112 stdout equ 09h ; MODE 16X2 STOP: NO PARITY
112 stdout equ 27h ; COMMAND, ENABLE TXRX
112 stdout equ 02h ; READY BIT
112 stdout equ 02h ; MONITOR SUBROUTINE EQUATES
112 stdout equ 02h ; STORAGE DEFINITIONS:
112 stdout equ 02h ; LOC DEF
112 stdout equ 02h ; COMMAND BUFFER
112 stdout equ 02h ; RESULT BUFFER LOAD POINTER
112 stdout equ 02h ; RESULT BUFFER CONSOLE POINTER
112 stdout equ 02h ; PROMPT CHARACTER STORAGE
112 stdout equ 02h ; POLL MODE INDICATOR
112 stdout equ 02h ; BAUD RATE LSB FOR SELF-TEST
112 stdout equ 02h ; BAUD RATE LSB FOR SELF-TEST
112 stdout equ 02h ; SPARE
112 stdout equ 02h ; RESPONSE COMMAND BUFFER FOR POLL MODE
112 stdout equ 02h ; RESULT BUFFER
0800 31C020 151 START, LXI SP, STKSRT : INITIALIZE SP
0803 03E0 192 MVI A, MOUNT : 6253 MODE SET
0805 03F0 193 OUT Modes : 6253 MODE PORT
0807 3A1720 194 LDA LKBR1 : GET 8273 BAUD RATE LSB
080A 03C0 195 OUT CNT853 : USING COUNTER 0 AS BAUD RATE GEN
080C 3A1520 196 LDA LKBR2 : GET 8273 BAUD RATE MSB
080F 03C0 197 OUT CNT853 : COUNTER 0
0811 C14880 198 CALL ROMPA : INITIALIZE 8257 RX DMA CHANNEL
0814 035580 199 CALL ROMPA : INITIALIZE 8257 TX DMA CHANNEL
0817 0380 200 MVI A, A : OUTPUT 1 FOLLOWED BY A 0
0819 0392 201 OUT TEST73 : TO TEST MODE REGISTER
081B 03A0 202 MVI A, A : TO RESET THE 8273
081D 03B0 203 OUT TEST73 :
081F 03C0 204 MVI A, A : NORMAL MODE PROMPT CHAR
0821 211520 205 STA PAMPT : PUT IN STORAGE
0824 03D0 206 MVI A, A : TX POLL RESPONSE INDICATOR
0826 321E20 207 STA POLIN : 0 MEANS NO SPECIAL TX
0829 222720 208 STA DEMODE : CLEAR DEMODE
082C 21A3C 212 LXI H, S : SIGNAL MESSAGEADR
082F 03500C 213 CALL TIMMSG : DISPLAY SIGNAL
082F 03500C 214 , MONITOR USES JUMPS IN RAM TO DIRECT INTERRUPTS
0832 21D420 215 LXI H, RST75 : RST75 JUMP LOCATION USED BY MONITOR
0835 01000C 216 LXI B, RXI : ADDRESS OF RX INT ROUTINE
0838 36C0 217 MVI A, B'03H : LOAD '03H OP CODE
083A 22 218 INX H : INC POINTER
083B 71 219 MOV M, C : LOAD RXI LSB
083C 22 220 INX H : INC POINTER
083D 70 221 MOV M, B : LOAD RXI MSB
083E 21E220 222 LXI H, RST65 : RST65 JUMP LOCATION USED BY MONITOR
0841 01300C 223 LXI B, TXI : ADDRESS OF TX INT ROUTINE
0844 36C0 224 MVI A, B'03H : LOAD '03H OP CODE
0846 22 225 INX H : INC POINTER
0847 71 226 MOV M, C : LOAD TXI LSB
0848 22 227 INX H : INC POINTER
0849 70 228 MOV M, B : LOAD TXI MSB
084B E210 231 MVI A, A'18H : CLEAR SET TO RESET INTERRUPTS
084D 38 232 SIM : SEND INTERRUPTS
084F 78 233 EI : CLEAR INTERRUPTS
0851 C1A320 234 LXI H, RESBUF : set RESULT BUFFER POINTERS
0854 221220 235 SHLD CHNR : RESULT CONSOLE POINTER
0857 221220 236 SHLD LSOR : RESULT LOAD POINTER
085A C1A320 237 LXI H, RESBUF : set RESULT BUFFER POINTERS
085D 221220 238 SHLD CHNR : RESULT CONSOLE POINTER
085E C1A320 239 LXI H, RESBUF : set RESULT BUFFER POINTERS
0861 221220 240 SHLD CHNR : RESULT CONSOLE POINTER
0864 221220 241 , MAIN PROGRAM LOOP - CHECKS FOR CHANGE IN RESULT POINTERS, USRT STATUS,
0867 221220 242 , OR Poll STATUS

611001-48
<table>
<thead>
<tr>
<th>Address</th>
<th>Instruction</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0057</td>
<td>CDRECF</td>
<td>CALL CRDF, DISPLAY CR</td>
</tr>
<tr>
<td>005A</td>
<td>381528</td>
<td>CALL PRMP, GET CURRENT</td>
</tr>
<tr>
<td>005C</td>
<td>4F</td>
<td>MOV C.R, MOVE TO C</td>
</tr>
<tr>
<td>005E</td>
<td>CDF85</td>
<td>CALL ECHO, DISPLAY IT</td>
</tr>
<tr>
<td>0061</td>
<td>245208</td>
<td>LOOP IT, HLDB CHDR,</td>
</tr>
<tr>
<td>0064</td>
<td>7D</td>
<td>MOV R.L, SAVE POINTER LS</td>
</tr>
<tr>
<td>0065</td>
<td>264208</td>
<td>MOV L, SAME LSB</td>
</tr>
<tr>
<td>0069</td>
<td>C239A</td>
<td>JNZ DISP, NO RESULTS</td>
</tr>
<tr>
<td>006C</td>
<td>268B08</td>
<td>IN STS:R, YES, CHECK</td>
</tr>
<tr>
<td>006E</td>
<td>592</td>
<td>ANI ROV, CHR RECEIVED?</td>
</tr>
<tr>
<td>0070</td>
<td>270B08</td>
<td>JNZ GETCMD, MUST BE CH</td>
</tr>
<tr>
<td>0073</td>
<td>361620</td>
<td>LDA POLIN, GET POLL</td>
</tr>
<tr>
<td>0076</td>
<td>87</td>
<td>ANA A, IS IT 0?</td>
</tr>
<tr>
<td>0077</td>
<td>2C40B</td>
<td>JNZ TXPOL, NO THEN POL</td>
</tr>
<tr>
<td>007A</td>
<td>366188</td>
<td>JMP LOOPIT, YES TRY</td>
</tr>
<tr>
<td>007D</td>
<td>CD1F96</td>
<td>CALL GETCMD, GET CHAR</td>
</tr>
<tr>
<td>0080</td>
<td>CD5A85</td>
<td>CALL ECH, ECHO IT</td>
</tr>
<tr>
<td>0083</td>
<td>79</td>
<td>MOV A.C, SETUP FOR COMPARE</td>
</tr>
<tr>
<td>0084</td>
<td>F52</td>
<td>CPI 'R', R?</td>
</tr>
<tr>
<td>0086</td>
<td>C0FA85</td>
<td>JZ ROV, GET MORE</td>
</tr>
<tr>
<td>0088</td>
<td>CD07B08</td>
<td>JZ TXN, GET MORE</td>
</tr>
<tr>
<td>008E</td>
<td>FE47</td>
<td>CPI 'O', O?</td>
</tr>
<tr>
<td>0090</td>
<td>CD0F80</td>
<td>JZ TXN, GET MORE</td>
</tr>
<tr>
<td>0093</td>
<td>F54</td>
<td>CPI 'T', T?</td>
</tr>
<tr>
<td>0095</td>
<td>C0EA09</td>
<td>JZ TXN, GET MORE</td>
</tr>
<tr>
<td>009B</td>
<td>FE41</td>
<td>CPI 'R', R?</td>
</tr>
<tr>
<td>009A</td>
<td>CD2609</td>
<td>JZ TXN, GET MORE</td>
</tr>
<tr>
<td>009D</td>
<td>FE99</td>
<td>CPI 'Z', Z?</td>
</tr>
<tr>
<td>009F</td>
<td>CD1089</td>
<td>JZ CHNDE, YES GO CHANGE</td>
</tr>
<tr>
<td>00A2</td>
<td>FE81</td>
<td>CPI CNTLC, CNTL-C</td>
</tr>
<tr>
<td>00A4</td>
<td>CD0000</td>
<td>JZ MONTOR, EXIT TO MON</td>
</tr>
<tr>
<td>00A7</td>
<td>B3F</td>
<td>ILLEG MVI C, &quot;?&quot; PRINT</td>
</tr>
<tr>
<td>00A9</td>
<td>CD0B85</td>
<td>CALL ECHO, DISPLAY IT</td>
</tr>
<tr>
<td>00AC</td>
<td>C357B08</td>
<td>JMP CRDF, LOOP FOR COMMAND</td>
</tr>
<tr>
<td>00AF</td>
<td>CD5F86</td>
<td>CALL GETCMD, GET NEXT CH</td>
</tr>
<tr>
<td>00B2</td>
<td>CD5F85</td>
<td>CALL ECH, ECHO IT</td>
</tr>
<tr>
<td>00B5</td>
<td>79</td>
<td>MOV A.C, SETUP FOR COMPARE</td>
</tr>
<tr>
<td>0086</td>
<td>FE4F</td>
<td>CPI 'O', O?</td>
</tr>
<tr>
<td>0088</td>
<td>C05D89</td>
<td>JZ RCONI, RO COMMAND</td>
</tr>
<tr>
<td>0088</td>
<td>F53</td>
<td>CPI 'S', S?</td>
</tr>
<tr>
<td>008B</td>
<td>CD6789</td>
<td>JZ RCONI, RS COMMAND</td>
</tr>
<tr>
<td>008B</td>
<td>FE44</td>
<td>CPI 'O', O?</td>
</tr>
<tr>
<td>00CB</td>
<td>CD2589</td>
<td>JZ RCONI, RO COMMAND</td>
</tr>
<tr>
<td>00C5</td>
<td>FE58</td>
<td>CPI 'P', P?</td>
</tr>
<tr>
<td>00CD</td>
<td>CD0889</td>
<td>JZ RCONI, RO COMMAND</td>
</tr>
<tr>
<td>00C9</td>
<td>FE52</td>
<td>CPI 'R', R?</td>
</tr>
<tr>
<td>00CC</td>
<td>CD8888</td>
<td>JZ START, START OVER</td>
</tr>
<tr>
<td>00CF</td>
<td>FE42</td>
<td>CPI 'B', B?</td>
</tr>
<tr>
<td>00D1</td>
<td>CD7B89</td>
<td>JZ RCONI, RO COMMAND</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Address</th>
<th>Instruction</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>244;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>245;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>246;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>247;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>248;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>249;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>250;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>251;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>252;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>253;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>254;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>255;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>256;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>257;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>258;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>259;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>260;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>261;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>262;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>263;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>264;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>265;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>266;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>267;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>268;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>269;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>270;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>271;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>272;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>273;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>274;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>275;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>276;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>277;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>278;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>279;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>280;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>281;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>282;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>283;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>284;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>285;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>286;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>287;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>288;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>289;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>290;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>291;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>292;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>293;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>294;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>295;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>296;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>297;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>298;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>299;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>300;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>301;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>302;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>303;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>304;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>305;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>306;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>307;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>308;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>309;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>310;</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

611001-49
373. TRANSMIT ANSWER TO POLL SETUP

374,

094C 3E80 382 TXPOL MVI A.80H ; CLEAR POLL INDICATOR
094E 321620 384 STA POLIN ; INDICATOR ADDR.
0951 21610H 385 LXI H.LOOPIT ; SETUP STACK FOR COMMAND OUTPUT
0954 E3 386 PUSH H ; PUT RETURN TO CMDRECV ON STACK
0955 8684 387 MVI B.84H ; GET # OF PARAMETERS READY
0957 21620H 388 LXI H.CMDHFL ; POINT TO SPECIAL BUFFER
0959 C3880A 389 JMP COMM2 ; JUMP TO COMMAND OUTPUTER

380,

381,

382 ; COMMAND IMPLEMENTING ROUTINES

383,

384,

385 ; RO - RESET OPERATING MODE

386,

0959 8681 396 RCMD MVI B.81H ; # OF PARAMETERS
095F 8581 399 MVI C.81H ; COMMAND
0961 C0680A 400 CALL COMM ; GET PARAMETERS AND ISSUE COMMAND
0964 C57806 401 JMP CMDRECV ; GET NEXT COMMAND

387 ;

388 ; RS - RESET SERIAL I/O MODE COMMAND

389,

0967 8681 409 RCMD MVI B.81H ; # OF PARAMETERS
0969 8581 40A MVI C.81H ; COMMAND
0968 C0680A 40B CALL COMM ; GET PARAMETERS AND ISSUE COMMAND
096E C57806 40C JMP CMDRECV ; GET NEXT COMMAND

389 ;

401 ; RD - RECEIVER DISABLE COMMAND

402,

0971 8688 412 RCMD MVI B.88H ; # OF PARAMETERS
0973 8585 413 MVI C.85H ; COMMAND
0975 C0680A 414 CALL COMM ; ISSUE COMMAND
0978 C57806 415 JMP CMDRECV ; GET NEXT COMMAND

403 ;

404 ; RB - RESET ONE BIT DELAY COMMAND

405,

0978 8681 419 RCMD MVI B.81H ; # OF PARAMETERS
097D 8584 428 MVI C.84H ; COMMAND
097F C0680A 421 CALL COMM ; GET PARAMETER AND ISSUE COMMAND
0982 C57806 422 JMP CMDRECV ; GET NEXT COMMAND

406 ;

407 ; SB - SET ONE BIT DELAY COMMAND

408,

0985 8681 426 SCDMD MVI B.81H ; # OF PARAMETERS
0987 8584 427 MVI C.84H ; COMMAND
0989 C0680A 428 CALL COMM ; GET PARAMETER AND ISSUE COMMAND
098C C57806 429 JMP CMDRECV ; GET NEXT COMMAND

409 ;

410 ; SL - SELECTIVE LOOP RECEIVE COMMAND

411,

098F 8684 432 SCDMD MVI B.84H ; # OF PARAMETERS
0991 8582 434 MVI C.82H ; COMMAND
0993 C0680A 435 CALL COMM ; GET PARAMETERS AND ISSUE COMMAND
0996 C57806 436 JMP CMDRECV ; GET NEXT COMMAND

412 ;

413 ; TL - TRANSMIT LOOP COMMAND
SET COMMAND BUFFER POINTER
LOAD PARAMETER COUNTER
LOAD COMMAND INTO BUFFER
POINT AT ADD AND CTRL POSITIONS
FINISH OFF COMMAND IN TF ROUTINE

SET OPERATING MODE COMMAND

SET SERIAL I/O COMMAND

SELECTIVE RECEIVE COMMAND

GENERAL RECEIVE COMMAND

NO PARAMETERS

SET PORT COMMAND

RESET PORT COMMAND

SP - SET PORT COMMAND

TF - TRANSMIT FRAME COMMAND
<table>
<thead>
<tr>
<th>Line</th>
<th>Instruction</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>0032 0E28</td>
<td>MV1 C,/</td>
<td>SP CHAR</td>
</tr>
<tr>
<td>0054 0F88</td>
<td>CALL ECHO</td>
<td>DISPLAY IT</td>
</tr>
<tr>
<td>0057 2C</td>
<td>INR L</td>
<td>INC BUFFER POINTER</td>
</tr>
<tr>
<td>0059 15</td>
<td>DCR D</td>
<td>DEC RESULT COUNTER</td>
</tr>
<tr>
<td>0059 C2E8</td>
<td>JSNZ DISPM1</td>
<td>NOT DONE</td>
</tr>
<tr>
<td>005C 22130</td>
<td>SHLD CNHOR</td>
<td>UPDATE CONSOLE POINTER</td>
</tr>
<tr>
<td>005F C3786</td>
<td>JMP CNHREC</td>
<td>RETURN TO LOOP</td>
</tr>
<tr>
<td>0062 21088</td>
<td>RXS1: LXI H.RXMSG</td>
<td>RX INT MESSAGEADR</td>
</tr>
<tr>
<td>0063 C928</td>
<td>CALL TM5G</td>
<td>DISPLAY MESSAGE</td>
</tr>
<tr>
<td>0068 E1</td>
<td>POP H</td>
<td>RESTORE CONSOLE POINTER</td>
</tr>
<tr>
<td>0069 7E</td>
<td>RXS1: MOV A.N</td>
<td>RETRIEVE RESULT FROM BUFFER</td>
</tr>
<tr>
<td>006A C076</td>
<td>CALL KMOUT</td>
<td>CONVERT AND DISPLAY IT</td>
</tr>
<tr>
<td>006D 0E28</td>
<td>MV1 C,/</td>
<td>ASCII SP</td>
</tr>
<tr>
<td>006F C0F8</td>
<td>CALL ECHO</td>
<td>DISPLAY IT</td>
</tr>
<tr>
<td>0072 2C</td>
<td>INR L</td>
<td>INC CONSOLE POINTER</td>
</tr>
<tr>
<td>0073 15</td>
<td>DCR D</td>
<td>DEC RESULT COUNTER</td>
</tr>
<tr>
<td>0074 7A</td>
<td>MOV A.D</td>
<td>GET SET TO TEST COUNTER</td>
</tr>
<tr>
<td>0075 FE04</td>
<td>CPI 0AH</td>
<td>IS THE RESULT RB?</td>
</tr>
<tr>
<td>0077 C920A</td>
<td>JZ RAFT</td>
<td>YES, GO SAVE IT</td>
</tr>
<tr>
<td>007A FE03</td>
<td>CPI 03H</td>
<td>IS THE RESULT RI?</td>
</tr>
<tr>
<td>007C C920A</td>
<td>JZ RFT</td>
<td>YES, GO SAVE IT</td>
</tr>
<tr>
<td>007F 67</td>
<td>RXS2: ANR A</td>
<td>TEST RESULT COUNTER</td>
</tr>
<tr>
<td>0080 C928</td>
<td>JNZ RXS1: MOV A.N</td>
<td>NOT DONE YET. GET NEXT RESULT</td>
</tr>
<tr>
<td>0082 22130</td>
<td>SHLD CNHOR</td>
<td>DONE, SO UPDATE CONSOLE POINTER</td>
</tr>
<tr>
<td>0086 C0E8</td>
<td>CALL CRLF</td>
<td>.DISPLAY CR</td>
</tr>
<tr>
<td>0089 21086</td>
<td>LXI H.RXBUF</td>
<td>POINT AT RX BUFFER</td>
</tr>
<tr>
<td>008C C1</td>
<td>POP B</td>
<td>RETRIEVE RECEIVED COUNT</td>
</tr>
<tr>
<td>008D 70</td>
<td>RXS3: MOV A.B</td>
<td>IS COUNT #?</td>
</tr>
<tr>
<td>008E 61</td>
<td>ORR C</td>
<td></td>
</tr>
<tr>
<td>008F C978</td>
<td>JZ CNHREC: MOV A.R</td>
<td>YES, GO BACK TO LOOP</td>
</tr>
<tr>
<td>0092 7E</td>
<td>MOV A.R</td>
<td>NO, GET CHAR</td>
</tr>
<tr>
<td>0093 C5</td>
<td>PUSH B</td>
<td>SAVE BC</td>
</tr>
<tr>
<td>0094 C076</td>
<td>CALL KMOUT</td>
<td>CONVERT AND DISPLAY CHAR</td>
</tr>
<tr>
<td>0097 0E28</td>
<td>MV1 C,/</td>
<td>ASCII SP</td>
</tr>
<tr>
<td>0099 C0F8</td>
<td>CALL ECHO</td>
<td>DISPLAY IT TO SEPARATE DATA</td>
</tr>
<tr>
<td>009C C1</td>
<td>POP B</td>
<td>RESTORE BC</td>
</tr>
<tr>
<td>009D 8B</td>
<td>DCX B</td>
<td>DEC COUNT</td>
</tr>
<tr>
<td>009E 23</td>
<td>INX H</td>
<td>INC POINTER</td>
</tr>
<tr>
<td>009F C300A</td>
<td>JMP RXS3</td>
<td>GET NEXT CHAR</td>
</tr>
<tr>
<td>00A2 4E</td>
<td>RAFT MOV C.M</td>
<td>GET R0 FOR RESULT BUFFER</td>
</tr>
<tr>
<td>00A3 C5</td>
<td>PUSH B</td>
<td>SAVE IT</td>
</tr>
<tr>
<td>00A4 C3F8A</td>
<td>JMP RXS2</td>
<td>RETURN</td>
</tr>
<tr>
<td>00A7 C1</td>
<td>RAFT MOV B</td>
<td>GET R0</td>
</tr>
<tr>
<td>00AB 46</td>
<td>MOV B.M</td>
<td>GET R1 FOR RESULT BUFFER</td>
</tr>
<tr>
<td>00A9 C5</td>
<td>PUSH B</td>
<td>SAVE IT</td>
</tr>
<tr>
<td>00AA C3F8A</td>
<td>JMP RXS2</td>
<td></td>
</tr>
<tr>
<td>00AB ;</td>
<td>611 ;</td>
<td>PARAMETER INPUT - PARAMETER RETURNED IN E REGISTER</td>
</tr>
<tr>
<td>611001-54</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
613;
614 PARI: PUSH B ; SAVE BC
615 MOV AX, DX ; SET CHR COUNTER
616 CALL GETCH ; GET CHR
617 CALL ECHO ; ECHO IT
618 MOV A, C ; PUT CHR IN A
619 CPI ; SP?
620 CALL ECHO ; ECHO IT
621 CALL VALDS ; IS IT A VALID CHR?
622 CALL PARIN ; NO. ILLEGAL TRY AGAIN
623 CALL GETCH ; GET CHR OF PARAMETER
624 CALL ECHO ; ECHO IT
625 CALL DATA ; IS IT A VALID CHR?
626 MOV A, C ; TRY AGAIN
627 MOV A, E ; IS IT A?
628 JZ FINISH ; YES DO WITH THIS PARAMETER
629 JZ FINISH ; YES, DONE WITH THIS PARAMETER
630 DCR D ; DEC CHR COUNTER
631 JXZ RPR ; CLEAR CARRY
632 MOV A, C ; RECOVER 1ST CHR
633 RAL ; ROTATE LEFT 4 PLACES
634 RAL ;
635 RAL ;
636 RAL ;
637 MOV E, A ; SAVE IT IN E
638 JMP PARIN ; GET NEXT CHR
639 MOV A, C ; 2ND CHR IN A
640 ORA E ; COMBINE BOTH CHARS
641 POP B ; RESTORE BC
642 RET ; RETURN TO CALLING PROGRAM
643 MOV A, C ; PUT ILLEGAL CHR IN A
644 STC ; SET CARRY AS ILLEGAL STATUS
645 POP B ; RESTORE BC
646 RET ; RETURN TO CALLING PROGRAM
647,
648,
649 JUMP HERE IF BUFFER FULL
650,
651 BUFFUL DB 10FH ; EXIT TO MONITOR
652,
653 ;
654 :COMMAND DISPATCHER
655 ;
656 ;
657 COMM LXI M.COMBUF ; SET POINTER
658 PUSH B ; SAVE BC
659 MOV M, C ; LOAD COMMAND INTO BUFFER
660 COMM LXI M.COMM ; CHECK PARAMETER COUNTER
661 MOV A, B ; NO PARAMETER
662 CALL PARIN ; SET PARAMETER
663 CALL DATA ; GET PARAMETER
664 JXZ FINISH ; YES GO ISSUE COMMAND
665 INX H ; INC BUFFER POINTER
666 DCR B ; DEC PARAMETER COUNTER
667 MOV M, A ; PARAMETER TO BUFFER
668 JMP COMM ; SET NEXT PARAMETER
669 CIDOUT: LXI M.COMBUF ; REPOINT POINTER
670 POP B ; RESTORE PARAMETER COUNT

61001-55

2-339
<table>
<thead>
<tr>
<th>Address</th>
<th>Instruction</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>671-672</td>
<td>MOV  A, M</td>
<td>Move Accumulator to Memory</td>
</tr>
<tr>
<td>673</td>
<td>MOV  M, A</td>
<td>Move Memory to Accumulator</td>
</tr>
<tr>
<td>674-675</td>
<td>OUT M.3</td>
<td>Output Memory into Carry</td>
</tr>
<tr>
<td>676-677</td>
<td>MOV  A, B</td>
<td>Move Accumulator to Second Accumulator</td>
</tr>
<tr>
<td>678</td>
<td>ADD  M</td>
<td>Add Memory to Memory</td>
</tr>
<tr>
<td>679-680</td>
<td>OUT M.3</td>
<td>Output Memory into Carry</td>
</tr>
</tbody>
</table>

- **INITIALIZE AND ENABLING RX DMA CHANNEL**
  - 689-690: Disable RX DMA Channel
  - 691-692: Disable RX DMA Channel
  - 693-694: Output Mode 57
  - 695-696: RX Buffer Start Address
  - 697-698: RX Buffer LSB
  - 699-700: RX Buffer MSB
  - 701-702: RX Terminal Count LSB
  - 703-704: RX Terminal Count MSB
  - 705-706: RX Terminal Count Port
  - 707-708: RX Terminal Count Port
  - 709-710: Initialize and Enable RX DMA Channel
  - 711-712: Initialize and Enable RX DMA Channel

- **INITIALIZE AND ENABLING TX DMA CHANNEL**
  - 713-714: Disable TX DMA Channel
  - 715-716: TX Buffer Start Address
  - 717-718: TX Buffer LSB
  - 719-720: TX Buffer MSB
  - 721-722: TX Terminal Count LSB
  - 723-724: TX Terminal Count MSB
  - 725-726: TX Terminal Count Port
  - 727-728: Initialize and Enable TX DMA Channel

---

**Note:** The above instructions are for demonstration purposes and may not accurately represent the original content of the document.
INTEGRANT PROCESSING SECTION

OC00  ORG 0C00H

OC00  E5
OC01  F5
OC02  C5
OC03  D5
OC04  3E2
OC05  E98
OC06  3E18
OC07  30
OC08  184
OC09  291020
OC10  E3
OC11  E5
OC12  45
OC13  3A120
OC14  04
OC15  7B
OC16  8D
OC17  CA640A
OC18  15
OC19  2168C
OC20  685
OC21  E09
OC22  E98
OC23  0R38C
OC24  9A9
OC25  0R389
OC26  0R9A9
OC27  0R38C
OC28  862
OC29  3220C
OC30  862
OC31  892
OC32  77
OC33  2C
OC34  15
OC35  C329C
OC36  79
OC37  87
OC38  C450C
OC39  3608
OC40  2C
OC41  15
OC42  C339C
OC43  28182B
OC44  38182B
OC45  FE2D
OC46  CA50C
OC47  E5
OC48  7E

729 : RECEIVED INTERRUPT – RST 7 5 (LOC 3Ch)

730 731

732

733

734

735

736  RX1  PUSH H  SAVE HL
737  PUSH PSW  SAVE PSW
738  PUSH B  SAVE BC
739  PUSH D  SAVE DE
740  3E2  MVI A.DRAM  DISABLE RX DMA
741  E98  OUT MODE7  6257 NODE PORT
742  3E18  MVI A.18H  RESET RST 5 F/F
743  30  SIN
744  184  MVI D.8AH  D IS RESULT COUNTER
745  291020  LHLD LAADR  GET LOAD POINTER
746  E3  PUSH H  SAVE HL
747  E5  PUSH H  SAVE IT AGAIN
748  45  MOV B.L  SAVE LSB
749  3A120  LHLD LAADR  GET CONSOLE POINTER
750  04  RX1  INR B  BUMP LOAD POINTER LSB
751  7B  MOV A.B  GET SET TO TEST
752  8D  CMP L  LOAD=CONSOLE?
753  CA640A  JZ  RX1  YES, BUFFER FULL
754  15  DCR D  DEC COUNTER
755  2168C  JNZ RX1  NOT DONE, TRY AGAIN
756  685  MVI D.0FH  RESET COUNTER
757  E09  POP H  RESTORE LOAD POINTER
758  E98  RX1  IN  STAT73  READ STATUS
759  E98  ANI RXINT  TEST RX INT BIT
760  0R38C  JZ RX1  DONE, GO FINISH UP
761  0R9A9  IN STAT73  READ STATUS AGAIN
762  0R62  ANI RXINT  IS RESULT READY?
763  3220C  JZ RX1  NO, TEST AGAIN
764  862  RX1  IN RXINT  YES, READ RESULT
765  77  MOV M.A  STORE IN BUFFER
766  2C  INR L  INC BUFFER POINTER
767  15  DCR D  DEC COUNTER
768  C329C  JMP RX1  GET MORE RESULTS
769  9A9  MOV A.D  GET SET TO TEST
770  87  MVI A.7FH  ALL RESULTS?
771  C450C  RX1  IN  RXINT  YES, 50 FINISH UP
772  3608  MVI M.8AH  AND LOAD @ TIL DONE
773  2C  INR L  BUMP POINTER
774  15  DCR D  DEC COUNTER
775  C339C  JMP RX1  GO AGAIN
776  28182B  RX1  8ALD LDAR  UPDATE LOAD POINTER
777  38182B  LHVA PAPSH  GET MODE INDICATOR
778  FE2D  CPI ~  NORMAL MODE?
779  CA50C  RX1  JZ RX16  YES, CLEAN UP BEFORE RETURN
780  E5  POP H  GET PREVIOUS LOAD ADDR POINTER
781  7E  MOV A.M  GET I/O BYTE FROM BUFFER

782  15  DCR D  DEC COUNTER
783  87  MVI A.7FH  ALL RESULTS?
784  C450C  RX1  IN  RXINT  YES, 50 FINISH UP
785  C329C  JMP RX1  GET MORE RESULTS
786  9A9  MOV A.D  GET SET TO TEST
787  87  MVI A.7FH  ALL RESULTS?
788  C450C  RX1  IN  RXINT  YES, 50 FINISH UP
789  E5  POP H  GET PREVIOUS LOAD ADDR POINTER
790  7E  MOV A.M  GET I/O BYTE FROM BUFFER

791  15  DCR D  DEC COUNTER
792  87  MVI A.7FH  ALL RESULTS?
793  C450C  RX1  IN  RXINT  YES, 50 FINISH UP
794  E5  POP H  GET PREVIOUS LOAD ADDR POINTER
795  7E  MOV A.M  GET I/O BYTE FROM BUFFER

796  15  DCR D  DEC COUNTER
797  87  MVI A.7FH  ALL RESULTS?
798  C450C  RX1  IN  RXINT  YES, 50 FINISH UP
799  E5  POP H  GET PREVIOUS LOAD ADDR POINTER
800  7E  MOV A.M  GET I/O BYTE FROM BUFFER

611001-57
8C32 ES1E 787 ANI 1EH ;LOOK AT GOOD FRAME BITS
8C34 C2898C 788 NJA RX1S ; IF NOT 0, INTERRUPT WASN'T FROM A GOOD FRAME
8C37 2C 789 INR L ; BYPASS R9 AND R1 IN BUFFER
8C38 2C 790 INR L
8C39 2C 791 INR L
8C3A 55 792 MOV D,M ; GETADR BYTE AND SAVE IT IN D
8C3B 2C 793 INR L
8C3C 7E 794 MOV A,M ; GET CNL BYTE FROM BUFFER
8C3D FE93 795 CPI SNRP ; WAS IT SNRP-P?
8C3F C66C0C 796 JZ TL ; YES, GO SET RESPONSE
8C40 FE11 797 CPI RARP ; WAS IT RR()-P?
8C41 C2898C 798 RZ RX15 ; YES, GO SET RESPONSE, OTHERWISE RETURN
8C42 E111 799 MVI E,RR8F ; RR()-P SO SET RESPONSE TO RR()-F
8C43 C3698C 800 JMP TARE ; GO FINISH LOADING SPECIAL BUFFER
8C44 1873 801 TL, MVI E,RR8F ; SNRP-P SO SET RESPONSE TO RR()-F
8C45 212828 802 TARE ;LXI M,CMDF1 ; SPECIAL BUFFER ADDR
8C46 36C8 806 MVI M,BCH ; LOAD TX FRAME COMMAND
8C47 23 808 INX H ; INC POINTER
8C48 3688 809 MVI M,00H ; LB=0
8C49 23 810 INX H ; INC POINTER
8C4A 3688 811 MVI M,00H ; LB=0
8C4B 23 812 INX H ; INC POINTER
8C4C 72 813 MOV M,D ; LOAD ROVD ADDR BYTE
8C4D 23 814 INX H ; INC POINTER
8C4E 73 815 MOV M,E ; LOAD RESPONSE CNL BYTE
8C4F 36B1 816 MVI A,01H ; SET POLL INDICATOR NOT 0
8C50 321620 817 STA POLIN ; LOAD POLL INDICATOR
8C51 C3698C 818 JMP RX15 ; RETURN
8C52 E1 819 820 RX16 ; POP H ; CLEAN UP STACK IF NORMAL MODE
8C53 C6698C 821 JMP RX15 ; RETURN
8C54 22
8C55 0D A28B 822 RX15 ; CALL RXMHA ; RESET DMA CHANNEL
8C56 01 823 POP D ; RESTORE REGISTERS
8C57 08 824 POP B
8C58 F1 825 POP PSW
8C59 E1 826 POP H
8C5A FB 828 EI ; ENABLE INTERRUPTS
8C5B 09 829 RET ; RETURN
8C5C 39 830
8C5D 32 ; MESSAGE TYPE - ASSUMES MESSAGE STARTS AT HL
8C5E 33 ;
8C5F 34 ;
8C60 C5 832 TMSGL ; PUSH B ; SAVE BC
8C61 7E 833 TMSGL2; MOV A,M ; GET ASCII CHAR
8C62 23 834 INX H ; INC POINTER
8C63 FEFF 835 CPI 0FFH ; STOP?
8C64 C6698C 836 JZ TMSGL1 ; YES, GET SET FOR EXIT
8C65 6F 837 MOV C,A ; SET UP FOR DISPLAY
8C66 C80805 840 CALL ECHO ; DISPLAY CHAR
8C67 C3698C 841 JMP TMSGL2 ; GET NEXT CHAR
8C68 61 843 TMSGL2 ; POP B ; RESTORE BC
8C69 C9 844 RET ; RETURN
8C6A 45 ;
8C6B 46 ;
8C6C 47 ; SIGHON MESSAGE
8C6D 48 ;
OCR3 80 049 SIGNON DB CR.'8273 MONITOR VS L'.CR.0FFH
OCR4 382273
OCR5 822355
OCR6 49545F32
OCR7 20220561
OCR8 2E31
OCR9 00
OCR8 FF
OCR8 ; 050 ;
OCR8 ; 051 ;
OCR8 ; 052 ;
OCR8 ; 053 ; RECEIVER INTERRUPT MESSAGES
OCR8 ; 054 ;
OCR8 ; 055 ;
OCR10 00 056 RXMSG DB CR.‘RX INT - .0FFH
OCR11 52562049
OCR12 4E542020
OCR1 28
OCR12 FF
OCR12 ; 057 ;
OCR12 ; 058 ; TRANSMITTER INTERRUPT MESSAGES
OCR12 ; 059 ;
OCR13 00 060 TXMSG DB CR.‘TX INT - .0FFH
OCR14 54562049
OCR15 4E542020
OCR16 20
OCR16 FF
OCR16 ; 061 ;
OCR16 ; 062 ;
OCR16 ; 063 ; TRANSMITTER INTERRUPT ROUTINE
OCR16 ; 064 ;
OCR16 065 TXI. PUSH H ; SAVE HL
OCR16 066 PUSH PSW ; SAVE PSW
OCR16 067 PUSH B ; SAVE BC
OCR16 068 PUSH D ; SAVE DE
OCR16 069 MVI A.DTMRA ; DISABLE TX DMA
OCR16 070 OUT MODE57 ; 8257 MODE PORT
OCR16 071 MVI D.04H ; SET COUNTER
OCR16 072 LHLD LOADR ; GET LOAD POINTER
OCR16 073 PUSH H ; SAVE IT
OCR16 074 MOV B.L ; SAVE LSB IN B
OCR16 075 LHLD CHADDR ; GET CONSOLE POINTER
OCR16 076 TXII. INC R B ; INC POINTER
OCR16 077 MOV A.B ; GET SET TO TEST
OCR16 078 CMP L ; LOAD-CONSIME?
OCR16 079 JZ BUFFUL ; YES. BUFFER FULL
OCR16 080 DCR D ; IND. TEST NEXT LOCATION
OCR16 081 JNZ TXI1 ; TRY AGAIN
OCR16 082 POP H ; RESTORE LOAD POINTER
OCR16 083 IN TXI173 ; READ RESULT
OCR16 084 MOV M.A ; STORE IN BUFFER
OCR16 085 INC L ; INC POINTER
OCR16 086 MVI M.00H ; EXTRA RESULT SPOTS 0
OCR16 087 INC L
OCR16 088 MVI M.00H
OCR16 089 INC L
OCR16 08A MVI M.00H
OCR16 08B INC L
OCR16 08C MVI M.00H
OCR16 08D INC L
OCR16 08E MVI M.00H
OCR16 08F INC L
OCR16 090 MVI M.00H
OCR16 091 INC L

611001-59

2-343
Asynchronous Communication with the 8274 Multiple-Protocol Serial Controller
INTRODUCTION

The 8274 Multiprotocol serial controller (MPSC) is a sophisticated dual-channel communications controller that interfaces microprocessor systems to high-speed serial data links (at speeds to 880K bits per second) using synchronous or asynchronous protocols. The 8274 interfaces easily to most common microprocessors (e.g., 8048, 8051, 8085, 8086, and 8088), to DMA controllers such as the 8237 and 8257, and to the 8089 I/O processor. Both MPSC communication channels are completely independent and can operate in a full-duplex communication mode (simultaneous data transmission and reception).

Communication Functions

The 8274 performs many communications-oriented functions, including:

- Converting data bytes from a microprocessor system into a serial bit stream for transmission over the data link to a receiving system.
- Receiving serial bit streams and reconverting the data into parallel data bytes that can easily be processed by the microprocessor system.
- Performing error checking during data transfers. Error checking functions include computing/transmitting error codes (such as parity bits or CRC bytes) and using these codes to check the validity of received data.
- Operating independently of the system processor in a manner designed to reduce the system overhead involved in data transfers.

System Interface

The MPSC system interface is extremely flexible, supporting the following data transfer modes:

1. Polled Mode. The system processor periodically reads (polls) an 8274 status register to determine when a character has been received, when a character is needed for transmission, and when transmission errors are detected.

2. Interrupt Mode. The MPSC interrupts the system processor when a character has been received, when a character is needed for transmission, and when transmission errors are detected.

3. DMA Mode. The MPSC automatically requests data transfers from system memory for both transmit and receive functions by means of two DMA request signals per serial channel. These DMA request signals may be directly interfaced to an 8237 or 8257 DMA controller or to an 8089 I/O processor.

4. WAIT Mode. The MPSC ready signal is used to synchronize processor data transfers by forcing the processor to enter wait states until the 8274 is ready for another data byte. This feature enables the 8274 to interface directly to an 8086 or 8088 processor by means of string I/O instructions for very high-speed data links.

Scope

This application note describes the use of the 8274 in asynchronous communication modes. Asynchronous communication is typically used to transfer data to/from video display terminals, modems, printers, and other low-to-medium-speed peripheral devices. Use of the 8274 in both interrupt-driven and polled system environments is described. Use of the DMA and WAIT modes are not described since these modes are employed mainly in synchronous communication systems where extremely high data rates are common. Programming examples are written in PL/M-86 (Appendix B and Appendix C). PL/M-86 is executed by the iAPX-86 and iAPX-88 processor families. In addition, PL/M-86 is very similar to PL/M-80 (executed by the MCS-80 and MCS-85 processor families). In addition, Appendix D describes a simple application example using an SDK-86 in an iAPX-86/88 environment.

SERIAL-ASYNCHRONOUS DATA LINKS

A serial asynchronous interface is a method of data transmission in which the receiving and transmitting systems need not be synchronized. Instead of transmitting clocking information with the data, locally generated clocks (16, 32 or 64 times as fast as the data transmission rate) are used by the transmitting and receiving systems. When a character of information is sent by the transmitting system, the character data is framed (preceded and followed) by special START and STOP bits. This framing information permits the receiving system to temporarily synchronize with the data transmission. (Refer to Figure 1 during the following discussion of asynchronous data transmission.)
Normally the data link is in an idle or marking state, continuously transmitting a "mark" (binary 1). When a character is to be sent, the character data bits are immediately preceded by a "space" (binary 0 START bit). The mark-to-space transition informs the receiving system that a character of information will immediately follow the start bit. Figure 1 illustrates the transmission of a 7-bit ASCII character (upper case S) with even parity. Note that the character is transmitted immediately following the start bit. Data bits within the character are transmitted from least-significant to most-significant. The parity bit is transmitted immediately following the character data bits and the STOP framing bit (binary 1) signifies the end of the character.

Asynchronous interfaces are often used with human interface devices such as CRT/keyboard units where the time between data transmissions is extremely variable.

Characters

In asynchronous mode, characters may vary in length from five to eight bits. The character length depends on the coding method used. For example, five-bit characters are used when transmitting Baudot Code, seven-bit characters are required for ASCII data, and eight-bit characters are needed for EBCDIC and binary data. To transmit messages composed of multiple characters, each character is framed and transmitted separately (Figure 2).

This framing method ensures that the receiving system can easily synchronize with the start and stop bits of each character, preventing receiver synchronization errors. In addition, this synchronization method makes both transmitting and receiving systems insensitive to possible time delays between character transmissions.
Framing

Character framing is accomplished by the START and STOP bits described previously. When the START bit transition (mark-to-space) is detected, the receiving system assumes that a character of data will follow. In order to test this assumption (and isolate noise pulses on the data link), the receiving system waits one-half bit time and samples the data link again. If the link has returned to the marking state, noise is assumed, and the receiver waits for another START bit transition.

When a valid START bit is detected, the receiver samples the data link for each bit of the following character. Character data bits and the parity bit (if required) are sampled at their nominal centers until all required characters are received. Immediately following the data bits, the receiver samples the data link for the STOP bit, indicating the end of the character. Most systems permit specification of 1, 1½, or 2 stop bits.

Timing

The transmitter and receiver in an asynchronous data link arrangement are clocked independently. Normally, each clock is generated locally and the clocks are not synchronized. In fact, each clock may be a slightly different frequency. (In practice, the frequency difference should not exceed a few percent. If the transmitter and receiver clock rates vary substantially, errors will occur because data bits may be incorrectly identified as START or STOP framing bits.) These clocks are designed to operate at 16, 32, or 64 times the communications data rate. These clock speeds allow the receiving device to correctly sample the incoming bit stream.

Serial-interface data rates are measured in bits/second. The term "baud" is used to specify the number of times per second that the transmitted signal level can change states. In general, the baud is not equal to the bit rate. Only when the transmitted signal has two states (electrical levels) is the baud rate equal to the bit rate. Most point-to-point serial data links use RS-232-C, RS-422, or RS-423 electrical interfaces. These specifications call for two electrical signal levels (the baud is equal to the bit rate). Modem interfaces, however, may often have differing bit and baud rates.

While there are generally no limitations on the data transmission rates used in an asynchronous data link, a limited set of rates has been standardized to promote equipment interconnection. These rates vary from 75 bits per second to 38,400 bits per second. Table 1 illustrates typical asynchronous data rates and the associated clock frequencies required for the transmitter and receiver circuits.

<table>
<thead>
<tr>
<th>Data Rate (Bits/Second)</th>
<th>Clock Rate (kHz)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>X16</td>
</tr>
<tr>
<td>75</td>
<td>1.2</td>
</tr>
<tr>
<td>150</td>
<td>2.4</td>
</tr>
<tr>
<td>300</td>
<td>4.8</td>
</tr>
<tr>
<td>600</td>
<td>9.6</td>
</tr>
<tr>
<td>1200</td>
<td>19.2</td>
</tr>
<tr>
<td>2400</td>
<td>38.4</td>
</tr>
<tr>
<td>4800</td>
<td>76.8</td>
</tr>
<tr>
<td>9600</td>
<td>153.6</td>
</tr>
<tr>
<td>19200</td>
<td>307.2</td>
</tr>
<tr>
<td>38400</td>
<td>614.4</td>
</tr>
</tbody>
</table>

Parity

In order to detect transmission errors, a parity bit may be added to the character data as it is transferred over the data link. The parity bit is set or cleared to make the total number of "one" bits in the character even (even parity) or odd (odd parity). For example, the letter "A" is represented by the seven-bit ASCII code 1000001 (41H). The transmitted data code (with parity) for this character contains eight bits; 01000001 (41H) for even parity and 11000001 (OC1H) for odd parity. Note that a single bit error changes the parity of the received character and is therefore easily detected. The 8274 supports both odd and even parity checking as well as a parity disable mode to support binary data transfers.

Communication Modes

Serial data transmission between two devices can occur in one of three modes. In the simplex transmission mode, a data link can transmit data in one direction only. In the half-duplex mode, the data link can transmit data in both directions, but not simultaneously. In the full-duplex mode (the most common), the data link can transmit data in both directions simultaneously. The 8274 directly supports the full-duplex mode and will interface to simplex and half-duplex communication data links with appropriate software controls.
BREAK Condition

Asynchronous data links often include a special sequence known as a break condition. A break condition is initiated when the transmitting device forces the data link to a spacing state (binary 0) for an extended length of time (typically 150 milliseconds). Many terminals contain keys to initiate a break sequence. Under software control, the 8274 can initiate a break sequence when transmitting data and detect a break sequence when receiving data.

MPSC SYSTEM INTERFACE

Hardware Environment

The 8274 MPSC interfaces to the system processor over an 8-bit data bus. Each serial I/O channel responds to two I/O or memory addresses as shown in Table 2. In addition, the MPSC supports non-vectored and vectored interrupts.

The 8274 may be configured for memory-mapped or I/O-mapped operation.

The 8274-processor hardware interface can be configured in a flexible manner, depending on the operating mode selected—polled, interrupt-driven, DMA, or WAIT. Figure 3 illustrates typical MPSC configurations for use with an 8088 microprocessor in the polled and interrupt-driven modes.

All serial-to-parallel conversion, parallel-to-serial conversion, and parity checking required during asynchronous serial I/O operation is automatically performed by the MPSC.

Operational Interface

Command, parameter, and status information is stored in 21 registers within the MPSC (8 writable registers and 2 readable registers for each channel, plus the interrupt vector register). These registers are all accessed by means of the command/status ports for each channel. An internal pointer register selects which of the command or status registers will be written or read during a command/status access of an MPSC channel. Figure 4 diagrams the command/status register architecture for each serial channel. In the following discussion, the writable registers will be referred to as WR0 through WR7 and the readable registers will be referred to as RR0 through RR2.

Table 2. 8274 Addressing

<table>
<thead>
<tr>
<th>CS</th>
<th>A1</th>
<th>A0</th>
<th>Read Operation</th>
<th>Write Operation</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>Ch. A Data Read</td>
<td>Ch. A Data Write</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>Ch. A Status Read</td>
<td>Ch. A Command/Parameter</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>Ch. B Data Read</td>
<td>Ch. B Data Write</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>Ch. B Status Read</td>
<td>Ch. B Command/Parameter</td>
</tr>
<tr>
<td>1</td>
<td>X</td>
<td>X</td>
<td>High Impedance</td>
<td>High Impedance</td>
</tr>
</tbody>
</table>
The least-significant three bits of WR0 are automatically loaded into the pointer register every time WR0 is written. After reset, WR0 is set to zero so that the first write to a command register causes the data to be loaded into WR0 (thereby setting the pointer register). After WR0 is written, the following read or write accesses the register selected by the pointer. The pointer is reset after the read or write operation is completed. In this manner, reading or writing an arbitrary MPSC channel register requires two I/O accesses. The first access is always a write command. This write command is used to set the pointer register. The second access is either a read or a write command; the pointer register (previously set) will ensure that the correct internal register is read or written. After this second access, the pointer register is automatically reset. Note that writing WR0 and reading RR0 does not require presetting of the pointer register.

During initialization and normal MPSC operation, various registers are read and/or written by the system processor. These actions are discussed in detail in the following paragraphs. Note that WR6 and WR7 are not used in the asynchronous communication modes.

**RESET**

When the 8274RESET line is activated, both MPSC channels enter the idle state. The serial output lines are forced to the marking state (high) and the modem interface signals (RTS, DTR) are forced high. In addition, the pointer register is set to zero.
External/Status Latches

The MPSC continuously monitors the state of four external/status conditions:
1. CTS—clear-to-send input pin.
2. CD—carrier-detect input pin.
3. SYNDET—sync-detect input pin. This pin may be used as a general-purpose input in the asynchronous communication mode.
4. BREAK—a break condition (series of space bits on the receiver input pin).

A change of state in any of these monitored conditions will cause the associated status bit in RR0 (Appendix A) to be latched (and optionally cause an interrupt).

Error Reporting

Three error conditions may be encountered during data reception in the asynchronous mode:

1. Parity. If parity bits are computed and transmitted with each character and the MPSC is set to check parity (bit 0 in WR4 is set), a parity error will occur whenever the number of "1" bits within the character (including the parity bit) does not match the odd/even setting of the parity check flag (bit 1 in WR4).
2. Framing. A framing error will occur if a stop bit is not detected immediately following the parity bit (if parity checking is enabled) or immediately following the most-significant data bit (if parity checking is not enabled).

3. Overrun. If an input character has been assembled but the receiver buffers are full (because the previously received characters have not been read by the system processor), an overrun error will occur. When an overrun error occurs, the input character that has just been received will overwrite the immediately preceding character.

Transmitter/Receiver Initialization

In order to operate in the asynchronous mode, each MPSC channel must be initialized with the following information:

1. Clock Rate. This parameter is specified by bits 6 and 7 of WR4. The clock rate may be set to 16, 32, or 64 times the data-link bit rate. (See Appendix A for WR4 details.)

2. Number of Stop Bits. This parameter is specified by bits 2 and 3 of WR4. The number of stop bits may be set to 1, 1½, or 2. (See Appendix A for WR4 details.)

3. Parity Selection. Parity may be set for odd, even, or no parity by bits 0 and 1 of WR4. (See Appendix A for WR4 details.)

4. Receiver Character Length. This parameter sets the length of received characters to 5, 6, 7, or 8 bits. This parameter is specified by bits 6 and 7 of WR3. (See Appendix A for WR3 details.)

5. Receiver Enable. The serial-channel receiver operation may be enabled or disabled by setting or clearing bit 0 of WR3. (See Appendix A for WR3 details.)

6. Transmitter Character Length. This parameter sets the length of transmitted characters to 5, 6, 7, or 8 bits. This parameter is specified by bits 5 and 6 of WR5. (See Appendix A for WR5 details.) Characters of less than 5 bits in length may be transmitted by setting the transmitted length to five bits (set bits 5 and 6 of WR5 to 1).

The MPSC then determines the actual number of bits to be transmitted from the character data byte. The bits to be transmitted must be right justified in the data byte, the next three bits must be set to 0 and all remaining bits must be set to 1. The following table illustrates the data formats for transmission of 1 to 5 bits of data:

<table>
<thead>
<tr>
<th>Number of Bits Transmitted</th>
<th>D7 D6 D5 D4 D3 D2 D1 D0 (Character Length)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1 1 1 1 0 0 0 c 1</td>
</tr>
<tr>
<td>2</td>
<td>1 1 1 0 0 0 c c 2</td>
</tr>
<tr>
<td>3</td>
<td>1 0 0 0 c c c 3</td>
</tr>
<tr>
<td>4</td>
<td>0 0 0 c c c 4</td>
</tr>
<tr>
<td>5</td>
<td></td>
</tr>
</tbody>
</table>

7. Transmitter Enable. The serial channel transmitter operation may be enabled or disabled by setting or clearing bit 3 of WR5. (See Appendix A for WR5 details.)

For data transmissions via a modem or RS-232-C interface, the following information must also be specified:

1. Request-to-Send/Data-Terminal-Ready. Must be set to indicate status of data terminal equipment. Request-to-send is controlled by bit 1 of WR5 and data terminal ready is controlled by bit 7. (See Appendix A for WR5 details.)

2. Auto Enable. May be set to allow the MPSC to automatically enable the channel transmitter when the clear-to-send signal is active and to automatically enable the receiver when the carrier-detect signal is active. Auto Enable is controlled by bit 5 of WR3. (See Appendix A for WR3 details.)

During initialization, it is desirable to guarantee that the external/status latches reflect the latest interface information. Since up to two state changes are internally stored by the MPSC, at least two Reset External/Status Interrupt commands must be issued. This procedure is most easily accomplished by simply issuing this reset command whenever the pointer register is set during initialization.

An MPSC initialization procedure (MPSC$RX$INIT) for asynchronous communication is listed in Appendix B. Figure 5 illustrates typical MPSC initialization parameters for use with this procedure.

call MPSC$RX$INIT(41, 1,1,0,1, 3,1,1, 3,1,1,0,1);

initializes the 8274 at address 41 as follows:

- X16 clock rate: Enable transmitter and receiver
- 1 stop bit
- Odd parity: Auto enable set
- 8-bit characters: DTR and RTS set
- (Tx and Rx): Break transmission disabled

Figure 5. Sample 8274 Initialization Procedure for Polled Operation
Polled Operation

In the polled mode, the processor must monitor the MPSC status by testing the appropriate bits in the read register. Data available, status, and error conditions are represented in RR0 and RR1 for channels A and B. An example of MPSC-polled transmitter/receiver routines are given in Appendix B. The following routines are detailed:

1. **MPSC$POLL$RCV$CHARACTER**—This procedure receives a character from the serial data link. The routine waits until the character-available flag in RR0 has been set. When this flag indicates that a character is available, RR1 is checked for errors (overrun, parity, or framing). If an error is detected, the character in the MPSC receive buffer must be read and discarded and the error routine (RECEIVE$ERROR) is called. If no receive errors have been detected, the character is input from the 8274 data port and returned to the calling program.

The routine waits until the transmitter-buffer-empty flag is set. When this flag indicates that the transmitter buffer is empty, the following eight interrupts are differentiated automatically by the 8274 hardware:

- **Channel B Transmitter Buffer Empty.**
- **Channel B External/Status Transition.**
- **Channel B Character Available.**
- **Channel B Receive Error.**
- **Channel A Transmitter Buffer Empty.**
- **Channel A External/Status Transition.**
- **Channel A Character Available.**
- **Channel A Receive Error.**

2. **MPSC$POLL$TRANS$CHARACTER**—This procedure transmits a character to the serial data link. The routine waits until the transmitter-buffer-empty flag has been set in RR0 before writing the character to the 8274.

3. **RECEIVE$ERROR**—This procedure processes receiver errors. First, an Error Reset command is written to the affected channel. All additional error processing is dependent on the specific application. For example, the receiving device may immediately request retransmission of the character or wait until a message has been completed.

Serial I/O by writing the first character of a message to the MPSC. The MPSC interrupts the system processor whenever the next character is required (i.e., when the transmitter buffer is empty) and the processor responds by writing the next character of the message to the MPSC data port for the appropriate channel.

By using interrupt-driven I/O, the MPSC proceeds independently of the system processor, signalling the processor only when characters are required for transmission, when characters are received from the data link, or when errors occur. In this manner, the system processor may continue execution of other tasks while serial I/O is performed concurrently.

Interrupt Configurations

The 8274 is designed to interface to 8085- and 8086-type processors in much the same manner as the 8259A is designed. When operating in the 8085 mode, the 8274 causes a “call” to a prespecified, interrupt-service routine location. In the 8086 mode, the 8274 presents the processor with a one-byte interrupt-type number. This interrupt-type number is used to “vector” through the 8086 interrupt service table. In either case, the interrupt service address or interrupt-type number is specified during MPSC initialization.

To shorten interrupt latency, the 8274 can be programmed to modify the prespecified interrupt vector so that no software overhead is required to determine the cause of an interrupt. When this “status affects vector” mode is enabled, the following eight interrupts are differentiated automatically by the 8274 hardware:

1. Channel B Transmitter Buffer Empty.
2. Channel B External/Status Transition.
3. Channel B Character Available.
5. Channel A Transmitter Buffer Empty.
6. Channel A External/Status Transition.
7. Channel A Character Available.
8. Channel A Receive Error.

Interrupt Sources/Priorities

The 8274 has three interrupt sources for each channel:

1. Receiver (RxA, RxB). An interrupt is initiated when a character is available in the receiver buffer or when a receiver error (parity, framing, or overrun) is detected.
2. Transmitter (TxA, TxB). An interrupt is initiated when the transmitter buffer is empty and the 8274 is ready to accept another character for transmission.
3. External/Status (ExTA, ExTB). An interrupt is initiated when one of the external/status conditions (CDE, CTS, SYNDET, BREAK) changes state.

The 8274 supports two interrupt priority orderings (selectable during MPSC initialization) as detailed in Appendix A, WR2, CH-A.

**Interrupt Initialization**

In addition to the initialization parameters required for polled operation, the following parameters must be supplied to the 8274 to specify interrupt operation:

1. Transmit Interrupt Enable. Transmitter-buffer-empty interrupts are separately enabled by bit 1 of WR1. (See Appendix A for WR1 details.)

2. Receive Interrupt Enable. Receiver interrupts are separately enabled in one of three modes: a) interrupt on first received character only and on receive errors (used for message-oriented transmission systems), b) interrupt on all received characters and on receive errors, but do not interrupt on parity errors, and c) interrupt on all received characters and on receive errors (including parity errors). The ability to separately disable parity interrupts can be extremely useful when transmitting messages. Since the parity error bit in RR1 is latched, it will not be reset until an error reset operation is performed. Therefore, the parity error bit will be set if any parity errors were detected in a multi-character message. If this mode is used, the serial I/O software must poll the parity error bit at the completion of a message and issue an error reset if appropriate. The receiver interrupt mode is controlled by bits 3 and 4 of WR1. (See Appendix A for WR1 details.)

3. External/Status Interrupts. External/Status interrupts can be separately enabled by bit 0 of WR1. (See Appendix A for WR1 details.)

4. Interrupt Vector. An eight-bit interrupt-service routine location (8085) or interrupt type (8086) is specified through WR2 of channel B. (See Appendix A for WR2 details.) Table 3 lists interrupt vector addresses generated by the 8274 in the “status affects vector” mode.

5. “Status Affects Vector” Mode. The 8274 will automatically modify the interrupt vector if bit 3 of WR1 is set. (See Appendix A for WR1 details.)

6. System Configuration. Specifies the 8274 data transfer mode. Three configuration modes are available: a) interrupt-driven operation for both channels, b) DMA operation for both channels, and c) DMA operation for channel A, interrupt-driven operation for channel B. The system configuration is specified by means of bits 0 and 1 of WR2 (channel A). (See Appendix A for WR2 details.)

7. Interrupt Priorities. The 8274 permits software specification of receive/transmit priorities by means of bit 2 of WR2 (channel A). (See Appendix A for WR2 details.)

8. Interrupt Mode. Specifies whether the MPSC is to operate in a non-vectored mode (for use with an external interrupt controller), in an 8086-vectored mode, or in an 8085-vectored mode. This parameter is specified through bits 3 and 4 of WR2 (channel A). (See Appendix A for WR2 details.)

An MPSC interrupt initialization procedure (MPSC$INT$INIT$IT$) is listed in Appendix C.
Table 3. MPSC-Generated Interrupt Vectors in "Status Affects Vector" Mode

<table>
<thead>
<tr>
<th>V7</th>
<th>V6</th>
<th>V5</th>
<th>V4</th>
<th>V3</th>
<th>V2</th>
<th>V1</th>
<th>V0</th>
<th>Original Vector (Specified during Initialization)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td><strong>8086 Interrupt Type</strong></td>
</tr>
<tr>
<td>V7</td>
<td>V6</td>
<td>V5</td>
<td>V4</td>
<td>V3</td>
<td>V2</td>
<td>V1</td>
<td>V0</td>
<td>V7</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Channel B Transmitter Buffer Empty</td>
</tr>
<tr>
<td>V7</td>
<td>V6</td>
<td>V5</td>
<td>V4</td>
<td>V3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>V7</td>
</tr>
<tr>
<td>V7</td>
<td>V6</td>
<td>V5</td>
<td>V4</td>
<td>V3</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>V7</td>
</tr>
<tr>
<td>V7</td>
<td>V6</td>
<td>V5</td>
<td>V4</td>
<td>V3</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>V7</td>
</tr>
<tr>
<td>V7</td>
<td>V6</td>
<td>V5</td>
<td>V4</td>
<td>V3</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>V7</td>
</tr>
<tr>
<td>V7</td>
<td>V6</td>
<td>V5</td>
<td>V4</td>
<td>V3</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>V7</td>
</tr>
<tr>
<td>V7</td>
<td>V6</td>
<td>V5</td>
<td>V4</td>
<td>V3</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>V7</td>
</tr>
<tr>
<td>V7</td>
<td>V6</td>
<td>V5</td>
<td>V4</td>
<td>V3</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>V7</td>
</tr>
</tbody>
</table>

**Interrupt Service Routines**

Appendix C lists four interrupt service procedures, a buffer transmission procedure, and a buffer reception procedure that illustrate the use of the 8274 in interrupt-driven environments. Use of these procedures assumes that the 8086/8088 interrupt vector is set to 20H and that channel B is used with the "status affects vector" mode enabled.

1. TRANSMIT$BUFFER—This procedure begins serial transmission of a data buffer. Two parameters are required—a pointer to the buffer (buf$ptr) and the length of the buffer (buf$length). The procedure first sets the global buffer pointer, buffer length, and initial index for the transmitter-interrupt service routine and initiates transmission by writing the first character of the buffer to the 8274. The procedure then enters a wait loop until the I/O completion status is set by the transmit-interrupt service routine (MPSC$TRANSMIT$CHARACTERS$INT).

2. RECEIVE$BUFFER—This procedure inputs a line (terminated by a line feed) from a serial I/O port. Two parameters are required—a pointer to the input buffer (buf$ptr) and a pointer to the buffer length variable (buf$length$ptr). The buffer length will be set by this procedure when the complete line has been input. The procedure first sets the global buffer pointer and initial index for the receiver interrupt service routine. RECEIVE$BUFFER then enters a wait loop until the I/O completion status is set by the receive interrupt routine (MPSC$RECEIVE$CHARACTERS$INT).

3. MPSC$TRANSMIT$CHARACTERS$INT—This procedure is executed when the MPSC Tx-buffer-empty interrupt is acknowledged. If the current transmit buffer index is less than the buffer length, the next character in the buffer is written to the MPSC data port and the buffer pointer is updated. Otherwise, the transmission complete status is posted.

4. MPSC$RECEIVE$CHARACTERS$INT—This procedure is executed when a character has been assembled by the MPSC and the MPSC has issued a character-available interrupt. If no input buffer has been set up by RECEIVE$BUFFER, the character is ignored. If a buffer has been set up, but it is full, a receive overrun error is posted. Otherwise, the received character is read from the MPSC data port and the buffer index is updated. Finally, if the received character is a line feed, the reception complete status is posted.

5. RECEIVESERRORS$INT—This procedure is executed when a receive error is detected. First, the error conditions are read from RR1 and the character currently in the MPSC receive buffer is read and discarded. Next, an Error Reset command is written to the affected channel. All additional error processing is application dependent.

6. EXTERNAL$STATUS$CHANGES$INT—This procedure is executed when an external status condition change is detected. The status conditions are read from RR0 and a Reset External/Status Interrupt command is issued. Further error processing is application dependent.
DATA LINK INTERFACE

Serial Data Interface

Each serial I/O channel within the 8274 MPSC interfaces to two data link lines—one line for transmitting data and one for receiving data. During transmission, characters are converted from parallel data format (as supplied by the system processor or DMA device) into a serial bit stream (with START and STOP bits) and clocked out on the TxD pin. During reception, a serial bit stream is input on the RxD pin, framing bits are stripped out of the data stream, and the resulting character is converted to parallel data format and passed to the system processor or DMA device.

Data Clocking

As discussed previously, the frequency of data transmission/reception on the data link is controlled by the MPSC clock in conjunction with the programmed clock divider (in register WR4). The 8274 is designed to permit all four serial interface lines (TxD and RxD for each channel) to operate at different data rates. Four clock input pins (TxC and RxC for each channel) are available for this function. Note that the clock rate divider specified in WR4 is used for both RxC and TxC on the appropriate channel; clock rate dividers for each channel are independent.

Modem Control

The following four modem interface signals may be connected to the 8274:

1. Data Terminal Ready (DTR). This interface signal (output by the 8274) is software controlled through bit 7 of WR5. When active, DTR indicates that the data terminal/computer equipment is active and ready to interact with the data communications channel. In addition, this signal prepares the modem for connection to the communication channel and maintains connections previously established (e.g., manual call origination).

2. Request To Send (RTS). This interface signal (output by the 8274) is software controlled through bit 1 of WR5. When active, RTS indicates that the data terminal/computer equipment is ready to transmit data. When the RTS bit is reset in asynchronous mode, the signal does not go high until the transmitter is empty.

3. Clear To Send (CTS). This interface signal (input to the 8274) is supplied by the modem in response to an active RTS signal. CTS indicates that the data terminal/computer equipment is permitted to transmit data. The state of CTS is available to the programmer as bit 5 of RR0. In addition, if the auto enable control is set (bit 5 of WR3), the 8274 will not transmit data bytes until CTS has been activated. If CTS becomes inactive during transmission of a character, the current character transmission is completed before the transmitter is disabled.

4. Carrier Detect (CD). This interface signal (input to the 8274) is supplied by the modem to indicate that a data carrier signal has been detected and that a valid data signal is present on the RxD line. The state of CD is available to the programmer as bit 3 of RR0. In addition, if the auto enable control is set (bit 5 of WR3), the 8274 will not enable the serial receiver until CD has been activated. If the CD signal becomes inactive during reception of a character, the receiver is disabled, and the partially received character is lost.

In addition to the above modem interface signals, the 8274 SYNDET input pin for channel A may be used as a general-purpose input in the asynchronous communication mode. The status of this signal is available to the programmer as bit 4 of status register RR0.
**APPENDIX A**

**COMMAND/STATUS DETAILS FOR ASYNCHRONOUS COMMUNICATION**

**Write Register 0 (WR0):**

<table>
<thead>
<tr>
<th>Register Pointer Bits</th>
<th>Command Status Pointer Bits</th>
</tr>
</thead>
<tbody>
<tr>
<td>D7 D6 D5 D4 D3 D2 D1 D0</td>
<td>NOT USED IN ASYNCHRONOUS MODES</td>
</tr>
<tr>
<td>0 0 0 0</td>
<td>NULL CODE</td>
</tr>
<tr>
<td>0 0 1 0</td>
<td>NOT USED IN ASYNCHRONOUS MODES</td>
</tr>
<tr>
<td>0 1 0 0</td>
<td>CHANNEL RESET</td>
</tr>
<tr>
<td>0 1 1 0</td>
<td>ENABLE INTERRUPT ON NEXT RX CHARACTER</td>
</tr>
<tr>
<td>1 0 0 0</td>
<td>RESET Rx INT PENDING</td>
</tr>
<tr>
<td>1 1 0 0</td>
<td>END OF INTERRUPT</td>
</tr>
</tbody>
</table>

**Write Register 1 (WR1):**

<table>
<thead>
<tr>
<th>D0</th>
<th>External/Status Interrupt Enable</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>allows interrupt to occur as the result of transitions on the CD, CTS or SYNDET inputs. Also allows interrupts as the result of a Break/Abort detection and termination, or at the beginning of CRC, or sync character transmission when the Transmit Underrun/EOM latch becomes set.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>D1</th>
<th>Transmitter Interrupt/DMA Enable</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>allows the MPSC to interrupt or request a DMA transfer when the transmitter buffer becomes empty.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>D2</th>
<th>Status Affects Vector</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>(WR1, D2 active in channel B only.) If this bit is not set, then the fixed vector, programmed in WR2, is returned from an interrupt acknowledge sequence. If the bit is set, then the vector returned from an interrupt acknowledge is variable as shown in the Interrupt Vector Table.</td>
</tr>
</tbody>
</table>
Write Register 1 (WR1):

<table>
<thead>
<tr>
<th>D7</th>
<th>D6</th>
<th>D5</th>
<th>D4</th>
<th>D3</th>
<th>D2</th>
<th>D1</th>
<th>D0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

- **MSB**
  - **D7**: EXT INTERRUPT ENABLE
  - **D6**: RxINT/DMA DISABLE
  - **D5**: INT ON FIRST CHAR OR SPECIAL CONDITION
  - **D4**: INT ON ALL Rx CHAR (PARITY AFFECTS VECTOR) OR SPECIAL CONDITION
  - **D3**: INT ON ALL Rx CHAR (PARITY DOES NOT AFFECT VECTOR) OR SPECIAL CONDITION
  - **D2**: WAIT ON Rx, 0 = WAIT ON Tx
  - **D1**: WAIT ENABLE
  - **D0**: MUST BE ZERO

- **LSB**
  - **D7**: MUST BE ZERO
  - **D6**: RXINT/DMA DISABLE
  - **D5**: INT ON FIRST CHAR OR SPECIAL CONDITION
  - **D4**: INT ON ALL Rx CHAR (PARITY AFFECTS VECTOR) OR SPECIAL CONDITION
  - **D3**: INT ON ALL Rx CHAR (PARITY DOES NOT AFFECT VECTOR) OR SPECIAL CONDITION
  - **D2**: WAIT ON Rx, 0 = WAIT ON Tx
  - **D1**: WAIT ENABLE
  - **D0**: MUST BE ZERO

Write Register 2 (WR2): Channel A

<table>
<thead>
<tr>
<th>D7</th>
<th>D6</th>
<th>D5</th>
<th>D4</th>
<th>D3</th>
<th>D2</th>
<th>D1</th>
<th>D0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

- **MSB**
  - **D7**: BOTH INTERRUPT
  - **D6**: BOTH DMA
  - **D5**: ILLEGAL
  - **D4**: 8085 MODE 1
  - **D3**: 8085 MODE 2
  - **D2**: 8086/88 MODE
  - **D1**: ILLEGAL
  - **D0**: NON VECTORED INTERRUPT

- **LSB**
  - **D7**: PIN 10 = SYNDET
  - **D6**: PIN 10 = RTS
  - **D5**: 1 = VECTORED INTERRUPT
  - **D4**: MUST BE ZERO
  - **D3**: 0 = NON VECTORED INTERRUPT
  - **D2**: 1 = VECTORED INTERRUPT
  - **D1**: MUST BE ZERO
  - **D0**: PIN 10 = RTS

**NOTE:** *External Status Interrupt—only if EXT Interrupt Enable (WR1; D0) is set.

**D4,D3** Receive Interrupt Mode.
0 0 Receive Interrupts/DMA Disabled.
0 1 Receive Interrupt on First Character Only or Special Condition.
1 0 Interrupt on All Receive Characters of Special Condition (Parity Error is a Special Receive Condition).
1 1 Interrupt on All Receive Characters or Special Condition (Parity Error is not a Special Receive Condition).

**D5** Wait on Receive/Transmit—when the following conditions are met, the RDY pin is activated, otherwise it is held in the High-Z state. (Conditions: Interrupt Enabled Mode, Wait Enabled, CS = 0, A0 = 0/1, and A1 = 0). The RDY pin is pulled low when the transmitter buffer is full or the receiver buffer is empty and it is driven High when the transmitter buffer is empty or the receiver buffer is full. The RDY_A and RDY_B may be wired or connected since only one signal is active at any one time while the other is in the High Z state.

**D6** Must be Zero.
**D7** Wait Enable—enables the wait function.

**D1,D0** System Configuration—These specify the data transfer from MPSC channels to the CPU, either interrupt or DMA based.
0 0 Channel A and Channel B both use interrupts.
0 1 Channel A uses DMA, Channel B uses interrupt.
1 0 Channel A and Channel B both use DMA.
1 1 Illegal Code.

**D2** Priority—this bit specifies the relative priorities of the internal MPSC interrupt/DMA sources.
0 (Highest) RxA, TxA, RxA, RxB, TxBExTA, ExTB (Lowest).
1 (Highest) RxA, RxB, TxA, TxB, ExTA, ExTB (Lowest).
D5,D4,D3 Interrupt Code—specifies the behavior of the MPSC when it receives an interrupt acknowledge sequence from the CPU. (See Interrupt Vector Mode Table.)

0 X X Non-vectored interrupts—intended for use with an external interrupt controller such as the 8259A.

1 0 0 8085 Vector Mode 1—intended for use as the primary MPSC in a daisy-chained priority structure.

1 0 1 8085 Vector Mode 2—intended for use as any secondary MPSC in a daisy-chained priority structure.

1 1 0 8086/88 Vector Mode—intended for use as either a primary or secondary in a daisy-chained priority structure.

D6 Must be Zero.

D7

0 Pin 10 = RTSB.

1 Pin 10 = SYNDETB.

Write Register 2 (WR2): Channel B

Write Register 3 (WR3):

D0 Receiver Enable—a one enables the receiver to begin. This bit should be set only after the receiver has been initialized.

D5 Auto Enables—a one written to this bit causes CD to be an automatic enable signal for the receiver and CTS to be an automatic enable signal for the transmitter. A zero written to this bit limits the effect of CD and CTS signals to setting/resetting their corresponding bits in the status register (RR0).

D7,D6 Receiver Character length.

0 0 Receive 5 Data bits/character.

0 1 Receive 7 Data bits/character.

1 0 Receive 6 Data bits/character.

1 1 Receive 8 Data bits/character.

D7-D0 Interrupt vector—this register contains the value of the interrupt vector placed on the data bus during acknowledge sequences.
**Write Register 4 (WR4):**

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>D7</td>
<td>Parity—a one in this bit causes a parity bit to be added to the programmed number of data bits per character for both the transmitted and received character. If the MPSC is programmed to receive 8 bits per character, the parity bit is not transferred to the microprocessor. With other receiver character lengths, the parity bit is transferred to the microprocessor.</td>
</tr>
<tr>
<td>D6</td>
<td>Even/Odd Parity—if parity is enabled, a one in this bit causes the MPSC to transmit and expect even parity, and zero causes it to send and expect odd parity.</td>
</tr>
<tr>
<td>D5</td>
<td>Stop Bits.</td>
</tr>
<tr>
<td>D4</td>
<td>0 0 Selects synchronous modes.</td>
</tr>
<tr>
<td>D3</td>
<td>0 1 Async mode, 1 stop bit/character.</td>
</tr>
<tr>
<td>D2</td>
<td>1 0 Async mode, 1½ stop bits/character.</td>
</tr>
<tr>
<td>D1</td>
<td>1 1 Async mode, 2 stop bits/character.</td>
</tr>
<tr>
<td>D0</td>
<td>Clock mode—selects the clock/data rate multiplier for both the receiver and the transmitter. If the 1x mode is selected, bit synchronization must be done externally.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 0</td>
<td>Clock rate = Data rate × 1.</td>
</tr>
<tr>
<td>0 1</td>
<td>Clock rate = Data rate × 16.</td>
</tr>
<tr>
<td>1 0</td>
<td>Clock rate = Data rate × 32.</td>
</tr>
<tr>
<td>1 1</td>
<td>Clock rate = Data rate × 64.</td>
</tr>
</tbody>
</table>

**Write Register 5 (WR5):**

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>D7</td>
<td>Request to Send—a one in this bit forces the RTS pin active (low) and zero in this bit forces the RTS pin inactive (high). When the RTS bit is reset in asynchronous mode, the signal does not go inactive until the transmitter is empty.</td>
</tr>
<tr>
<td>D6</td>
<td>Transmitter Enable—a zero in this bit forces a marking state on the transmitter output. If this bit is set to zero during data or sync character transmission, the marking state is entered after the character has been sent. If this bit is set to zero during transmission of a CRC character, sync or flag bits are substituted for the remainder of the CRC bits.</td>
</tr>
<tr>
<td>D5</td>
<td>Send Break—a one in this bit forces the transmit data low. A zero in this bit allows normal transmitter operation.</td>
</tr>
<tr>
<td>D4</td>
<td>Transmit Character length.</td>
</tr>
<tr>
<td>D3</td>
<td>0 0 Transmit 5 or less bits/character.</td>
</tr>
<tr>
<td>D2</td>
<td>0 1 Transmit 7 bits/character.</td>
</tr>
<tr>
<td>D1</td>
<td>1 0 Transmit 6 bits/character.</td>
</tr>
<tr>
<td>D0</td>
<td>1 1 Transmit 8 bits/character.</td>
</tr>
</tbody>
</table>

Bits to be sent must be right justified, least-significant bit first, e.g.:

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>D7</td>
<td>Clock rate = Data rate × 1.</td>
</tr>
<tr>
<td>D6</td>
<td>Clock rate = Data rate × 16.</td>
</tr>
<tr>
<td>D5</td>
<td>Clock rate = Data rate × 32.</td>
</tr>
<tr>
<td>D4</td>
<td>Clock rate = Data rate × 64.</td>
</tr>
<tr>
<td>D3</td>
<td>0 0 B5 B4 B3 B2 B1 B0</td>
</tr>
</tbody>
</table>
Read Register 0 (RR0):

D0  Receive Character Available—this bit is set when the receive FIFO contains data and is reset when the FIFO is empty.

D1  Interrupt Pending—This Interrupt-Pending bit is reset when an E01 command is issued and there is no other interrupt request pending at that time. In vector mode, this bit is set at the falling edge of the second INTA in an INTA cycle for an internal interrupt request. In non-vector mode, this bit is set at the falling edge of RD input after pointer 2 is specified. This bit is always zero in Channel B.

D2  Transmit Buffer Empty—This bit is set whenever the transmit buffer is empty except when CRC characters are being sent in a synchronous mode. This bit is reset when the transmit buffer is loaded. This bit is set after an MPSC reset.

D3  Carrier Detect—This bit contains the state of the CD pin at the time of the last change of any of the External/Status bits (CD, CTS, Sync/Hunt, Break/Abort, or Tx Underrun/EOM). Any change of state of the CD pin causes the CD bit to be latched and causes an External/Status interrupt. This bit indicates current state of the CD pin immediately following a Reset External/Status Interrupt command.

D4  SYNDET—In asynchronous modes, the operation of this bit is similar to the CD status bit, except that it shows the state of the SYNDET input. Any High-to-Low transition on the SYNDET pin sets this bit, and causes an External/Status interrupt (if enabled). The Reset External/Status Interrupt command is issued to clear the interrupt. A Low-to-High transition clears this bit and sets the External/Status interrupt. When the External/Status interrupt is set by the change in state of any other input or condition, this bit shows the inverted state of the SYNDET pin at time of the change. This bit must be read immediately following a Reset External/Status Interrupt command to read the current state of the SYNDET input.

D5  Clear to Send—this bit contains the inverted state of the CTS pin at the time of the last change of any of the External/Status bits (CD, CTS, Sync/Hunt, Break/Abort, or Tx Underrun/EOM). Any change of state of the CTS pin causes the CTS bit to be latched and causes an External/Status interrupt. This bit indicates the inverse of the current state of the CTS pin immediately following a Reset External/Status Interrupt command.

D7  Break—in the Asynchronous Receive mode, this bit is set when a Break sequence (null character plus framing error) is detected in the data stream. The External/Status interrupt, if enabled, is set when break is detected. The interrupt service routine must issue the Reset External/Status Interrupt command (WR0, Command 2) to the break detection logic so the Break sequence termination can be recognized.
Read Register 1 (RR1):

The Break bit is reset when the termination of the Break sequence is detected in the incoming data stream. The termination of the Break sequence also causes the External/Status interrupt to be set. The Reset External/Status Interrupt command must be issued to enable the break detection logic to look for the next Break sequence. A single, extraneous null character is present in the receiver after the termination of a break; it should be read and discarded.

**D0** All sent—this bit is set when all characters have been sent. It is reset when characters are in the transmitter. In synchronous modes, this bit is always set.

**D4** Parity Error—if parity is enabled, this bit is set for received characters whose parity does not match the programmed sense (Even/Odd). This bit is latched. Once an error occurs, it remains set until the Error Reset command is written.

**D5** Receive Overrun Error—this bit indicates that the receive FIFO has been overloaded by the receiver. The last character in the FIFO is overwritten and flagged with this error. Once the overwritten character is read, this error condition is latched until reset by the Error Reset command. If the MPSC is in the “status affects vector” mode, the overrun causes a Special Receive Error Vector.

**D6** Framing Error—in async modes, a one in this bit indicates a receive framing error. It can be reset by issuing an Error Reset command.

**RR2** Channel B

Read Register 2 (RR2):

**D7–D0** Interrupt vector—contains the interrupt vector programmed into WR2. If the “status affects vector” mode is selected, it contains the modified vector. (See WR2.) RR2 contains the modified vector for the highest priority interrupt pending. If no interrupts are pending, the variable bits in the vector are set to one. May be read from Channel B only.
APPENDIX B
MPSC-POLLED TRANSMIT/RECEIVE CHARACTER ROUTINES

MPSC$RX$INIT: procedure (cmd$port,
clock$rate, stop$bits, parity$type, parity$enable,
rx$char$length, rx$enable, auto$enable,
tx$char$length, tx$enable, dtr, brk, rts);

declare cmd$port byte,
clock$rate byte,
stop$bits byte,
parity$type byte,
parity$enable byte,
rx$char$length byte,
rx$enable byte,
auto$enable byte,
rx$char$length byte,
tx$enable byte,
dtr byte,
brk byte,
rts byte;

output(cmd$port)=30H; /* channel reset */
output(cmd$port)=14H; /* point to WR4 */
/* set clock rate, stop bits, and parity information */
output(cmd$port)=shl(clock$rate,6) or shl(stop$bits,2) or shl(parity$type,1)
or parity$enable;
output(cmd$port)=13H; /* point to WR3 */
/* set up receiver parameters */
output(cmd$port)=shl(rx$char$length,6) or rx$enable or shl(auto$enable,5);
output(cmd$port)=15H; /* point to WR5 */
/* set up transmitter parameters */
output(cmd$port)=shl(tx$char$length,5) or shl(tx$enable,3) or shl(dtr,7)
or shl(brk,4) or shl(rts,1);
end MPSC$RX$INIT;
MPSC$POLL$RCV$CHARACTER: procedure(data$port,cmd$port,character$ptr) byte;

declare data$port byte,
    cmd$port byte,
    character$ptr pointer,
    character based character$ptr byte,
    status byte;

declare char$avail literally '1',
    rcv$error literally '70H';

/* wait for input character ready */
while (input(cmd$port) and char$avail) <> 0 do;
/* check for errors in received character */
output(cmd$port)=1;
/* point to RR1 */
if (status=input(cmd$port) and rcv$error) then do;
    character=input(data$port);
/* read character to clear MPSC */
    call RECEIVE$ERROR(cmd$port,status);
/* clear receiver errors */
    return 0;
end;
else do;
    character=input(data$port);
/* good return - character avail */
    return OFFH;
end;
end MPSC$POLL$RCV$CHARACTER;

MPSC$POLL$TRAN$CHARACTER: procedure(data$port,cmd$port,character);

declare data$port byte,
    cmd$port byte,
    character byte;

declare tx$buffer$empty literally '4';
/* wait for transmitter buffer empty */
while not (input(cmd$port) and tx$buffer$empty) do;
/* output character */
output(data$port)=character;
end MPSC$POLL$TRAN$CHARACTER;

RECEIVE$ERROR: procedure(cmd$port,status);

declare cmd$port byte,
    status byte;

output(cmd$port)=3OH; /* error reset */

/* *** other application dependent error processing should be placed here *** */
end RECEIVE$ERROR;
TRANSMIT BUFFER: procedure(buf$ptr,buf$length)

declare
  buf$ptr pointer,
  buf$length byte;

/* set up transmit buffer pointer and buffer length in global variables */
  tx$buffer$ptr=buf$ptr;
  transmit$length=buf$length;

transmit$status=not$complete; /* setup status for not complete */
  output(data$port)=transmit$buffer(0); /* transmit first character */
  transmit$index=1; /* first character transmitted */

/* wait until transmission complete or error detected */
while transmit$status = not$complete do; end;
if transmit$status <> complete then return false;
else return true;
end TRANSMIT$BUFFER;

RECEIVE BUFFER: procedure (buf$ptr,buf$length$ptr);

declare
  buf$ptr pointer,
  buf$length$ptr pointer,
  buf$length based buf$length$ptr byte;

/* set up receive buffer pointer in global variable for interrupt service */
  rx$buffer$ptr=buf$ptr;
  receive$index=0;

receive$status=not$complete; /* set status to not complete */
/* wait until buffer received */
while receive$status = not$complete do; end;
if receive$status = complete then return true;
else return false;
end RECEIVE BUFFER;
APPENDIX C
INTERRUPT-DRIVEN TRANSMIT/RECEIVE SOFTWARE

declare
/* global variables for buffer manipulation */
rx$buffer$ptr pointer, /* pointer to receive buffer */
receive$buffer based rx$buffer$ptr(128) byte,
receive$status byte initial(0), /* indicates receive buffer status */
receive$index byte, /* current index into receive buffer */
receive$length byte, /* length of final receive buffer */

tx$buffer$ptr pointer, /* pointer to transmit buffer */
transmit$buffer based tx$buffer$ptr(128) byte,
transmit$status byte initial(0), /* indicates transmit buffer status */
transmit$index byte, /* current index into transmit buffer */
transmit$length byte, /* length of buffer to be transmitted */

cmd$port literally "43H", 
data$port literally "41H", 
a$cmd$port literally "42H", 
b$cmd$port literally "43H", 
linefeed literally "0AH", 
not$complete literally "0", 
complete literally "OFFH", 
overrun literally "1", 

channel$reset literally "18H", 
error$reset literally "30H", 
reset$ext$status literally "10H";
MPSC$INT$INIT: procedure (clock$rate, stop$bits, parity$type, parity$enable,
    rx$char$length, rx$enable, auto$enable,
    tx$char$length, tx$enable, dtr, brk, rts,
    ext$en, tx$en, rx$en, stat$aff$vector,
    config, priority, vector$int$mode, int$vector);

declare
    clock$rate byte,        /* 2-bit code for clock rate divisor */
    stop$bits byte,        /* 2-bit code for number of stop bits */
    parity$type byte,      /* 1-bit parity type */
    parity$enable byte,    /* 1-bit parity enable */
    rx$char$length byte,   /* 2-bit receive character length */
    rx$enable byte,        /* 1-bit receiver enable */
    auto$enable byte,      /* 1-bit auto enable flag */
    tx$char$length byte,   /* 2-bit transmit character length */
    tx$enable byte,        /* 1-bit transmitter enable */
    dtr byte,              /* 1-bit status of DTR pin */
    brk byte,              /* 1-bit data link break enable */
    rts byte,              /* 1-bit status of RTS pin */
    ext$en byte,           /* 1-bit external/status enable */
    rx$en byte,            /* 2-bit Rx interrupt enable */
    stat$aff$vector byte,  /* 1-bit status affects vector flag */
    config byte,           /* 2-bit system config - int/DMA */
    priority byte,         /* 1-bit priority flag */
    vector$int$mode byte,  /* 3-bit interrupt mode code */
    int$vector byte;       /* 8-bit interrupt type code */

output(b$cmd$port)=channel$reset;  /* channel reset */
output(b$cmd$port)=14H;               /* point to WR4 */
/* set clock rate, stop bits, and parity information */
output(b$cmd$port)=shl(clock$rate,6) or shl(stop$bits,2) or shl(parity$type,1)
    or parity$enable;
output(b$cmd$port)=13H;               /* point to WR3 */
/* set up receiver parameters */
output(b$cmd$port)=shl(rx$char$length,6) or rx$enable or shl(auto$enable,5);
output(b$cmd$port)=15H;               /* point to WR5 */
/* set up transmitter parameters */
output(b$cmd$port)=shl(tx$char$length,5) or shl(tx$enable,3) or shl(dtr,7)
    or shl(brk,4) or shl(rts,1);
output(b$cmd$port)=12H;               /* point to WR2 */
/* set up interrupt vector */
output(b$cmd$port)=int$vector;
output(a$cmd$port)=12H;               /* point to WR2, channel A */
/* set up interrupt modes */
output(a$cmd$port)=shl(vector$int$mode,3) or shl(priority,2) or config;
output(b$cmd$port)=11H;               /* point to WR1 */
/* set up interrupt enables */
output(b$cmd$port)=shl(rx$en,3) or shl(stat$aff$vector,2) or shl(tx$en,1)
    or ext$en;
end MPSC$INT$INIT;
AP-134

MPSC$RECEIVES$CHARACTER$INT: procedure interrupt 22H:

/* ignore input if no open buffer */
if receive$status <> not$complete then return;

/* check for receive buffer overrun */
if receive$index = 128
then receive$status=overrun
else do;
/* read character from MPSC and place in buffer - note that the
parity of the character must be masked off during this step if
the character is less than 8 bits (e.g., ASCII) */
receive$buffer(receive$index),character=input(data$port) and 7FH;
receive$index=receive$index+1; /* update receive buffer index */
/* check for line feed to end line */
if character = line$feed
then do; receive$length=receive$index; receive$status=complete; end;
end;
end MPSC$RECEIVES$CHARACTER$INT;

MPSC$TRANSMIT$CHARACTER$INT: procedure interrupt 20H:

/* check for more characters to transfer */
if transmit$index < transmit$length
then do;
/* write next character from buffer to MPSC */
output(data$port)=transmit$buffer(transmit$index);
transmit$index=transmit$index+1; /* update transmit buffer index */
end;
else transmit$status=complete:
end MPSC$TRANSMIT$CHARACTER$INT;

RECEIVE$ERROR$INT: procedure interrupt 23H:

declare

  temp   byte;  /* temporary character storage */
output(cmd$port)=1; /* point to RRI */
receive$status=input(cmd$port);  /* discard character */
temp=input(data$port);
output(cmd$port)=error$reset; /* send error reset */
/* *** other application dependent error processing should be placed here *** */
end RECEIVE$ERROR$INT;

EXTERNAL$STATUS$CHANGES$INT: procedure interrupt 21H:

transmit$status=input(cmd$port)  /* input status change information */
output(cmd$port)=reset$ext$status;
/* *** other application dependent error processing should be placed here *** */
end EXTERNAL$STATUS$CHANGES$INT;
APPENDIX D
APPLICATION EXAMPLE USING SDK-86

This application example shows the 8274 in a simple iAPX-86/88 system. The 8274 controls two separate asynchronous channels using its internal interrupt controller to request all data transfers. The 8274 driver software is described which transmits and receives data buffers provided by the CPU. Also, status registers are maintained in system memory to allow the CPU to monitor progress of the buffers and error conditions.

THE HARDWARE INTERFACE

Nothing could be easier than the hardware design of an interrupt-driven 8274 system. Simply connect the data bus lines, a few bus control lines, supply a timing clock for baud rate and, voila, it's done! For this example, the ubiquitous SDK-86 is used as the host CPU system. The 8274 interface is constructed on the wire-wrap area provided. While discussing the hardware interface, please refer to Diagram 1.

Placing the 8274 on the lower 8 bits of the 8086 data bus allows byte-wide data transfers at even I/O addresses. For simplicity, the 8274's CS input is generated by combining the M/IO select line with address line A7 via a 7432. This places the 8274 address range in multiple spots within the 8086 I/O address space. (While fine for this example, a more complete address decoding is recommended for actual prototype systems.) The 8086's A1 and A2 address lines are connected to the A0 and A1 8274 register select inputs respectively. Although other port assignments are possible because of the overlapping address spaces, the following I/O port assignments are used in this example:

<table>
<thead>
<tr>
<th>Port Function</th>
<th>I/O Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data channel A</td>
<td>0000H</td>
</tr>
<tr>
<td>Command/status A</td>
<td>0002H</td>
</tr>
<tr>
<td>Data channel B</td>
<td>0004H</td>
</tr>
<tr>
<td>Command/status B</td>
<td>0006H</td>
</tr>
</tbody>
</table>

To connect the 8274’s interrupt controller into the system an inverter and pull-up resistor are needed to convert the 8274’s active-low, interrupt-request output, INT, into the correct polarity for the 8086’s INTR interrupt input. The 8274 recognizes interrupt-acknowledge bus cycles by connecting the INTA (INTerrupt Acknowledge) lines of the 8274 and 8086 together.

The 8274 Read and Write lines directly connect to the respective 8086 lines. The RESET line requires an inverter. The system clock for the 8274 is provided by the PCLK (peripheral clock) output of the 8284A clock generator.

On the 8274’s serial side, traditional 1488 and 1489 RS-232 drivers and receivers are used for the serial interface. The onboard baud rate generator supplies the channel baud rate timing. In this example, both sides of both channels operate at the same baud rate although this certainly is not a requirement. (On the SDK-86, the baud rate selection is hard-wired thru jumpers. A more flexible approach would be to incorporate an 8253 Programmable Interval Timer to allow software-configurable baud rate selection.)

That's all there is to it. This hardware interface is completely general-purpose and supports all of the 8274 features except the DMA data transfer mode which requires an external DMA controller. Now let's look at the software interface.

SOFTWARE INTERFACE

In this example, it is assumed that the 8086 has better things to do rather than continuously run a serial channel. Presenting the software as a group of callable procedures lets the designer include them in the main body of another program. The interrupt-driven data transfers give the effect that the serial channels are handled in the background while the main program is executing in the foreground. There are five basic procedures: a serial channel initialization routine and buffer handling routines for the transmit and receive data buffers of each channel. Appendix D-1 shows the entire software listing. Listing line numbers are referenced as each major routing is discussed.

The channel initialization routine (INITIAL 8274), starting with line #203, simply sets each channel into a particular operating mode by loading the command registers of the 8274. In normal operation, once these registers are loaded, they are rarely changed. (Although this example assumes a simple asynchronous operating mode, the concept is easily extended for the byte- and bit-synchronous modes.)
(For detailed description on SDK-86, refer to SDK-86 MCS-86 System Design Kit Assembly Manual.)
The channel operating modes are contained in two tables starting with line #163. As the 8274 has only one command register per channel, the remaining seven registers are loaded indirectly through the WR0 (Write Register 0) register. The first byte of each table entry is the register pointer value which is loaded into WR0 and the second byte is the value for that particular register.

The indicated modes set the 8274 for asynchronous operation with data characters 8 bits long, no parity, and 2 stop bits. An X16 baud rate clock is assumed. Also selected is the “interrupt on all RX character” mode with a variable interrupt vector compatible with the 8086/8088. The transmitters are enabled and all model control lines are put in their active state.

In addition to initializing the 8274, this routine also sets up the appropriate interrupt vectors. The 8086 assumes the first 1K bytes of memory contain up to 256 separate interrupt vectors. On the SDK-86 the initial 2K bytes of memory is RAM and therefore must be initialized with the appropriate vectors. (In a prototype system, this initial memory is probably ROM, thus the vector set-up is not needed.) The 8274 supplies up to eight different interrupt vectors. These vectors are developed from internal conditions such as data requests, status changes, or error conditions for each channel. The initialization routine arbitrarily assumes that the initial 8274 vector corresponds to 8086 vector location 80H (memory location 200H). This choice is arbitrary since the 8274 initial vector location is programmable.
Finally, the initialization routine sets up the status and flag in RAM. The meaning and use of these locations are discussed later.

Following the initialization routine are those for the transmit commands (starting with line #268). These commands assume that the host CPU has initialized the publicly declared variables for the transmit buffer pointer, TX_POINTER_CHx, and the buffer length, TX_LENGTH_CHx. The transmit command routines simply clear the transmitter empty flag, TX_EMPTY_CHx, and load the first character of the buffer into the transmitter. It is necessary to load the first character in this manner since transmitter interrupts are generated only when the 8274's transmit data buffer becomes empty. It is the act of becoming empty which generates the interrupt not simply the buffer being empty, thus the transmitter needs one character to start.

The host CPU can monitor the transmitter empty flag, TX_EMPTY_CHx, in order to determine when transmission of the buffer is complete. Obviously, the CPU should only call the command routine after first checking that the empty flag is set.

After returning to the main program, all transmitter data transfers are handled via the transmitter-interrupt service routines starting at lines #360 and #443. These routines start by issuing an End-Of-Interrupt command to the 8274. (This command resets the internal-interrupt controller logic of the 8274 for this particular vector and opens the logic for other internal interrupt requests. The routines next check the length count. If the buffer is completely transmitted, the transmitter empty flag, TXEMPTY_CHx, is set and a command is issued to the 8274 to reset its interrupt line. Assuming that the buffer is not completely transmitted, the next character is output to the transmitter. In either case, an interrupt return is executed to return to the main CPU program.

The receiver commands start at line #314. Like the transmit commands, it is assumed that the CPU has initialized the receive-buffer-pointer public variable, RX_POINTER_CHx. This variable points to the first location in an empty receive buffer. The command routines clear the receiver ready flag, RX_READY_CHx, and then set the receiver enable bit in the 8274 WR3 register. With the receiver now enabled, any received characters are placed in the receive buffer using interrupt-driven data transfers.

The received data service routines, starting at lines #402 and #485, simply place the received character in the buffer after first issuing the EOI command. The character is then compared to an ASCII CR. An ASCII CR causes the routine to set the receiver ready flag, RX READY_CHx, and to disable the receiver. The CPU can interrogate this flag to determine when the buffer contains a new line of data. The receive buffer pointer, RX_POINTER_CHx, points to the last received character and the receive counter, RX_COUNTER_CHx, contains the length.

That completes our discussion of the command routines and their associated interrupt service routines. Although not used by the commands, two additional service routines are included for completeness. These routines handle the error and status-change interrupt vectors.

The error service routines, starting at lines #427 and #510, are vectored to if a special receive condition is detected by the 8274. These special receive conditions include parity, receiver overrun, and framing errors. When this vector is generated, the error condition is indicated in RR1 (Read Register 1). The error service routine issues an EOI command, reads RR1 and places it in the ERROR_MSG_CHx variable, and then issues a reset error command to the 8274. The CPU can monitor the error message location to detect error conditions. The designer, of course, can supply his own error service routine.

Similarly, the status-change routines (starting lines #386 and #469) are initiated by a change in the modem-control status lines CTS/, CD/, or SYND/. (Note that WR2 bit 0 controls whether the 8274 generates interrupts based upon changes in these lines. Our WR2 parameter is such that the 8274 is programmed to ignore changes for these inputs.) The service routines simply read RR0, place its contents in the STATUS_MSG_CHx variable and then issue a reset external status command. Read Register 0 contains the state of the modem inputs at the point of the last change.

Well, that's it. This application example has presented useful, albeit very simple, routines showing how the 8274 might be used to transmit and receive buffers using an asynchronous serial format. Extensions for byte- or bit-synchronous formats would require no hardware changes due to the highly programmable nature of the 8274's serial formats.
8274 APPLICATION BRIEF PROGRAM

**APPLICATION BRIEF**

**PROGRAM**

**ISIS-II**

**Twelve (12)**

**8274**

**APPLICATION BRIEF**

**OBJECT MODULE PLACED IN** F1 ENHANCED ODB

**ASSEMBLER INVOKED BY** ASM86 F1 ASMCB SRC

<table>
<thead>
<tr>
<th>LOC</th>
<th>GBT</th>
<th>LINE</th>
<th>SOURCE</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td>8274 APPLICATION BRIEF PROGRAM</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td></td>
<td>THE 8274 IS INITIALIZED FOR SIMPLE ASYNCHRONOUS SERIAL</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td></td>
<td>FORMAT AND VECTOR INTERRUPT-DRIVEN DATA TRANSFERS</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td></td>
<td>THE INITIAL PROTOCOL ALLOWS THE B896'S INTERRUPT</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td></td>
<td>VECTOR TABLE FROM THE CODE SEGMENT INTO LOW RAM ON THE</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td></td>
<td>80K-86 THE TRANSMITTER AND RECEIVER ARE LEFT ENABLED</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>13</td>
<td></td>
<td>FOR TRANSMIT, THE CPU PASSES IN MEMORY THE POINTER OF A</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td></td>
<td>BUFFER TO TRANSMIT AND THE BYTE LENGTH OF THE BUFFER</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td></td>
<td>THE DATA TRANSFER PROCEEDS USING INTERRUPT-DRIVEN TRANSFERS</td>
<td></td>
</tr>
<tr>
<td>16</td>
<td></td>
<td>A STATUS BIT IN MEMORY IS SET WHEN IF BUFFERS IS EMPTY</td>
<td></td>
</tr>
<tr>
<td>17</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>18</td>
<td></td>
<td>FOR RECEIVE, THE CPU PASSES THE POINTER OF A BUFFER TO FILL</td>
<td></td>
</tr>
<tr>
<td>19</td>
<td></td>
<td>THE BUFFER IS FILLED UNTIL A &quot;CLR&quot; CHARACTER IS RECEIVED</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td></td>
<td>A STATUS BIT IS SET AND THE CPU MAY READ THE RX POINTER TO</td>
<td></td>
</tr>
<tr>
<td>21</td>
<td></td>
<td>DETERMINE THE LOCATION OF THE LAST CHARACTER</td>
<td></td>
</tr>
<tr>
<td>22</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>23</td>
<td></td>
<td>ALL ROUTINES ARE ASSUMED TO EXIST IN THE SAME CODE SEGMENT</td>
<td></td>
</tr>
<tr>
<td>24</td>
<td></td>
<td>CALL'S TO THE SERVICE ROUTINES ARE ASSUMED TO BE &quot;SHORT&quot; OR</td>
<td></td>
</tr>
<tr>
<td>25</td>
<td></td>
<td>INTRASEGMENT (ONLY THE RETURN ADDRESS IP IS ON THE STACK)</td>
<td></td>
</tr>
<tr>
<td>26</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>27</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>28</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>29</td>
<td></td>
<td>B</td>
<td></td>
</tr>
<tr>
<td>30</td>
<td></td>
<td>BAD PROGRAM BRIEF</td>
<td></td>
</tr>
</tbody>
</table>

**210311-24**
PUBLIC DECLARATIONS FOR COMMAND ROUTINES

PUBLIC INITIALIZE, initialization routine
PUBLIC TXCOMMAND.CH, TX BUFFER COMMAND CHANNEL B
PUBLIC TXCOMMAND.CH, TX BUFFER COMMAND CHANNEL A
PUBLIC RXCOMMAND.CH, RX BUFFER COMMAND CHANNEL B
PUBLIC RXCOMMAND.CH, RX BUFFER COMMAND CHANNEL A

PUBLIC DECLARATIONS FOR STATUS VARIABLES

PUBLIC RXREADY.CH, RX READY FLAG CH
PUBLIC RXREADY.CH, RX READY FLAG CH
PUBLIC TXEMPTY.CH, TX EMPTY FLAG CH
PUBLIC RXEMPTY.CH, RX EMPTY FLAG CH
PUBLIC RXCOUNT.CH, RX BUFFER COUNTER CH
PUBLIC RXBUFFER.CH, RX BUFFER COUNTER CH
PUBLIC ERROR.MSG.CH, ERROR FLAG CH
PUBLIC ERROR.MSG.CH, ERROR FLAG CH
PUBLIC STATUS.MSG.CH, STATUS FLAG CH
PUBLIC STATUS.MSG.CH, STATUS FLAG CH

PUBLIC DECLARATIONS FOR VARIABLES PASSED TO THE TRANSMIT

PUBLIC TCPPTR.CH, TX BUFFER POINTER FOR CH
PUBLIC TCPTR.CH, TX BUFFER POINTER FOR CH
PUBLIC TXLEN.CH, TX LENGTH OF BUFFER FOR CH
PUBLIC TCPTR.CH, TX BUFFER POINTER FOR CH
PUBLIC RXPOINTER.CH, RX BUFFER POINTER FOR CH
PUBLIC RXPOINTER.CH, RX BUFFER POINTER FOR CH

PUBLIC PORT ASSIGNMENTS

DATA_PORT.CH, EQU 0, DATA I/O PORT
COMMAND_PORT.CH, EQU 2, COMMAND PORT
STATUS_PORT.CH, EQU COMMAND_PORT.CH, STATUS PORT

CHANNEL B PORT ASSIGNMENTS

DATA_PORT.CH, EQU 4, DATA I/O PORT
COMMAND_PORT.CH, EQU 6, COMMAND PORT
STATUS_PORT.CH, EQU COMMAND_PORT.CH, STATUS PORT

MISC SYSTEM VARIABLES

CP.CH, EQU 80H, ASCII OR CHARACTER CODE
INT_TABLE.BASE, EQU 200H, INT VECTOR BASE ADDRESS
CODE.START, EQU 500H, START LOCATION FOR CODE

DATA SEGMENT

DATA SEGMENT

210311-25
LOC. OBJ. | LINE | SOURCE
--- | --- | ---
91 | VECTOR INTERRUPT TABLE - ASSUME INITIAL 8274 INTERRUPT
92 | VECTOR IS NUMBER 80 (180H) FOR EACH VECTOR. THE TABLE
93 | CONTAINS START LOCATION AND CODE SEGMENT REGISTER VALUE
94 | THE TABLE IS LONGED FROM PROM
95
0200 | 96 | ORG INIT.TABLE.BASE
97
0200 0000 | 98 | TX.VECTR.CHB DB 0 | .TX INTERRUPT VECTOR FOR CHB
0200 0000 | 99 | TX.SS.CHB DB 0
100
0204 0000 | 101 | STS.VECTR.CHB DB 0 | .STATUS INTERRUPT VECTOR FOR CHB
0204 0000 | 102 | STS.SS.CHB DB 0
103
0208 0000 | 104 | RX.VECTR.CHB DB 0 | .RX INTERRUPT VECTOR FOR CHB
0208 0000 | 105 | RX.SS.CHB DB 0
106
020C 0000 | 107 | ERR.VECTR.CHB DB 0 | .ERROR INTERRUPT VECTOR FOR CHB
020C 0000 | 108 | ERR.SS.CHB DB 0
109
0210 0000 | 110 | TX.VECTR.CHB DB 0 | .TX INTERRUPT VECTOR FOR CHA
0210 0000 | 111 | TX.SS.CHB DB 0
112
0214 0000 | 113 | STS.VECTR.CHB DB 0 | .STATUS INTERRUPT VECTOR FOR CHA
0214 0000 | 114 | STS.SS.CHB DB 0
115
0218 0000 | 116 | RX.VECTR.CHB DB 0 | .RX INTERRUPT VECTOR FOR CHA
0218 0000 | 117 | RX.SS.CHB DB 0
118
021C 0000 | 119 | ERR.VECTR.CHB DB 0 | .ERROR INTERRUPT VECTOR FOR CHA
021C 0000 | 120 | ERR.SS.CHB DB 0
121
0220 0000 | 122 | MISC PAM LOCATIONS FOR CHANNEL STATUS AND POINTERS
123
0220 0000 | 124 | .CHANNEL B POINTERS AND STATUS
125
0220 0000 | 126 | TX.POINTER.CHB DB 0 | .TX BUFFER POINTER FOR CHB
0220 0000 | 127 | TX.LENGTH.CHB DB 0 | .TX BUFFER LENGTH FOR CHB
0224 0000 | 128 | RX.POINTER.CHB DB 0 | .RX BUFFER POINTER FOR CHB
0224 0000 | 129 | RX.LENGTH.CHB DB 0 | .RX BUFFER LENGTH FOR CHB
0226 0000 | 130 | TX.EMPTY.CHB DB 0 | .TX DONE FLAG
0228 0000 | 131 | RX.REDY.CHB DB 0 | READY FLAG = 1 IF OR CHB RECEIVED- ELSE 0
022A 0000 | 132 | STATUS.MSG.CHB DB 0 | .STATUS CHANGE MESSAGE
022A 0000 | 133 | ERROR.MSG.CHB DB 0 | .ERROR STATUS LOCATION = 0 IF NO ERROR
134
022C 0000 | 135 | CHANNEL A POINTERS AND STATUS
136
022C 0000 | 137 | TX.POINTER.CHB DB 0 | .TX BUFFER POINTER FOR CHA
022C 0000 | 138 | TX.LENGTH.CHB DB 0 | .TX BUFFER LENGTH FOR CHA
0230 0000 | 139 | RX.POINTER.CHB DB 0 | .RX BUFFER POINTER FOR CHA
0230 0000 | 140 | RX.LENGTH.CHB DB 0 | .RX BUFFER LENGTH FOR CHA
0232 0000 | 141 | TX.EMPTY.CHB DB 0 | .TX DONE FLAG
0234 0000 | 142 | RX.REDY.CHB DB 0 | READY FLAG = 1 IF OR CHA RECEIVED- ELSE 0
0236 0000 | 143 | STATUS.MSG.CHB DB 0 | .STATUS CHANGE MESSAGE
0238 0000 | 144 | ERROR.MSG.CHB DB 0 | .ERROR STATUS LOCATION = 0 IF NO ERROR
145
146 | DATA ENDS
147
148 | REPEAT

210311-26
MS-DOS MACRO ASSEMBLER
ASYNCH

LOC OBJ LINE SOURCE

----
149
150 ABC SEGMENT
151 ASSUME CS ABC,DS DATA:ES DATA
152 ORG CODE:START
153
154******************************************************************************
155 *
156 *
157 *
158 ******************************************************************************
159
160 CHANNEL B PARAMETERS
161
162 .WPL - INTERRUPT ON ALL RX CHAR. VARIABLE INT VECTOR, TX INT ENABLE
163 CMSTAR DB 1:16H
164
165
166 .W3 - RX 8 BITS/CHAR. RX DISABLE
167 DB 3:00H
168
169 .W4 - X16 CLOCK: 2 STOP BITS: NO PARITY
170 DB 4:40H
171
172 .W5 AND W7 NOT REQUIRED FOR ASYN
173 DB B:00H
174
175 CHANNEL A PARAMETERS
176
177 .W1 - INTERRUPT ON ALL RX CHAR. TX INT ENABLE
178 CMSTAR DB 1:12H
179
180 .W2 - VECTORED INTERRUPT FOR BBS
181 DB 2:30H
182
183 .W3 - RX 8 BITS/CHAR. RX DISABLE
184 DB 3:00H
185
186 .W4 - X16 CLOCK: 2 STOP BITS: NO PARITY
187 DB 4:40H
188
189 .W5 AND W7 NOT REQUIRED FOR ASYN
190 DB B.00H
191
192 +1 REJECT

210311-27
LOC 0B1

LINE 121

SOURCE

191 - START OF COMMAND ROUTINES

193

195 *

196 * INITIALIZATION COMMAND FOR THE 8274 - THE 8274 *

197 * IS SETUP ACCORDING TO THE PARAMETERS STORED IN *

198 * FROM ABOVE STARTING AT 85548 FOR CHANNEL B AND *

199 * COUNTER FOR CHANNEL A *

200 *

201

202

0516

204 INITIALIZE 8274

0518 C70600000006

205 MOV TLVECTOR,CXB OFFSET VMTHINE - TX DATA VECTOR CHB

0522 C70600425086

206 MOV TLCS,CHB CS

0528 C0BE0002

207 MOV STSVECTOR,CXB OFFSET STAINB - STATUS VECTOR CHB

052C C70600894086

208 MOV STS,CS,CHB CS

0532 C0BE0002

209 MOV RXVECTOR,CXB OFFSET RYVNE - RX DATA VECTOR CHB

053C C70601077086

210 MOV RXCS,CHB CS

0542 C08E0002

211 MOV ERFVECTOR,CXB OFFSET ERRINB - ERROR VECTOR CHB

054C C08E0002

212 MOV RXCS,CHB CS

0554 C706011B086

213 MOV RXVECTOR,CXB OFFSET RXDATA VECTOR

0558 C08E0002

214 MOV TLCS,CHB CS

055C C706012B086

215 MOV RXVECTOR,CXB OFFSET RXDATA VECTOR

055F C08E0002

216 MOV STS,CS,CHB CS

0563 C706014C8086

217 MOV RXVECTOR,CXB OFFSET RXDATA VECTOR

0565 C7060150F086

218 MOV RXCS,CHB CS

0567 C70601601086

219 MOV ERFVECTOR,CXB OFFSET ERRINB - ERROR VECTOR CHB

0569 C08E0002

220 MOV RXCS,CHB CS

056A

221 .COPY SETUP TABLE PARAMETERS INTO 8274

056B BF8000

222 MOV DI, OFFSET CMDSTP, ; INITIALIZE CHB

056D B89000

223 MOV DL, COMMAND, PORT, CHB

056E EB6000

224 CALL SETUP ; COPY CHB PARAMETERS

0571 BF8000

225 MOV DL, OFFSET CMDSTP, ; INITIALIZE CHB

0574 B49000

226 MOV DL, COMMAND, PORT, CHB

0577 EB2000

227 CALL SETUP ; COPY CHB PARAMETERS

0578

228 . INITIALIZE STATUS BYTES AND FLAGS

057B 880000

229 MOV RX, 8

057C 828802

230 MOV ERRNSLCS, AL, ; CLEAR ERROR FLAG CHB

057D 827702

231 MOV ERRNSLCS, AL, ; CLEAR ERROR FLAG CHB

057F 822802

232 MOV STATUSLCS, AL, ; CLEAR STATUS FLAG CHB

0583 822802

233 MOV STATUSLCS, AL, ; CLEAR STATUS FLAG CHB

0585 822802

234 MOV RXCOUNT, RX, ; CLEAR RX COUNTER CHB

0587 822802

235 MOV RXCOUNT, RX, ; CLEAR RX COUNTER CHB

058F B001

236 MOV RX, 1

0591 822802

237 MOV RXCOUNT, RX, ; SET RX DONE FLAG CHB

0593 822802

238 MOV RXCOUNT, RX, ; SET RX DONE FLAG CHB

0595 822802

239 MOV RXCOUNT, RX, ; SET RX DONE FLAG CHB

0597 822802

240 MOV RXCOUNT, RX, ; SET RX DONE FLAG CHB

0599 822802

241 MOV RXCOUNT, RX, ; ENABLE INTERRUPTS

059C C0

242 RET ; RETURN - DONE WITH SETUP

059E 8900

243 SETUP MOV AL [DI] ; PARAMETER COPYING ROUTINE

05A1 3000

244 CMP AL, 8

05A3 7404

245 JE DONE

210311-28

2-377
AP-134

LOC GBT
LINE SOURCE

251 OUT DX, AL : OUTPUT PARAMETER

252 INC DI : POINT AT NEXT PARAMETER

253 JMP SETUP : GO LONE IT

254 DONE RET : DONE - 5Q RETURN

255 * RETURN

256 * TX CHANNEL B COMMAND ROUTINE - ROUTINE IS CALLED TO

257 * TRANSMIT A BUFFER THE BUFFER STARTING ADDRESS,

258 * TX_POINTER.CH.B AND THE BUFFER LENGTH, TX_LENGTH.CH.B.

259 * MUST BE INITIALIZED BY THE CALLING PROGRAM

260 * BOTH ITEMS ARE WORD VARIABLES

261 *

262 *******************************************************

263 *******************************************************

264 * TX CHANNEL A COMMAND ROUTINE - ROUTINE IS CALLED TO

265 * TRANSMIT A BUFFER THE BUFFER STARTING ADDRESS,

266 * TX_POINTER.CH.A AND THE BUFFER LENGTH, TX_LENGTH.CH.A.

267 * MUST BE INITIALIZED BY THE CALLING PROGRAM

268 * BOTH ITEMS ARE WORD VARIABLES

269 *

270 *******************************************************

271 *******************************************************

272 *******************************************************

273 *******************************************************

274 *******************************************************

275 *******************************************************

276 *******************************************************

277 *******************************************************

278 *******************************************************

279 *******************************************************

280 *******************************************************

281 *******************************************************

282 *******************************************************

283 *******************************************************

284 *******************************************************

285 *******************************************************

286 *******************************************************

287 *******************************************************

288 *******************************************************

289 *******************************************************

290 *******************************************************

291 *******************************************************

292 *******************************************************

293 *******************************************************

294 *******************************************************

295 *******************************************************

296 *******************************************************

297 *******************************************************

298 *******************************************************

299 *******************************************************

300 *******************************************************

301 *******************************************************

302 *******************************************************

303 *******************************************************

304 *******************************************************

305 *******************************************************

306 *******************************************************

307 *******************************************************

308 *******************************************************

309 *******************************************************

310 *******************************************************

311 *******************************************************

312 *******************************************************

313 *******************************************************

314 *******************************************************

315 *******************************************************

316 *******************************************************

317 *******************************************************

318 *******************************************************

319 *******************************************************

320 *******************************************************

321 *******************************************************

322 *******************************************************

323 *******************************************************

324 *******************************************************

325 *******************************************************

326 *******************************************************

327 *******************************************************

328 *******************************************************

329 *******************************************************

330 *******************************************************

331 *******************************************************

332 *******************************************************

333 *******************************************************

334 *******************************************************

335 *******************************************************

336 *******************************************************

337 *******************************************************

338 *******************************************************

339 *******************************************************

340 *******************************************************

341 *******************************************************

342 *******************************************************

343 *******************************************************

344 *******************************************************

345 *******************************************************

346 *******************************************************

347 *******************************************************

348 *******************************************************

349 *******************************************************

350 *******************************************************

351 *******************************************************

352 *******************************************************

353 *******************************************************

354 *******************************************************

355 *******************************************************

356 *******************************************************

357 *******************************************************

358 *******************************************************

359 *******************************************************

360 *******************************************************

361 *******************************************************

362 *******************************************************

363 *******************************************************

364 *******************************************************

365 *******************************************************

366 *******************************************************

367 *******************************************************

368 *******************************************************

369 *******************************************************

370 *******************************************************

371 *******************************************************

372 *******************************************************

373 *******************************************************

374 *******************************************************

375 *******************************************************

376 *******************************************************

377 *******************************************************

378 *******************************************************

20311-29
LOC 081  LINE  SOURCE

111  *  
112  *  
113  *  
114  *RX.COMMAND.CH
115  .PUSH AX  .SAVE REGISTERS
116  .PUSH DX
117  .MOV RX.RECV.CH, 0  .CLEAR RX.RECV FLAG
118  .MOV RX.COUNT.CH, 0  .CLEAR RX COUNTER
119  .MOV RX.COMMAND.PORT.CH, 0  .POINT AT COMMAND PORT
120  .MOV AL, 0  .SET UP FOR MP
121  .OUT DX, AL
122  .POP DX
123  .POP AX
124  .RET  .RETURN

128  *  
129  *RX.COMMAND.CH
130  .PUSH AX  .SAVE REGISTERS
131  .PUSH DX
132  .MOV RX.RECV.CH, 0  .CLEAR RX.RECV FLAG
133  .MOV RX.COUNT.CH, 0  .CLEAR RX COUNTER
134  .MOV RX.COMMAND.PORT.CH, 0  .POINT AT COMMAND PORT
135  .MOV AL, 0  .SET UP FOR MP
136  .OUT DX, AL
137  .POP DX
138  .POP AX
139  .RET  .RETURN

358  *REJECT
359  .START OF INTERRUPT SERVICE ROUTINES
360  .CHANNEL B TRANSMIT DATA SERVICE ROUTINE

0062  .XMIT.CH  .PUSH DX  .SAVE REGISTERS
0063  .PUSH AX
0064  .PUSH DX
0065  .CLI
0066  .FEND  .SEND EOI COMMAND TO 8274
0067  .INC TL.POINTER.CH  .POINT TO NEXT CHARACTER
0068  .DEC TL.LENGTH.CH  .DEC LENGTH COUNTER
0069  .JE TLXIB  .TEST IF DONE
0070  .BND  .DATA.PORT.CH  .NOT DONE - GET NEXT CHARACTER
0071  .DEC TL.POINTER.CH
0072  .MOV AL, [DI]  .PUT CHARACTER IN AL
0073  .OUT DX, AL  .OUTPUT IT TO 8274

210311-30

2-379
<table>
<thead>
<tr>
<th>LOC OBJ</th>
<th>LINE</th>
<th>SOURCE</th>
</tr>
</thead>
<tbody>
<tr>
<td>431</td>
<td>MOV AL, 1</td>
<td>POINT AT RX</td>
</tr>
<tr>
<td>432</td>
<td>OUT DX, AL</td>
<td>READ RX</td>
</tr>
<tr>
<td>433</td>
<td>IN AL, DX</td>
<td>READ RX</td>
</tr>
<tr>
<td>434</td>
<td>MOV AL, CB</td>
<td>SAVE IT IN ERROR FLAG</td>
</tr>
<tr>
<td>435</td>
<td>MOV AL, 2BH</td>
<td>SEND RESET ERROR COMMAND TO 8274</td>
</tr>
<tr>
<td>436</td>
<td>OUT DX, AL</td>
<td></td>
</tr>
<tr>
<td>437</td>
<td>POP AX</td>
<td>RESTORE REGISTERS</td>
</tr>
<tr>
<td>438</td>
<td>POP DX</td>
<td></td>
</tr>
<tr>
<td>439</td>
<td>IRET</td>
<td>RETURN TO FOREGROUND</td>
</tr>
</tbody>
</table>

- **CHANNEL A TRANSFER DATA SERVICE ROUTINE**

<table>
<thead>
<tr>
<th>LOC OBJ</th>
<th>LINE</th>
<th>SOURCE</th>
</tr>
</thead>
<tbody>
<tr>
<td>441</td>
<td>MOV DX, DATA_PORT.CHAR</td>
<td>NOT DONE - GET NEXT CHARACTER</td>
</tr>
<tr>
<td>442</td>
<td>MOV DX, DATA_PORT.CHAR</td>
<td>NOT DONE - GET NEXT CHARACTER</td>
</tr>
<tr>
<td>443</td>
<td>MOV DX, DATA_PORT.CHAR</td>
<td>NOT DONE - GET NEXT CHARACTER</td>
</tr>
<tr>
<td>444</td>
<td>MOV DX, DATA_PORT.CHAR</td>
<td>NOT DONE - GET NEXT CHARACTER</td>
</tr>
<tr>
<td>445</td>
<td>MOV DX, DATA_PORT.CHAR</td>
<td>NOT DONE - GET NEXT CHARACTER</td>
</tr>
<tr>
<td>446</td>
<td>MOV DX, DATA_PORT.CHAR</td>
<td>NOT DONE - GET NEXT CHARACTER</td>
</tr>
<tr>
<td>447</td>
<td>MOV DX, DATA_PORT.CHAR</td>
<td>NOT DONE - GET NEXT CHARACTER</td>
</tr>
<tr>
<td>448</td>
<td>MOV DX, DATA_PORT.CHAR</td>
<td>NOT DONE - GET NEXT CHARACTER</td>
</tr>
<tr>
<td>449</td>
<td>MOV AX, DATA_PORT.CHAR</td>
<td>NOT DONE - GET NEXT CHARACTER</td>
</tr>
<tr>
<td>450</td>
<td>MOV AX, DATA_PORT.CHAR</td>
<td>NOT DONE - GET NEXT CHARACTER</td>
</tr>
<tr>
<td>451</td>
<td>MOV AX, DATA_PORT.CHAR</td>
<td>NOT DONE - GET NEXT CHARACTER</td>
</tr>
<tr>
<td>452</td>
<td>MOV AX, DATA_PORT.CHAR</td>
<td>NOT DONE - GET NEXT CHARACTER</td>
</tr>
<tr>
<td>453</td>
<td>MOV AX, DATA_PORT.CHAR</td>
<td>NOT DONE - GET NEXT CHARACTER</td>
</tr>
<tr>
<td>454</td>
<td>MOV AX, DATA_PORT.CHAR</td>
<td>NOT DONE - GET NEXT CHARACTER</td>
</tr>
<tr>
<td>455</td>
<td>MOV AX, DATA_PORT.CHAR</td>
<td>NOT DONE - GET NEXT CHARACTER</td>
</tr>
<tr>
<td>456</td>
<td>MOV AX, DATA_PORT.CHAR</td>
<td>NOT DONE - GET NEXT CHARACTER</td>
</tr>
</tbody>
</table>

  - **CHANNEL A STATUS CHANGE SERVICE ROUTINE**

<table>
<thead>
<tr>
<th>LOC OBJ</th>
<th>LINE</th>
<th>SOURCE</th>
</tr>
</thead>
<tbody>
<tr>
<td>461</td>
<td>MOV DX, DATA_PORT.CHAR</td>
<td>ALL CHARACTERS HAVE BEEN SENT</td>
</tr>
<tr>
<td>462</td>
<td>MOV DX, DATA_PORT.CHAR</td>
<td>ALL CHARACTERS HAVE BEEN SENT</td>
</tr>
<tr>
<td>463</td>
<td>MOV AX, DATA_PORT.CHAR</td>
<td>ALL CHARACTERS HAVE BEEN SENT</td>
</tr>
<tr>
<td>464</td>
<td>MOV AX, DATA_PORT.CHAR</td>
<td>ALL CHARACTERS HAVE BEEN SENT</td>
</tr>
<tr>
<td>465</td>
<td>MOV AX, DATA_PORT.CHAR</td>
<td>ALL CHARACTERS HAVE BEEN SENT</td>
</tr>
<tr>
<td>466</td>
<td>MOV AX, DATA_PORT.CHAR</td>
<td>ALL CHARACTERS HAVE BEEN SENT</td>
</tr>
</tbody>
</table>

  - **CHANNEL A RECEIVE DATA SERVICE ROUTINE**

<table>
<thead>
<tr>
<th>LOC OBJ</th>
<th>LINE</th>
<th>SOURCE</th>
</tr>
</thead>
<tbody>
<tr>
<td>469</td>
<td>STRAIA</td>
<td>SAVE REGISTERS</td>
</tr>
<tr>
<td>470</td>
<td>PUSH DX</td>
<td>SAVE REGISTERS</td>
</tr>
<tr>
<td>471</td>
<td>PUSH DX</td>
<td>SAVE REGISTERS</td>
</tr>
<tr>
<td>472</td>
<td>PUSH AX</td>
<td>SAVE REGISTERS</td>
</tr>
<tr>
<td>473</td>
<td>POP AX</td>
<td>SAVE REGISTERS</td>
</tr>
<tr>
<td>474</td>
<td>POP AX</td>
<td>SAVE REGISTERS</td>
</tr>
</tbody>
</table>

  - **CHANNEL A RECEIVE DATA SERVICE ROUTINE**

<table>
<thead>
<tr>
<th>LOC OBJ</th>
<th>LINE</th>
<th>SOURCE</th>
</tr>
</thead>
<tbody>
<tr>
<td>480</td>
<td>MOV AX, DATA_PORT.CHAR</td>
<td>SEND EOI COMMAND TO 8274</td>
</tr>
<tr>
<td>481</td>
<td>MOV AX, DATA_PORT.CHAR</td>
<td>SEND EOI COMMAND TO 8274</td>
</tr>
<tr>
<td>482</td>
<td>MOV AX, DATA_PORT.CHAR</td>
<td>SEND EOI COMMAND TO 8274</td>
</tr>
<tr>
<td>483</td>
<td>MOV AX, DATA_PORT.CHAR</td>
<td>SEND EOI COMMAND TO 8274</td>
</tr>
<tr>
<td>484</td>
<td>MOV AX, DATA_PORT.CHAR</td>
<td>SEND EOI COMMAND TO 8274</td>
</tr>
<tr>
<td>485</td>
<td>MOV AX, DATA_PORT.CHAR</td>
<td>SEND EOI COMMAND TO 8274</td>
</tr>
<tr>
<td>486</td>
<td>MOV AX, DATA_PORT.CHAR</td>
<td>SEND EOI COMMAND TO 8274</td>
</tr>
<tr>
<td>487</td>
<td>MOV AX, DATA_PORT.CHAR</td>
<td>SEND EOI COMMAND TO 8274</td>
</tr>
<tr>
<td>488</td>
<td>MOV AX, DATA_PORT.CHAR</td>
<td>SEND EOI COMMAND TO 8274</td>
</tr>
<tr>
<td>489</td>
<td>MOV AX, DATA_PORT.CHAR</td>
<td>SEND EOI COMMAND TO 8274</td>
</tr>
</tbody>
</table>

  - **CHANNEL A RECEIVE DATA SERVICE ROUTINE**

<table>
<thead>
<tr>
<th>LOC OBJ</th>
<th>LINE</th>
<th>SOURCE</th>
</tr>
</thead>
<tbody>
<tr>
<td>490</td>
<td>MOV AX, DATA_PORT.CHAR</td>
<td>SEND EOI COMMAND TO 8274</td>
</tr>
<tr>
<td>491</td>
<td>MOV AX, DATA_PORT.CHAR</td>
<td>SEND EOI COMMAND TO 8274</td>
</tr>
<tr>
<td>492</td>
<td>MOV AX, DATA_PORT.CHAR</td>
<td>SEND EOI COMMAND TO 8274</td>
</tr>
<tr>
<td>493</td>
<td>MOV AX, DATA_PORT.CHAR</td>
<td>SEND EOI COMMAND TO 8274</td>
</tr>
<tr>
<td>494</td>
<td>MOV AX, DATA_PORT.CHAR</td>
<td>SEND EOI COMMAND TO 8274</td>
</tr>
<tr>
<td>495</td>
<td>MOV AX, DATA_PORT.CHAR</td>
<td>SEND EOI COMMAND TO 8274</td>
</tr>
</tbody>
</table>

  - **CHANNEL A RECEIVE DATA SERVICE ROUTINE**

<table>
<thead>
<tr>
<th>LOC OBJ</th>
<th>LINE</th>
<th>SOURCE</th>
</tr>
</thead>
<tbody>
<tr>
<td>496</td>
<td>MOV AX, DATA_PORT.CHAR</td>
<td>SEND EOI COMMAND TO 8274</td>
</tr>
<tr>
<td>497</td>
<td>MOV AX, DATA_PORT.CHAR</td>
<td>SEND EOI COMMAND TO 8274</td>
</tr>
<tr>
<td>498</td>
<td>MOV AX, DATA_PORT.CHAR</td>
<td>SEND EOI COMMAND TO 8274</td>
</tr>
<tr>
<td>499</td>
<td>MOV AX, DATA_PORT.CHAR</td>
<td>SEND EOI COMMAND TO 8274</td>
</tr>
<tr>
<td>500</td>
<td>MOV AX, DATA_PORT.CHAR</td>
<td>SEND EOI COMMAND TO 8274</td>
</tr>
<tr>
<td>501</td>
<td>MOV AX, DATA_PORT.CHAR</td>
<td>SEND EOI COMMAND TO 8274</td>
</tr>
</tbody>
</table>

The document appears to be a technical specification or documentation related to a specific service routine, involving data transfer and status handling, possibly for a microcontroller or similar device. The code examples are in assembly language, indicating that the document is technical in nature and aimed at engineers or technicians working with embedded systems.
<table>
<thead>
<tr>
<th>LOC</th>
<th>OBJ</th>
<th>LINE</th>
<th>SOURCE</th>
</tr>
</thead>
<tbody>
<tr>
<td>0604</td>
<td>EC</td>
<td>491</td>
<td>IN AL, DX ; READ CHARACTER</td>
</tr>
<tr>
<td>0605</td>
<td>B805</td>
<td>492</td>
<td>MOV [DI], AL ; STORE IN BUFFER</td>
</tr>
<tr>
<td>0606</td>
<td>FF0C</td>
<td>493</td>
<td>INC RX POINTER, Char ; BUMP THE BUFFER POINTER</td>
</tr>
<tr>
<td>0601</td>
<td>FFAC</td>
<td>494</td>
<td>INC RX COUNT, Char ; BUMP THE COUNTER</td>
</tr>
<tr>
<td>0603</td>
<td>3C0D</td>
<td>495</td>
<td>CMP AL, OR CHAR ; TEST IF LAST CHARACTER TO BE RECEIVED</td>
</tr>
<tr>
<td>0607</td>
<td>750E</td>
<td>496</td>
<td>JNE RIA</td>
</tr>
<tr>
<td>0609</td>
<td>C005</td>
<td>497</td>
<td>MOV RL, RX-Char ; 1; YES, SET READY FLAG</td>
</tr>
<tr>
<td>060A</td>
<td>809B</td>
<td>498</td>
<td>MOV DX, COMMAND-PORT, Char ; POINT AT COMMAND PORT</td>
</tr>
<tr>
<td>060F</td>
<td>B800</td>
<td>499</td>
<td>MOV AL, J ; POINT AT J #</td>
</tr>
<tr>
<td>0613</td>
<td>EE</td>
<td>500</td>
<td>OUT DX, AL</td>
</tr>
<tr>
<td>0614</td>
<td>80C0</td>
<td>501</td>
<td>MOV AL, RX CHAR ; DISABLE RX</td>
</tr>
<tr>
<td>0615</td>
<td>EE</td>
<td>502</td>
<td>OUT DX, AL</td>
</tr>
<tr>
<td>0617</td>
<td>58</td>
<td>503</td>
<td>RIA POP AX ; EITHER WAY, RESTORE REGISTERS</td>
</tr>
<tr>
<td>0618</td>
<td>5F</td>
<td>504</td>
<td>POP DI</td>
</tr>
<tr>
<td>0619</td>
<td>5A</td>
<td>505</td>
<td>POP DX</td>
</tr>
<tr>
<td>061A</td>
<td>CF</td>
<td>506</td>
<td>RET ; RETURN TO FOREGROUND</td>
</tr>
<tr>
<td>061B</td>
<td>5A1</td>
<td>507</td>
<td>CHANNEL A ERROR SERVICE ROUTINE</td>
</tr>
<tr>
<td>061C</td>
<td>5A2</td>
<td>508</td>
<td>ERROR PUSH DX ; SAVE REGISTERS</td>
</tr>
<tr>
<td>061D</td>
<td>5A3</td>
<td>509</td>
<td>PUSH AX</td>
</tr>
<tr>
<td>061E</td>
<td>E900</td>
<td>510</td>
<td>CALL EDI ; SEND EDI COMMAND TO 8274</td>
</tr>
<tr>
<td>0620</td>
<td>B800</td>
<td>511</td>
<td>MOV DX, COMMAND PORT CHAR</td>
</tr>
<tr>
<td>0624</td>
<td>B801</td>
<td>512</td>
<td>MOV AL, J ; POINT AT J #</td>
</tr>
<tr>
<td>0625</td>
<td>EE</td>
<td>513</td>
<td>OUT DX, AL</td>
</tr>
<tr>
<td>0626</td>
<td>56</td>
<td>514</td>
<td>IN AL, DX ; READ RX</td>
</tr>
<tr>
<td>0627</td>
<td>AE70</td>
<td>515</td>
<td>MOV ERROR RX CHAR, AL ; SAVE IT IN ERROR FLAG</td>
</tr>
<tr>
<td>0628</td>
<td>AE78</td>
<td>516</td>
<td>MOV AL, RX CHAR ; SEND RESET ERROR COMMAND TO 8274</td>
</tr>
<tr>
<td>0629</td>
<td>EE</td>
<td>517</td>
<td>OUT DX, AL</td>
</tr>
<tr>
<td>062A</td>
<td>5A</td>
<td>518</td>
<td>POP AX ; RESTORE REGISTERS</td>
</tr>
<tr>
<td>062B</td>
<td>59</td>
<td>519</td>
<td>POP DX</td>
</tr>
<tr>
<td>062C</td>
<td>CF</td>
<td>520</td>
<td>RET ; RETURN TO FOREGROUND</td>
</tr>
<tr>
<td>062D</td>
<td>5A2</td>
<td>521</td>
<td>EDI PUSH AX ; SAVE REGISTERS</td>
</tr>
<tr>
<td>062E</td>
<td>5A3</td>
<td>522</td>
<td>PUSH DX</td>
</tr>
<tr>
<td>062F</td>
<td>B800</td>
<td>523</td>
<td>MOV DX, COMMAND PORT CHAR ; ALWAYS FOR CHANNEL A ** **</td>
</tr>
<tr>
<td>0632</td>
<td>58</td>
<td>524</td>
<td>MOV AL, RX CHAR</td>
</tr>
<tr>
<td>0633</td>
<td>EE</td>
<td>525</td>
<td>OUT DX, AL</td>
</tr>
<tr>
<td>0634</td>
<td>5A</td>
<td>526</td>
<td>POP AX</td>
</tr>
<tr>
<td>0635</td>
<td>59</td>
<td>527</td>
<td>POP DX</td>
</tr>
<tr>
<td>0636</td>
<td>CF</td>
<td>528</td>
<td>RET</td>
</tr>
<tr>
<td>0637</td>
<td>5A2</td>
<td>529</td>
<td>ERROR ENDS</td>
</tr>
<tr>
<td>576</td>
<td>END</td>
<td></td>
<td>ASSEMBLY COMPLETE; NO ERRORS FOUND</td>
</tr>
</tbody>
</table>

**REFERENCES**

Synchronous Communication with the 8274 Multiple Protocol Serial Controller

SIKANDAR NAQVI
APPLICATION ENGINEER
INTRODUCTION

The INTEL 8274 is a Multi-Protocol Serial Controller, capable of handling both asynchronous and synchronous communication protocols. Its programmable features allow it to be configured in various operating modes, providing optimization to given data communication applications.

This application note describes the features of the MPSC in Synchronous Communication applications only. It is strongly recommended that the reader read the 8274 Data Sheet and Application Note AP134 “Asynchronous Communication with the 8274 Multi-Protocol Serial Controller” before reading this Application Note. This Application note assumes that the reader is familiar with the basic structure of the MPSC, in terms of pin description, Read/Write registers and asynchronous communication with the 8274. Appendix A contains the software listings of the Application Example and Appendix B shows the MPSC Read/Write Registers for quick reference.

The first section of this application note presents an overview of the various synchronous protocols. The second section discusses the block diagram description of the MPSC. This is followed by the description of MPSC interrupt structure and mode of operation in the third and fourth sections. The fifth section describes a hardware/software example, using the INTEL single board computer ISBC88/45 as the hardware vehicle. The sixth section consists of some specialized applications of the MPSC. Finally, in section seven, some useful programming hints are summarized.

SYNCHRONOUS PROTOCOL OVERVIEW

This section presents an overview of various synchronous protocols. The contents of this section are fairly tutorial and may be skipped by the more knowledgeable reader.

Bit Oriented Protocols Overview

Bit oriented protocols have been defined to manage the flow of information on data communication links. One of the most widely known protocols is the one defined by the International Standards Organization: HDLC (High Level Data Link Control). The American Standards Association’s protocol, ADCCP is similar to HDLC. CCITT Recommendation X.25 layer 2 is also an acceptable version of HDLC. Finally, IBM’s SDLC (Synchronous Data Link Control) is also a subset of the HDLC.

In this section, we will concentrate most of our discussion on HDLC. Figure 1 shows a basic HDLC frame format.

A frame consists of five basic fields: Flag, Address, Control, Data and Error Detection. A frame is bounded by flags—opening and closing flags. An address field is 8 bits wide, extendable to 2 or more bytes. The control field is also 8 bits wide, extendable to two bytes. The data field or information field may be any number of bits. The data field may or may not be on an 8-bit boundary. A powerful error detection code called Frame Check Sequence contains the calculated CRC (Cycle Redundancy Code) for all the bits between the flags.

ZERO BIT INSERTION

The flag has a unique binary bit pattern: 7E HEX. To eliminate the possibility of the data field containing a 7E HEX pattern, a bit stuffing technique called Zero Bit Insertion is used. This technique specifies that during transmission, a binary 0 be inserted by the transmitter after any succession of five contiguous binary 1’s. This will ensure that no pattern of 0 1 1 1 1 1 1 0 is ever transmitted between flags. On the receiving side, after receiving the flag, the receiver hardware automatically deletes any 0 following five consecutive 1’s. The 8274 performs zero bit insertion and deletion automatically in the SDLC/HDLC mode. The zero-bit stuffing ensures periodic transitions in the data stream. These transitions are necessary for a phase lock circuit, which may be used at the receiver end to generate a receive clock which is in phase to the received data. The inserted and deleted 0’s are not included in the CRC checking. The address field is used to address a given secondary station. The control field contains the link-level control information which includes implied acknowledgment, supervisory commands and responses, etc. A more detailed discussion of higher level protocol functions is beyond the scope of this application note. Interested readers may refer to the references at the end of this application note.

<table>
<thead>
<tr>
<th>Opening Flag Byte</th>
<th>Address* Field (A)</th>
<th>Control** Field (C)</th>
<th>Data Field</th>
<th>Frame Check Sequence</th>
<th>Closing Flag Byte</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

*Extendable to 2 or More Bytes. **Extendable to 2 Bytes.

Figure 1. HDLC/SDLC Frame Format
The data field may be of any length and content in HDLC. Note that SDLC specifies that data field be a multiple of bytes only. In data communications, it is generally desirable to transmit data which may be of any content. This requires that data field should not contain characters which are defined to assist the transmission protocol (like opening flag 7EH in HDLC/SDLC communications). This property is referred to as "data transparency". In HDLC/SDLC, this code transparency is made possible by Zero Bit Insertion discussed earlier and the bit oriented nature of the protocol.

The last field is the FCS (Frame Check Sequence). The FCS uses the error detecting techniques called Cyclic Redundancy Check. In SDLC/HDLC, the CCITT-CRC must be used.

NON-RETURN TO ZERO INVERTED (NRZI)

NRZI is a method of clock and data encoding that is well suited to the HDLC protocol. It allows HDLC protocols to be used with low cost asynchronous modems. NRZI coding is done at the transmitter to enable clock recovery from the data at the receiver terminal by using standard digital phase locked loop techniques. NRZI coding specifies that the signal condition does not change for transmitting a 1, while a 0 causes a change of state. NRZI coding ensures that an active data line will have transition at least every 5-bit times (recall Zero Bit Insertion), while contiguous 0's will cause a change of state. Thus, ZBI and NRZI encoding makes it possible for a phase lock circuit at the receiver end to derive a receive clock (from received data) which is synchronized to the received data and at the same time ensure data transparency.

Byte Synchronous Communication

As the name implies, Byte Synchronous Communication is a synchronous communication protocol which means that the transmitting station is synchronized to the receiving station through the recognition of a special sync character or characters. Two examples of Byte Synchronous protocol are the IBM Bisync and Monosync. Bisync has two starting sync characters per message while monosync has only one sync character. For the sake of brevity, we will only discuss Bisync here. All the discussion is valid for Monosync also. Any exceptions will be noted. Figure 2 shows a typical Bisync message format.

The Bisync protocol is defined for half duplex communication between two or more stations over point to point or multipoint communication lines. Special characters control link access, transmission of data and termination of transmission operations for the system. A detailed discussion of these special control characters (SYN, ENQ, STX, ITB, ETB, ETX, DLE, SOH, ACK0, ACK1, WACK, NAK and EOT, etc) is beyond the scope of this Application Note. Readers interested in more detailed discussion are directed to the references listed at the end of this Application Note.

As shown in Figure 2, each message is preceded by two sync characters. Since the sync characters are defined at the beginning of the message only, the transmitter must insert fill characters (sync) in order to maintain synchronization with the receiver when no data is being transmitted.

TRANSPARENT TRANSMISSION

Bisync protocol requires special control characters to maintain the communication link over the line. If the data is EBCDIC encoded, then transparency is ensured by the fact that the field will not contain any of the bisync control characters. However, if data does not conform to standard character encoding techniques, transparency in bisync is achieved by inserting a special character DLE (Data Link Escape) before and after a string of characters which are to be transmitted transparently. This ensures that any data characters which match any of the special characters are not confused for special characters. An example of a transparent block is shown in Figure 3.

In a transparent mode, it is required that the CRC (BCC) is not performed on special characters. Later on, we will show how the 8274 can be used to achieve transparent transmission in Bisync mode.
BLOCK DIAGRAM

This section discusses the block diagram view of the 8274. The CPU interface and serial interface is discussed separately. This will be followed by a hardware example in the fifth section, which will show how to interface the 8274 with the Intel CPU 8088. The 8274 block diagram is shown in Figure 4.

CPU Interface

The CPU interface to the system interface logic block utilizes the A0, A1, CS, RD and WR inputs to communicate with the internal registers of the 8274. Figure 5 shows the address of the internal registers. The DMA interface is achieved by utilizing DMA request lines for each channel: TxDRQA, TxDRQB, RxDRQA, RxDRQB. Note that TxDRQB and RxDRQB become IPO and IPI respectively in non-DMA mode. IPI is the Interrupt Priority Input and IPO is the Interrupt Priority Output. These two pins can be used for connecting multiple MPSCs in a daisy chain. If the Wait Mode is programmed, then TxRDQA and RxRDQA pins become RDYB and RDYA pins. These pins can be wire-OR’ed and are usually hooked up to the CPU RDY line to synchronize the CPU for block transfers. The INT pin is activated whenever the MPSC requires CPU attention. The INTA may be used to utilize the powerful vectored mode feature of the 8274. Detailed discussion on these subjects will be done later in this Application Note. The RESET pin may be used for hardware reset while the clock is required to click the internal logic on the MPSC.
Serial Interface

On the serial side, there are two completely independent channels: Channel A and Channel B. Each channel consists of a transmitter block, receiver block and a set of read/write registers which are used to initialize the device. In addition, a control logic block provides the modem interface pins. Channel B serial interface logic is a mirror image of Channel A serial interface logic, except for one exception: there is only one pin for RTSB and SYNDETB.

At a given time, this pin is either RTSB or SYNDETB. This mode is programmable through one of the internal registers on the MPSC.

Transmit and Receive Data Path

Figure 6 shows a block diagram for transmit and receive data path. Without describing each block on the diagram, a brief discussion of the block diagram will be presented here.

TRANSMIT DATA PATH

The transmit data is transferred to the twenty-bit serial shift register. The twenty bits are needed to store two bytes of sync characters in bisync mode. The last three bits of the shift register are used to indicate to the internal control logic that the current data byte has been shifted out of the shift register. The transmit data in the

<table>
<thead>
<tr>
<th>CS</th>
<th>A1</th>
<th>A0</th>
<th>Read Operation</th>
<th>Write Operation</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>CHA DATA READ</td>
<td>CHA DATA WRITE</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>CHA STATUS REGISTER (RR0,RR1)</td>
<td>CHA COMMAND/PARAMETER (WR0–WR7)</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>CHB DATA READ</td>
<td>CHB DATA WRITE</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>CHB STATUS REGISTER (RR0,RR1,RR2)</td>
<td>CHB COMMAND/PARAMETER (WR0–WR7)</td>
</tr>
<tr>
<td>1</td>
<td>X</td>
<td>X</td>
<td>HIGH Z</td>
<td>HIGH Z</td>
</tr>
</tbody>
</table>

Figure 5. Bus Interface

Figure 6. Transmit and Receive Data Path
transmit shift register is shifted out through a two bit delay onto the TxData line. This two bit delay is used to synchronize the internal shift clock with the external transmit clock. The data in the shift register is also presented to zero bit insertion logic which inserts a zero after sensing five contiguous ones in the data stream. In parallel to all this activity, the CRC-generator is computing CRC on the transmitted data and appends the frame with CRC bytes at the end of the data transmission.

**RECEIVE DATA PATH**

The received data is passed through a one bit delay before it is presented for flag/sync comparison. In bisync mode, after the synchronization is achieved, the incoming data bypasses the sync register and enters directly into the three bit buffer on its way to receive shift register. In SDLC mode, the incoming data always passes through the sync register where the data pattern is continuously monitored for contiguous ones for the

![Diagram](image-url)
zero deletion logic. The data then enters the three bit buffer and the receive shift register. From the receive shift register, the data is transferred to the three byte deep FIFO. The data is transferred to the top of the FIFO at the chip clock rate (not the receiver clock). It takes three chip clock/periods to transfer data from the serial shift register to the top of the FIFO. The three bit deep Receive Error FIFO shifts any error condition which may have occurred during a frame reception. While all this is happening, the CRC checker is checking the CRC on the incoming data. The computed CRC is checked with the CRC bytes attached to the incoming frame and an error generated under a no-check condition. Note that the bisync data is presented to the CRC checker with an 8-bit delay. This is necessary to achieve transparency in bisync mode as will be shown later in this Application Note.

**MULTI-PROTOCOL SERIAL CONTROLLER (MPSC) INTERRUPT STRUCTURE**

The MPSC offers a very powerful interrupt structure, which helps in responding to an interrupt condition very quickly. There are multiple sources of interrupts within the MPSC. However, the MPSC resolves the priority between various interrupting sources and interrupts the CPU for service through the interrupt line. This section presents a comprehensive discussion of all the 8247 interrupts and the priority resolution between these interrupts.

All the sources of interrupts on the 8274 can be grouped into three distinct categories. (See Figure 7.)

1. Receive Interrupts
2. Transmit Interrupts
3. External/Status Interrupts.

An internal interrupt priority structure sets the priority between the interrupts. There are two programmable options available on the MPSC. The priority is set by WR2A, D2 (Figure 8).

**RECEIVE INTERRUPT ON RECEIVE CHARACTER**

A receive interrupt is generated when a character is received by the MPSC. However, as will be discussed later, this is a programmable feature on the MPSC. A Rx character available interrupt is generated by the MPSC after the receive character has been assembled by the MPSC. It may be noted that in DMA transfer mode too, a receive interrupt on the first receive character should be programmed. In SDLC mode, if address search mode has been programmed, this interrupt will be generated only after a valid address match has occurred. In bisync mode, this interrupt is generated on receipt of a character after at least two valid sync characters. In monosync mode, a character followed after at least a single valid sync character will generate this interrupt. An interrupt on first receive character signifies the beginning of a valid frame. An end of the frame is characterized by an “End of Frame” Interrupt (RR1:D7).* This bit (RR1:D7) is set in SDLC/HDLC mode only and signifies that a valid ending flag (7EH) has been received. This bit gets reset either by an “Error Reset” command (WR8:D5D4D3 = 110) or upon reception of the first character of the next frame. In multiframe reception, on receiving the interrupt at the “End of Frame” the CPU may issue an Error Reset command which will reset the interrupt. In DMA mode, the interrupt on first receive character is accompanied by a RxDRQ (Receiver DMA request) on the appropriate channel. At the end of the frame, an End of Frame interrupt is generated. The CPU may use this interrupt to jump into a routine which may redefine the receive buffer for the next incoming frame.

*NOTE: RR1:D7 is bit D7 in Read Register 1.

**SPECIAL RECEIVE CONDITION INTERRUPTS**

So far, we have assumed that the reception is error free. But this is not ‘typical’ in most real life applications. Any error condition during a frame reception generates yet another interrupt—special receive condition interrupt. There are four different error conditions which can generate this interrupt.

(i) Parity error
(ii) Receive Overrun error
(iii) Framing error
(iv) End of Frame

(i) Parity error: Parity error is encountered in asynchronous (start-stop bits) and in bisync/monosync protocols. Both odd or even parity can be programmed. A parity error in a received byte will generate a special receive condition interrupt and sets bit 4 in RR1.
(ii) Receive Overrun error: If the CPU or the DMA controller (in DMA mode) fails to read a received character within three byte times after the received character interrupt (or DMA request) was generated, the receiver buffer will overflow and this will generate a special receive condition interrupt and sets bit 5 in RRI.

(iii) Framing error: In asynchronous mode, a framing error will generate a special receive interrupt and set bit D6 in RRI. This bit is not latched and is updated on the next received character.

(iv) End of frame: This interrupt is encountered in SDLC/HDLC mode only. When the MPSC receives the closing flag, it generates the special receive condition interrupt and sets bit D7 in RRI.

All the special receive condition interrupts may be reset by issuing an Error Reset Command.

CRC Error: In SDLC/HDLC and synchronous modes, a CRC error is indicated by bit D6 in RRI. When used to check CRC error, this bit is normally set until a correct CRC match is obtained which resets this bit. After receiving a frame, the CPU must read this bit (RRI:D6) to determine if a valid CRC check had occurred. It may be noted that a CRC error does not generate an interrupt.

It may also be pointed out that in SDLC/HDLC mode, receive DMA requests are disabled by a special receive condition and can only be re-enabled by issuing an Error Reset Command.

Transmit Interrupt

A transmit buffer empty generates a transmit interrupt. This has been discussed earlier under "Transmit in Interrupt Mode" and it would be sufficient to note here that a transmit buffer empty interrupt is generated only when the transmit buffer gets empty—assuming it had a data character loaded into it earlier. This is why on starting a frame transmission, the first data character is loaded by the CPU without a transmit empty interrupt (or DMA request in DMA mode). After this character is loaded into the serial shift register, the buffer becomes empty, and an interrupt (or DMA request) is generated. This interrupt is reset by a "Reset Tx Interrupt/DMA Pending" command (WR0: D5 D4 D3 = 101).

External/Status Interrupt

Continuing our discussion on transmit interrupt, if the transmit buffer is empty and the transmit serial shift register also becomes empty (due to the data character shifted out of the MPSC), a transmit under-run interrupt will be generated. This interrupt may be reset by "Reset External/Status Interrupt" command (WR0: D5 D4 D3 = 101).

The External Status Interrupt can be caused by five different conditions:

(i) CD Transition
(ii) CTS Transition
(iii) Sync/Hunt Transition
(iv) Tx under-run/EOM condition
(v) Break/Abort Detection.

CD, CTS TRANSITION

Any transition on these inputs on the serial interface will generate an External/Status interrupt and set the corresponding bits in status register RR0. This interrupt will also be generated in DMA as well as in Wait Mode. In order to find out the state of the CTS or CD pins before the transition had occurred, RR0 must be read before issuing a Reset External/Status Command through WR0. A read of RR0 after the Reset External/Status Command will give the condition of CTS or CD pins after the transition had occurred. Note that bit D5 in RR0 gives the complement of the state of CTS while D3 in RR0 reflects the actual state of the CD pin.

SYNC HUNT TRANSITION

Any transition of the SYNDET input generates an interrupt. However, sync input has different functions in different modes and we shall discuss them individually.

SDLC Mode

In SDLC mode, the SYNDET pin is an output. Status register RR1, D4 contains the state of the SYNDET pin. The Enter Hunt Mode initially sets this bit in R0. An opening flag in a received SDLC frame resets this bit and generates an external status interrupt. Every time the receiver is enabled or the Enter Hunt Code Command is issued, an external status interrupt will be generated on receiving a valid flag followed by a valid address/data character. This interrupt may be reset by the "Reset External/Status Interrupt" command.

External SYNC Mode

The MPSC can be programmed into External Sync Mode by setting WR4, D5 D4 = 11. The SYNDET pin is an input in this case and must be held high until an external character synchronization is established. However, the External Sync mode is enabled by the Enter Hunt Mode control bit (WR3: D4). A high at the SYNDET pin holds the Sync/Hunt bit (RR0,D4) in the reset state. When external synchronization is established, SYNDET must be driven low on second rising
edge of RxC after the rising edge of RxC on which the last bit of sync character was received. This high to low transition sets the Sync/Hunt bit and generates an external/status interrupt, which must be reset by the Reset External/Status command. If the SYNDET input goes high again, another External Status Interrupt is generated, which may be cleared by Reset External/Status command.

Mono-Sync/Bisync Mode
SYNDET pin acts as an output in this case. The Enter Hunt Mode sets the Sync/Hunt bit in R0. Sync/Hunt bit is reset when the MPSC achieves character synchronization. This high to low transition will generate an external status interrupt. The SYNDET pin goes active every time a sync pattern is detected in the data stream. Once again, the external status interrupt may be reset by the Reset External/Status command.

Tx UNDER-RUN/END OF MESSAGE (EOM)
The transmitter logic includes a transmit buffer and a transmit serial shift register. The CPU loads the character into the transmit buffer which is transferred into the transmit shift register to be shifted out of the MPSC. If the transmit buffer gets empty, a transmit buffer empty interrupt is generated (as discussed earlier). However, if the transmit buffer gets empty and the serial shift register gets empty, a transmit under-run condition will be created. This generates an External Status Interrupt and the interrupt can be cleared by the Reset External Status command. The status register RR0, D6 bit is set when the transmitter under-runs. This bit plays an important role in controlling a transmit operation, as will be discussed later in this application note.

BREAK/ABORT DETECTION
In asynchronous mode, bit D7 in RR0 is set when a break condition is detected on the receive data line. This also generates an External/Status interrupt which may be reset by issuing a Reset External/Status Interrupt command to the MPSC. Bit D7 in RR0 is reset when the break condition is terminated on the receive data line and this causes another External/Status interrupt to go generated. Again, a Reset External/Status Interrupt command will reset this interrupt and will enable the break detection logic to look for the next break sequence.

In SDLC Receive Mode, an Abort sequence (seven or more 1's) detection on the receive data line will generate an External/Status interrupt and set RR0,D7. A Reset External/Status command will clear this interrupt. However, a termination of the Abort sequence will generate another interrupt and set RR0,D7 again. Once again, it may be cleared by issuing Reset External/Status Command.

This concludes our discussion on External Status Interrupts.

Interrupt Priority Resolution
The internal interrupt priority between various interrupt sources is resolved by an internal priority logic circuit, according to the priority set in WR2A. We will now discuss the interrupt timings during the priority resolution. Figures 9 and 10 show the timing diagrams for vectored and non-vectored modes.

VECTORED MODE
We shall assume that the MPSC accepted an internal request for an interrupt by activating the internal INT signal. This leads to generating an external interrupt signal on the INT pin. The CPU responds with an interrupt acknowledge (INTA) sequence. The leading edge of the first INTA pulse sets an internal interrupt acknowledge (INTA) sequence. The leading edge of the first INTA pulse sets an internal interrupt acknowledge signal (we will call it Internal INTA). Internal INTA is reset by the high going edge of the third INTA pulse. The MPSC will not accept any internal requests for an interrupt during the period when Internal INTA is active (high). The MPSC resolves the priority during various existing internal interrupt requests during the Interrupt Request Priority Resolve Time, which is defined as the time between the leading edge of the first INTA and the leading edge of the second INTA from the CPU. Once the internal priorities have been resolved, an internal Interrupt-in-Service Latch is set. The external INT is also deactivated when the Interrupt-in-Service Latch is set.

The lower priority interrupt requests are not accepted internally until an EOI (WR0: D5 D4 D3 = 111 Ch. A only) command is issued by the CPU. The EOI command enables the lower priority interrupts. However, a higher priority interrupt request will still be accepted (except during the period when internal INTA is active) even though the Internal-in-Service Latch is set.
Figure 9. 8274 in 8085 Vectored Mode Priority Resolution Time

Figure 10. 8274 Non Vectored Mode Priority Resolve Time
This higher priority request will generate another external INT and will have to be handled by the CPU according to how the CPU is set up. If the CPU is set up to respond to this interrupt, a new INTA cycle will be repeated as discussed earlier. It may also be noted that a transmitter buffer empty and receive character available interrupts are cleared by loading a character into the MPSC and by reading the character received by the MPSC respectively.

NON-VECTORED MODE

Figure 10 shows the timing of interrupt sequence in non-vectored mode. The explanation of non-vectored is similar to the vector mode, except for the following exceptions.

- No internal priority requests are accepted during the time when pointer 2 for Channel B is specified.
- The interrupt request priority resolution time is the time between the leading edge of pointer 2 and leading edge of RD active. It may be pointed out that in non-vectored mode, it is assumed that the status affects vector mode is used to expedite interrupt response.

On getting an interrupt in non-vectored mode, the CPU must read status register RR2 to find out the cause of the interrupt. In order to do so, first a pointer to status register RR2 is specified and then the status read from RR2. It may be noted here that after specifying the pointer, the CPU must read status register RR2 otherwise, no new interrupt requests will be accepted internally.

Just like the vectored mode, no lower internal priority requests are accepted until an EOI command is issued by the CPU. A higher priority request can still interrupt the CPU (except during the priority request inhibit time). It is important to note here that if the CPU does not perform a read operation after specifying the pointer 2 for Channel B, the interrupt request accepted before the pointer 2 was activated will remain valid and no other request (high or low priority) will be accepted internally. In order to complete a correct priority resolution, it is advised that a read operation be done after specifying the pointer 2B.

IPI and IPO

So far, we have ignored the IPI and IPO signals shown in Figures 9 and 10. We may recall that IPI is the Interrupt-Priority-Input to the MPSC. In conjunction with the IPO (Interrupt Priority Output), it is used to daisy chain multiple MPSCs. MPSC daisy chaining will be discussed in detail later in this application note.

EOI Command

The EOI command as explained earlier, enables the lower priority interrupts by resetting the internal In-Service-Latch, which consequently resets the IPO output to a low state. See Figures 9 and 10 for details. Note that before issuing any EOI command, the internal interrupting source must be satisfied otherwise, same source will interrupt again. The Internal Interrupt is the signal which gets reset when the internal interrupting source is satisfied (see Figure 9).

This concludes our discussion on the MPSC Interrupt Structure.

MULTI-PROTOCOL SERIAL CONTROLLER (MPSC) MODES OF OPERATION

The MPSC provides two fully independent channels that may be configured in various modes of operations. Each channel can be configured into full duplex mode and may operate in a mode or protocol different from the other channel. This feature will be very efficient in an application which requires two data link channels operating in different protocols and possibly at different data rates. This section presents a detailed discussion on all the 8274 modes and shows how to configure it into these modes.

Interrupt Driven Mode

In the interrupt mode, all the transmitter and receiver operations are reported to the processor through interrupts. Interrupts are generated by the MPSC whenever it requires service. In the following discussion, we will discuss how to transmit and receive in interrupt driven mode.

TRANSMIT IN INTERRUPT MODE

The MPSC can be configured into interrupt mode by appropriately setting the bits in WR2 A (Write Register 2, Channel A). Figure 11 shows the modes of operation.
We will limit our discussion to SDLC transmit and receive only. However, exceptions for other synchronous protocols will be pointed out. To initiate a frame transmission, the first data character must be loaded from the CPU, in all cases. (DMA Mode too, as you will notice later in this application note). Note that in SDLC mode, this first data character may be the address of the station addressed by the MPSC. The transmit buffer consists of a transmit buffer and a serial shift register. When the character is transferred from the buffer into the serial shift register, an interrupt due to transmit buffer empty is generated. The CPU has one byte time to service this interrupt and load another character into the transmitter buffer. The MPSC will generate an interrupt due to transmit buffer underrun condition if the CPU does not service the Transmit Buffer Empty Interrupt within one byte time.

This process will continue until the CPU is out of any more data characters to be sent. At this point, the CPU does not respond to the interrupt with a character, but simply issues a Reset Tx INT/DMA pending command (WR0: D5 D4 D3 = 101). The MPSC will ultimately underrun, which simply means that both the transmit buffer and transmit shift registers are empty. At this point, flag character (7EH) or CRC byte is loaded into the transmit shift register. This sets the transmit underrun bit in RR0 and generates “Transmit Underrun/EOM” interrupt (RR0: D6 = 1).

You will recall that an SDLC frame has two CRC bytes after the data field. 8274 generates the CRC on all the data that is loaded from the CPU. During initialization, there is a choice of selecting a CRC-16 or CCITT-CRC (WR5: D2). In SDLC/HDLC operation, CCITT-CRC must be selected. We will now see how the CRC gets inserted at the end of the data field. Here we have a choice of having the CRC attached to the data field or sending the frame without the CRC bytes. During transmission, a “Reset Tx Underrun/EOM Latch” command (WR0: D7 D6 D5 = 11) will ensure that at the end of the frame when the transmitter underrun, CRC bytes will be automatically inserted at the end of the data field. If the “Reset Tx Underrun/EOM Latch” command was not issued during the transmission of data characters, no CRC would be inserted and the MPSC will transmit flags (7EH) instead.

However, in case of CRC transmission, the CRC transmission sets the Tx Underrun/EOM bit and generates a Transmitter Underrun/EOM Interrupt as discussed earlier. This will have to be reset in the next frame to ensure CRC insertion in the next frame. It is recommended that Tx Underrun/EOM latch be reset very early in the transmission mode, preferably after loading the first character. It may be noted here that Tx Underrun EOM latch cannot be reset if there is no data in the transmit buffer. This means that at least one character has to be loaded into the MPSC before a “Reset Transmit Underrun/EOM Latch” command will be accepted by the MPSC.

When the transmitter is underrun, an interrupt is generated. This interrupt is generated at the beginning of the CRC transmission, thus giving the user enough time (minimum 22 transmit clock cycles) to issue an Abort command (WR0: D5 D4 D3 = 0 0 1) in case if the transmitted data had an error. The Abort Command will ensure that the MPSC transmits at least eight 1's but less than fourteen 1's before the line reverts to continuous flags. The receiver will scratch this frame because of bad CRC.

However, assuming the transmission was good (no Abort Command issued), after the CRC bytes have been transmitted, closing flag (7EH) is loaded into the transmit buffer. When the flag (7EH) byte is transferred to the serial shift register, a transmit buffer empty interrupt is generated. If another frame has to be transmitted, a new data character has to be loaded into the transmit buffer and the complete transmit sequence repeated. If no more frames are to be transmitted, a “Reset Transmit INT/DMA Pending” command (WR0: D5 D4 D3 = 101) will reset the transmit buffer empty interrupt.

For character oriented protocols (Bisync, Monosync), the same discussion is valid, except that during transmit underrun condition and transmit underrun/EOM bit in set state, instead of flags, filler sync characters are transmitted.

**CRC Generation**

The transmit CRC enable bit (WR5: D0) must be set before loading any data into the MPSC. The CRC generator must be reset to all 1's at the beginning of each frame before CRC computation has begun. The CRC computation starts on the first data character loaded from the CPU and continues until the last data character. The CRC generated is inverted before it is sent on the Tx Data line.

**Transmit Termination**

A successful transmission can be terminated by issuing a “Reset Transmit Interrupt/DMA Pending” command, as discussed earlier. However, the transmitter may be disabled any time during the transmission and the results will be as shown in Figure 12.

**RECEIVE IN INTERRUPT MODE**

The receiver has to be initialized into the appropriate receive mode (see sample program later in this application note). The receiver must be programmed into Hunt Mode (WR3: D4) before it is enabled (WR3: D0). The receiver will remain in the Hunt Mode until a flag (or sync character) is received. While in the SDLC/Bisync/Monosync mode, the receiver does not enter the Hunt Mode unless the Hunt bit (WR3, D4) is set again or the receiver is enabled again.
SDLC Address byte is stored in WR6. A global address (FFH) has been hardwired on the MPSC. In address search mode (WR3: D2 = 1), any frame with address matching with the address in WR6 will be received by the MPSC. Frames with global address (FFH) will also be received, irrespective of the condition of address search mode bit (WR3: D2). In general receive mode (WR3: D2 = 0), all frames will be received.

<table>
<thead>
<tr>
<th>Transmitter Disabled during</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Data Transmission</td>
<td>Tx Data will send idle characters* which will be zero inserted.</td>
</tr>
<tr>
<td>2. CRC Transmission</td>
<td>16 bit transmission, corresponding to 16 bits of CRC will be completed. However, flag bits will be substituted in the CRC field.</td>
</tr>
<tr>
<td>3. Immediately after issuing ABORT command.</td>
<td>Abort will still be transmitted—output will be in the mark state.</td>
</tr>
</tbody>
</table>

Figure 12. Transmitter Disabled During Transmission

*NOTE:
Idle characters are defined as a string of 15 or more contiguous ones.

Since the MPSC only recognizes single byte address field, extended address recognition will have to be done by the CPU on the data passed on by the MPSC. If the first address byte is checked by the MPSC, and the CPU determines that the second address byte does not have the correct address field, it must set the Hunt Mode (WR3: D2 = 1) and the MPSC will start searching for a new address byte preceded by a flag.

Programmable Interrupts

The receiver may be programmed into any one of the four modes. See Figure 13 for details.

<table>
<thead>
<tr>
<th>WR1, D4, D3</th>
<th>Rx Interrupt Mode</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 0</td>
<td>Rx INT/DMA disable</td>
</tr>
<tr>
<td>0 1</td>
<td>Rx INT on first character</td>
</tr>
<tr>
<td>1 0</td>
<td>INT on all Rx characters (Parity affects vector)</td>
</tr>
<tr>
<td>1 1</td>
<td>INT on all Rx characters (Parity does not affect vector)</td>
</tr>
</tbody>
</table>

Figure 13. Receiver Interrupt Modes

All receiver interrupts can be disabled by WR1: D4 D3 = 00. Receiver interrupt on first character is normally used to start a DMA transfer or a block transfer sequence using WAIT to synchronize the data transfer to received or transmitted data.

External Status Interrupts

Any change in CD input or Abort detection in the received data, will generate an interrupt if External Status Interrupt was enabled (WR1: D0).

Special Receive Conditions

The receiver buffer is quadruply buffered. If the CPU fails to respond to “receive character” available interrupt within a period of three byte times (received bytes), the receiver buffer will overflow and generate an interrupt. Finally, at the end of the received frame, an interrupt will be generated when a valid ending flag has been detected.

Receive Character Length

The receive character length (6, 7 or 8 bits/character) may be changed during reception. However, to ensure that the change is effective on the next received character, this must be done fast enough such that the bits specified for the next character have not been assembled.

CRC Checking

The opening flag in the frame resets the receive CRC generator and any field between the opening and closing flag is checked for the CRC. In case of a CRC error, the CRC/Framing Error bit in status register 1 is set (RR1: D6 = 1). Receiver CRC may be disabled/enabled by WR3, D3. The CRC bytes on the received frame are passed on to the CPU just like data, and may be discarded by the CPU.

Receive Terminator

An end of frame is indicated by End of Frame interrupt. The CPU may issue an “Error Reset” command to reset this interrupt.

DMA (Direct Memory Access) Mode

The 8274 can be interfaced directly to the Intel DMA Controllers 8237A, 8257A and Intel I/O Processor 8089. The 8274 can be programmed into DMA mode by setting appropriate bits in WR2A. See Figure 11 for details.
TRANSMIT IN DMA MODE

After initializing the 8274 into the DMA mode, the first character must be loaded from the CPU to start the DMA cycle. When the first data character (may be the address byte in SDLC) is transferred from the transmit buffer to the transmit serial shift register, the transmit buffer gets empty and a transmit DMA request (TxDRQ) is generated for the channel. Just like the interrupt mode, to ensure that the CRC bytes are included in the frame, the transmit under-run/EOM latch must be reset. This should preferably be done after loading the first character from the CPU. The DMA will progress without any CPU intervention. When the DMA controller reaches the terminal count, it will not respond to the DMA request, thus letting the MPSC under-run. This will ensure CRC transmission. However, the under-run condition will generate an interrupt due to the Tx under-run/EOM bit getting set (RR0: D6). The CPU should issue a "Reset TxInt/DRQ pending" command to reset TxDRQ and issue a "Reset External Status" command to reset Tx Under-run/EOM interrupt. Following the CRC transmission, flag (7EH) will be loaded into the transmit buffer. This will also generate the TxDRQ since the transmit buffer is empty following the transmission of the CRC bytes. The CPU may issue a "Reset TxINT/DRQ pending" command to reset the TxDRQ. "Reset TxINT/DRQ pending" command must be issued before setting up the transmit DMA channel on the DMA Controller, otherwise the MPSC will start the DMA transfer immediately after the DMA channel is set up.

RECEIVE IN DMA MODE

The receiver must be programmed in RxINT on first receive character mode (WR1: D4 D3 = 0 1). Upon receiving the first character, which may be the address byte in SDLC, the MPSC generates an interrupt and also generates a Rx DMA Request (Rx DRQ) for the appropriate channel. The CPU has three byte times to service this interrupt (enable the DMA controller, etc.) before the receiver buffer will overflow. It is advisable to initialize the DMA controller before receiving the first character. In case of high bit rates, the CPU will have to service the interrupt very fast in order to avoid receiver over-run.

Once the DMA is enabled, the received data is transferred to the memory under DMA control. Any received error conditions or external status change condition will generate an interrupt as in the interrupt driven mode. The End of Frame is indicated by the End of Frame interrupt which is generated on reception of the closing flag of the SDLC frame. This End of Frame condition also disables the Receive DMA request. The End of Frame interrupt may be reset by issuing an "Error Reset" command to the MPSC. The "Error Reset" command also re-enables the Receive DMA request. It may be noted that the End of Frame condition sets bit D7 in RR1. This bit gets reset by "Error Reset" command. However, End of Frame bit (RR1: D7) can also be reset by the flag of the next incoming frame. For proper operation, Error Reset Command should be issued "after" the End of Frame Bit (RR1: D7) is set. In a more general case, "Error Reset" command should be issued after End of Frame, Receive over-run or Receive parity bit is set in RR1.

Wait Mode

The wait mode is normally used for block transfer by synchronizing the data transfer through the Ready output from the MPSC, which may be connected to the Ready input of the CPU. The mode can be programmed by WR 1, D7 D5 and may be programmed separately and independently on CH A and CH B. The Wait Mode will be operative if the following conditions are satisfied.

(i) Interrupts are enabled.
(ii) Wait Mode is enabled (WR1: D7)
(iii) CS = 0, A1 = 0

The RDY output becomes active when the transmitter buffer is full or receiver buffer is empty. This way the R DY output from the MPSC can be used to extend the CPU read and write cycle by inserting WAIT states. R DY_A or R DY_B are in high impedance state when the corresponding channel is not selected. This makes it possible to connect R DY_A and R DY_B outputs in wired OR configuration. Caution must be exercised here in using the R DY outputs of the MPSC or else the CPU may hang up for indefinite period. For example, let us assume that transmitter buffer is full and R DY_A is active, forcing the CPU into a wait state. If the CTS goes inactive during this period, the R DY_A will remain active for indefinite period and CPU will continue to insert wait states.

Vectored/Non-Vectored Mode

The MPSC is capable of providing an interrupt vector in response to the interrupt acknowledge sequence from the CPU. WR2, CH B contains this vector and the vector can be read in status register RR2. WR2, CH A (bit D5) can program the MPSC in vectored or non-vectored mode. See Figure 14 for details.
In both cases, WR2 may still have the vector stored in it. However, in vectored mode, the MPSC will put the vector on the data bus in response to the INTA (Interrupt Acknowledge) sequence as shown in Figure 15. In non-vectored mode, the MPSC will not respond to the INTA sequence. However, the CPU can read the vector by polling Status Register RR2. WR2A, D4 and D3 can be programmed to respond to 8085 or 8086 INTA sequence. It may be noted here that IPI (Interrupt Priority In) pin on the MPSC must be active for the vector to appear on the data bus.

<table>
<thead>
<tr>
<th>WR2A, D5</th>
<th>Interrupt Mode</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Non-vectored Interrupt</td>
</tr>
<tr>
<td>1</td>
<td>Vectored Interrupt</td>
</tr>
</tbody>
</table>

Figure 14. Vectored Interrupt

<table>
<thead>
<tr>
<th>D5</th>
<th>D4</th>
<th>D3</th>
<th>IPI</th>
<th>Mode</th>
<th>1st INTA</th>
<th>2nd INTA</th>
<th>3rd INTA</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>Non-Vectored</td>
<td>HI-Z</td>
<td>HI-Z</td>
<td>HI-Z</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>8085-1</td>
<td>1100 1101</td>
<td>V7 V6 V5 V4 V3 V2 V1 V0</td>
<td>0000 0000</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>8085-1</td>
<td>1100 1101</td>
<td>HI-Z</td>
<td>HI-Z</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>8085-2</td>
<td>HI-Z</td>
<td>V7 V6 V5 V4 V3 V2 V1 V0</td>
<td>0000 0000</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>8085-2</td>
<td>HI-Z</td>
<td>HI-Z</td>
<td>HI-Z</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>8086</td>
<td>HI-Z</td>
<td>V7 V6 V5 V4 V3 V2 V1 V0</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>8086</td>
<td>HI-Z</td>
<td>HI-Z</td>
<td>HI-Z</td>
</tr>
</tbody>
</table>

Figure 15. MPSC Vectored Interrupts

<table>
<thead>
<tr>
<th>(8085)</th>
<th>V4</th>
<th>V3</th>
<th>V2</th>
<th>V0</th>
<th>Channel</th>
<th>Interrupt Source</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>B</td>
<td>Tx Buffer Empty</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td></td>
<td>EXT/STAT Change</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>RX CHAR Available</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td>Special Rx Condition</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>A</td>
<td>Tx Buffer Empty</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td></td>
<td>EXT/STAT Change</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td></td>
<td>RX CHAR Available</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td>Special Rx Condition</td>
<td></td>
</tr>
</tbody>
</table>

Rx Special Condition: Parity Error, Framing Error, Rx Over-run Error, EOF (SDLC).
EXT/STAT Change: Change in Modem Control Pin Status: CTS, DCD, SYNC, EOM, Break/Abort Detection.

Figure 16. Status Affect Vector Mode
APPLICATION EXAMPLE

This section describes the hardware and software of an 8274/8088 system. The hardware vehicle used is the INTEL Single Board Computer iSBC 88/45—Advanced Communication Controller. The software which exercises the 8274 is written in PLM 86. This example will demonstrate how 8274 can be configured into the SDLC mode and transfer data through DMA control. The hardware example will help the reader configure his hardware and the software examples will help in developing an application software. Most software examples closely approximate real data link controller software in the SDLC communication and may be used with very little modification.

iSBC® 88/45

A brief description of the iSBC 88/45 board will be presented here. For more detailed information on the board and the schematics, refer to Hardware Manual for the iSBC 88/45, Advanced Communication Controller. iSBC 88/45 is an intelligent slave/multimaster communication board based on the 8088 processor, the 8274 and the 8273 SDLC/HDLC controller. Figure 17 shows the functional block diagram of the board. The iSBC 88/45 has the following features.

- 8 MHz processor
- 16K bytes of static RAM (12K dual port)
- Multimaster/Intelligent Slave Multibus Interface
- Nine Interrupt Levels 8259A
- Two serial channels through 8274
- One Serial channel through 8273
- S/W programmable baud rate generator
- Interfaces: RS232, RS422/449, CCITT V.24
- 8237A DMA controller
- Baud Rate to 800K Baud
**INITIALIZE_B274** PROCEDURE PUBLIC,

```
/* INITIALIZE THE B274 FOR SDLC MODE */
/* 1 RESET CHANNEL */
/* 2 EXTERNAL INTERRUPTS ENABLED */
/* 3 NO WAIT */
/* 4 PIN 10 = RTS */
/* 5 NON-VECTORED INTERRUPT-B0B6 MODE */
/* 6 CHANNEL A DMA, CH B INT */
/* 7 TX AND RX = B BITS/CHAR */
/* 9 ADDRESS SEARCH MODE */
/* 10 CD AND CTS AUTO ENABLE */
/* 11 X1 CLOCK */
/* 12 NO PARITY */
/* 13 SDLC/HDLC MODE */
/* 14 RTS AND DTR */
/* 15 CCITT - CRC */
/* 16 TRANSMITTER AND RECEIVER ENABLED */
/* 17 7EH = FLAG */
```

DECLARE C BYTE.

```
/* TABLE TO INITIALIZE THE B274 CHANNEL A AND B */
/* FORMAT IS WRITE REGISTER, REGISTER DATA */
/* INITIALIZE CHANNEL A ONLY */
```

DECLARE TABLE_74_A(*) BYTE DATA

```
(00H, 1AH, /* CHANNEL RESET */
00H, 80H, /* RESET TX CRC */
02H, 11H, /* PIN 10=RTS, A DMA, B INT */
04H, 20H, /* SDLC/HDLC MODE, NO PARITY */
07H, 07EH, /* SDLC FLAG */
01H, 0BH, /* RX DMA ENABLE */
03H, 03BH, /* DTR, RTS, B TX BITS, TX ENABLE */
06H, 55H, /* SDLC CRC, TX CRC ENABLE */
03H, 03FH, /* B RX BITS, AUTO ENABLES, HUNT MODE */
OFFH, /* END OF INITIALIZATION TABLE */
```

DECLARE TABLE_74_B(*) BYTE DATA

```
(02H, 00H, /* INTERRUPT VECTOR */
0IH, 1CH, /* STATUS AFFECTS VECTOR */
OFFH, /* END */
```

/* INITIALIZE THE B274 */

C=0,
DO WHILE TABLE_74_B(C) <> OFFH,
  OUTPUT(COMMAND_B_74) = TABLE_74_B(C),
  C=C+1,
  OUTPUT(COMMAND_B_74) = TABLE_74_B(C),
  C=C+1,
END,
C=0,
DO WHILE TABLE_74_A(C) <> OFFH,
  OUTPUT(COMMAND_A_74) = TABLE_74_A(C),
  C=C+1,
  OUTPUT(COMMAND_A_74) = TABLE_74_A(C),
  C=C+1,
END,
RETURN,
END INITIALIZE_B274.
```

Figure 18. Typical MPSC SDLC Initialization Sequence
For this application, the CPU is run at 8 MHz. The board is configured to operate the 8274 in SDLC operation with the data transfer in DMA mode using the 8237A. 8274 is configured first in non-vectored mode in which case the INTEL Priority Interrupt Controller 8259A is used to resolve priority between various interrupting sources on the board and subsequently interrupt the CPU. However, the vectored mode of the 8274 is also verified by disabling the 8259A and reading the vectors from the 8274. Software examples for each case will be shown later.

The application example is interrupt driven and uses DMA for all data transfers under 8237A control. The 8254 provides the transmit and receive clocks for the 8274. The 8274 was run at 400K baud with a local loopback (jumper wire) on Channel A data. The board was also run at 800K baud by modifying the software as will be discussed later in the Special Applications section. One detail to note is that the Rx Channel DMA request line from the 8274 has higher priority than the Tx Channel DMA request line. The 8274 master clock was 4.0 MHz. The on-board RAM is used to define transmit and receive data buffers. In this application, the data is read from memory location 800H through 810H and transferred to memory location 900H through 910H through the 8274 Serial Link. The operation is full duplex. 8274 modem control pins, CTS and CD have been tied low (active).

Software

The software consists of a monitor program and a program to exercise the 8274 in the SDLC mode. Appendix A contains the entire program listing. For the sake of clarity, each source module has been rewritten in a simple language and will be discussed here individually. Note that some labels in the actual listings in the Appendix will not match with the labels here. Also the listing in the Appendix sets up some flags to communicate with the monitor. Some of these flags are not explained in detail for the reason that they are not pertinent to this discussion. The monitor takes the command from a keyboard and executes this program, logging any error condition which might occur.

8274 Initialization

The MPSC is initialized in the SDLC mode for Channel A. Channel B is disabled. See Figure 18 for the initialization routine. Note that WR4 is initialized before setting up the transmitter and receive parameters. However, it may also be pointed out that other than WR4, all the other registers may be programmed in any order. Also SDLC-CRC has been programmed for correct operation. An incorrect CRC selection will result in incorrect operation. Also note that receive interrupt on first receive character has been programmed although Channel A is in the DMA mode.

Interrupt Routines

The 8274 interrupt routines will be discussed here. On an 8274 interrupt, program branches off to the "Main Interrupt Routine". In main interrupt routine, status register RR2 is read. RR2 contains the modified vector. The cause of the interrupt is determined by reading the modified bits of the vector. Note that the 8274 has been programmed in the non-vectored mode and status affects vector bit has been set. Depending on the value of the modified bits, the appropriate interrupt routine is called. See Figure 19 for the flow diagram and Figure 20 for the source code. Note that an End of Interrupt Command is issued after servicing the interrupt. This is necessary to enable the lower priority interrupts.

Figure 21 shows all the interrupt routines called by the Main Interrupt Routine. "Ignore-Interrupt" as the name implies, ignores any interrupts and sets the FAIL flag. This is done because this program is for Channel A only and we are ignoring any Channel B interrupts. The important thing to note is the Channel A Receiver Character available routine. This routine is called after receiving the first character in the SDLC frame. Since the transfer mode is DMA, we have a maximum of three character times to service this interrupt by enabling the DMA controller.
/* MAIN INTERRUPT ROUTINE */

// SET POINTER TO 2/
OUTPUT(COMMAND_B_74) = 2;
// READ INTERRUPT VECTOR */
TEMP = INPUT(STATUS_B_74) AND 07H.
// CHECK FOR CH 3 INTERRUPTS ONLY*/

// FOR THIS APPLICATION CH 3 INTERRUPTS ARE IGNORED*/

DO CASE TEMP:
    CALL IGNORE_INT; /* V2VIVO = 000*/
    CALL CHB_RX_CHAR; /* V2VIVO = 010*/
    CALL CHB_RX_CHAR; /* V2VIVO = 011*/
    CALL CHA_EXTERNAL_CHANGE; /* V2VIVO = 101*/
    CALL CHA_RX_CHAR; /* V2VIVO = 110*/
    CALL CHA_RX_SPECIAL; /* V2VIVO = 111*/
END;

OUTPUT(COMMAND_A_74) = 3BH; /* END OF INTERRUPT FOR 8274 */
RETURN;
END INTERRUPT_B274;

Figure 20. Typical Main Interrupt Routine

/* CHANNEL A EXTERNAL/STATUS CHANGE INTERRUPT HANDLER */
CHA_EXTERNAL_CHANGE: PROCEDURE;

TEMP = INPUT(STATUS_A_74); /* STATUS REG 1*/
IF (TEMP AND END_OF_TX_MESSAGE) = END_OF_TX_MESSAGE THEN
    TXDONE_S = DONE;
ELSE DO.
    TXDONE_S = DONE;
    RESULTS_S = FAIL;
END;
OUTPUT(COMMAND_A_74) = 10H; /* RESET EXT/STATUS INTERRUPTS */
RETURN.
END CHA_EXTERNAL_CHANGE;

Figure 21. 8274 Typical Interrupt Handling Routines
It may be recalled that the receiver buffer is three bytes deep in addition to the receiver shift register. At very high data rates, it may not be possible to have enough time to read RR2, enable the DMA controller without overrunning the receiver. In a case like this, the DMA controller may be left enabled before receiving the Receive Character Interrupt. Remember, the Rx DMA request and interrupt for the receive character appears at the same time. If the DMA controller is enabled, it would service the DMA request by reading the received character. This will make the 8274 interrupt line go inactive. However, the 8259A has latched the interrupt and a regular interrupt acknowledge sequence still occurs after the DMA controller has completed the transfer and given up the bus. The 8259A will return Level 7 interrupt since the 8274 interrupt has gone away. The user software must take this into account, otherwise the CPU will hang up.

The procedure shown for the Special Receive Condition Interrupt checks if the interrupt is due to the End of Frame. If this is not TRUE, the FAIL flag is set and the program aborted. For a real life system, this must be followed up by error recovery procedures which obviously are beyond the scope of this Application Note.

The transmission is terminated when the End of Message (RR0, D6) interrupt is generated. This interrupt is serviced in the Channel A External/Status Change interrupt procedure. For any other change in external status conditions, the program is aborted and a FAIL flag set.

Main Program

Finally, we will briefly discuss the main program. Figure 22 shows the source program. It may be noted that the Transmit Under-run latch is reset after loading the first character into the 8274. This is done to ensure CRC transmission at the end of the frame. Also, the first character is loaded from the CPU to start DMA transfer of subsequent data. This concludes our discussion on hardware and software example. Appendix A also includes the software written to exercise the 8274 in the vectored mode by disabling the 8259A.

```
CHA_SDLC_TEST. PROCEDURE BYTE PUBLIC;

CALL ENABLE_INTERRUPTS_S,
CALL INIT_8274_SDLC_S,
ENABLE;
OUTPUT(COMMAND_A_74) = 28H; /* RESET TX INT/DMA */
OUTPUT(COMMAND_B_74) = 28H; /* BEFORE INITIALIZING 8237*/
CALL INIT_8237_S,
OUTPUT(DAT A_74) = 55H; /* LOAD FIRST CHARACTER FROM */
/*CPU */
/* TO ENSURE CRC TRANSMISSION, RESET TX UNDERRUN LATCH */
OUTPUT(COMMAND_A_74) = 00H;
RXDONE_S.TXDONE_S=NOT_DONE, /* CLEAR ALL FLAGS */
RESULTS_S=PASS, /* FLAG SET FOR MONITOR */
DO WHILE RXDONE_S=NOT_DONE, /* DO UNTIL TERMINAL COUNT */
END;
DO WHILE (INPUT(STATUS_A_74) AND 04H) O 04H,
/* WAIT FOR CRC TO GET TRANSMITTED */
/* TEST FOR TX BUFFER EMPTY TO VERIFY THIS */
END;
DO WHILE RXDONE_S=NOT_DONE, /* DO UNTIL TERMINAL COUNT */
END;
CALL STOP_8237_S;
END CHA_SDLC_TEST;
```

Figure 22. Typical 8274 Transmit/Receive Set-Up in SDLC Mode
SPECIAL APPLICATIONS

In this section, some special application issues will be discussed. This will be useful to a user who may be using a mode which is possible with the 8274 but not explicitly explained in the data sheet.

MPSC Daisy Chain Operation

Multiple MPSCs can be connected in a daisy-chain configuration (see Figure 23). This feature may be useful in an application where multiple communication channels may be required and because of high data rates, conventional interrupt controller is not used to avoid long interrupt response times. To configure the MPSCs for the daisy chain operation, the interrupt priority input pins (IPI) and interrupt priority output pins (IPO) of the MPSC should be connected as shown. The highest priority device has its IPI pin connected to ground. Each MPSC is programmed in a vectored mode with status affects vector bit set. In the 8085 basic systems, only one MPSC should be programmed in the 8085 Mode 1. This is the MPSC which will put the call vector (CD Hex) on the data bus in response to the first INTA pulse (see Figure 15). It may be pointed out that the MPSC in 8085 Mode 1 will provide the call vector irrespective of the state of IPI pin. Once a higher priority MPSC generates an interrupt, its IPO pin goes inactive thus preventing lower priority MPSCs from interrupting the CPU. Preferably the highest priority MPSC should be programmed in 8085 Mode 1. It may be recalled that the Priority Resolve Time on a given MPSC extends from the falling edge of the first INTA pulse to the falling edge of the second INTA pulse. During this period, no new internal interrupt requests are accepted. The maximum number of the MPSCs that can be connected in a daisy chain is limited by the Priority Resolution Time. Figure 24 shows a maximum number of MPSCs that can be connected in various CPU systems.

It may be pointed out that IPO to IPI delay time specification is 100 ns.

<table>
<thead>
<tr>
<th>System Configuration</th>
<th>Priority Resolution Time Min (ns)</th>
<th>Number of 8274s Daisy Chained (Max)</th>
</tr>
</thead>
<tbody>
<tr>
<td>8086-1</td>
<td>400</td>
<td>4</td>
</tr>
<tr>
<td>8086-2</td>
<td>500</td>
<td>5</td>
</tr>
<tr>
<td>8086</td>
<td>800</td>
<td>8</td>
</tr>
<tr>
<td>8088</td>
<td>800</td>
<td>8</td>
</tr>
<tr>
<td>8085-2</td>
<td>1200</td>
<td>12</td>
</tr>
<tr>
<td>8085A</td>
<td>1920</td>
<td>19</td>
</tr>
</tbody>
</table>

NOTE: Zero wait states have been assumed.

Figure 24. 8274 Daisy Chain Operation

Bisync Transparent Communication

Bisync applications generally require that data transparency be established during communication. This requires that the special control characters may not be included in the CRC accumulation. Refer to the Synchronous Protocol Overview section for a more detailed discussion on data transparency. The 8274 can be used for transparent communication in Bisync communications. This is made possible by the capability of the MPSC to selectively turn on/turff off the CRC accumulation while transmitting or receiving. In bisync transparent transmit mode, the special characters (DLE, DLE SYN, etc) are excluded from CRC calculation. This can be easily accomplished by turning off the transmit CRC calculation (WR5: D5 = 0) before loading the special character into the transmit buffer. If the next character is to be included in the CRC accumulation, then the CRC can be enabled (WR5: D5 = 1). See Figure 25 for a typical flow diagram.
During reception, it is possible to exclude received character from CRC calculation by turning off the Receive CRC after reading the special character. This is made possible by the fact that the received data is presented to receive CRC checker 8 bit times after the character has been received. During this 8 bit times, the CPU must read the character and decide if it wants to be included in the CRC calculation. Figure 26 shows the typical flow diagram to achieve this.

It should be noted that the CRC generator must be enabled during CRC reception. Also, after reading the CRC bytes, two more characters (SYNC) must be read before checking for CRC check result in RR1.

**Auto Enable Mode**

In some data communication applications, it may be required to enable the transmitter or the receiver when the CTS or the CD lines respectively, are activated by the modems. This may be done very easily by programming the 8274 into the Auto Enable Mode. The auto enable mode is set by writing a '1' to WR3:D5. The function of this mode is to enable the transmitter automatically when the CTS goes active. The receiver is enabled when CD goes active. An in-active state of CTS or CD pin will disable the transmitter or the receiver respectively. However, the Transmit Enable bit (WR5:D3) and Receive Enable bit (WR3:D1) must be set in order to use the auto enable mode. In non-auto mode, the transmitter or receiver is enabled if the corresponding bits are set in WR5 and WR3, irrespective of the state CTS or CD pins. It may be recalled that any transition on CTS or CD pin will generate External/Status Interrupt with the corresponding bits set in RR1. This interrupt can be cleared by issuing a Reset External/Status interrupt command as discussed earlier.

Note that in auto enable mode, the character to be transmitted must be loaded into the transmit buffer after the CTS becomes active, not before. Any character loaded into the transmit buffer before the CTS became active will not be transmitted.

**High Speed DMA Operation**

In the section titled Application Example, the MPSC has been programmed to operate in DMA mode and receiver is programmed to generate an interrupt on the first receive character. You may recall that the receive FIFO is three bytes deep. On receiving the interrupt on the first receive character, the CPU must enable the DMA controller within three received byte times to avoid receiver over-run condition. In the application example, at 400K baud, the CPU had approximately 60 µs to enable the DMA controller to avoid receiver buffer overflow. However, at higher baud rates, the CPU may not have enough time to enable the DMA controller in time. For example, at 1M baud, the CPU should enable the DMA controller within approximately 24 µs to avoid receiver buffer overrun. In most applications, this is not sufficient time. To solve this problem, the DMA controller should be left enabled before getting the interrupt on the first receive character (which is accompanied by the Rx DMA request for the appropriate channel). This will allow he DMA controller to start DMA transfer as soon as the Rx DMA request becomes active without giving the CPU enough time to respond to the interrupt on the first receive character. The CPU will respond to the interrupt after the DMA transfer has been completed and will find the 8259A (see Application Example) responding with interrupt level 7, the lowest priority level. Note that the 8274 interrupt request was satisfied by the DMA controller, hence the interrupt on the first receive character was cleared and the 8259A had no pending interrupt. Because of no pending interrupt, the 8259A returned interrupt level 7 in response to the INTA sequence from the CPU. The user software should take care of this interrupt.
PROGRAMMING HINTS

This section will describe some useful programming hints which may be useful in program development.

Asynchronous Operation

At the end of transmission, the CPU must issue “Reset Transmit Interrupt/DMA Pending” command in WR0 to reset the last transmit empty request which was not satisfied. Failing to do so will result in the MPSC locking up in a transmit empty state forever.

Non-Vectored Mode

In non-vectored mode, the Interrupt Acknowledge pin (INTA) on the MPSC must be tied high through a pull-up resistor. Failing to do so will result in unpredictable response from the 8274.

HDLC/SDLC Mode

When receiving data in SDLC mode, the CRC bytes must be read by the CPU (or DMA controller) just like any other data field. Failing to do so will result in receiver buffer overflow. Also, the End of Frame Interrupt indicates that the entire frame has been received. At this point, the CRC result (RR1:D6) and residue code (RR1:D3, D2, D1) may be checked.

Status Register RR2

ChB RR2 contains the vector which gets modified to indicate the source of interrupt (see the section titled MPSC Modes of Operation). However, the state of the vector does not change if no new interrupts are generated. The contents of ChB RR2 are only changed when a new interrupt is generated. In order to get the correct information, RR2 must be read only after an interrupt is generated, otherwise it will indicate the previous state.

Initialization Sequence

The MPSC initialization routine must issue a channel Reset Command at the beginning. WR4 should be defined before other registers. At the end of the initialization sequence, Reset External/Status and Error Reset commands should be issued to clear any spurious interrupts which may have been caused at power up.

Transmit Under-Run/EOM Latch

In SDLC/HDLC, bisync and monosync mode, the transmit underrun/EOM must be reset to enable the CRC check bytes to be appended to the transmit frame or transmit message. The transmit under-run/EOM latch can be reset only after the first character is loaded into the transmit buffer. When the transmitter underruns at the end of the frame, CRC check bytes are appended to the frame/message. The transmit underrun/EOM latch can be reset at any time during the transmission after the first character. However, it should be reset before the transmitter underruns otherwise, both bytes of the CRC may not be appended to the frame/message. In the receive mode in bisync operation, the CPU must read the CRC bytes and two more SYNC characters before checking for valid CRC result in RR1.

Sync Character Load Inhibit

In bisync/monosync mode only, it is possible to prevent loading sync characters into the receive buffers by setting the sync character load inhibit bit (WR3:D1 = 1). Caution must be exercised in using this option. It may be possible to get a CRC character in the received message which may match the sync character and not get transferred to the receive buffer. However, sync character load inhibit should be enabled during all pre-frame sync characters so the software routine does not have to read them from the MPSC.

In SDLC/HDLC mode, sync character load inhibit bit must be reset to zero for proper operation.

EOI Command

EOI command can only be issued through channel A irrespective of which channel had generated the interrupt.

Priority in DMA Mode

There is no priority in DMA mode between the following four signals: TxDRQ(CHA), RxDRQ(CHA), TxDRQ(CHB), RxDRQ(CHB). The priority between these four signals must be resolved by the DMA controller. At any given time, all four DMA channels from the 8274 are capable of going active.
INIT_B274_S: DD:
*INCLUDE (.FI: PORTS PLM)

DECLARE LIT LITERALLY 'LITERALLY':

DECLARE CHO_ADDR LIT 'OB0H',
    CHO_COUNT LIT 'OB1H',
    CH1_ADDR LIT 'OB2H',
    CH1_COUNT LIT 'OB3H',
    CH2_ADDR LIT 'OB4H',
    CH2_COUNT LIT 'OB5H',
    CH3_ADDR LIT 'OB6H',
    CH3_COUNT LIT 'OB7H',
    STATUS_37 LIT 'OB8H',
    COMMAND_37 LIT 'OB9H',
    REQUEST_REQ_37 LIT 'OBAH',
    SINGLE_MASK LIT 'OB0H',
    MODE_REG_37 LIT 'OB0H',

DECLARE CTR_O0 LIT '090H',
    CTR_O1 LIT '091H',
    CTR_O2 LIT '092H',

DECLARE CHO_ADDR LIT 'OB0H',
    CHO_COUNT LIT 'OB1H',
    CH1_ADDR LIT 'OB2H',
    CH1_COUNT LIT 'OB3H',
    CH2_ADDR LIT 'OB4H',
    CH2_COUNT LIT 'OB5H',
    CH3_ADDR LIT 'OB6H',
    CH3_COUNT LIT 'OB7H',
    STATUS_37 LIT 'OB8H',
    COMMAND_37 LIT 'OB9H',
    REQUEST_REQ_37 LIT 'OBAH',
    SINGLE_MASK LIT 'OB0H',
    MODE_REG_37 LIT 'OB0H',

DECLARE CTR_O0 LIT '090H',
    CTR_O1 LIT '091H',
    CTR_O2 LIT '092H',

PL/M-86 COMPILER  ISBC 88/45 8274 CHANNEL A SDLC TEST

SERIES-III PL/M-86 V2 0 COMPILATION OF MODULE INIT_B274_S
OBJECT MODULE PLACED IN .FI: SBC874 OBJ
COMPILER INVOKED BY PLMB6 B6 FI: SBC874 PLM TITLE:(ISBC 88/45 8274 CHANNEL
A SDLC TEST) COMPACT NOINTVECTOR ROM

INIT_B274_S:
DO:
*INITIALIZE THE B274 FOR SDLC MODE
*/
*/ 1 RESET CHANNEL
*/ 2. EXTERNAL INTERRUPTS ENABLED
*/ 3. NO WAIT
*/ 4. PIN 10 = RTS
*/ 5. NON-VECTORED INTERRUPT-B0B6 MODE
*/ 6. CHANNEL A DMA, CH B INT
*/ 7. TX AND RX = 8 BITS/CHAR
*/ 9. ADDRESS SEARCH MODE
*/ 10. CD AND CTS AUTO ENABLE
*/ 11. X1 CLOCK
*/ 12. NO PARITY
*/ 13. SDLC/HDLC MODE
*/ 14. RTS AND DTR
*/ 15. CCIT - CRC
*/ 16. TRANSMITTER AND RECEIVER ENABLED
*/ 17. 7EH = flag
*/

APPENDIX A
APPLICATION EXAMPLE: SOFTWARE LISTINGS
```plaintext
= CONTROL0_54 LIT '093H',
= STATUS0_54 LIT '093H',
= CTR_10 LIT '09BH',
= CTR_11 LIT '09BH',
= CTR2 LIT '09AH',
= CONTROL1_54 LIT '09BH',
= STATUS1_54 LIT '09BH',

= /* 8255 PORTS */
5 1 = DECLARE PORTA_55 LIT '0A0H',
= PORTB_55 LIT '0A1H',
= PORTC_55 LIT '0A2H',
= CONTROL_55 LIT '0A3H',

= /* 8274 PORTS */
6 1 = DECLARE DATA_A_74 LIT '0D0H',
= DATA_B_74 LIT '0D1H',
= STATUS_A_74 LIT '0D2H',
= COMMAND_A_74 LIT '0D2H',
= STATUS_B_74 LIT '0D3H',
= COMMAND_B_74 LIT '0D3H',

= /* 8259A PORTS */
7 1 = DECLARE STATUS_POLL_59 LIT '0E0H',
= ICW1_59 LIT '0E0H',
= DCW2_59 LIT '0E0H',
= DCW3_59 LIT '0E0H',
= DCW4_59 LIT '0E0H',
= ICW2_59 LIT '0E1H',
= ICW3_59 LIT '0E1H',
= ICW4_59 LIT '0E1H',

= /* 8274 REGISTER BIT ASSIGNMENTS */
= /* READ REGISTER 0 */
8 1 = DECLARE RX_AVAIL LIT '01H',
= INT_PENDING LIT '02H',
= TX_EMPTV LIT '04H',
= CARRIÆR_DETECT LIT '08H',
= SYNC_HUNT LIT '10H',
= CLEAR_TO_SEND LIT '20H',

PL/M-86 Compiler ISBC 88/45 8274 CHANNEL A SDLC TEST

= END_OF_TX_MESSAGE LIT '40H',
= BREAK_ABORT LIT '80H',

= /* READ REGISTER 1 */
9 1 = DECLARE ALL_SENT LIT '01H',
= PARITY_ERROR LIT '10H',
= RX_OVERRUN LIT '20H',
= CRC_ERROR LIT '40H',
= END_OF_FRAME LIT '80H',

= /* READ REGISTER 2 */
10 1 = DECLARE TX_B_EMPTY LIT '00H',
= EXT_B_CHANGE LIT '01H',
= RX_B_AVAIL LIT '02H',
= RX_B_SPECIAL LIT '03H',
= TX_A_EMPTY LIT '04H',
= EXT_A_CHANGE LIT '05H',
= RX_A_AVAIL LIT '06H',
= RX_A_SPECIAL LIT '07H',
```

210403-16
declare cho_sel  = lit '00h';
declare ch1_sel  = lit '01h';
declare ch2_sel  = lit '02h';
declare ch3_sel  = lit '03h';

declare write_xfer = lit '04h';

declare read_xfer = lit '05h';

declare demand_mode = lit '06h';

declare single_mode = lit '07h';

declare block_mode = lit '08h';

declare set_mask  = lit '09h';

delay_s: procedure public;
  declare d word;
  d=0;
  do while d<000h;
  d=d+1;
  end;
  end delay_s;

init_9274 sdlc_s: procedure public.
  eject

/* table to initialize the 8274 channel a and b */
/* format is: write register, register data */
/* initialize channel only */

 declare table_74_a(*) byte data
 (00h,02h,07h,01h,0ah,04h,03h,0eh,0ch,0fh);/* channel reset */
(00h,00h,00h,00h,00h,00h,00h,00h,00h,00h); /* reset tx crc */
(02h,01h,00h,00h,00h,00h,00h,00h,00h,00h); /* pin 10=rtsc, a dma, b int */
(04h,02h,00h,00h,00h,00h,00h,00h,00h,00h); /* sdlc/hdlc mode, no parity */
(07h,07eh,00h,00h,00h,00h,00h,00h,00h,00h); /* sdlc flag */
(01h,08h,00h,00h,00h,00h,00h,00h,00h,00h); /* rts, rts, tx enable, tx crc enable */
(04h,05h,00h,00h,00h,00h,00h,00h,00h,00h); /* default address */
(03h,08h,00h,00h,00h,00h,00h,00h,00h,00h); /* rx crc enable */
(offh),/* end of initialization table */

 declare table_74_b(*) byte data
 (02h,00h,00h,00h,00h,00h,00h,00h,00h,00h); /* interrupt vector */
(01h,0ch,00h,00h,00h,00h,00h,00h,00h,00h); /* status affects vector */
(offh);/* end */

/* initialize the 8274 */

output(command_b_74)=table_74_b(c);/* baud rate = 4096baud*/

output(command_b_74)=table_74_b(c);/* baud rate = 4096baud*/

/* initialize the 8274 */

output(command_b_74)=table_74_b(c); /* baud rate = 4096baud*/

output(command_b_74)=table_74_b(c); /* baud rate = 4096baud*/

C=0;

while table_74_b(c)<offh; /* output(command_b_74)=table_74_b(c); */

C=C+1;

output(command_b_74)=table_74_b(c);

C=C+1;

end;
33 2 C=0.
34 2 DO WHILE TABLE_74_A(C) <> OFFH.
35 3 OUTPUT(COMMAND_74) = TABLE_74_A(C).
36 3 C=C+1.
37 3 OUTPUT(COMMAND_74) = TABLE_74_A(C).
38 3 C=C+1.
39 3 END.
40 2 CALL DELAY_S.
41 2 RETURN.
42 2 END INIT_B274_SDLC_S.
43 1 END INIT_B274_S.

PL/M-86 COMPILER :SBC BB/45 B274 CHANNEL A SDL TEST

MODULE INFORMATION

CODE AREA SIZE = 00A8H 168D
CONSTANT AREA SIZE = 0000H 0B
VARIABLE AREA SIZE = 0003H 3D
MAXIMUM STACK SIZE = 0006H 6D
213 LINES READ
0 PROGRAM WARNINGS
0 PROGRAM ERRORS

END OF PL/M-86 COMPILATION

PL/M-86 COMPILER :SBC BB/45 B274 CHANNEL A SDL TEST

SERIES-III PL/M-86 V2 0 COMPILATION OF MODULE INIT_B237_CHA
OBJECT MODULE PLACED IN .FI SINI37 OBJ
COMPILER INVOKED BY: PLM86.B6 .FI SINI37 PLM TITLE: (SBC BB/45 B274 CHANNEL A SDL TEST) COMPACT NointVECTOR ROM

慑陲LaunchInitializationRoutineForDMATransfer

1 INIT_B237_CHA, DD.
$NDLIST

12 1 INIT_B237_CHA PROCEDURE PUBLIC.

13 2 OUTPUT(MASTER_CLEAR_37) = 0.
14 2 OUTPUT(COMMAND_37) = 20H. */ EXTENDED WRITE */
15 2 OUTPUT(ALL_MASK_37) = OFFH. */ MASK ALL REQUESTS */
16 2 OUTPUT(MODE_REQ_37) = (SINGLE_MODE OR WRITE_XFER OR CHO_SEL).
17 2 OUTPUT(MODE_REG_37) = (SINGLE_MODE OR READ_XFER OR CHI_SEL).
18 2 OUTPUT(CLR_BYTE_PTR_37) = 0.
19 2 OUTPUT(CHO_ADDR) = 00. */ RECEIVE BUFF AT 900H */
20 2 OUTPUT(CHO_ADDR) = 09H.
21 2 OUTPUT(CHO_COUNT) = 0H.
22 2 OUTPUT(CHO_COUNT) = 01.
23 2 OUTPUT(CHO_ADDR) = 00. */ TRANSMIT BUFF AT 800H */
24 2 OUTPUT(CHO_ADDR) = 08H.
25 2 OUTPUT(CHO_COUNT) = 010H.
26 2 OUTPUT(CHO_COUNT) = 00H.

210403-18
/* ENABLE TRANSFER */
27 2  OUTPUT(SINGLE_MASK) = CHI_SEL,  /* ENABLE TX DMA */
28 2  RETURN;
29 2  END INIT_B237_S;
30 1  /* TURN OFF THE B237 CHANNELS 0 AND 1 */
31 2  STOP_B237_S; PROCEDURE PUBLIC,
32 2  OUTPUT(SINGLE_MASK) = CHI_SEL OR SET_MASK;
33 2  RETURN;
34 2  END STOP_B237_S;
35 1  END INIT_B237_CHAN;

MODULE INFORMATION:
CODE AREA SIZE = 004CH 74D
CONSTANT AREA SIZE = 0000H 0D
VARIABLE AREA SIZE = 0000H 0D

PL/M-86 COMPILER 15BC B8/45 B274 CHANNEL A SDLC TEST
MAXIMUM STACK SIZE = 0002H 2D
163 LINES READ
0 PROGRAM WARNINGS
0 PROGRAM ERRORS
END OF PL/M-86 COMPILATION

PL/M-86 COMPILER 15BC B8/45 B274 CHANNEL A SDLC TEST
SHERIE-III PL/M-86 V2.0 COMPILATION OF MODULE INTR_B274_S
OBJECT MODULE PLACED IN F1 SINTR OBJ
COMPILER INVOKED BY PLMB6 B6 F1 SINTR PLM TITLE;15BC B8/45 B274 CHANNEL
A SDLC TEST) COMPACT NOINVECTOR ROM
/
*******************************************************************************/
/*/ 8274 INTERRUPT ROUTINE */
/*/*******************************************************************************/
1 1 INTR_B274_S DO.
ENDLIST
12 1 DECLARE TEMP BYTE.
13 1 DECLARE (RESULTS_S, TXDONE_S, RXDONE_S) BYTE EXTERNAL.
14 1 DECLARE INT_VEC POINTER AT (140).
15 1 DECLARE INT_VEC_STORE POINTER.
16 1 DECLARE MAG_S BYTE.
17 1 DECLARE DONE. LIT 'OFFH',
18 1 DECLARE NT_DONE LIT 'OOFH',
19 1 DECLARE PAGE LIT 'OFFH',
20 1 DECLARE FAIL LIT 'OOFH',

*******************************************************************************/
/* IGNORE INTERRUPT HANDLER */
*******************************************************************************/
18 1 IGNORE_INT PROCEDURE.
19 2 RESULTS_S = FAIL.
20 2 RETURN.
21 2 END IGNORE_INT.

2-410
AP-145

PL/M-86 COMPILER :SBC 88/45 8274 CHANNEL A SDLC TEST

CHAN_EXTERNAL_CHANGE PROCEDURE.
22 1
23 2 TEMP = INPUT(Status_A_74),  /* STATUS REG */
24 2 IF (TEMP AND END_OF_TX_MESSAGE) = END_OF_TX_MESSAGE THEN
25 2 TXDONE_S = DONE.
26 2 ELSE DO.
27 3 TXDONE_S = DONE.
28 3 RESULT_S = FAIL.
29 3 END.
30 2 OUTPUT(Command_A_74) = 10h. /* RESET EXT/STATUS INTERRUPTS */
31 2 RETURN.
32 2 END CHA_EXTERNAL_CHANGE.

CHA_RX_SPECIAL PROCEDURE.
33 1
34 2 OUTPUT(Command_A_74) = 1.
35 2 TEMP = INPUT(Status_A_74).
36 2 IF (TEMP AND END_OF_FRAME) = END_OF_FRAME THEN
37 2 DO.
38 3 IF (TEMP AND 040H) = 040H THEN
39 3 RESULTS_S = FAIL.  /* CRC ERROR */
40 3 RXDONE_S = DONE.
41 3 OUTPUT(Command_A_74) = 30h. /* ERROR RESET */
42 3 END.
43 2 ELSE DO.
44 3 IF (TEMP AND 20H) = 20H THEN DO.
45 4 RESULTS_S = FAIL.  /* RX OVERRUN ERROR */
46 4 RXDONE_S = DONE.
47 4 OUTPUT(Command_A_74) = 30h. /* ERROR RESET */
48 4 END.
49 3 END.
50 2 RETURN.
51 2 END CHA_RX_SPECIAL.

CHA_RX_CHAR PROCEDURE.
52 1
53 2 OUTPUT(SINGLE_MASK) = CHO_SEL.  /* ENABLE RX DMA CHANNEL */
54 2 RETURN.
55 2 END CHA_RX_CHAR.

PL/M-86 COMPILER :SBC 88/45 8274 CHANNEL A SDLC TEST

/* ENABLE 8274 INTERRUPTS - SET UP THE 8259A */
57 1 ENABLE_INTERRUPTS_S PROCEDURE PUBLIC.
58 2 DECLARE CHA_INT_ON LIT 'OF7H'.
59 2 DISABLE.
60 2 CALL SET_INTERRUPT(39, INT_39).
INT_VEC_STORE = INT_VEC.
INT_VEC = INTERRUPT_PTR(INT_B274_S).
MASK_59 = INPUT(OCWI_59).
OUTPUT(OCWI_59) = MASK_59 AND CHA_INT_ON.
RETURN.
END ENABLE_INTERRUPTS_S.

DISABLE_INTERRUPTS_S PROCEDURE PUBLIC.
DISABLE.
INT_VEC = INT_VEC_STORE.
OUTPUT(OCWI_59) = MASK_59.
ENABLE.
RETURN.
END DISABLE_INTERRUPTS_S.

CHB_RX_CHAR PROCEDURE.
TEMP = INPUT(DATA_B_74).
OUTPUT(COMMAND_B_74) = 3BH.
RETURN.
END CHB_RX_CHAR.

DISABLE_INTERRUPTS_S

MAIN INTERRUPT ROUTINE

INT_B274_S PROCEDURE INTERRUPT 35 PUBLIC.
OUTPUT(COMMAND_B_74) = 2. /* SET POINTER TO 2*/
TEMP = INPUT(STATUS_B_74) AND 07H. /* READ INTERRUPT VECTOR */
/* CHECK FOR CHA INT ONLY*/
/* FOR THIS APPLICATION CH B INTERRUPTS ARE IGNORED*/
DO CASE TEMP.
CALL IGNORE_INT.
CALL CHO_RX_CHAR.
CALL IGNORE_Int.
CALL CHO_EXTERNAL_CHANGE.
CALL CHA_RX_CHAR.
CALL CHA_RX_SPECIAL.
RETURN.
END.

END OF INTERRUPT FOR B274 */

OUTPUT(COMMAND_A_74) = 3BH. /* END OF INTERRUPT FOR B274 */
OUTPUT(COMMAND_B_74) = 3BH. /* B259 A EOI */
OUTPUT(COMMAND_A_74) = INPUT(OCW1_59) AND 0F7H.
RETURN.
END INT_B274_S.

DEFAULT INTERRUPT ROUTINE - B259A INTERRUPT 7 */
/* REQUIRED ONLY WHEN DMA CONTROLLER IS ENABLED */
/* BEFORE RECEIVING FIRST CHARACTER WHICH IS */
/* AT HIGH BAUD RATES LIKE 9600 BAUD READ APP */
/* NOTE SECTION 6 FOR DETAILS */
INT_39 PROCEDURE INTERRUPT 39.
OUTPUT(OCW2_59) = 20H. /* NON-SPECIFIC EOI */
OUTPUT(OCW1_59) = INPUT(OCW1_59) AND OFFH.
END INT_39.

END OF PL/M-86 COMPILATION.

DELAY_5 PROCEDURE EXTERNAL.
END DELAY_5.

ENABLE_INTERRUPTS_S PROCEDURE EXTERNAL.
END ENABLE_INTERRUPTS_S.

DISABLE_INTERRUPTS_S PROCEDURE EXTERNAL.
END DISABLE_INTERRUPTS_S.

INIT_8274_SDLC_5 PROCEDURE EXTERNAL.
END INIT_8274_SDLC_5.

STOP_8237_5 PROCEDURE EXTERNAL.
END STOP_8237_5.

VERIFY_TRANSFER_S PROCEDURE EXTERNAL.
END VERIFY_TRANSFER_S.

INT_8274_5 PROCEDURE INTERRUPT 35 EXTERNAL.
END INT_8274_5.

DECLARE (RESULTS_S, TXDONE_S, RXDONE_S) BYTE PUBLIC.
DECLARE DONE LIT 'OFFH'.
DECLARE NOT_DONE LIT '00H'.
DECLARE PASS LIT 'OFFH'.
DECLARE FAIL LIT '00H'.

PL/M-86 COMPILER I5BC 88/45 8274 CHANNEL A SDLC TEST

SERIES-III PL/M-86 V2.0 COMPILED OF MODULE STEST
OBJECT MODULE PLACED IN F1 STEST OBJ
COMPILER INVOKED BY: PLMBb.66 F1 STEST PLM TITLE plurality.
PL/M-86 COMPILER I5BC 88/45 8274 CHANNEL A SDLC TEST

CODE AREA SIZE = 01BFH 4470
CONSTANT AREA SIZE = 0000H 0D
VARIABLE AREA SIZE = 0006H 6D
MAXIMUM STACK SIZE = 0020H 340
295 LINES READ
0 PROGRAM WARNINGS
0 PROGRAM ERRORS.

DECLARE (RESULTS_S, TXDONE_S, RXDONE_S) BYTE PUBLIC.
DECLARE DONE LIT 'OFFH'.
DECLARE NOT_DONE LIT '00H'.
DECLARE PASS LIT 'OFFH'.
DECLARE FAIL LIT '00H'.

END OF PL/M-86 COMPILATION.

SERIES-III PL/M-86 V2.0 COMPILED OF MODULE STEST
OBJECT MODULE PLACED IN F1 STEST OBJ
COMPILER INVOKED BY: PLMBb.66 F1 STEST PLM TITLE plurality.
PL/M-86 COMPILER I5BC 88/45 8274 CHANNEL A SDLC TEST

DECLARE (RESULTS_S, TXDONE_S, RXDONE_S) BYTE PUBLIC.
DECLARE DONE LIT 'OFFH'.
DECLARE NOT_DONE LIT '00H'.
DECLARE PASS LIT 'OFFH'.
DECLARE FAIL LIT '00H'.

END OF PL/M-86 COMPILATION.

DECLARE (RESULTS_S, TXDONE_S, RXDONE_S) BYTE PUBLIC.
DECLARE DONE LIT 'OFFH'.
DECLARE NOT_DONE LIT '00H'.
DECLARE PASS LIT 'OFFH'.
DECLARE FAIL LIT '00H'.

END OF PL/M-86 COMPILATION.
OBJECT

PL/I-S6 COMPILER ISBC BB/45 8274 CHANNEL A SDLC TEST

30 1  CHA_SDLC_TEST: PROCEDURE BYTE PUBLIC;

31 2  CALL   ENABLE_INTERRUPTS_S;
32 2  CALL   INIT_8274_SDLC_S;
33 2  ENABLE;
34 2  OUTPUT(COMMAND_A_74) = 2BH;  /* RESET TX INT/DMA */
35 2  OUTPUT(COMMAND_B_74) = 2BH;  /* BEFORE INITIALIZING 8237 */
36 2  CALL   INIT_8237_S;
37 2  OUTPUT(DATA_A_74) = 25H;  /* LOAD FIRST CHARACTER FROM CPU */

/* TO ENSURE CRC TRANSMISSION RESET TX UNDERRUN LATCH */
38 2  OUTPUT(COMMAND_A_74) = 00H;
39 2  RXDONE_S.TXDONE_S=NOT_DONE;  /* CLEAR ALL FLAGS */
40 2  RESULTS_S=PASS;  /* FLAG SET FOR MONITOR */

41 2  DO WHILE TXDONE_S=NOT_DONE;  /* DO UNTIL TERMINAL COUNT */
42 3  END;

43 2  DO WHILE INPUT (STATUS_A_74) AND 04H) <> 04H;
/* WAIT FOR CRC TO GET TRANSMITTED */
/* TEST FOR TX BUFFER EMPTY TO VERIFY THIS */
44 3  END;
45 2  DO WHILE RXDONE_S=NOT_DONE;  /* DO UNTIL TERMINAL COUNT */
46 3  END;
47 2  CALL   STOP_8237_S;
48 2  CALL   DISABLE_INTERRUPTS_S;
49 2  CALL   VERIFY_TRANSFER_S;
50 2  RETURN RESULTS_S;
51 2  END CHA_SDLC_TEST;
52 1  END TEST;

MODULE INFORMATION:

  CODE AREA SIZE = 0063H 99D
  CONSTANT AREA SIZE = 0000H 0D
  VARIABLE AREA SIZE = 0003H 3D
  MAXIMUM STACK SIZE = 0004H 4D

  198 LINES READ
  0 PROGRAM WARNINGS
  0 PROGRAM ERRORS

END OF PL/I-S6 COMPILATION

PL/I-S6 COMPILER ISBC BB/45 8274 CHANNEL A SDLC TEST

SERIES-III PL/I-S6 V2.0 COMPILATION OF MODULE VECTOR_NODE
OBJECT MODULE PLACED IN : F1 VECTOR OBJ
COMPILED INVOKED BY : PLM68_B6 F1:VECTOR.PL1 TITLE(ISBC BB/45 8274 CHANNEL A SDLC TEST)

******************************************************************************
/*
/ * 8274 INTERRUPT HANDLING ROUTINE FOR */
/ * 8274 VECTOR NODE */
/ * STATUS AFFECTS VECTOR */
******************************************************************************

210403-23
/* THIS IS AN EXAMPLE OF HOW 8274 CAN BE USED IN VECTORED MODE */
/* THE 18B84/45 BOARD WAS REQUIRED TO DISABLE THE PIT B259A AND */
/* ENABLE THE 8274 TO PLACE ITS VECTOR ON THE DATA BUS IN RESPONSE */
/* TO THE INTA SEQUENCE FROM THE B259. OTHER MODIFICATIONS INCLUDED */
/* CHANGES TO 8274 INITIALIZATION PROGRAM (B1174) TO PROGRAM 8274 */
/* INTO VECTORED MODE (WRITE REGISTER 2A D5=1) */

VECTOR_MODE DO;
#NOLIST

DECLARE TEMP BYTE;
DECLARE (RESULT_S, TXDONE, RXDONE) BYTE EXTERNAL;
DECLARE DONE LITERALLY 'OFFH',
     NOT_DONE LITERALLY '00H',
     PASS LITERALLY 'OFFH',
     FAIL LITERALLY '00H';

*******************************************************************************/
/* TRANSMIT INTERRUPT CHANNEL A INTERRUPT WILL NOT BE SEEN IN THE */
/* DMA OPERATION */
*******************************************************************************/

TX_INTERRUPT_CHAN PROCEDURE INTERRUPT 84;
  OUTPUT(COMMAND_A_74) = 00101000B;  /*RESET TXINT PENDING*/
  OUTPUT(COMMAND_A_74) = 00111000B;  /*ED1*/
END TX_INTERRUPT_CHAN;

*******************************************************************************/
/* EXTERNAL/STATUS INTERRUPT PROCEDURE: CHECKS FOR END OF MESSAGE */
/* ONLY. IF THIS IS NOT TRUE THEN THE FAIL FLAG IS SET HOWEVER. */
/* A USER PROGRAM SHOULD CHECK FOR OTHER EX/STAT CONDITIONS */
/* ALSO IN R11 AND THEN TAKE APPROPRIATE ACTION BASED ON THE */
/* APPLICATION */
*******************************************************************************/

EXT_STAT_CHANGE_CHAN PROCEDURE INTERRUPT 85;
  TEMP = INPUT(STATUS_A_74);
  IF (TEMP AND END_OF_TX_MESSAGE) = END_OF_TX_MESSAGE THEN
    TXDONE = DONE.
  ELSE DO;
    TXDONE = DONE.
END EXT_STAT_CHANGE_CHAN;

PL/86 Compiler: ISBC 8B/45 8274 CHANNEL A SDLT TEST

RESULTS_S = FAIL;
END.

OUTPUT(COMMAND_A_74) = 00010000B;  /*RESET EXT STAT INT*/
OUTPUT(COMMAND_A_74) = 00111000B;  /*ED1*/
RETURN.

RX_CHAR_AVAILABLE_CHAN PROCEDURE INTERRUPT 86;
  OUTPUT(COMMAND_A_74) = 00111000B;  /*ED1*/
RETURN.
END RX_CHAR_AVAILABLE_CHAN.

2-415
SPECIAL_RX_CONDITION_CHAN PROCEDURE INTERRUPT 87.

36 2 OUTPUT(COMMAND_A_74) = 0.1. /*POINTER 1*/
37 2 TEMP = INPUTSTATUS_A_74).
38 2 IF (TEMP AND END_OF_FRAME) = END_OF_FRAME THEN
39 2 RXDONE = DONE.
40 2 ELSE DO.
41 3 RXDONE = DONE.
42 3 RESULTS_S = FAIL.
43 3 END.
44 2 OUTPUT(COMMAND_A_74) = 00110008. /*ERROR RESET*/
45 2 OUTPUT(COMMAND_A_74) = 00111008. /*ED!*/
46 2 RETURN.
47 2 END SPECIAL_RX_CONDITION_CHAN.

48 1 ENABLE_INTERRUPTS.PROCEDURE PUBLIC.
49 2 DISABLE.
50 2 CALL SET_INTERRUPT(86.TX_INTERRUPT_CHAN).
51 2 CALL SET_INTERRUPT(85.EXT_STAT_CHANGE_CHAN).
52 2 CALL SET_INTERRUPT(86.RX_CHAN_AVAILABLE_CHAN).
53 2 CALL SET_INTERRUPT(87.SPECIAL_RX_CONDITION_CHAN).
54 2 RETURN.
55 2 END ENABLE_INTERRUPTS.
56 1 END VECTOR_MODE.

MODULE INFORMATION

CODE AREA SIZE = 012EH 302D
CONSTANT AREA SIZE = 0000H 0D
VARIABLE AREA SIZE = 0001H 1D
MAXIMUM STACK SIZE = 001EH 30D
226 LINES READ
0 PROGRAM WARNINGS
0 PROGRAM ERRORS

END OF PL/M-86 COMPILATION
APPENDIX B
MPSC READ/WRITE REGISTER DESCRIPTIONS

WRITE REGISTER 0 (WR0)

<table>
<thead>
<tr>
<th>MSB</th>
<th>LSB</th>
</tr>
</thead>
<tbody>
<tr>
<td>D7</td>
<td>D6</td>
</tr>
<tr>
<td>D5</td>
<td>D4</td>
</tr>
<tr>
<td>D3</td>
<td>D2</td>
</tr>
<tr>
<td>D1</td>
<td>D0</td>
</tr>
</tbody>
</table>

COMMAND STATUS POINTER
REGISTER POINTER

- NULL CODE
- SEND ABORT (SDLC)
- RESET EXT STATUS INTERRUPTS
- CHANNEL RESET
- ENABLE INTERRUPT ON NEXT RX CHARACTER
- RESET TDXINT DMA PENDING
- ERROR RESET
- END OF INTERRUPT (Ch. A only)

<table>
<thead>
<tr>
<th>Bits</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 0 0</td>
<td>NULL CODE</td>
</tr>
<tr>
<td>0 0 1</td>
<td>SEND ABORT (SDLC)</td>
</tr>
<tr>
<td>0 1 0</td>
<td>RESET EXT STATUS INTERRUPTS</td>
</tr>
<tr>
<td>0 1 1</td>
<td>CHANNEL RESET</td>
</tr>
<tr>
<td>1 0 0</td>
<td>ENABLE INTERRUPT ON NEXT RX CHARACTER</td>
</tr>
<tr>
<td>1 0 1</td>
<td>RESET TDXINT DMA PENDING</td>
</tr>
<tr>
<td>1 1 0</td>
<td>ERROR RESET</td>
</tr>
<tr>
<td>1 1 1</td>
<td>END OF INTERRUPT (Ch. A)</td>
</tr>
</tbody>
</table>

WRITE REGISTER 1 (WR1)

<table>
<thead>
<tr>
<th>MSB</th>
<th>LSB</th>
</tr>
</thead>
<tbody>
<tr>
<td>D7</td>
<td>D6</td>
</tr>
<tr>
<td>D5</td>
<td>D4</td>
</tr>
<tr>
<td>D3</td>
<td>D2</td>
</tr>
<tr>
<td>D1</td>
<td>D0</td>
</tr>
</tbody>
</table>

EXT INTERRUPT ENABLE
Tx INTERRUPT DMA ENABLE
STATUS AFFECTS VECTOR
VARIES VECTOR (CHB ONLY)
NULL CODE CH A

<table>
<thead>
<tr>
<th>Bits</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 0 0</td>
<td>RxINT/DMA DISABLE</td>
</tr>
<tr>
<td>0 0 1</td>
<td>RxINT ON FIRST CHAR OR SPECIAL CONDITION</td>
</tr>
<tr>
<td>1 0 0</td>
<td>INT ON ALL Rx CHAR (PARITY AFFECTS VECTOR) OR SPECIAL CONDITION</td>
</tr>
<tr>
<td>1 1 0</td>
<td>INT ON ALL Rx CHAR (PARITY DOES NOT AFFECT VECTOR) OR SPECIAL CONDITION</td>
</tr>
<tr>
<td>1 1 1</td>
<td>INT ON ALL Rx CHAR (PARITY DOES NOT AFFECT VECTOR) OR SPECIAL CONDITION</td>
</tr>
</tbody>
</table>

- 1 WAIT ON Rx, 0 WAIT ON Tx
- MUST BE ZERO
- WAIT ENABLE, 1 ENABLE, 0 DISABLE

210403–26
WRITE REGISTER 2 (WR2): CHANNEL A

```
<table>
<thead>
<tr>
<th>D7</th>
<th>D6</th>
<th>D5</th>
<th>D4</th>
<th>D3</th>
<th>D2</th>
<th>D1</th>
<th>D0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

0 PRIORITY RxA > RxB > TxA > TxB > EXTA > EXTB
1 PRIORITY RxA > TxA > RxB > TxB > EXTA > EXTB
```

0 0 BOTH INTERRUPT
0 1 A DMA B INT
1 0 BOTH DMA
1 1 ILLEGAL

0 0 8085 MODE 1
0 1 8085 MODE 2
1 0 8088/86 MODE
1 1 ILLEGAL

1 VECTORED INTERRUPT
0 NON VECTORED INTERRUPT

MUST BE ZERO
1 PIN 10 SYNDETg
0 PIN 10 RTSg

*External Status interrupt only if EXT Interrupt Enable (WR1: D0) is set.

WRITE REGISTER 2 (WR2): CHANNEL B

WRITE REGISTER 3 (WR3)

```
<table>
<thead>
<tr>
<th>D7</th>
<th>D6</th>
<th>D5</th>
<th>D4</th>
<th>D3</th>
<th>D2</th>
<th>D1</th>
<th>D0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

0 Rx5 BITS/CHAR
0 1 Rx7 BITS/CHAR
1 0 Rx6 BITS/CHAR
1 1 Rx8 BITS/CHAR
```

210403-28

210403-29

210403-30
WRITE REGISTER 4 (WR4)

<table>
<thead>
<tr>
<th>MSB</th>
<th>LSB</th>
</tr>
</thead>
<tbody>
<tr>
<td>D7</td>
<td>D6</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

1 ENABLE PARITY
0 DISABLE PARITY
1 EVEN PARITY
0 ODD PARITY
0 0 ENABLE SYNC MODES
0 1 1 STOP BIT
1 0 1.5 STOP BITS
1 1 2 STOP BITS
0 0 8 BIT SYNC CHAR
0 1 16 BIT SYNC CHAR
1 0 SDLC/HDLC(01111110)FLAG
1 1 1 EXTERNAL SYNC MODE
0 0 X1 CLOCK
0 1 X16 CLOCK
1 0 X32 CLOCK
1 1 X64 CLOCK

WRITE REGISTER 5 (WR5)

<table>
<thead>
<tr>
<th>MSB</th>
<th>LSB</th>
</tr>
</thead>
<tbody>
<tr>
<td>D7</td>
<td>D6</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Tx CRC ENABLE
RTS
SDLC/CRC -16 (CRC MODE)
Tx ENABLE
SEND BREAK
0 0 Tx5 BITS OR LESS/CHAR
0 1 Tx7 BITS/CHAR
1 0 Tx8 BITS/CHAR
1 1 Tx8 BITS/CHAR

WRITE REGISTER 6 (WR6)

<table>
<thead>
<tr>
<th>MSB</th>
<th>LSB</th>
</tr>
</thead>
<tbody>
<tr>
<td>D7</td>
<td>D6</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

LEAST SIGNIFICANT SYNC BYTE (ADDRESS IN SDLC/HDLC MODE)

WRITE REGISTER (WR7)

<table>
<thead>
<tr>
<th>MSB</th>
<th>LSB</th>
</tr>
</thead>
<tbody>
<tr>
<td>D7</td>
<td>D6</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

MOST SIGNIFICANT
SYNC BYTE (7EH IN SDLC/HDLC MODE)

READ REGISTER 0 (RR0)

<table>
<thead>
<tr>
<th>MSB</th>
<th>LSB</th>
</tr>
</thead>
<tbody>
<tr>
<td>D7</td>
<td>D6</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Rx CHAR AVAILABLE
INT PENDING (CHA ONLY)
Tx BUFFER EMPTY
CARRIER DETECT
SYNC/HUNT
CTS
Tx UNDERRUN/EOM
BREAK/ABORT
EXTERNAL STATUS INTERRUPT MODE

210403-31
210403-32
210403-33
210403-34
210403-35
READ REGISTER 1 (RR1): (SPECIAL RECEIVE CONDITION MODE)

READ REGISTER 2 (RR2) CHANNEL B ONLY

REFERENCES
1. IBM Document No. GA27-3004-2: General Information—Binary Synchronous Communications
3. 8274 MPSC Data Sheet, Intel Corporation, Ca.
5. Computer Networks and Distributed Processing by James Martin. Prentice Hall, Inc., N.J.
Asynchronous and SDLC Communications with 82530
INTRODUCTION

INTEL's 82530, Serial Communications Controller (SCC), is a dual channel, multi-protocol data communications peripheral. It is designed to interface to high speed communications lines using asynchronous, byte synchronous, and bit synchronous protocols. It runs up to 1.5 Mbits/sec, has on-chip baud rate generators and on-chip NRZI encoding and decoding circuits—very useful for SOLC communication. This application note shows how to write I/O drivers for the 82530 to do initialization and data links using asynchronous (ASYNC) and SOLC protocols. The appendix includes sections to show how the on-chip baud rate generators could be programmed, how the modem control pins could be used, and how the 82530 could be interfaced to INTEL's 80186/188 processors.

This article deals with the software for the following:
1. SCC port definition
2. Accessing the SCC registers
3. Initialization for ASYNC communication
4. ASYNC communication in polling mode
5. ASYNC communication in interrupt mode
6. Initialization for SDLC communication
7. SDLC frame reception
8. SDLC frame transmission
9. SDLC interrupt routines

The description is written around illustrations of the actual software written in PLM86 for a 80186 - 82530 system.

I. SCC Port Definition

The Figure 1 shows how the 4 ports (2 per channel) of the SCC can be defined. Note that the sequence of ports in the ascending order of addresses is not the one that is normally expected. In the ascending order it is: command (B), data (B), command (A) and data (A). In an 80186 - 82530 system, the interconnection is as follows:

<table>
<thead>
<tr>
<th>PCSn</th>
<th>CS</th>
</tr>
</thead>
<tbody>
<tr>
<td>A1</td>
<td>D/C</td>
</tr>
<tr>
<td>80186 pins</td>
<td>A2</td>
</tr>
<tr>
<td></td>
<td>A/B</td>
</tr>
<tr>
<td></td>
<td>82530 pins</td>
</tr>
<tr>
<td>RD</td>
<td>RD</td>
</tr>
<tr>
<td>WR</td>
<td>WR</td>
</tr>
</tbody>
</table>

2. Accessing the SCC Registers

The SCC has 16 registers on each of the channels (A and B). For each channel there is only one port, the command port, to access all the registers. The register #0 can be always accessed directly through the command port. All other registers are accessed indirectly through register #0. First, the number of the register to be accessed is written to the register #0 - see the statement, in Figure 2: 'output (ch_a_command) = re8no and Ofh'. Then, the desired register is written to or read out. The Figure 2 shows 4 procedures: rra and wra, for reading and writing channel A registers; rrb and wrb, for reading and writing channel B registers. The read procedures are of the type 'byte' - they return the contents of the register being read. The write procedures require two parameters - the register number and the value to be written.

```c
/*-----------------------------------------------*/
declare ch_b_command literally 'pcs5 + 0'; /* scc channel_b command word*/
ch_b_data literally 'pcs5 + 2'; /* scc channel_b data word */
ch_a_command literally 'pcs5 + 4'; /* scc channel_a command word */
ch_a_data literally 'pcs5 + 6'; /* scc channel_a data word */
/*-----------------------------------------------*/
```

Figure 1. SCC Port Definition
```c
/* read selected SCC register */
rra: procedure (reg_no) bytel
    declare reg_no byte;
    if (reg_no and Ofh) <> 0
        then output(ch_a_command) = reg_no and Ofh;
        return input(ch_a_command);
end rra;

rrb: procedure (reg_no) bytel
    declare reg_no byte;
    if (reg_no and Ofh) <> 0
        then output(ch_b_command) = reg_no and Ofh;
        return input(ch_b_command);
end rrb;

/* write selected SCC register */
wra: procedure (reg_no, value);
    declare reg_no byte;
    declare value byte;
    if (reg_no and Ofh) <> 0
        then output(ch_a_command) = reg_no and Ofh;
        output(ch_a_command) = value;
end wra;

wrb: procedure (reg_no, value);
    declare reg_no byte;
    declare value byte;
    if (reg_no and Ofh) <> 0
        then output(ch_b_command) = reg_no and Ofh;
        output(ch_b_command) = value;
end wrb;

/* Figure 2: Accessing the SCC Registers */
```
3. Initialization for ASYNC Operation

In the following example, channel B of the SCC is used to perform ASYNC communication. Figure 3 shows how the channel B is initialized and configured for ASYNC operation. This is done by writing the various channel B registers with the proper parameters as shown. The comments in the program show what is achieved by each statement. After a software reset of the channel, register #4 should be written before writing to the other registers. The on-chip Baud Rate Generator is used to generate a 1200 bits/sec clock for both the transmitter and the receiver. The interrupts for transmitter and/or receiver are enabled only for the interrupt mode of operation; for polling, interrupts must be kept disabled.

```
/*-----------------------------------------------*/

scc_init_b: procedure:
/* scc ch B register initialization for ASYNC mode */

    call wrb(09, 01000000b);    /* channel B reset */
    call wrb(04, 11001110b);    /* 2 stop, no parity, brf = 64x */
    call wrb(02, 00100000b);    /* vector = 20h */
    call wrb(03, 11000000b);    /* rx 8 bits/char, no auto-enable */
    call wrb(05, 01100000b);    /* tx 8 bits/char */
    call wrb(06, 00000000b);
    call wrb(07, 00000000b);
    call wrb(09, 0000001b);     /* vector includes status */
    call wrb(10, 00000000b);
    call wrb(11, 01010110b);    /* rxc = txc = BRG, trxc = BRG out */
    call wrb(12, 00011000b);    /* to generate 1200 baud, x64 @ 4 mhz */
    call wrb(13, 00000000b);
    call wrb(14, 0000011b);     /* BRG source = SYS CLK, enable BRG */
    call wrb(15, 00000000b);    /* all ext status interrupts off */

/* enables */

    call wrb(03, 1100001b);     /* scc-b receive enable */
    call wrb(05, 11101010b);    /* scc-b transmit enable, dtr on, rts on */

/* enable interrupts - only for interrupt driven ASYNC I/O */

    call wrb(09, 0001001b);    /* master IE, vector includes status */
    call wrb(01, 0001001b);    /* tx,rx, ext interrupts enable */

end scc_init_b;
/*-----------------------------------------------*/
```

Figure 3. Initialization for ASYNC Communication

4. ASYNC Communication in Polling Mode

Figure 4 shows the procedures for reading in a received character from the 82530 (scc__in) and for writing out a character to the 82530 (scc__out) in the polling mode.

The scc__in procedure returns a byte value which is the character read in. The receiver is polled to find if a character has been received by the SCC. Only when a character has been received, the character is read in from the data port of the SCC channel B.

The scc__out procedure requires a byte parameter which is the character being written out. The transmit-
Figure 4. ASYNC Communication in Polling Mode

The character is polled for being ready to transmit the next character before writing the character out to the data port of SCC channel B.

Typical calls to these procedures are:

```
abc__variable = scc_in;
call scc_out (xyz__variable);
```

5. ASYNC Communication in Interrupt Mode

In contrast to polling for the receiver and/or the transmitter to be ready with/for the next character, the 82530 can be made to interrupt when it is ready to do receive or transmit.

The on-chip interrupt controller of the SCC can be made to operate in the vectored mode. In this mode, it generates interrupt vectors that are characteristic of the event causing the interrupt. For the example here, the vector base is programmed at 20h and ‘Vector Includes Status’ (VIS) mode is set - WR9 = XXX0XX01. Vectors and the associated events are:

<table>
<thead>
<tr>
<th>Vector</th>
<th>Procedure</th>
<th>Event Causing Interrupt</th>
</tr>
</thead>
<tbody>
<tr>
<td>20h</td>
<td>txintr_b</td>
<td>ch_b - transmit buffer empty</td>
</tr>
<tr>
<td>22h</td>
<td>esi_b</td>
<td>ch_b - external/status change</td>
</tr>
<tr>
<td>24h</td>
<td>rxintr_b</td>
<td>ch_b - receive character available</td>
</tr>
<tr>
<td>26h</td>
<td>src_b</td>
<td>ch_b - special receive condition</td>
</tr>
<tr>
<td>28h</td>
<td>txintr_a</td>
<td>ch_a - transmit buffer empty</td>
</tr>
<tr>
<td>2ah</td>
<td>esi_a</td>
<td>ch_a - external/status change</td>
</tr>
<tr>
<td>2ch</td>
<td>rxintr_a</td>
<td>ch_a - receive character available</td>
</tr>
<tr>
<td>2eh</td>
<td>src_a</td>
<td>ch_a - special receive condition</td>
</tr>
</tbody>
</table>

**NOTE:**
Odd vector numbers do not exist.

Figure 5 shows the interrupt procedures for the channel B operating in ASYNC mode. The transmitter buffer empty interrupt occurs when the transmitter can accept one more character to output. In the interrupt procedure for transmit, the byte char_out_530 is output. Following this, is an epilogue that is common to all the
**Interrupt procedures**: the first statement is an end of interrupt command to the 82530; note that it is issued to channel A - and the second is an End of Interrupt (EOI) command to the 80186 interrupt controller which is, in fact, receiving the interrupt from the 82530.

The receive buffer full interrupt occurs when the receiver has at least one character in its buffer, waiting to be read in by the CPU.

The esi_b is not enabled to occur and src_b cannot occur in the ASYNC mode unless the receiver is overrun or a parity error occurs.

```c
/* channel B interrupt procedures */
taxintr_b: procedure interrupt 20h;
   output (ch_b_data) = char_out_330;
   call wra(00, 3Bh); /* reset highest IUS */
   output (eoir_1B6) = 8000h; /* non specific EOI */
   return;
end taxintr_b;
esi_b:    procedure interrupt 22h;
   call wrb(00, 10h); /* reset ESI */
   call wra(00, 3Bh); /* reset highest IUS */
   output (eoir_1B6) = 8000h; /* non specific EOI */
   return;
end esi_b;
rxintr_b: procedure interrupt 24h;
   char_in_330 = input (ch_b_data);
   call wra(00, 3Bh); /* reset highest IUS */
   output (eoir_1B6) = 8000h; /* non specific EOI */
   return;
end rxiintr_b;
src_b:    procedure interrupt 26h;
   call wrb(00, 30h); /* error reset */
   call wra(00, 3Bh); /* reset highest IUS */
   output (eoir_1B6) = 8000h; /* non specific EOI */
   return;
end src_b;
```

Figure 5. ASYNC Communication in Interrupt Mode

2-426
6. Initialization for SDLC Communication

Channel A of the SCC is programmed for being used for SDLC operation. It uses the DMA channels on the 80186. Figure 6 shows the initialization procedure for channel A. The comments in the software show the effect of each statement. The on-chip Baud Rate Generator is used to generate a clock of 125 kHz both for reception and transmission. This procedure is just to prepare the channel A for SDLC operation. The actual transmission and reception of frames is done using the procedures described further.

7. SDLC Frame Reception

Figure 7 shows the entire set-up necessary to receive a SDLC frame. First the DMA controller is programmed with the receive buffer address (@Rx_buff), byte count, mode etc and is also enabled. Then a flag indicating reception of the frame is reset. An Error Reset command is issued to clear up any pending error conditions. The receive interrupt is enabled to occur at the end of frame reception (Special Receive Condition); lastly, the receiver is enabled and put in the Hunt mode (to detect the SDLC flag). When the first flag is detected on the RxDA pin, it goes from the Hunt to the Sync mode. It receives the frame and the end of frame interrupt (src_b, vector = 2eh) occurs.

8. SDLC Frame Transmission

Figure 8 shows the procedure for transmitting a SDLC frame once channel A is initialized. The DMA controller is initialized with the transmit buffer address (@tx_buff) - note, it is the second byte of the transmit buffer - and the byte count - again one less than the total buffer length. This is done because the first byte in the buffer is output directly using an I/O instruction and not by DMA. Then the flag indicating frame transmitted is reset. The events following are very critical in sequence:

a. Reset external status interrupts
b. Enable the transmitter
c. Reset transmit CRC
d. Enable transmitter underrun interrupt
e. Enable the DMA controller
f. Output first byte of the transmit block to data port
g. Reset Transmit Underrun Latch

```c
/*-----------------------------------------------*/
scc_init_a: procedure;
/* scc ch A register initialization for SDLC mode */
call wra(09, 10000000b);  /* channel A reset */
call wra(04, 00100000b);  /* SDLC mode */
call wra(01, 01100000b);  /* DMA for Rx */
call wra(03, 11000000b);  /* 8 bit Rx char, Rx disable */
call wra(05, 01100000b);  /* 8 bit Tx char, Tx disable */
call wra(06, 01010101b);  /* node address */
call wra(07, 01111110b);  /* SDLC flag */
call wra(11, 01010110b);  /* rxc = txc = BRG, trxc = BRQ out */
call wra(12, 00011110b);  /* to generate 125 Kbaud, xl @ 4 mhz */
call wra(13, 00000000b);  /* preset CRC, NRZ encoding */
call wra(14, 00001110b);  /* BRG source = SYS CLK, DMA for Tx */
call wra(15, 00000000b);  /* all ext status interrupts off */
/* enables */
call wra(14, 00001111b);  /* enable : BRG */
call wra(01, 11100000b);  /* enable : draq */
call wra(09, 00010010b);  /* master IE, vector includes status */
end scc_init_a;
/*-----------------------------------------------*/
```

Figure 6. Initialization for SDLC Communication
/*-----------------------------------------------*/
rx_init: procedure;

declare dma_0_mode literally '1010001001000000b'; /* src=IO, dest=M(inc), sync=src, TC, noint, priority, byte */
outword(dma_0_dpl) = low16(rx_buff);
outword(dma_0_dph) = high16(rx_buff);
outword(dma_0_sp1) = ch_a_data;
outword(dma_0_sph) = 0;
outword(dma_0_tc) = block_length + 2; /* +2 for CRC */
outword(dma_0_cw) = dma_0_mode or 0006h; /* start DMA channel 0 */
frame_rec = 0, /* reset frame received flag */
call wra(00, 30h); /* error reset */
call wra(01, 11111001b); /* sp. cond intr only, ext int enable */
call wra(03, 11010001b); /* enable receiver, enter hunt mode */
end rx_init;

Figure 7. SDLC-DMA Frame Reception

/*-----------------------------------------------*/

tx_init: procedure;

declare dma_1_mode literally '00010101001000000b'; /* src=M(inc), dest=IO, sync=dest, TC, noint, noprior, byte */
outword(dma_1_spl) = low16(tx_buff(1));
outword(dma_1_sph) = high16(tx_buff(1));
outword(dma_1_dpl) = ch_a_data;
outword(dma_1_dph) = 0;
outword(dma_1_tc) = block_length - 1; /* -1 for first byte */
frame_tx = 0; /* reset frame transmitted flag */
call wra(00, 00010000b); /* reset ESI */
call wra(05, 01010111b); /* enable transmitter */
call wra(00, 10101000b); /* reset tx CRC, TxINT pending */
call wra(15, 01000000b); /* enable : TxU int */
outword(dma_1_cw) = dma_1_mode or 0006h; /* start DMA channel 1 */
output(ch_a_data) = tx_buff(0); /* first byte - address field */
call wra(00, 11000000b); /* Reset Tx Underrun latch */
end tx_init;

Figure 8. SDLC-DMA Frame Transmission

2-428
/* channel A interrupt procedures */

txintr_a: procedure interrupt 28h;
  call wra(00,38h); /* reset highest IUS */
  output (eoir_186) = 8000h; /* non specific EDI */
  return;
end txintr_a;

esi_a: procedure interrupt 2ah;
  call wra(00,10h); /* reset ESI */
  tx_stat = rra(0); /* read in status */
  frame_tx = 0ffh; /* set frame transmitted flag */
  call wra(00,38h); /* reset highest IUS */
  output (eoir_186) = 8000h; /* non specific EDI */
  return;
end esi_a;

rxintr_a: procedure interrupt 2ch;
  call wra(00,38h); /* reset highest IUS */
  output (eoir_186) = 8000h; /* non specific EDI */
  return;
end rxintr_a;

src_a: procedure interrupt 2eh;
  rx_stat = rra(1); /* error reset */
  call wra(00,30h); /* disable rx */
  frame_recd = 0ffh; /* set frame received flag */
  call wra(00,38h); /* reset highest IUS */
  output (eoir_186) = 8000h; /* non specific EDI */
  return;
end src_a;

/* ---------------------------------------------*/

Figure 9. SDLC-DMA Interrupt Routines
The frame gets transmitted out with all bytes, except the first one, being fetched by the SCC using the DMA controller. At the end of the block the DMA controller stops supplying bytes to the SCC. This makes the transmitter underrun. Since the Transmitter Underrun Latch is in the reset state at this moment, the CRC bytes are appended by the SCC at the end of the transmit block going out. An External Status Change interrupt (esi_a, vector = 2ah) is generated with the bit for transmitter underrun set in RR0 register. This interrupt occurs when the CRC is being transmitted out and not when the frame is completely transmitted out.

9. SDLC Interrupt Routines

Figure 9 shows all the interrupt procedures for channel A when operating in the SDLC mode. The procedures of significance here are esi_a and src_a.

The end of frame reception results in the src_a procedure getting executed. Here the status in register RR1 is stored in a variable rx_stat for future examination. Any error bits set in status are reset, receiver is disabled and the flag indicating reception of a new frame is set.

The esi_a procedure is executed when CRC of the transmitted frame is just going out of the SCC. Reset External Status Interrupt command is executed, the external status is stored in a variable tx_stat for future examination and the flag indicating transmission of the frame is set.

End of frame processing is required after both of these interrupt procedures. It involves looking at rx_stat and tx_stat and checking if the desired operation was successful. The buffers used, may have to be recovered or new ones obtained to start another frame transmission or reception.

CONCLUSIONS

This article should ease the process of writing a complete data link driver for ASYNC and SDLC modes since most of the hardware dependent procedures are illustrated here. It was a conscious decision to make the procedures as small and easy to understand as possible. This had to be done at the expense of making the procedures general and not dealing with various exception conditions that can occur.

REFERENCES

1. 82530 Data Sheet, Order #230834-001
2. 82530 SCC Technical Manual, Order #230925-001
APPENDIX A
82530—BAUD RATE GENERATORS

The 82530 has two Baud Rate Generators (BRG) on chip—one for each channel. They are used to provide the baud rate or serial clock for receive and transmit operations. This article describes how the BRG can be programmed and used.

The BRG for each channel is totally independent of each other and have to be programmed separately for each channel. This article describes how any one of the two BRGs can be programmed for operation. To use the BRG, four steps have to be performed:

1. Determine the Baud Rate Time Constant (BRTC) to be programmed into registers WR12 (LSB) and WR13 (MSB).
2. Program in register WR11, to specify where the output of the BRG must go to.
3. Program the clock source to the BRG in register WR14.
4. Enable the BRG.

**Step 1: Baud Rate Time Constant (BRTC)**

The BRTC is determined by a simple formula:

\[
BRTC = \frac{\text{Serial Clock Frequency}}{2 \times (\text{Baud Rate} \times \text{Baud Rate Factor})} - 2
\]

**Example:**

For Serial Clock Frequency = 4 MHz

- Baud Rate = 9600
- Baud Rate Factor = 16

\[
BRTC = \frac{4000000}{2 \times (9600 \times 16)} - 2
\]

\[
= 13.021 - 2 = 11.021
\]

---

**Figure 1. Write Register 11**
Table 1. BRTC - Baud Rate Time Constant

<table>
<thead>
<tr>
<th>Baud Rate Factor</th>
<th>1</th>
<th>16</th>
<th>32</th>
<th>64</th>
</tr>
</thead>
<tbody>
<tr>
<td>9600</td>
<td>206.333</td>
<td>11.021</td>
<td>4.510</td>
<td>1.255</td>
</tr>
<tr>
<td>4800</td>
<td>414.667</td>
<td>24.042</td>
<td>11.021</td>
<td>4.510</td>
</tr>
<tr>
<td>2400</td>
<td>831.333</td>
<td>50.083</td>
<td>24.042</td>
<td>11.021</td>
</tr>
<tr>
<td>1200</td>
<td>1664.667</td>
<td>102.167</td>
<td>50.083</td>
<td>24.042</td>
</tr>
<tr>
<td>600</td>
<td>3331.333</td>
<td>206.333</td>
<td>102.167</td>
<td>50.083</td>
</tr>
<tr>
<td>300</td>
<td>6664.667</td>
<td>414.667</td>
<td>206.333</td>
<td>102.167</td>
</tr>
</tbody>
</table>

Since only integers can be written into the registers WR12/WR13 this will have to be rounded off to 11 and it will result in an error of:

\[
\text{fraction} \times 100 = \frac{0.021}{11.021} \times 100 = 0.19\%
\]

This error indicates that the baud rate signal generated by the BRG does not provide the exact frequency required by the system. This error is more serious for smaller baud rate factors. For asynchronous systems, errors up to 5% are considered acceptable.

Note that for BRTC = 0, BRG output frequency = 1/4 × Serial Clock Freq.

Table 1 shows the BRTC for a 4 MHz serial clock with various baud rates on the Y - axis and baud rate factors on the X - axis. The constant that is really programmed into registers WR12/WR13 is the integer closest to the BRTC value shown in the table.

**Step 2: BRG Output**

The output of the BRG can be directed to the Receiver, Transmitter and the TRxC output. This is programmed by setting bits D6 D5, bits D4 D3, and bits D1 D0 in register WR11 to 10. See Figure 1. The output of the BRG can also be directed to the Digital Phase Locked Loop (DPLL) for the on-chip decoding of the NRZI encoded received data signal. This is done by writing 100 into bits D7 D6 D5 of register WR14 as shown in Figure 2.

**Step 3: BRG Source Clock**

Register WR14 is used to select the input clock to the BRG. See Figure 2.

**Step 4: BRG Enable**

This is the last step where bit D0 of WR14 is set to start the BRG. The BRG can also be disabled by resetting this bit.
APPENDIX B
MODEM CONTROL PINS ON THE 82530

Introduction

This article describes how the CTS and CD pins on the 82530 behave and how to write software to service these pins. The article explains when the External Status Interrupt occurs and how and when to issue the Reset External/Status Interrupt command to reliably determine the state of these pins.

Bits D3 and D5 of register RR0 show the inverted state of logic levels on CD and CTS pins respectively. It is important to note that the register RR0 does not always reflect the current state of the CD and CTS pins. Whenever a Reset External/Status Interrupt (RESI) command is issued, the (inverted) states of the CD and the CTS pins get updated and latched into the RR0 register and the register RR0 then reflect the inverted state of the CD and CTS pins at the time of the write operation to the chip. On channel or chip reset, the inverted state of CD and CTS pins get latched into RR0 register.

Normally, a transition on any of the pins does not necessarily change the corresponding bit(s) in RR0. In certain situations it does and in some cases it does not. A sure way of knowing the current state of the pins is to read the register RR0 after a RESI command.

There are two cases:
I. External/Status Interrupt (ESI) enabled.
II. Polling (ESI disabled).

Case I: External Status Interrupt (ESI) Enabled

Whenever ESI is enabled, an interrupt can occur whenever there is a transition on CD or CTS pins - the IE bits for CD and/or CTS must also be set in WR15 for the interrupt to be enabled.

In this case, the first transition on any of these pins will cause an interrupt to occur and the corresponding bit in RR0 to change (even without the RESI command). A RESI command resets the interrupt line and also latches in the current state of both the CD and the CTS pins. If there was just one transition the RESI does not really change the contents of RR0.

If there are more than one transitions, either on the same pin or one each on both pins or multiple on both pins, the interrupt would get activated on the first transition and stay active. The bit in RR0 corresponding only to the very first transition is changed. All subsequent transitions have no effect on RR0. The first transition, in effect, freezes all changes in RR0. The first RESI command, as could be expected, latches the final (inverted) state of the CD and CTS pins into the RR0 register. Note that all the intermediate transitions on the pins are lost (because the response to the interrupt was not fast enough). The interrupt line gets reset for only a brief moment following the first RESI command. This brief moment is approximately 500 ns for the 82530. After that the interrupt becomes active again. A second RESI command is necessary to reset the interrupt. Two RESI commands resets the interrupt line independent of the number of transitions occurred.

Whenever operating with ESI enabled, it is recommendable to issue two back-to-back RESI commands and then read the RR0 register to reliably determine the state of the CD and CTS pins and also to reset the interrupt line in case multiple transitions may have occurred.

State Diagram

![State Diagram Image]
Case II: Polling RR0 for CD and CTS Pins

If RR0 is polled for determining the state of the CD and CTS pins, then the External/Status Interrupt (ESI) is kept disabled. In this case the bits in RR0 may not change even for the first transition. The best way to handle this case is to always issue a RESI command before reading in the RR0 register to determine the state of CD and CTS pins. Note, however, if two back-to-back RESI commands were to be issued every time before reading in the RR0 register, the first subsequent transition will change the corresponding bit in RR0.

The state diagram above illustrates how each transition on CD and CTS pins affect the 82530 and what effect the RESI command has.

State 0

It is entered on reset. No ESI due to CTS or CD are pending in this state. Any transition on CTS or CD pins lead to the state 1 accompanied by an immediate change in the RR0 register.

State 1

Interrupt is active (if enabled). If a RESI command is issued, state 0 is reached where interrupt is again inactive. However, a further transition on CTS or CD pin leads to state 2 without an immediate change in RR0 register.

State 2

Interrupt is active (if enabled). Any further transitions have no effect. A RESI command leads to state 1, temporarily making the interrupt inactive.

CONCLUSIONS

Register RR0 does not always reflect the current (inverted) state of the CD and CTS pins. The most reliable way to determine the state of the pins in interrupt or polling mode is to issue two back-to-back RESI commands and then read RR0. While polling, the second RESI is redundant but harmless. When issuing the back-to-back RESI commands to 82530 note that the separation between the two write cycles should be at least 6 CLK + 200 ns; otherwise the second RESI will be ignored.
APPENDIX C
THE 82530 SCC - 80186 INTERFACE AP BRIEF

INTRODUCTION

The object of this document is to give the 82530 system designer an in-depth worst case design analysis of the typical interface to a 80186 based system. This document has been revised to include the new specifications for the 6 MHz 82530. The new specifications yield better margins and a 1 wait state interface to the CPU (2 wait states are required for DMA cycles). These new specifications will appear in the 1987 data sheet and advanced specification information can be obtained from your local Intel sales office. The following analysis includes a discussion of how the interface TTL is utilized to meet the timing requirements of the 80186 and the 82530. In addition, several optional interface configurations are also considered.

INTERFACE OVERVIEW

The 82530 - 80186 interface requires the TTL circuitry illustrated in Figure 1. Using five 14 pin TTL packages, 74LS74, 74AS74, 74AS08, 74AS04, and 74LS32, the following operational modes are supported:

- Polled
- Interrupt in vectored mode
- Interrupt in non-vectored mode
- Half-duplex DMA on both channels
- Full-duplex DMA on channel A

A brief description of the interface functional requirements during the five possible BUS operations follows below.

\[
\text{Figure 1. 82530-80186 Interface}
\]
READ CYCLE: The 80186 read cycle requirements are met without any additional logic, Figure 2. At least one wait state is required to meet the 82530 tAD access time.

WRITE CYCLE: The 82530 requires that data must be valid while the WR pulse is low, Figure 3. A D Flip-Flop delays the leading edge of WR until the falling edge of CLOCKOUT when data is guaranteed valid and WR is guaranteed active. The CLOCKOUT signal is inverted to assure that WR is active low before the D Flip-Flop is clocked. No wait states are necessary to meet the 82530's WR cycle requirements, but one is assumed from the RD cycle.

INTA CYCLE: During an interrupt acknowledge cycle, the 80186 provides two INTA pulses, one per bus cycle, separated by two idle states. The 82530 expects only one long INTA pulse with a RD pulse occurring only after the 82530 IEI/IEO daisy chain settles. As
illustrated in Figure 4, the INTA signal is sampled on the rising edge of CLK (82530). Two D Flip-Flops and two TTL gates, U2 and U5, are implemented to generate the proper INTA and RD pulses. Also, the INT signal is passively pulled high, through a 1 k resistor, and inverted through U3 to meet the 80186's active high requirement.

DMA CYCLE: Conveniently, the 80186 DMA cycle timings are the same as generic read and write operations. Therefore, with two wait states, only two modifications to the DMA request signals are necessary. First, the RDYREQA signal is inverted through U3 similar to the INT signal, and second the DTR/REQA signal is conditioned through a D Flip-Flop to prevent inadvertent back to back DMA cycles. Because the 82530 DTR/REQA signal remains active low for over five CLK (82530)'s, an additional DMA cycle could occur. This uncertain condition is corrected when U4 resets the DTR/REQ signal inactive high. Full Duplex on both DMA channels can easily be supported with one extra D Flip-Flop and an inverter.

RESET: The 82530 does not have a dedicated RESET input. Instead, the simultaneous assertion of both RD and WR causes a hardware reset. This hardware reset is implemented through U2, U3, and U4.

ALTERNATIVE INTERFACE CONFIGURATIONS

Due to its wide range of applications, the 82530 interface can have many varying configurations. In most of these applications the supported modes of operation need not be as extensive as the typical interface used in this analysis. Two alternative configurations are discussed below.

8288 BUS CONTROLLER: An 80186 based system implementing an 8288 bus controller will not require the preconditioning of the WR signal through the D Flip-Flop U4. When utilizing an 8288, the control signal IOWC does not go active until data is valid, therefore, meeting the timing requirements of the 82530. In such a configuration, it will be necessary to logically OR the IOWC with reset to accommodate a hardware reset operation.

NON-VECTORED INTERRUPTS: If the 82530 is to be operated in the non-vectored interrupt mode (B step only), the interface will not require U1 or U5. Instead, INTA on the 82530 should be pulled high, and pin 3 of U2 (RD AND RESET) should be fed directly into the RD input of the SCC. Obviously, the amount of required interface logic is application dependent and in many cases can be considerably less than required by the typical configuration, supporting all modes of SCC operation.

DESIGN ANALYSIS

This design analysis is for a typical microprocessor system, pictured in Figure 5. The Timing analysis assumes an 8 MHz 80186 and a 6 MHz 82530 being clocked at 4 MHz. The 4 MHz clock is the 80186 CLKOUT divided by two by a flip-flop (U6). Also, included in the analysis are bus loading, and TTL-MOS compatibility considerations.
**Bus Loading and Voltage Level Compatibilities**

The data and address lines do not exceed the drive capability of either 80186 or the 82530. There are several control lines that drive more than one TTL equivalent input. The drive capability of these lines are detailed below.

**WR:** The WR signal drives U3 and U4.

- $\text{lof (2.0 mA)} > \text{lii } (-0.4 \text{ mA } - 0.5 \text{ mA})$
- $\text{loh } (-400 \mu\text{A}) > \text{lih } (20 \mu\text{A} + 20 \mu\text{A})$

**PC55:** The PC55 signal drives U2 and U4.

- $\text{lof (2.0 mA)} > \text{lii } (-0.5 \text{ mA } - 0.5 \text{ mA})$
- $\text{loh } (-400 \mu\text{A}) > \text{lih } (20 \mu\text{A} + 20 \mu\text{A})$

**INTA:** The INTA signal drives 2(U1) and U5.

- $\text{lof (2.0 mA)} > \text{lii } (-0.4 \text{ mA } - 0.8 \text{ mA } - 0.4 \text{ mA})$
- $\text{loh } (-400 \mu\text{A}) > \text{lih } (20 \mu\text{A} + 40 \mu\text{A} + 20 \mu\text{A})$

All the 82530 I/O pins are TTL voltage level compatible.

**TIMING ANALYSIS**

Certain symbolic conventions are adhered to throughout the analysis below and are introduced for clarity.

1. All timing variables with a lower case first letter are 82530 timing requirements or responses (i.e., $t_{RR}$).
2. All timing variables with Upper case first letters are 80186 timing responses or requirements unless preceded by another device's alpha-numeric code (i.e., Tclvl or '373 Tpd).
3. In the write cycle analysis, the timing variable $T_{PDW186-WR530}$ represents the propagation delay between the leading or trailing edge of the WR signal leaving the 80186 and the WR edge arrival at the 82530 WR input.

**Read Cycle**

1. $t_{AR}$: Address valid to RD active set up time for the 82530. Since the propagation delay is the worst case path in the assumed typical system, the margin is calculated only for a propagation delay constrained and not an ALE limited path. The spec value is 0 ns minimum.

   $\text{lof} T_{TCOL} - T_{TCOL}^{\text{max}} = 245 T_{PD}^{\text{max}} + T_{TCR}^{\text{min}} + 2 T_{U2}^{\text{max}} - t_{AR}^{\text{min}}$

   $= 125 - 55 + 10 + 2(2) - 0 = 63.2$ ns margin
2. tRA: Address to RD inactive hold time. The ALE delay is the worst case path and the 82530 requires 0 ns minimum.
   \[ T_{AD} = T_{Cl} + T_{Chl} + T_{R} + 375 \text{ ns} \]
   \[ T_{Cl} = T_{pd(min)} - T_{pd(max)} \]
   \[ T_{Chl} = 2(U2) T_{pd(max)} \]
   \[ T_{R} = 82530 \text{ ns} \]
   \[ T_{AD} = 55 + 5 + 8 - 2(5.5) = 2 \text{ ns margin} \]

3. tCLR: CS active low to RD active low set up time. The 82530 spec value is 0 ns minimum.
   \[ T_{CLR} = T_{pd(max)} - T_{pd(min)} \]
   \[ T_{pd(max)} = U2 T_{pd(max)} \]
   \[ T_{pd(min)} = U2 T_{pd(min)} \]
   \[ T_{CLR} = 55 + 5 + 1 + 2 = 62 \text{ ns} \]

4. tRCS: RD inactive to CS inactive hold time. The 82530 spec calls for 0 ns minimum.
   \[ T_{RCS} = T_{pd(min)} - T_{pd(max)} \]
   \[ T_{pd(max)} = U2 T_{pd(max)} \]
   \[ T_{pd(min)} = U2 T_{pd(min)} \]
   \[ T_{RCS} = 35 - 1 - 5.5 = 28.5 \text{ ns margin} \]

5. tCHR: CS inactive to RD active set up time. The 82530 requires 5 ns minimum.
   \[ T_{CHR} = T_{pd(max)} - T_{pd(min)} \]
   \[ T_{pd(max)} = U2 T_{pd(max)} \]
   \[ T_{pd(min)} = U2 T_{pd(min)} \]
   \[ T_{CHR} = 125 + 55 - 10 - 1 + 2 = 131 \text{ ns margin} \]

6. tWR: RD pulse active low time. One 80186 wait state is included to meet the 150 ns minimum timing requirements of the 82530.
   \[ T_{WR} = T_{Chl} + T_{Chl} + T_{Chl} + T_{pd(max)} \]
   \[ T_{Chl} = T_{Cl} + T_{Cl} + T_{Cl} \]
   \[ T_{Chl} = 2(U2) T_{pd(max)} \]
   \[ T_{WR} = 125 + 55 + 10 - 1 - 2 + 5 = 131 \text{ ns margin} \]

7. tRDV: RD active low to data valid maximum delay for 80186 read data set up time (Tdvcl = 20 ns). The margin is calculated on the Propagation delay path (worst case).
   \[ T_{RDV} = T_{Cl} + T_{Cl} + T_{Cl} + T_{pd(max)} \]
   \[ T_{Cl} = T_{pd(min)} - T_{pd(max)} \]
   \[ T_{Cl} = 2(U2) T_{pd(max)} \]
   \[ T_{RDV} = 55 + 5 + 10 - 11 - 70 = 179 \text{ ns margin} \]

8. tDF: RD inactive to data output float delay. The margin is calculated to DEN active low of next cycle.
   \[ T_{DF} = T_{Cl} + T_{Cl} + T_{Cl} + T_{Cl} \]
   \[ T_{Cl} = T_{pd(max)} - T_{pd(min)} \]
   \[ T_{Cl} = 2(U2) T_{pd(max)} \]
   \[ T_{DF} = 250 + 55 + 10 + 11 - 70 = 179 \text{ ns margin} \]

9. tAD: Address required valid to read data valid maximum delay. The 82530 spec value is 325 ns maximum.
   \[ T_{AD} = T_{Cl} + T_{Cl} + T_{Cl} + T_{Cl} \]
   \[ T_{Cl} = T_{pd(max)} - T_{pd(min)} \]
   \[ T_{Cl} = 2(U2) T_{pd(max)} \]
   \[ T_{AD} = 375 + 125 + 55 + 20.8 + 15 - 14.2 - 20 - 325 = 65 \text{ ns margin} \]

**Write Cycle**

1. tAW: Address required valid to WR active low set up time. The 82530 spec is 0 ns minimum.
   \[ T_{AW} = T_{Cl} + T_{Cl} + T_{Cl} \]
   \[ T_{Cl} = T_{Cl} + T_{Cl} \]
   \[ T_{Cl} = U2 T_{Cl} \]
   \[ T_{AW} = 125 + 55 = 20.8 + [125 + 5 + 1 + 4.4] = 70.6 \text{ ns margin} \]

2. tWA: WR inactive to address invalid hold time. The 82530 spec is 0 ns.
   \[ T_{WA} = T_{Cl} + T_{Cl} + T_{Cl} \]
   \[ T_{Cl} = U2 T_{Cl} \]
   \[ T_{WA} = 150 + 5 + 8 - [5.5 + 3 + 7.1] = 264 \text{ ns margin} \]

3. tCL: Chip select active low to WR active low hold time. The 82530 spec is 0 ns.
   \[ T_{CL} = T_{Cl} + T_{Cl} \]
   \[ T_{Cl} = U2 T_{Cl} \]
   \[ T_{CL} = 150 + 5 + 8 - [5.5 + 3 + 7.1] = 375 + 5 + 8 = 150 \text{ ns margin} \]

4. tCS: WR invalid to Chip Select invalid hold time. The 82530 spec is 0 ns.
   \[ T_{CS} = T_{Cl} + T_{Cl} \]
   \[ T_{Cl} = U2 T_{Cl} \]
   \[ T_{CS} = 150 + 5 + 8 - [5.5 + 3 + 7.1] = 183.9 \text{ ns margin} \]

5. tCHW: Chip Select inactive high to WR active low set up time. The 82530 spec is 5 ns.
   \[ T_{CHW} = T_{Cl} + T_{Cl} + T_{Cl} \]
   \[ T_{Cl} = U2 T_{Cl} \]
   \[ T_{CHW} = 150 + 5 + 8 - [5.5 + 3 + 7.1] = 20.9 \text{ ns margin} \]

6. tTW: WR active low pulse. 82530 requires a minimum of 60 ns from the falling to the rising edge of WR. This includes one wait state.
**INTA Cycle:**

1. **tIC:** This 82530 spec implies that the INTA signal is latched internally on the rising edge of CLK (82530). Therefore the maximum delay between the 80186 asserting INTA active low or inactive high and the 82530 internally recognizing the new state of INTA is the propagation delay through U1 plus the 82530 CLK period.

   \[ T_{IC} = U_1 \text{Tpd(max)} + 82530 \text{ CLK period} \]

   = 45 + 250 = 295 ns

2. **tCI:** rising edge of CLK to INTA hold time. This spec requires that the state of INTA remains constant for 100 ns after the rising edge of CLK. If this spec is violated any change in the state of INTA may not be internally latched in the 82530. \( t_{CI} \) becomes critical at the end of an INTA cycle when INTA goes inactive. When calculating margins with \( t_{CI} \), an extra 82530 CLK period must be added to the INTA inactive delay.

3. **tIW:** \( t_{IW} \) inactive high to \( t_{WR} \) active low minimum setup time. The spec pertains only to 82530 RD cycles and has a value of 55 ns. The margin is calculated in the same manner as \( t_{IW} \).

   \[ T_{IW} = 125 - 55 - (5.5 + 7.1) + 5 + 10 = 69.4 \text{ ns margin} \]

4. **tWI:** \( t_{WI} \) inactive high to \( t_{WR} \) active low minimum hold time. The spec is 0 ns and the margin assumes CLK coincident with INTA.

   \[ T_{IW} = 125 - 55 - (5.5 + 3.71) + 5 + 10 = 69.4 \text{ ns margin} \]

5. **tIR:** INTA inactive high to RD active low minimum setup time. Since the INTA inactive high to WR active low setup time. The spec is 0 ns and the margin assumes CLK coincident with INTA.

   \[ T_{IR} = 125 - 55 - (5.5 + 3.71) + 5 + 10 = 69.4 \text{ ns margin} \]

6. **tRI:** RD inactive high to INTA active low minimum hold time. The spec is 0 ns and the margin assumes CLK coincident with INTA.

   \[ T_{RI} = 125 - 55 - (5.5 + 3.71) + 5 + 10 = 69.4 \text{ ns margin} \]

7. **tID:** INTA active low to RD active low minimum setup time. This parameter is system dependent. For any SCC in the daisy chain, tID must be greater than the sum of tCEQ for the highest priority device in the daisy chain, tEI for this particular SCC, and tEIEO for each device separating them in the daisy chain. The typical system with only 1 SCC requires tID to be greater than tCEQ. Since tEI occurs coincidently with tCEQ and it is smaller it can be neglected. Additionally, tEIEO does not have any relevance to a system with only one SCC. Therefore \( t_{ID} > t_{CEQ} = 250 \text{ ns} \).

   \[ T_{ID} = 500 + 250 - 45 + 250 + 5 + 6 + 2 - 250 = 148 \text{ ns margin} \]

8. **tIDV:** RD active low to interrupt vector valid delay. The 80186 expects the interrupt vector to be valid on the data bus a minimum of 20 ns before T4 of the second acknowledge cycle (Tdvcl). \( t_{IDV} \) spec is 100 ns maximum.

   \[ T_{IDV} = 375 - 70 - 25 - 5.5 - 100 - 14.2 - 20 = 140.3 \text{ ns margin} \]
9. tII: RD pulse low time. The 82530 requires a minimum of 125 ns.

\[ 3 \text{Tclcl} - \text{Tcvctv(max)} - \text{U5 Tpd(max)} - \text{U2 Tpd(max)} + \text{Tcvtx(min)} + \text{U5 Tpd(min)} + \text{U2 Tpd(min)} - \text{tII(min)} = 375 - 70 - 25 - 5.5 + 5 + 6 + 1.5 - 125 = 162 \text{ ns margin} \]

**DMA Cycle**

Fortunately, the 80186 DMA controller emulates CPU read and write cycle operation during DMA transfers. The DMA transfer timings are satisfied using the above analysis. Because of the 80186 DMA request input requirements, two wait states are necessary to prevent inadvertent DMA cycles. There are also CPUDMA intracycle timing considerations that need to be addressed.

1. tDRD: RD inactive high to DTRREQ (REQUEST) inactive high delay. Unlike the READYREQ signal, DTRREQ does not immediately go inactive after the requested DMA transfer begins. Instead, the DTRREQ remains active for a maximum of 5 \( t\text{CY} \) + 300 ns. This delayed request pulse could trigger a second DMA transfer. To avoid this undesirable condition, a D Flip Flop is implemented to reset the DTRREQ signal inactive low following the initiation of the requested DMA transfer. To determine if back to back DMA transfers are required in a source synchronized configuration, the 80186 DMA controller samples the service request line 25 ns before T1 of the deposit cycle, the second cycle of the transfer.

\[ 4 \text{Tclcl} - \text{Tclcsv(max)} - \text{U4 Tpd(max)} - \text{Tdrqcl(min)} = 500 - 66 - 10.5 - 25 = 398.5 \text{ ns margin} \]

2. tRRI: 82530 RD active low to REQ inactive high delay. Assuming source synchronized DMA transfers, the 80186 requires only one wait state to meet the tRRI spec of 200 ns. Two are included for consistency with tWRI.

\[ 2 \text{Tclcl} + 2(\text{Tclcwait state}) - \text{Tcrl(max)} - 2(\text{U2}) \text{Tpd(max)} - \text{Tdrqcl} - \text{tRRI} = 2(125) + 2(125) - 70 - 2(5.5) - 200 = 219 \text{ ns margin} \]

3. tWRI: 82530 WR active low to REQ inactive high delay. Assuming destination synchronized DMA transfers, the 80186 needs two wait states to meet the tWRI spec. This is because the 80186 DMA controller samples requests two clocks before the end of the deposit cycle. This leaves only 1 Tccl + n(wait states) minus WR active delay for the 82530 to inactivate its REQ signal.

\[ 4 \text{Tclcl} + 2(\text{Tclcwait state}) - \text{Tcvctv(min)} - \text{TpdWR}_{186} - \text{WR}_{530} - \text{(LOW)} ] \text{Tclcl} - \text{Tcvctv(min)} + \text{U3 Tpd(max)} + \text{U4 Tpd(max)} - \text{Tdrqcl} - \text{tWRI} = 375 - 5 - [125 - 5 + 4.5 + 9.21 - 25 - 200] = 11.3 \text{ ns margin} \]

**NOTE:**

If one wait state DMA interface is required, external logic, like that used on the DTRREQ signal, can be used to force the 82530 REQ signal inactive.

4. tREC: CLK recovery time. Due to the internal data path, a recovery period is required between SCC bus transactions to resolve metastable conditions internal to the SCC. The DMA request lines are masked from requesting service until after the tREC has elapsed. In addition, the CPU should not be allowed to violate this recovery period when interleaving DMA transfers and CPU bus cycles. Software drivers or external logic should orchestrate the CPU and DMA controller operation to prevent tREC violation. In this example circuit, tREC could be improved by clocking the '530 with a 6 MHz clock.

**Reset Operation**

During hardware reset, the system RESET signal is asserted high for a minimum of four 80186 clock cycles (1000 ns). The 82530 requires WR and RD to be simultaneously asserted low for a minimum of 250 ns.

\[ 4 \text{Tclcl} - \text{U3 Tpd(max)} - 2(\text{U2}) \text{Tpd(max)} + \text{U4 Tpd(min)} - \text{tREC} = 1000 - 17.5 - 2(5.5) + 3.5 - 250 = 725 \text{ ns margin} \]
Other Components
8291A
GPIB TALKER/LISTENER

- Designed to interface microprocessors (e.g., 8048/49, 8051, 8080/85, 8086/88) to an IEEE Standard 488 Digital Interface Bus
- Programmable Data Transfer Rate
- Complete Source and Acceptor Handshake
- Complete Talker and Listener Functions with Extended Addressing
- Service Request, Parallel Poll, Device Clear, Device Trigger, Remote/Local Functions
- Selectable Interrupts
- On-Chip Primary and Secondary Address Recognition
- Automatic Handling of Addressing and Handshake Protocol
- Provision for Software Implementation of Additional Features

- 1–8 MHz Clock Range
- 16 Registers (8 Read, 8 Write), 2 for Data Transfer, the Rest for Interface Function Control, Status, etc.
- Directly interfaces to external non-inverting transceivers for connection to the GPIB
- Provides three addressing modes, allowing the chip to be addressed either as a major or a minor Talker/Listener with primary or secondary addressing
- DMA Handshake Provision Allows for Bus Transfers without CPU Intervention
- Trigger Output Pin
- On-Chip EOS (End of Sequence) Message Recognition Facilitates Handling of Multi-Byte Transfers

The 8291A is an enhanced version of the 8291 GPIB Talker/Listener designed to interface microprocessors to an IEEE Standard 488 Instrumentation Interface Bus. It implements all of the Standard's interface functions except for the controller. The controller function can be added with the 8292 GPIB Controller, and the 8293 GPIB Transceiver performs the electrical interface for Talker/Listener and Talker/Listener/Controller configurations.
The 8291A is an improved design of the 8291 GPIB Talker/Listener. Most of the functions are identical to the 8291, and the pin configuration is unchanged.

The 8291A offers the following improvements to the 8291:

1. **EOI** is active with the data as a ninth data bit rather than as a control bit. This is to comply with some additions to the 1975 IEEE-488 Standard incorporated in the 1978 Standard.

2. The BO interrupt is not asserted until RFD is true. If the Controller asserts ATN synchronously, the data is guaranteed to be transmitted. If the Controller asserts ATN asynchronously, the SH (Source Handshake) will return to SIDS (Source Idle State), and the output data will be cleared. Then, if ATN is released while the 8291A is addressed to talk, a new BO interrupt will be generated. This change fixes 8291 problems which caused data to be lost or repeated and a problem with the RQS bit (sometimes cannot be asserted while talking).

3. LLOC and REMC interrupts are setting flipflops rather than toggling flipflops in the interrupt back-up register. This ensures that the CPU knows that these state changes have occurred. The actual state can be determined by checking the LLO and REM status bits in the upper nibble of the Interrupt Status 2 Register.

4. DREQ is cleared by DACK (RD + WR). DREQ on the 8291 was cleared only by DACK which is not compatible with the 8089 I/O Processor.

5. The INT bit in Interrupt Status 2 Register is duplicated in bit 7 of the Address 0 Register. If software polling is used to check for an interrupt, INT in the Address 0 Register should be polled rather than the Interrupt Status 2 Register. This ensures that no interrupts are lost due to asynchronous status reads and interrupts.

6. The 8291A's Send EOI Auxiliary Command works on any byte including the first byte of a message. The 8291 did not assert EOI after this command for a one-byte message nor on two consecutive bytes.

7. To avoid confusion between holdoff on DAV versus RFD if a device is readdressed from a talker to a listener role or vice-versa during a holdoff, the “Holdoff on Source Handshake” has been eliminated. Only “Holdoff on Acceptor Handshake” is available.

8. The rsv local message is cleared automatically upon exit from SPAS if (APRS:STRS:SPAS) occurred. The automatic resetting of the bit after the serial poll is complete simplifies the service request software.

9. The SPASC interrupt on the 8291 has been replaced by the SPC (Serial Poll Complete) interrupt on the 8291A. SPC interrupt is set on exit from SPAS if APRS:STRS:SPAS occurred, indicating that the controller has read the bus status byte after the 8291A requested service. The SPASC interrupt was ambiguous because a controller could enter SPAS and exit SPAS generating two SPASC interrupts without reading the serial poll status byte. The SPC interrupt also simplifies the CPU's software by eliminating the interrupt when the serial poll is half way done.

10. The rtl Auxiliary Command in the 8291 has been replaced by Set and Clear rtl Commands in the 8291A. Using the new commands, the CPU has the flexibility to extend the length of local mode or leave it as a short pulse as in the 8291.

11. A holdoff RFD on GET, SDC, and DCL feature has been added to prevent additional bus activity while the CPU is responding to any of these commands. The feature is enabled by a new bit (B4) in the Auxiliary Register B.

12. On the 8291, BO could cease to occur upon IFC going false if IFC occurred asynchronously. On the 8291A, BO continues to occur after IFC has gone false even if it arrived asynchronously.

13. User's software can distinguish between the 8291 and the 8291A as follows:
   a) pon (00H to register 5)
   b) RESET (02H to register 5)
   c) Read Interrupt Status 1 Register. If BO interrupt is set, the device is the 8291. If BO is clear, it is the 8291A.

This can be used to set a flag in the user's software which will permit special routines to be executed for each device. It could be included as part of a normal initialization procedure as the first step after a chip reset.
Table 1. Pin Description

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Pin No.</th>
<th>Type</th>
<th>Name and Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>D&lt;sub&gt;0&lt;/sub&gt;–D&lt;sub&gt;7&lt;/sub&gt;</td>
<td>12–19</td>
<td>I/O</td>
<td>DATA BUS PORT: To be connected to microprocessor data bus.</td>
</tr>
<tr>
<td>RS&lt;sub&gt;0&lt;/sub&gt;–RS&lt;sub&gt;2&lt;/sub&gt;</td>
<td>21–23</td>
<td>I</td>
<td>REGISTER SELECT: Inputs, to be connected to three nonmultiplexed microprocessor address bus lines. Select which of the 8 internal read (write) registers will be read from (written into) with the execution of RD (WR).</td>
</tr>
<tr>
<td>CS</td>
<td>8</td>
<td>I</td>
<td>CHIP SELECT: When low, enables reading from or writing into the register selected by RS&lt;sub&gt;0&lt;/sub&gt;–RS&lt;sub&gt;2&lt;/sub&gt;.</td>
</tr>
<tr>
<td>RD</td>
<td>9</td>
<td>I</td>
<td>READ STROBE: When low with CS or DACK low, selected register contents are read.</td>
</tr>
<tr>
<td>WR</td>
<td>10</td>
<td>I</td>
<td>WRITE STROBE: When low with CS or DACK low, data is written into the selected register.</td>
</tr>
<tr>
<td>INT (INT)</td>
<td>11</td>
<td>O</td>
<td>INTERRUPT REQUEST: To the microprocessor, set high for request and cleared when the appropriate register is accessed by the CPU. May be software configured to be active low.</td>
</tr>
<tr>
<td>DREQ</td>
<td>6</td>
<td>O</td>
<td>DMA REQUEST: Normally low, set high to indicate byte output or byte input in DMA mode; reset by DACK.</td>
</tr>
<tr>
<td>DACK</td>
<td>7</td>
<td>I</td>
<td>DMA ACKNOWLEDGE: When low, resets DREQ and selects data in/data out register for DMA data transfer (actual transfer done by RD/WR pulse). Must be high if DMA is not used.</td>
</tr>
<tr>
<td>TRIG</td>
<td>5</td>
<td>O</td>
<td>TRIGGER OUTPUT: Normally low; generates a triggering pulse with 1 μsec min. width in response to the GET bus command or Trigger auxiliary command.</td>
</tr>
<tr>
<td>CLOCK</td>
<td>3</td>
<td>I</td>
<td>EXTERNAL CLOCK: Input, used only for T, delay generator. May be any speed in 1–8 MHz range.</td>
</tr>
<tr>
<td>RESET</td>
<td>4</td>
<td>I</td>
<td>RESET INPUT: When high, forces the device into an &quot;idle&quot; (initialization) mode. The device will remain at &quot;idle&quot; until released by the microprocessor, with the &quot;Immediate Execute pon&quot; local message.</td>
</tr>
<tr>
<td>DIO&lt;sub&gt;1&lt;/sub&gt;–DIO&lt;sub&gt;8&lt;/sub&gt;</td>
<td>28–35</td>
<td>I/O</td>
<td>8-BIT GPIB DATA PORT: Used for bidirectional data byte transfer between 8291A and GPIB via non-inverting external line transceivers.</td>
</tr>
<tr>
<td>DAV</td>
<td>36</td>
<td>I/O</td>
<td>DATA VALID: GPIB handshake control line. Indicates the availability and validity of information on the DIO&lt;sub&gt;1&lt;/sub&gt;–DIO&lt;sub&gt;8&lt;/sub&gt; and EOI lines.</td>
</tr>
<tr>
<td>NRFD</td>
<td>37</td>
<td>I/O</td>
<td>NOT READY FOR DATA: GPIB handshake control line. Indicates the condition of readiness of device(s) connected to the bus to accept data.</td>
</tr>
<tr>
<td>NDAC</td>
<td>38</td>
<td>I/O</td>
<td>NOT DATA ACCEPTED: GPIB handshake control line. Indicates the condition of acceptance of data by the device(s) connected to the bus.</td>
</tr>
<tr>
<td>ATN</td>
<td>26</td>
<td>I</td>
<td>ATTENTION: GPIB command line. Specifies how data on DIO lines are to be interpreted.</td>
</tr>
</tbody>
</table>
Table 1. Pin Description (Continued)

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Pin No.</th>
<th>Type</th>
<th>Name and Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>IFC</td>
<td>24</td>
<td>I</td>
<td>INTERFACE CLEAR: GPIB command line. Places the interface functions in a known quiescent state.</td>
</tr>
<tr>
<td>SRQ</td>
<td>27</td>
<td>O</td>
<td>SERVICE REQUEST: GPIB command line. Indicates the need for attention and requests an interruption of the current sequence of events on the GPIB.</td>
</tr>
<tr>
<td>REN</td>
<td>25</td>
<td>I</td>
<td>REMOTE ENABLE: GPIB command line. Selects (in conjunction with other messages) remote or local control of the device.</td>
</tr>
<tr>
<td>EOI</td>
<td>39</td>
<td>I/O</td>
<td>END OR IDENTITY: GPIB command line. Indicates the end of a multiple byte transfer sequence or, in conjunction with ATN, addresses the device during a polling sequence.</td>
</tr>
<tr>
<td>T/R1</td>
<td>1</td>
<td>O</td>
<td>EXTERNAL TRANSCEIVERS CONTROL LINE: Set high to indicate output data/signals on the DIO1–DIO8 and DAV lines and input signals on the NRFD and NDAC lines (active source handshake). Set low to indicate input data/signals on the DIO1–DIO8 and DAV lines and output signals on the NRFD and NDAC lines (active acceptor handshake).</td>
</tr>
<tr>
<td>T/R2</td>
<td>2</td>
<td>O</td>
<td>EXTERNAL TRANSCEIVERS CONTROL LINE: Set to indicate output signals on the EOI line. Set low to indicate expected input signal on the EOI line during parallel poll.</td>
</tr>
<tr>
<td>VCC</td>
<td>40</td>
<td>P.S.</td>
<td>POSITIVE POWER SUPPLY: (5V ± 10%).</td>
</tr>
<tr>
<td>GND</td>
<td>20</td>
<td>P.S.</td>
<td>CIRCUIT GROUND POTENTIAL.</td>
</tr>
</tbody>
</table>

NOTE:
All signals on the 8291A pins are specified with positive logic. However, IEEE 488 specifies negative logic on its 16 signal lines. Thus, the data is inverted once from D0–D7 to DIO0–DIO8 and non-inverting bus transceivers should be used.

Figure 3. 8291A System Diagram
THE GENERAL PURPOSE INTERFACE BUS (GPIB)

The General Purpose Interface Bus (GPIB) is defined in the IEEE Standard 488-1978 "Digital Interface for Programmable Instrumentation." Although a knowledge of this standard is assumed, Figure 4 provides the bus structure for quick reference. Also, Tables 2 and 3 reference the interface mnemonics and the interface messages respectively. Modified state diagrams for the 8291A are presented in Appendix A.

General Description

The 8291A is a microprocessor-controlled device designed to interface microprocessors, e.g., 8048/49, 8051, 8080/85, 8086/88 to the GPIB. It implements all of the interface functions defined in the IEEE-488 Standard except for the controller function. If an implementation of the Standard’s Controller is desired, it can be connected with an Intel® 8292 to form a complete interface.

The 8291A handles communication between a microprocessor-controlled device and the GPIB. Its capabilities include data transfer, handshake protocol, talker/listener addressing procedures, device clearing and triggering, service request, and both serial and parallel polling. In most procedures, it does not disturb the microprocessor unless a byte has arrived (input buffer full) or has to be sent out (output buffer empty).

The 8291A architecture includes 16 registers. Eight of these registers may be written into by the microprocessor. The other eight registers may be read by the microprocessor. One each of these read and write registers is for direct data transfers. The rest of the write registers control the various features of the chip, while the rest of the read registers provide the microprocessor with a monitor of GPIB states, various bus conditions, and device conditions.

GPIB Addressing

Each device connected to the GPIB must have at least one address whereby the controller device in charge of the bus can configure it to talk, listen, or send status. An 8291A implementation of the GPIB offers the user three alternative addressing modes for which the device can be initialized for each application. The first of these modes allows for the device to have two separate primary addresses. The second mode allows the user to implement a single talker/listener with a two byte address (primary address + secondary address). The third mode again allows for two distinct addresses but in this instance, they can each have a ten-bit address (5 low-order bits of each of two bytes). However, this mode requires that the secondary addresses be passed to the microprocessor for verification. These three addressing schemes are described in more detail in the discussion of the Address Registers.

Figure 4. Interface Capabilities and Bus Structure
Table 2. IEEE 488 Interface State Mnemonics

<table>
<thead>
<tr>
<th>Mnemonic</th>
<th>State Represented</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACDS</td>
<td>Accept Data State</td>
</tr>
<tr>
<td>ACRS</td>
<td>Acceptor Ready State</td>
</tr>
<tr>
<td>AIDS</td>
<td>Acceptor Idle State</td>
</tr>
<tr>
<td>ANRS</td>
<td>Acceptor Not Ready State</td>
</tr>
<tr>
<td>APRS</td>
<td>Affirmative Poll Response State</td>
</tr>
<tr>
<td>AWNS</td>
<td>Acceptor Wait for New Cycle State</td>
</tr>
<tr>
<td>CACS</td>
<td>Controller Active State</td>
</tr>
<tr>
<td>CADS</td>
<td>Controller Addressed State</td>
</tr>
<tr>
<td>CAWS</td>
<td>Controller Active Wait State</td>
</tr>
<tr>
<td>CIDS</td>
<td>Controller Idle State</td>
</tr>
<tr>
<td>CPPS</td>
<td>Controller Parallel Poll State</td>
</tr>
<tr>
<td>CPWS</td>
<td>Controller Parallel Poll Wait State</td>
</tr>
<tr>
<td>CSBS</td>
<td>Controller Standby State</td>
</tr>
<tr>
<td>CSNS</td>
<td>Controller Service Not Requested State</td>
</tr>
<tr>
<td>CSRS</td>
<td>Controller Service Requested State</td>
</tr>
<tr>
<td>CSWS</td>
<td>Controller Synchronous Wait State</td>
</tr>
<tr>
<td>CTRS</td>
<td>Controller Transfer State</td>
</tr>
<tr>
<td>DCAS</td>
<td>Device Clear Active State</td>
</tr>
<tr>
<td>DCIS</td>
<td>Device Clear Idle State</td>
</tr>
<tr>
<td>DTAS</td>
<td>Device Trigger Active State</td>
</tr>
<tr>
<td>DTIS</td>
<td>Device Trigger Idle State</td>
</tr>
<tr>
<td>LACS</td>
<td>Listener Active State</td>
</tr>
<tr>
<td>LADS</td>
<td>Listener Addressed State</td>
</tr>
<tr>
<td>LIDS</td>
<td>Listener Idle State</td>
</tr>
<tr>
<td>LOCS</td>
<td>Local State</td>
</tr>
<tr>
<td>LPAS</td>
<td>Listener Primary Addressed State</td>
</tr>
<tr>
<td>LPIS</td>
<td>Listener Primary Idle State</td>
</tr>
<tr>
<td>LWLS</td>
<td>Local With Lockout State</td>
</tr>
<tr>
<td>NPRS</td>
<td>Negative Poll Response State</td>
</tr>
<tr>
<td>PACS</td>
<td>Parallel Poll Addressed to Configure State</td>
</tr>
<tr>
<td>PPAS</td>
<td>Parallel Poll Active State</td>
</tr>
<tr>
<td>PPIS</td>
<td>Parallel Poll Idle State</td>
</tr>
</tbody>
</table>

The Controller function is implemented on the Intel® 8292.

Table 3. IEEE 488 Interface Message Reference List

<table>
<thead>
<tr>
<th>Mnemonic</th>
<th>Message</th>
<th>Interface Function(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>gts(1)</td>
<td>go to standby</td>
<td>C</td>
</tr>
<tr>
<td>ist</td>
<td>individual status</td>
<td>PP</td>
</tr>
<tr>
<td>lon</td>
<td>listen only</td>
<td>L, LE</td>
</tr>
<tr>
<td>lpe</td>
<td>local poll enable</td>
<td>PP</td>
</tr>
<tr>
<td>nba</td>
<td>new byte available</td>
<td>SH</td>
</tr>
<tr>
<td>pon</td>
<td>power on</td>
<td>SH, AH, T, TE, L, LE, SR, RL, PP, C</td>
</tr>
<tr>
<td>rdy</td>
<td>ready</td>
<td>AH</td>
</tr>
<tr>
<td>rpp(1)</td>
<td>request parallel poll</td>
<td>C</td>
</tr>
<tr>
<td>rsc(1)</td>
<td>request system control</td>
<td>C</td>
</tr>
<tr>
<td>rsv</td>
<td>request service</td>
<td>SR</td>
</tr>
<tr>
<td>rtl</td>
<td>return to local</td>
<td>RL</td>
</tr>
<tr>
<td>sic(1)</td>
<td>send interface clear</td>
<td>C</td>
</tr>
<tr>
<td>sre(1)</td>
<td>send remote enable</td>
<td>C</td>
</tr>
<tr>
<td>tca(1)</td>
<td>take control asynchronously</td>
<td>C</td>
</tr>
</tbody>
</table>
Table 3. IEEE 488 Interface Message Reference List (Continued)

<table>
<thead>
<tr>
<th>Mnemonic</th>
<th>Message</th>
<th>Interface Function(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>tcs(1)</td>
<td>take control synchronously</td>
<td>AH, C</td>
</tr>
<tr>
<td>ton</td>
<td>talk only</td>
<td>T, TE</td>
</tr>
<tr>
<td>REMOTE MESSAGES RECEIVED</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ATN</td>
<td>Attention</td>
<td>SH, AH, T, TE, L, LE, PP, C</td>
</tr>
<tr>
<td>DAB</td>
<td>Data Byte</td>
<td>(Via L, LE)</td>
</tr>
<tr>
<td>DAC</td>
<td>Data Accepted</td>
<td>SH</td>
</tr>
<tr>
<td>DAV</td>
<td>Data Valid</td>
<td>AH</td>
</tr>
<tr>
<td>DCL</td>
<td>Device Clear</td>
<td>DC</td>
</tr>
<tr>
<td>END</td>
<td>End</td>
<td>(via L, LE)</td>
</tr>
<tr>
<td>GET</td>
<td>Group Execute Trigger</td>
<td>DT</td>
</tr>
<tr>
<td>GTL</td>
<td>Go to Local</td>
<td>RL</td>
</tr>
<tr>
<td>IDY</td>
<td>Identify</td>
<td>L, LE, PP</td>
</tr>
<tr>
<td>IFC</td>
<td>Interface Clear</td>
<td>T, TE, L, LE, C</td>
</tr>
<tr>
<td>LLO</td>
<td>Local Lockout</td>
<td>RL</td>
</tr>
<tr>
<td>MLA</td>
<td>My Listen Address</td>
<td>L, LE, RL, T, TE</td>
</tr>
<tr>
<td>MSA</td>
<td>My Secondary Address</td>
<td>TE, LE, RL</td>
</tr>
<tr>
<td>MTA</td>
<td>My Talk Address</td>
<td>T, TE, L, LE</td>
</tr>
<tr>
<td>OSA</td>
<td>Other Secondary Address</td>
<td>TE</td>
</tr>
<tr>
<td>OTA</td>
<td>Other Talk Address</td>
<td>T, TE</td>
</tr>
<tr>
<td>PCG</td>
<td>Primary Command Group</td>
<td>TE, LE, PP</td>
</tr>
<tr>
<td>PPC(2)</td>
<td>Parallel Poll Configure</td>
<td>PP</td>
</tr>
<tr>
<td><a href="2">PPD</a></td>
<td>Parallel Poll Disable</td>
<td>PP</td>
</tr>
<tr>
<td><a href="2">PPE</a></td>
<td>Parallel Poll Enable</td>
<td>PP</td>
</tr>
<tr>
<td>PPRN(1)</td>
<td>Parallel Poll Response N</td>
<td>(via C)</td>
</tr>
<tr>
<td>PPU(2)</td>
<td>Parallel Poll Unconfigure</td>
<td>PP</td>
</tr>
<tr>
<td>REN(2)</td>
<td>Remote Enable</td>
<td>RL</td>
</tr>
<tr>
<td>RFD</td>
<td>Ready for Data</td>
<td>SH</td>
</tr>
<tr>
<td>RQS</td>
<td>Request Service</td>
<td>(via L, LE)</td>
</tr>
<tr>
<td>[SDC]</td>
<td>Select Device Clear</td>
<td>DC</td>
</tr>
<tr>
<td>SPD</td>
<td>Serial Poll Disable</td>
<td>T, TE</td>
</tr>
<tr>
<td>SPE</td>
<td>Serial Poll Enable</td>
<td>T, TE</td>
</tr>
<tr>
<td>SQR(1)</td>
<td>Service Request</td>
<td>(via C)</td>
</tr>
<tr>
<td>STB</td>
<td>Status Byte</td>
<td>(via L, LE)</td>
</tr>
<tr>
<td>TCT or <a href="1">TCT</a></td>
<td>Take Control</td>
<td>C</td>
</tr>
<tr>
<td>UNL</td>
<td>Unlisten</td>
<td>L, LE</td>
</tr>
<tr>
<td>REMOTE MESSAGES SENT</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ATN</td>
<td>Attentions</td>
<td>C</td>
</tr>
<tr>
<td>DAB</td>
<td>Data Byte</td>
<td>(Via T, TE)</td>
</tr>
<tr>
<td>DAC</td>
<td>Data Accepted</td>
<td>AH</td>
</tr>
<tr>
<td>DAV</td>
<td>Data Valid</td>
<td>SH</td>
</tr>
<tr>
<td>DCL</td>
<td>Device Clear</td>
<td>(via C)</td>
</tr>
<tr>
<td>END</td>
<td>End</td>
<td>(via T)</td>
</tr>
<tr>
<td>GET</td>
<td>Group Execute Trigger</td>
<td>(via C)</td>
</tr>
<tr>
<td>GTL</td>
<td>Go to Local</td>
<td>(via C)</td>
</tr>
<tr>
<td>IDY</td>
<td>Identify</td>
<td>C</td>
</tr>
<tr>
<td>IFC</td>
<td>Interface Clear</td>
<td>C</td>
</tr>
<tr>
<td>LLO</td>
<td>Local Lockout</td>
<td>(via C)</td>
</tr>
<tr>
<td>MLA or [MLA]</td>
<td>My Listen Address</td>
<td>(via C)</td>
</tr>
<tr>
<td>MSA or [MSA]</td>
<td>My Secondary Address</td>
<td>(via C)</td>
</tr>
<tr>
<td>MTA or [MTA]</td>
<td>My Talk Address</td>
<td>(via C)</td>
</tr>
<tr>
<td>OSA</td>
<td>Other Secondary Address</td>
<td>(via C)</td>
</tr>
</tbody>
</table>
8291A

Table 3. IEEE 488 Interface Message Reference List (Continued)

<table>
<thead>
<tr>
<th>Mnemonic</th>
<th>Message</th>
<th>Interface Function(s)(3)</th>
</tr>
</thead>
<tbody>
<tr>
<td>OTA</td>
<td>Other Talk Address</td>
<td>(via C)</td>
</tr>
<tr>
<td>PCG</td>
<td>Primary Command Group</td>
<td>(via C)</td>
</tr>
<tr>
<td>PPC</td>
<td>Parallel Poll Configure</td>
<td>(via C)</td>
</tr>
<tr>
<td>[PPD]</td>
<td>Parallel Poll Disable</td>
<td>(via C)</td>
</tr>
<tr>
<td>[PEP]</td>
<td>Parallel Poll Enable</td>
<td>(via C)</td>
</tr>
<tr>
<td>PPRN</td>
<td>Parallel Poll Response N</td>
<td>PP</td>
</tr>
<tr>
<td>PPU</td>
<td>Parallel Poll Unconfigure</td>
<td>(via C)</td>
</tr>
<tr>
<td>REN</td>
<td>Remote Enable</td>
<td>C</td>
</tr>
<tr>
<td>RFD</td>
<td>Request for Data</td>
<td>AH</td>
</tr>
<tr>
<td>RQS</td>
<td>Request Service</td>
<td>T, TE</td>
</tr>
<tr>
<td>[SDC]</td>
<td>Selected Device Clear</td>
<td>(via C)</td>
</tr>
<tr>
<td>SPD</td>
<td>Serial Poll Disable</td>
<td>(via C)</td>
</tr>
<tr>
<td>SPE</td>
<td>Serial Poll Enable</td>
<td>(via C)</td>
</tr>
<tr>
<td>SRQ</td>
<td>Service Request</td>
<td>SR</td>
</tr>
<tr>
<td>STB</td>
<td>Status Byte</td>
<td>(via T, TE)</td>
</tr>
<tr>
<td>TCT</td>
<td>Take Control</td>
<td>(via C)</td>
</tr>
<tr>
<td>UNL</td>
<td>Unlisten</td>
<td>(via C)</td>
</tr>
</tbody>
</table>

NOTES:
1. These messages are handled only by Intel's 8292.
2. Undefined commands which may be passed to the microprocessor.
3. All Controller messages must be sent via Intel's 8292.

8291A Registers

A bit-by-bit map of the 16 registers on the 8291A is presented in Figure 5. A more detailed explanation of each of these registers and their functions follows. The access of these registers by the microprocessor is accomplished by using the CS, RD, WR, and RS0-RS2 pins.

<table>
<thead>
<tr>
<th>Register</th>
<th>CS</th>
<th>RD</th>
<th>WR</th>
<th>RS0-RS2</th>
</tr>
</thead>
<tbody>
<tr>
<td>All Read Registers</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>CCC</td>
</tr>
<tr>
<td>All Write Registers</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>CCC</td>
</tr>
<tr>
<td>High Impedance</td>
<td>1</td>
<td>d</td>
<td>d</td>
<td>ddd</td>
</tr>
</tbody>
</table>

Data Registers

<table>
<thead>
<tr>
<th>D17</th>
<th>D16</th>
<th>D15</th>
<th>D14</th>
<th>D13</th>
<th>D12</th>
<th>D11</th>
<th>D10</th>
</tr>
</thead>
<tbody>
<tr>
<td>DATA-IN REGISTER (0R)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DO7</td>
<td>DO6</td>
<td>DO5</td>
<td>DO4</td>
<td>DO3</td>
<td>DO2</td>
<td>DO1</td>
<td>DO0</td>
</tr>
<tr>
<td>DATA-OUT REGISTER (0W)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The Data-In Register is used to move data from the GPIB to the microprocessor or to memory when the 8291A is addressed to listen. Incoming information is separately latched by this register, and its contents are not destroyed by a write to the data-out register. The RFD (Ready for Data) message is held false until the byte is removed from the data in register, either by the microprocessor or by DMA. The 8291A then completes the handshake automatically. In RFD holdoff mode (see Auxiliary Register A), the handshake is not finished until a command is sent telling the 8291A to release the holdoff. In this way, the same byte may be read several times, or an over anxious talker may be held off until all available data has been processed.

When the 8291A is addressed to talk, it uses the data-out register to move data onto the GPIB. After the BO interrupt is received and a byte is written to this register, the 8291A initiates and completes the handshake while sending the byte out over the bus. In the BO interrupt disable mode, the user should wait until BO is active before writing to the register. (In the DMA mode, this will happen automatically.) A read of the Data-In Register does not destroy the information in the Data-Out Register.

Interrupt Registers

<table>
<thead>
<tr>
<th>CPT</th>
<th>APT</th>
<th>GET</th>
<th>END</th>
<th>DEC</th>
<th>ERR</th>
<th>BO</th>
<th>BI</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>INTERRUPT STATUS 1 (1R)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>INT</td>
<td>SPAS</td>
<td>LLO</td>
<td>REM</td>
<td>SPC</td>
<td>LLOC</td>
<td>REMC</td>
<td>ADSC</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>INTERRUPT STATUS 2 (2R)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CPT</td>
<td>APT</td>
<td>GET</td>
<td>END</td>
<td>DEC</td>
<td>ERR</td>
<td>BO</td>
<td>BI</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>INTERRUPT ENABLE 1 (1W)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>DMAO</td>
<td>DMA1</td>
<td>SPC</td>
<td>LLOC</td>
<td>REMC</td>
<td>ADSC</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>INTERRUPT ENABLE 2 (2W)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>INT</td>
<td>DT0</td>
<td>DL0</td>
<td>AD5-0</td>
<td>AD4-0</td>
<td>AD3-0</td>
<td>AD2-0</td>
<td>AD1-0</td>
</tr>
<tr>
<td>ADDRESS 0 REGISTER</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

3-8
The 8291A can be configured to generate an interrupt to the microprocessor upon the occurrence of any of 12 conditions or events on the GPIB. Upon receipt of an interrupt, the microprocessor must read the Interrupt Status Registers to determine which event has occurred, and then execute the appropriate service routine (if necessary). Each of the 12 interrupt status bits has a matching enable bit in the interrupt enable registers. These enable bits are used to select the events that will cause the INT pin to be asserted. Writing a logic "1" into any of these bits enables the corresponding interrupt status bits to generate an interrupt. Bits in the Interrupt Status Registers are set regardless of the states of the enable bits. The Interrupt Status Registers are then cleared upon being read or when a local pon (power-on) message is executed. If an event occurs while one of the Interrupt Status Registers is being read, the event is held until after its register is cleared and then placed in the register.

NOTE:
Reading the interrupt status registers clears the bits which were set. The software must examine all relevant bits in the interrupt status registers before disregarding the value or an important interrupt may be missed.

The mnemonics for each of the bits in these registers and a brief description of their respective functions appears in Table 4. This table also indicates how each of the interrupt bits is set.

NOTE:
The INT bit in the Address 0 Register is a duplicate of the INT bit in the Interrupt Status 2 Register. It is only a status bit. It does not generate interrupts and thus does not have a corresponding enable bit.

The BO and BI interrupts enable the user to perform data transfer cycles. BO indicates that a data byte should be written to the Data Out Register. It is set by TACS ⋅ (SWNS + SGNS) ⋅ RFD. It is reset when the data byte is written, ATN is asserted, or the 8291A exits TACS. Data should never be written to the Data Out Register before BO is set. Similarly, BI is set when an input byte is accepted into the 8291A and reset when the microprocessor reads the Data In Register. BO and BI are also reset by pon (power-on local message) and by a read of the Interrupt Status 1 Register. However, if it is so desired, data transfer cycles may be performed without reading the Interrupt Status 1 Register if all interrupts except for BO or BI are disabled; BO and BI will automatically reset after each byte is transferred.
Table 4. Interrupt Bits

<table>
<thead>
<tr>
<th>Interrupt Bit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPT</td>
<td>Set by (TPAS + LPAS) • SCG • ACDS • MODE 3</td>
</tr>
<tr>
<td>APT</td>
<td>Set by DTAS</td>
</tr>
<tr>
<td>GET</td>
<td>Set by (EOS + EOI) • LACS</td>
</tr>
<tr>
<td>END</td>
<td>Set by DCAS</td>
</tr>
<tr>
<td>DEC</td>
<td>Set by TACS • nba • DAC • RFD</td>
</tr>
<tr>
<td>ERR</td>
<td>TACS • (SWNS + SGNS)</td>
</tr>
<tr>
<td>BO</td>
<td>Set by LACS • ACDS</td>
</tr>
<tr>
<td>BI</td>
<td>Shows status of the INT pin</td>
</tr>
<tr>
<td>INT</td>
<td>The device has been enabled for a serial poll</td>
</tr>
<tr>
<td>SPAS</td>
<td>The device is in the local lock out state. (LWLS + RWLS)</td>
</tr>
<tr>
<td>LLO</td>
<td>The device is in a remote state. (REMS + RWLS)</td>
</tr>
<tr>
<td>REM</td>
<td>Serial Poll Complete interrupt.</td>
</tr>
<tr>
<td>ADSC</td>
<td>Address status change interrupt.</td>
</tr>
<tr>
<td>APRS:STRS:SPAS was true</td>
<td>If APRS:STRS:SPAS was true, then SPAS, LLO, Local, Addressed or Unaddressed.</td>
</tr>
</tbody>
</table>

NOTE:
1. In TRAN (talk-only) and LON (listen-only) modes, no ADSC interrupt is generated.

If the 8291A is used in the interrupt mode, the INT and DREQ pins can be dedicated to data input and output interrupts respectively by enabling BI and DMA0, provided that no other interrupts are enabled. This eliminates the need to read the interrupt status registers if a byte is received or transmitted.

The ERR bit is set to indicate the bus error condition when the 8291A is an active talker and tries to send a byte to the GPIB, but there are no active listeners (e.g., all devices on the GPIB are in AIDS). The logical equivalent of (nba • TACS • DAC • RFD) will set this bit.

The DEC bit is set whenever DCAS has occurred. The user must define a known state to which all device functions will return in DCAS. Typically this state will be a power-on state. However, the state of the device functions at DCAS is at the designer's discretion. It should be noted that DCAS has no effect on the interface functions which are returned to a known state by the IFC (interface clear) message or the pon local message.

The END interrupt bit may be used by the microprocessor to detect that a multi-byte transfer has been completed. The bit will be set when the 8291A is an active listener (LACS) and either EOS (provided the End on EOS Received feature is enabled in the Auxiliary Register A) or EOI is received. EOS will generate an interrupt when the byte in the Data In Register matches the byte in the EOS register. Otherwise the interrupt will be generated when a true input is detected on EOI.

The GET interrupt bit is used by the microprocessor to detect that DTAS has occurred. It is set by the 8291A when the GET message is received while it is addressed to listen. The TRIG output pin of the 8291A fires when the GET message is received. Thus, the basic operation of device trigger may be started without microprocessor software intervention.

The APT interrupt bit indicates to the processor that a secondary address is available in the CPT register for validation. This interrupt will only occur if Mode 3 addressing is in effect. (Refer to the section on addressing.) In Mode 2, secondary addresses will be recognized automatically on the 8291A. They will be ignored in Mode 1.
The CPT interrupt bit flags the occurrence of an undefined command and of all secondary commands following an undefined command. The Command Pass Through feature is enabled by the B0 bit of Auxiliary Register B. Any message not decoded by the 8291A (not included in the state diagrams in Appendix B) becomes an undefined command. Note that any addressed command is automatically ignored when the 8291A is not addressed.

Undefined commands are read by the CPU from the Command Pass Through register of the 8291A. This register reflects the logic levels present on the data lines at the time it is read. If the CPT feature is enabled, the 8291A will hold off the handshake until this register is read.

An especially useful feature of the 8291A is its ability to generate interrupts from state transitions in the interface functions. In particular, the lower 3 bits of the Interrupt Status 2 Register, if enabled by the corresponding enable bits, will cause an interrupt upon changes in the following states as defined in the IEEE 488 Standard.

Bit 0  ADSC  change in LIDS or TIDS or MJMN
Bit 1  REMC  change in LOCS or REMS
Bit 2  LLOC  change in LWLS or RWLS

The upper 4 bits of the Interrupt Status 2 Register are available to the processor as status bits. Thus, if one of the bits 0–2 generates an interrupt indicating a state change has taken place, the corresponding status bit (bits 3–5) may be read to determine what the new state is. To determine the nature of a change in addressed status (bit 0) the Address Status Register is available to be read. The SPC interrupt (bit 3 in Interrupt Status 2) is set upon exit from SPAS if APRS:STRS:SPAS occurred which indicates that the GPIB controller has read the bus serial poll status byte after the 8291A requested service (asserted SRQ). The SPC interrupt occurs once after the controller reads the status byte if service was requested. The controller may read the status byte later, and the byte will contain the last status the 8291A's CPU wrote to the Serial Poll Mode Register, but the SRQS bit will not be set and no interrupt will be generated. Finally, bit 7 monitors the state of the 8291A INT pin. Logically, it is an OR of all enabled interrupt status bits. One should note that bits 3–6 of the Interrupt Status 2 Register do not generate interrupts, but are available only to be read as status bits by the processor. Bit 7 in Interrupt Status 2 is duplicated in Address 0 Register, and the latter should be used when polling for interrupts to avoid losing one of the interrupts in Interrupt Status 2 Register.

Bits 4 and 5 (DMAI, DMAO) of the Interrupt Mask 2 Register are available to enable direct data transfers between memory and the GPIB; DMAI (DMA in) enables the DREQ (DMA request) pin of the 8291A to be asserted upon the occurrence of BI. Similarly, DMAO (DMA out) enables the DREQ pin to be asserted upon the occurrence of BO. One might note that the DREQ pin may be used as a second interrupt output pin, monitoring BI and/or BO and enabled by DMAI and DMAO. One should note that the DREQ pin is not affected by a read of the Interrupt Status 1 Register. It is reset whenever a byte is written to the Data Out Register or read from the Data In Register.

To ensure that an interrupt status bit will not be cleared without being read, and will not remain uncleared after being read, the 8291A implements a special interrupt handling procedure. When an enabled interrupt bit is set in either of the Interrupt Status Registers, the input of the registers are blocked until the set bit is read and reset by the microprocessor. Thus, potential problems arise when interrupt status changes while the register is being blocked. However, the 8291A stores all new interrupts in a temporary register and transfers them to the appropriate Interrupt Status Register after the interrupt has been reset. This transfer takes place only if the corresponding bits were read as zeroes.

### Serial Poll Registers

<table>
<thead>
<tr>
<th>S8</th>
<th>SRQS</th>
<th>S6</th>
<th>S5</th>
<th>S4</th>
<th>S3</th>
<th>S2</th>
<th>S1</th>
</tr>
</thead>
<tbody>
<tr>
<td>S8</td>
<td>rsv</td>
<td>S6</td>
<td>S5</td>
<td>S4</td>
<td>S3</td>
<td>S2</td>
<td>S1</td>
</tr>
</tbody>
</table>

**SERIAL POLL STATUS (3R)**

**SERIAL POLL MODE (3W)**

The Serial Poll Mode Register determines the status byte that the 8291A sends out on the GPIB data lines when it receives the SPE (Serial Poll Enable) message. Bit 6 of this register is reserved for the rsv (request service) local message. Setting this bit to 1 causes the 8291A to assert its SRQ line, indicating its need for attention from the controller-in-charge of the GPIB. The other bits of this register are available for sending status information over the GPIB. Sometimes after the microprocessor initiates a request for service by setting bit 6, the controller of the GPIB sends the SPE message and then addresses the 8291A to talk. At this point, one byte of status is returned by the 8291A via the Serial Poll Mode Register. After the status byte is read by the controller, rsv is automatically cleared by the 8291A and an SPC interrupt is generated. The CPU may request service again by writing another byte to the Serial Poll Mode Register with the rsv bit set. If the control-
I will be SRO line Serial Poll responds to The processor may check the 'status of a request for service by polling bit 6 of this register, which corresponds to SRQS (Service Request State). When a Serial Poll is conducted and the controller-in-charge reads the status byte, the SRQs bit is cleared. The SRQ line and the rsv bit are tied together.

**Address Registers**

<table>
<thead>
<tr>
<th>ton</th>
<th>ion</th>
<th>EOI</th>
<th>LPAS</th>
<th>TPAS</th>
<th>LA</th>
<th>TA</th>
<th>MJMN</th>
</tr>
</thead>
</table>

**ADDRESS STATUS (4R)**

<table>
<thead>
<tr>
<th>INT</th>
<th>DT0</th>
<th>DL0</th>
<th>AD5-0</th>
<th>AD4-0</th>
<th>AD3-0</th>
<th>AD2-0</th>
<th>AD1-0</th>
</tr>
</thead>
</table>

**ADDRESS 0 (6R)**

<table>
<thead>
<tr>
<th>X</th>
<th>DT1</th>
<th>DL1</th>
<th>AD5-1</th>
<th>AD4-1</th>
<th>AD3-1</th>
<th>AD2-1</th>
<th>AD1-1</th>
</tr>
</thead>
</table>

**ADDRESS 1 (7R)**

<table>
<thead>
<tr>
<th>TO</th>
<th>LO</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>ADM1</th>
<th>ADM0</th>
</tr>
</thead>
</table>

**ADDRESS MODE (4W)**

<table>
<thead>
<tr>
<th>ARS</th>
<th>DT</th>
<th>DL</th>
<th>AD5</th>
<th>AD4</th>
<th>AD3</th>
<th>AD2</th>
<th>AD1</th>
</tr>
</thead>
</table>

**ADDRESS 0/1 (6W)**

The Address Mode Register is used to select one of the five modes of addressing available on the 8291A. It determines the way in which the 8291A uses the information in the Address 0 and Address 1 Registers.

—in Mode 1, the contents of the Address 0 Register constitute the “Major” talker/listener address while the Address 1 Register represents the “Minor” talker/listener address. In applications where only one address is needed, the major talker/listener is used, and the minor talker/listener should be disabled. Loading an address via the Address 0/1 Register into Address Registers 0 and 1 enables the major and minor talker/listener functions respectively.

—in Mode 2 the 8291A recognizes two sequential address bytes: a primary followed by a secondary. Both address bytes must be received in order to enable the device to talk or listen. In this manner, Mode 2 addressing implements the extended talker and listener functions as defined in IEEE-488.

To use Mode 2 addressing the primary address must be loaded into the Address 0 Register, and the Secondary Address is placed in the Address 1 Register. With both primary and secondary addresses residing on chip, the 8291A can handle all addressing sequences without processor intervention.

—in Mode 3, the 8291A handles addressing just as it does in Mode 1, except that each Major or Minor primary address must be followed by a secondary address. All secondary addresses must be verified by the microprocessor when Mode 3 is used. When the 8291A is in TPAS or LPAS (talker/listener primary addressed state), and it does not recognize the byte on the DIO lines, an APT interrupt is generated (see section on Interrupt Registers) and the byte is available in the CPT (Command Pass-Through) Register. As part of its interrupt service routine, the microprocessor must read the CPT Register and write one of the following responses to the Auxiliary Mode Register:

1. 07H implies a non-valid secondary address
2. 0FH implies a valid secondary address

Setting the TO bit generates the local ton (talk-only) message and sets the 8291A to a talk-only mode. This mode allows the device to operate as a talker in an interface system without a controller.

Setting the LO bit generates the local ion (listen-only) message and sets the 8291A to a listen-only mode. This mode allows the device to operate as a listener in an interface system without a controller. The above bits may also be used by a controller-in-charge to set itself up for remote command or data communication.

The mode of addressing implemented by the 8291A may be selected by writing one of the following bytes to the Address Mode Register.

<table>
<thead>
<tr>
<th>Register</th>
<th>Contents</th>
<th>Mode</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>10000000</td>
<td>Enable talk only mode (ton)</td>
</tr>
<tr>
<td></td>
<td>01000000</td>
<td>Enable listen only mode (ion)</td>
</tr>
<tr>
<td></td>
<td>11000000</td>
<td>The 8291 may talk to itself</td>
</tr>
<tr>
<td></td>
<td>00000001</td>
<td>Mode 1, (Primary-Primary)</td>
</tr>
<tr>
<td></td>
<td>00000010</td>
<td>Mode 2 (Primary-Secondary)</td>
</tr>
<tr>
<td></td>
<td>00000111</td>
<td>Mode 3 (Primary/APT-Primary/APT)</td>
</tr>
</tbody>
</table>

The Address Status Register contains information used by the microprocessor to handle its own addressing. This information includes status bits that monitor the address state of each talker/listener, "ton" and "ion" flags which indicate the talk and listen only states, and an EOI bit which, when set, signifies that the END message came with the last data byte. LPAS and TPAS indicate that the listener
or talker primary address has been received. The microprocessor can use these bits when the secondary address is passed through to determine whether the 8291A is addressed to talk or listen. The LA (listener addressed) bit will be set when the 8291A is in LACS (Listener Active State) or in LADS (Listener Addressed State). Similarly, the TA (Talker Addressed bit) will be set to indicate TACS or TADS, but also to indicate SPAS (Serial Poll Active State). The MJMN bit is used to determine whether the information in the other bits applies to the Major or Minor talker/listener. It is set to “1” when the Minor talker/listener is addressed. It should be noted that only one talker/listener may be active at any one time. Thus, the MJMN bit will indicate which, if either, of the talker/listeners is addressed or active.

The Address 0/1 Register is used for specifying the device’s addresses according to the format selected in the Address Mode Register. Five bit addresses may be loaded into the Address 0 and Address 1 Registers by writing into the Address 0/1 Register. The ARS bit is used to select which of these registers the other seven bits will be loaded into. The DT and DL bits may be used to disable the talker or listener function at the address signified by the other five bits. When Mode 1 addressing is used and only one primary address is desired, both the talker and the listener should be disabled at the Minor address.

As an example of how the Address 0/1 Register might be used, consider an example where two primary addresses are needed in the device. The Major primary address will be selectable only as a talker and the Minor primary address will be selectable only as a listener. This configuration of the 8291A is formed by the following sequence of writes by the microprocessor.

<table>
<thead>
<tr>
<th>Operation</th>
<th>CS</th>
<th>RD</th>
<th>WR</th>
<th>Data</th>
<th>RS2−RS0</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Select addressing Mode 1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>00000001</td>
<td>100</td>
</tr>
<tr>
<td>2. Load major address into Address 0 Register with listener function disabled.</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>001AAAAA</td>
<td>110</td>
</tr>
<tr>
<td>3. Load minor address into Address 1 Register with talker function disabled.</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>110BBBBB</td>
<td>110</td>
</tr>
</tbody>
</table>

At this point, the addresses AAAAA and BBBBB are stored in the Address 0 and Address 1 Registers respectively, and are available to be read by the microprocessor. Thus, it is not necessary to store any address information elsewhere. Also, with the information stored in the Address 0 and Address 1 Registers, processor intervention is not required to recognize addressing by the controller. Only in Mode 3, where secondary addresses are passed through, must the processor intervene in the addressing sequence.

The Address 0 Register contains a copy of bit 7 of the Interrupt Status 2 Register (INT). This is to be used when polling for interrupts. Software should poll register 6 checking for INT (bit 7) to be set. When INT is set, the Interrupt Status Register should be read to determine which interrupt was received.

**Command Pass Through Register**

<table>
<thead>
<tr>
<th>CPT7</th>
<th>CPT6</th>
<th>CPT5</th>
<th>CPT4</th>
<th>CPT3</th>
<th>CPT2</th>
<th>CPT1</th>
<th>CPT0</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**COMMAND PASS THROUGH (5R)**

The Command Pass Through Register is used to transfer undefined 8-bit remote message codes from the GPIB to the microprocessor. When the CPT feature is enabled (bit BO in Auxiliary Register B), any message not decoded by the 8291A becomes an undefined command. When Mode 3 addressing is used, secondary addresses are also passed through the CPT Register. In either case, the 8291A will hold-off the handshake until the microprocessor reads this register and issues the VSCMD auxiliary command.

The CPT and APT interrupts flag the availability of undefined commands and secondary addresses in the CPT Register. The details of these interrupts are explained in the section on Interrupt Registers.

An added feature of the 8291A is its ability to handle undefined secondary commands following undefined primaries. Thus, the number of available commands for future IEEE-488 definition is increased; one undefined primary command followed by a sequence of as many as 32 secondary commands can be processed. The IEEE-488 Standard does not permit users to define their own commands, but upgrades of the standard are thus provided for.

The recommended use of the 8291A's undefined command capabilities is for a controller-configured Parallel Poll. The PPC message is an undefined primary command typically followed by PPE, and undefined secondary command. For details on this procedure, refer to the section on Parallel Poll Protocol.

**Auxiliary Mode Register**

<table>
<thead>
<tr>
<th>CNT2</th>
<th>CNT1</th>
<th>CNT0</th>
<th>COM4</th>
<th>COM3</th>
<th>COM2</th>
<th>COM1</th>
<th>COM0</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**AUX MODE (5W)**

CNT0–2:CONTROL BITS
COM0–4:COMMAND BITS
The Auxiliary Mode Register contains a three-bit control field and a five-bit command field. It is used for several purposes on the 8291A:
1. To load "hidden" auxiliary registers on the 8291A.
2. To issue commands from the microprocessor to the 8291A.
3. To preset an internal counter used to generate T1, delay in the Source Handshake function, as defined in IEEE-488.

Table 5 summarizes how these tasks are performed with the Auxiliary Mode Register. Note that the three control bits determine how the five command bits are interpreted.

<table>
<thead>
<tr>
<th>Code</th>
<th>Command</th>
<th>Command</th>
</tr>
</thead>
<tbody>
<tr>
<td>Control</td>
<td>Command</td>
<td></td>
</tr>
<tr>
<td>Bits</td>
<td>Bits</td>
<td></td>
</tr>
<tr>
<td>000 000</td>
<td>OCCCC</td>
<td>Execute auxiliary</td>
</tr>
<tr>
<td>000 001</td>
<td>ODDDD</td>
<td>command CCCC</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Preset internal</td>
</tr>
<tr>
<td></td>
<td></td>
<td>counter to</td>
</tr>
<tr>
<td></td>
<td></td>
<td>match external clock</td>
</tr>
<tr>
<td></td>
<td></td>
<td>frequency of DDDD MHz</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(DDDD binary</td>
</tr>
<tr>
<td></td>
<td></td>
<td>representation of</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 to 8 MHz)</td>
</tr>
<tr>
<td>010 100</td>
<td>DDDDD</td>
<td>Write DDDDD into</td>
</tr>
<tr>
<td></td>
<td></td>
<td>auxiliary register A</td>
</tr>
<tr>
<td>010 101</td>
<td>DDDDD</td>
<td>Write DDDDD into</td>
</tr>
<tr>
<td></td>
<td></td>
<td>auxiliary register B</td>
</tr>
<tr>
<td>011 011</td>
<td>USP2P2P1</td>
<td>Enable/disable parallel</td>
</tr>
<tr>
<td></td>
<td></td>
<td>poll either in response</td>
</tr>
<tr>
<td></td>
<td></td>
<td>to remote messages</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(PPC followed by PPE</td>
</tr>
<tr>
<td></td>
<td></td>
<td>or PPD) or as a local</td>
</tr>
<tr>
<td></td>
<td></td>
<td>lpe message. (Enable</td>
</tr>
<tr>
<td></td>
<td></td>
<td>if U = 0,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>disable if U = 1.)</td>
</tr>
</tbody>
</table>

**AUXILIARY COMMANDS**

Auxiliary commands are executed by the 8291A whenever 0000CCCC is written into the Auxiliary Mode Register, where CCCC is the 4-bit command code.

**0000**—Immediate Execute pon: This command resets the 8291A to a power up state (local pon message as defined in IEEE-488).

The following conditions constitute the power up state:
1. All talkers and listeners are disabled.
2. No interrupt status bits are set.

The 8291A is designed to power up in certain states as specified in the IEEE-488 state diagrams. Thus, the following states are in effect in the power up state: SIDS, AIDS, TIDS, LIDS, NPRS, LOCs, and PPIS.

The "0000" pon is an immediate execute command (a pon pulse). It is also used to release the "initialize" state generated by either an external reset pulse or the "0010" Chip Reset command.

**0010**—Chip Reset (Initialize): This command has the same effect as a pulse applied to the Reset pin. (Refer to the section on Reset Procedure.)

**0011**—Finish Handshake: This command finishes a handshake that was stopped because of a holdoff on RFD. (Refer to Auxiliary Register A.)

**0100**—Trigger: A "Group Execute Trigger" is forced by this command. It has the same effect as a GET command issued by the controller-in-charge of the GPIB, but does not cause a GET interrupt.

**0101, 1101**—Clear/Set rtl: These commands correspond to the local rtl message as defined by the IEEE-488. The 8291A will go into local mode when a Set rtl Auxiliary Command is received if local lockout is not in effect. The 8291A will exit local mode after receiving a Clear rtl Auxiliary Command if the 8291A is addressed to listen.

**0110**—Send EOI: The EOI line of the 8291A may be asserted with this command. The command causes EOI to go true with the next byte transmitted. The EOI line is then cleared upon completion of the handshake for that byte.

**0111, 1111**—Non Valid/Valid Secondary Address or Command (VSCMD): This command informs the 8291A that the secondary address received by the microprocessor was valid or invalid (0111 = invalid, 1111 = valid). If Mode 3 addressing is used, the processor must field each extended address and respond to it, or the GPIB will hang up. Note that the COM3 bit is the invalid/valid flag.

The valid (1111) command is also used to tell the 8291A to continue from the command-pass-through state, or from RFD holdoff on GET, SDC or DCL.

**1000**—pon: This command puts the 8291A into the pon (power on) state and holds it there. It is similar to a Chip Reset except none of the Auxiliary Mode Registers are cleared. In this state, the 8291A does not participate in any bus activity. An Immediate Execute pon releases the 8291A from the pon state and permits the device to participate in the bus activity again.
0001, 1001—Parallel Poll Flag (local "ist" message): This command sets (1001) or clears (0001) the parallel poll flag. A "1" is sent over the assigned data line (PRR = Parallel Poll Response true) only if the parallel poll flag matches the sense bit from the IPE local message (or indirectly from the PPE message). For a more complete description of the Parallel Poll features and procedures refer to the section on Parallel Poll Protocol.

INTERNAL COUNTER

The internal counter determines the delay time allowed for the setting of data on the DIO lines. This delay time is defined as $T_1$ in IEEE-488 and appears in the Source Handshake state diagram between the SDYS and STRS. As such, DAV is asserted $T_1$ after the DIO lines are driven. Consequently, $T_1$ is a major factor in determining the data transfer rate of the 8291A over the GPIB ($T_1 = T_{WIRDV2-WIRD15}$).

When open-collector transceivers are used for connection to the GPIB, $T_1$ is defined by IEEE-488 to be 2 $\mu$s. By writing 0010DDDD into the Auxiliary Mode Register, the counter is preset to match a $f_C$ MHz clock input, where DDDD is the binary representation of $N_F$ ($1 \leq N_F \leq 8$, $N_F = (DDDD)_2$). When $N_F = f_C$, a 2 $\mu$s $T_1$ delay will be generated before each DAV asserted.

$$T_1(\mu s) = \frac{2N_F}{f_C} + t_{SYNC}, 1 \leq N_F \leq 8$$

$t_{SYNC}$ is a synchronization error, greater than zero and smaller than the larger of $T_{clock}$ high and $T_{clock}$ low. (For a 50% duty cycle clock, $t_{SYNC}$ is less than half the clock cycle).

If it is necessary that $T_1$ be different from 2 $\mu$s, $N_F$ may be set to a value other than $f_C$. In this manner, data transfer rates may be programmed for a given system. In small systems, for example, where transfer rates exceeding GPIB specifications are required, one may set $N_F < f_C$ and decrease $T_1$.

When tri-state transceivers are used, IEEE-488 allows a higher transfer rate (lower $T_1$). Use of the 8291A with such transceivers is enabled by setting $B_2$ in Auxiliary Register B. In this case, setting $N_F = f_C$ causes a $T_1$ delay of 2$\mu$s to be generated for the first byte transmitted—all subsequent bytes will have a delay of 500 ns.

$$T_1 (High \ Speed) \ \mu s = \frac{N_F}{2f_C} + t_{SYNC}$$

Thus, the shortest $T_1$ is achieved by setting $N_F = 1$ using an 8 MHz clock with a 50% duty cycle clock ($t_{SYNC} < 63$ ns):

$$T_{1(HS)} = \frac{1}{2 \times 8} + 0.063 = 125 \text{ ns max.}$$

AUXILIARY REGISTER A

Auxiliary Register A is a "hidden" 5-bit register which is used to enable some of the 8291A features. Whenever a 100 $A_4A_3A_2A_1A_0$ byte is written into the Auxiliary Register, it is loaded with the data $A_4A_3A_2A_1A_0$. Setting the respective bits to "1" enables the following features.

$A_0$—RFD Holdoff on all Data: If the 8291A is listening, RFD will not be sent true until the "finish handshake" auxiliary command is issued by the microprocessor. The holdoff will be in effect for each data byte.

$A_1$—RFD Holdoff on End: This feature enables the holdoff on EOI or EOS (if enabled). However, no hold-off will be in effect on any other data bytes.

$A_2$—End on EOS Received: Whenever the byte in the Data In Register matches the byte in the EOS Register, the END interrupt bit will be set in the Interrupt Status 1 Register.

$A_3$—Output EOI on EOS Sent: Any occurrence of data in the Data Out Register matching the EOS Register causes the EOI line to be sent true along with the data.

$A_4$—EOS Binary Compare: Setting this bit causes the EOS Register to function as a full 8-bit word. When it is not set, the EOS Register is a 7-bit word (for ASCII characters).

If $A_0 = A_1 = 1$, a special "continuous Acceptor Handshake cycling" mode is enabled. This mode should be used only in a controller system configuration, where both the 8291A and the 8292 are used. It provides a continuous cycling through the Acceptor Handshake state diagram, requiring no local messages from the microprocessor; the rdy local message is automatically generated when in ANRS. As such, the 8291A Accept orphan Handshake serves as the controller Acceptor Handshake. Thus, the controller cycles through the Acceptor Handshake without delaying the data transfer in progress. When the tcs local message is executed, the 8291A should be taken out of the "continuous AH cycling" mode, the GPIB will hang up in ANRS, and a BI interrupt will be generated to indicate that control may be taken. A
simpler procedure may be used when a "tc's on end of block" is executed; the 8291A may stay in "continuous AH cycling". Upon the end of a block (EOI or EOS received), a holdoff is generated, the GPIB hangs up in ANRS, and control may be taken.

**AUXILIARY REGISTER B**

Auxiliary Register B is a "hidden" 4-bit register which is used to enable some of the features of the 8291A. Whenever a 101 B4B3B2B1B0 is written into the Auxiliary Mode Register, it is loaded with the data B4B3B2B1B0. Setting the respective bits to "1" enables the following features:

- **B0**—Enable Undefined Command Pass Through: This feature allows any commands not recognized by the 8291A to be handled in software. If enabled, this feature will cause the 8291A to holdoff the handshake when an undefined command is received. The microprocessor must then read the command from the Command Pass Through Register and send the VSCMD auxiliary command. Until the VSCMD command is sent, the handshake holdoff will be in effect.

- **B1**—Send EOI in SPAS: This bit enables EOI to be sent with the status byte; EOI is sent true in Serial Poll Active State. Otherwise, EOI is sent false in SPAS.

- **B2**—Enable High Speed Data Transfer: This feature may be enabled when tri-state external transceivers are used. The data transfer rate is limited by T1 delay time generated in the Source Handshake function, which is defined according to the type of transceivers used. When the "High Speed" feature is enabled, T1 = 2 microseconds is generated for the first byte transmitted after each true to false transition of ATN. For all subsequent bytes, T1 = 500 ns. Refer to the Internal Counter section for an explanation of T1 duration as a function of B2 and of clock frequency.

- **B3**—Enable Active Low Interrupt: Setting this bit causes the polarity of the INT pin to be reversed, providing an output signal compatible with Intel's MCS-48® Family. Interrupt registers are not affected by this bit.

- **B4**—Enable RFD Holdoff on GET or DEC: Setting this bit causes RFD to be held false until the "VSCMD" auxiliary command is written after GET, SDC, and DCL commands. This allows the device to hold off the bus until it has completed a clear or trigger similar to an unrecognized command.

**PARALLEL POLL PROTOCOL**

Writing a 011USBP2P1 into the Auxiliary Mode Register will enable (U = 0) or disable (U = 1) the 8291A for a parallel poll. When U = 0, this command is the "ipe" (local poll enable) local message as defined in IEEE-488. The "S" bit is the sense in which the 8291A is enabled; only if the Parallel Poll Flag ("ist" local message) matches this bit will the Parallel Poll Response, PPRN, be sent true (Response = S + ist). The bits P4P3P2P1 specify which of the eight data lines PPRN will be sent over. Thus, once the 8291A has been configured for Parallel Poll, whenever it senses both EOI and ATN true, it will automatically compare its PP flag with the sense bit and send PPRN true or false according to the comparison.

If a PP2* implementation is desired, the "ipe" and "ist" local messages are all that are needed. Typically, the user will configure the 8291A for Parallel Poll immediately after initialization. During normal operation the microprocessor will set or clear the Parallel Poll Flag (ist) according to the device's need for service. Consequently the 8291A will be set up to give the proper response to IDY (EOI • ATN) without directly involving the microprocessor.

If a PP1* implementation is desired, the undefined command features of the 8291A must be used. In PP1, the 8291A is indirectly configured for Parallel Poll by the active controller on the GPIB. The sequence at the 8291A being enabled or disabled remotely is as follows:

1. The PPC message is received and is loaded into the Command Pass Through Register as an undefined command. A CPT Interrupt is sent to the microprocessor; the handshake is automatically held off.
2. The microprocessor reads the CPT Register and sends VSCMD to the 8291A, releasing the handshake.
3. Having received an undefined primary command, the 8291A is set up to receive an undefined secondary command (the PPE or PPD message). This message is also received into the CPT Register, the handshake is held off, and the CPT interrupt is generated.
4. The microprocessor reads the PPE or PPD message and writes the command into the Auxiliary Mode Register (bit 7 should be cleared first). Finally, the microprocessor sends VSCMD and the handshake is released.

**NOTE:**

As defined in IEEE Standard 488.
**End of Sequence (EOS) Register**

<table>
<thead>
<tr>
<th>EC7</th>
<th>EC6</th>
<th>EC5</th>
<th>EC4</th>
<th>EC3</th>
<th>EC2</th>
<th>EC1</th>
<th>EC0</th>
</tr>
</thead>
</table>

The EOS Register and its features offer an alternative to the "Send EOI" auxiliary command. A seven or eight bit byte (ASCII or binary) may be placed in the register to flag the end of a block or read. The type of EOS byte to be used is selected in Auxiliary Register bit A4.

If the 8291A is a listener, and the "End on EOS Received" is enabled with bit A2, then an END interrupt is generated in the Interrupt Status 1 Register whenever the byte in the Data-In Register matches the byte in the EOS Register.

If the 8291A is a talker, and the "Output EOI on EOS Sent" is enabled with bit A3, then the EOI line is sent true with the next byte whenever the contents of the Data Out Register match the EOS register.

**Reset Procedure**

The 8291A is reset to an initialization state either by a pulse applied to its Reset pin, or by a reset auxiliary command (02H written into the Auxiliary Command Register). The following conditions are caused by a reset pulse (or local reset command):

1. A "pon" local message as defined by IEEE-488 is held true until the initialization state is released.
2. The Interrupt Status Registers are cleared (not Interrupt Enable Registers).
3. Auxiliary Registers A and B are cleared.
4. The Serial Poll Mode Register is cleared.
5. The Parallel Poll Flag is cleared.
6. The EOI bit in the Address Status Register is cleared.
7. Nc in the Internal Counter is set to 8 MHz. This setting causes the longest possible T1 delay to be generated in the Source Handshake (16 μs for 1 MHz clock).
8. The rdy local message is sent.

The initialization state is released by an "immediate execute pon" command (00H written into the Auxiliary Command Register).

The suggested initialization sequence is:

1. Apply a reset pulse or send the reset auxiliary command.
2. Set the desired initial conditions by writing into the Interrupt Enable, Serial Poll Mode, Address Mode, Address 0/1, and EOS Registers. Auxiliary Registers A and B, and the internal counter should also be initialized.
3. Send the "immediate execute pon" auxiliary command to release the initialization state.
4. If a PP2 Parallel Poll implementation is to be used the "Ipe" local message may be sent, enabling the 8291A for a Parallel Poll Response on an assigned line. (Refer to the section on Parallel Poll Protocol.)

**Using DMA**

The 8291A may be connected to the Intel® 8237 or 8257 DMA Controllers or the 8089 I/O Processor for DMA operation. The 8237 will be used to refer to any DMA controller. The DREQ pin of the 8291A requests a DMA byte transfer from the 8237. It is set by BO or BI flip flops, enabled by the DMAO and DMAIL bits in the Interrupt Enable 2 Register. (After reading, the INT1 register BO and BI interrupts will be cleared but not BO and BI in DREQ equation.)

The DACK pin is driven by the 8237 in response to the DMA request. When DACK is true (active low) it sets CS = RS0 = RS1 = RS2 = 0 such that the RD and WR signals sent by the 8237 refer to the Data In and Data Out Registers. Also, the DMA request line is reset by DACK (RD + WR).

DMA input sequence:

1. A data byte is accepted from the GPIB by the 8291A.
2. A BI interrupt is generated and DREQ is set.
3. DACK and RD are driven by the 8237, the contents of the Data In Register are transferred to the system bus, and DREQ is reset.
4. The 8291A sends RFD true on the GPIB and proceeds with the Acceptor Handshake protocol.

DMA output sequence:

1. A BO interrupt is generated (indicating that a byte should be output) and DREQ is asserted.
2. DACK and WR are driven by the 8237, a byte is transferred from the MCS bus into the Data Out Register, and DREQ is reset.
3. The 8291A sends DAV true on the GPIB and proceeds with the Source Handshake protocol.

It should be noted that each time the device is addressed (MTA + MLA + ton + lom), the Address Status Register should be read, and the 8237 should be initialized accordingly. (Refer to the 8237 or 8257 Data Sheets.)
Polling the 8291A

If polling is used to determine the 8291A’s service needs, the CPU must poll the INT bit in the address 0 register. All relevant interrupt status bits must be enabled during initialization for them to affect the INT status bit. The following flow chart illustrates the recommended polling algorithm.
APPLICATION BRIEF

System Configuration

MICROPROCESSOR BUS CONNECTION

The 8291A is 8048/49, 8051, 8080/85, and 8086/88 compatible. The three address pins (RS0, RS1, and RS2) should be connected to the non-multiplexed address bus (for example: A8, A9, A10). In case of 8080, any address lines may be used. If the three lowest address bits are used (A0, A1, A2), then they must be demultiplexed first.

EXTERNAL TRANSCEIVERS CONNECTION

The 8293 GPIO Transceiver interfaces the 8291A directly to the IEEE-488 bus. The 8291A and two 8293's can be configured as a talker/listener (see Figure 6) or with the 8292 as a talker/listener/controller (see Figure 7). Absolutely no active or passive external components are required to comply with the complete IEEE-488 electrical specification.
Figure 7. 8291A, 8292, and 8293 System Configuration

* = GPIB Bus Transceiver
† = See 8041A Data Sheet for alternate crystal configurations
†† = Can connect to system reset switch, see 8041A Data Sheet
Start-Up Procedures

The following section describes the steps needed to initialize a typical 8291A system implementing a talker/listener interface and an 8291A/8292 system implementing a talker/listener/controller interface.

TALKER/LISTENER SYSTEM

Assume a general system configuration with the following features: (i) Polled system interface; (ii) Mode 1 addressing; (iii) same address for talker and listener; (iv) ASCII carriage return as the end-of-sequence (EOS) character; (v) EOI sent true with the last byte; and, (vi) 8 MHz clock.

Initialization. Initialization is accomplished with the following steps:

1. Pulse the RESET input or write 02H to the Auxiliary Mode Register.
2. Write 00H to the Interrupt Enable Registers 1 and 2. This disables interrupt and DMA.
3. Write 01H to the Address Mode Register to select Mode 1 addressing.
4. Write 28H to the Auxiliary Mode Register. This loads 8H to the Auxiliary Register A matching the 8 MHz clock input to the internal T1 delay counter to generate the delay meeting the IEEE spec.
5. Write the talker/listener address to the Address 0/1 register. The three most significant bits are zero.
6. Write an ASCII carriage return (ODH) to the EOS register.
7. Write 84H to the Auxiliary Mode Register to allow EOI to be sent true when the EOS character is sent.
8. Write 00H to the Auxiliary Mode Register. This writes the "Immediate Execute pon" message and takes the 8291A from the initialization state into the idle state. The 8291A will remain idle until the controller initiates some activity by driving ATN true.

Communication. The local CPU now polls the 8291A to determine which controller command has been received.

The controller addresses the 8291A by driving ATN, placing MLA (My Listen Address) on the bus and driving DAV. If the lower five bits of the MLA message match the address programmed into the Address 0/1 register, the 8291A is addressed to listen. It would be addressed to talk if the controller sent the MTA message instead of MLA.

The ADSC bit in the Interrupt Status 2 Register indicates that the 8291A has been addressed or unaddressed. The TA and LA bits in the Address Status Register indicate whether the 8291A is talker (TA = 1), listener (LA = 1), both (TA = LA = 1) or unaddressed (TA = LA = 0).

If the 8291A is addressed to listen, the local CPU can read the Data-In Register whenever the BI (Byte In) interrupt occurs in the Interrupt Status 1 Register. If the END bit in the same register is also set, either EOI or a data byte matching the pattern in the EOS register has been received.

In the talker mode, the CPU writes data into the Byte-Out Register on BO (Byte Out) true.

TALKER/LISTENER/CONTROLLER SYSTEM

Combined with the Intel 8292, the 8291A executes a complete IEEE-488-1978 controller function. The 8291A talks and listens via the data and handshake lines (NRFD, NDAC and DAV). The 8292 controls four of the five bus management lines (IFC, SRQ, ATN and REN). EOI, the fifth line, is shared. The 8291A drives and receives EOI when EOI is used as an end-of-block indicator. The 8292 drives EOI along with ATN during a parallel poll command.

Once again, assume a general system configuration with the following features: (i) Polled system interface; (ii) 8292 as the system controller and controller-in-charge; (iii) ASCII carriage return (ODH) as the EOS identifier; (iv) EOI sent with the last character; and, (v) an external buffer (8282) used to monitor the TCI line.

Initialization. In order to send a command across the GPIB, the 8292 has to drive ATN, and the 8291A has to drive the data lines. Both devices therefore need initialization.

To initialize the 8292:

1. Pulse the RESET input. The 8292 will initially drive all outputs high. TCI, SPI, OBSI, IBFI and CLTH will then go low. The Interrupt Status, Interrupt Mask, Error Flag, Error Mask and Timeout registers will be cleared. The interrupt counter will be disabled and loaded with 255. The 8292 will then monitor the status of the SYC pin. If high, the 8292 will pulse IFC true for at least 100 µs in compliance with the IEEE-488-1978 standard. It will then take control by asserting ATN.

To initialize the 8291A, the following is necessary:

1. Write 00H to Interrupt Enable registers 1 and 2. This disables interrupt and DMA.
2. With the 8292 as the controller-in-charge, it is impossible to address the 8292 via the GPIB. Therefore, the ton or Ion modes of the 8291A must be used. To send commands, set the 8291A in the ton mode by writing 80H to the Address Mode Register.

3. Write 26H to the Auxiliary Mode Register to match the T1 data settling time to the 6 MHz clock input.

4. Write an ASCII carriage return (0DH) to the EOS Register.

5. Write 84H to the Auxiliary Mode Register in order to enable "Output EOI on EOS sent" and thus send EOI with the last character.

6. Write 00H—Immediate Execute pon—to the Auxiliary Mode Register to put the 8291A in the idle state.

**Communication.** Since the 8291A is in the ton mode, a BO interrupt is generated as soon as the immediate Execute pon command is written. The CPU writes the command into the Data Out Register, and repeats it on BO becoming true for as many commands as necessary. ATN remains continuously true unless the GTSB (Go To Standby) command is sent to the 8292.

ATN has to be false in order to send data rather than commands from the controller. To do this, the following steps are needed:

1. Enable the TCI interrupt if not already enabled.

2. Wait for IBF (Input Buffer Full) in the 8292 Interrupt Status Register to be reset.

3. Write the GTSB (F6H) command to the 8292 Command Field Register.

4. Read the 8282 and wait for TCI to be true.

5. Write the ton (80H) and pon (00H) command to the 8291A Address Mode Register and Auxiliary Mode Registers respectively.

6. Wait for the BO interrupt to be set in the 8291A.

7. Write the data to the 8291A Data-Out Register.

Identically, the user could command the controller to listen rather than talk. To do that, write Ion (40H) instead of ton into the Address Mode Register. Then wait for BI rather than BO to go true. Read the data Register.
ABSOLUTE MAXIMUM RATINGS

Ambient Temperature Under Bias ........... 0°C to 70°C
Storage Temperature ....................... -65°C to +150°C
Voltage on Any Pin With Respect to Ground ........... -0.5V to +7V
Power Dissipation .......................... 0.65 Watts

*Notice: Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions above those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

D.C. CHARACTERISTICS $V_{CC} = 5V \pm 10\%$, $T_A = 0°C$ to 70°C (Commercial)

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Parameter</th>
<th>Min</th>
<th>Max</th>
<th>Unit</th>
<th>Test Conditions</th>
</tr>
</thead>
<tbody>
<tr>
<td>$V_{IL}$</td>
<td>Input Low Voltage</td>
<td>-0.5</td>
<td>0.8</td>
<td>V</td>
<td></td>
</tr>
<tr>
<td>$V_{IH}$</td>
<td>Input High Voltage</td>
<td>2</td>
<td>$V_{CC}$ + 0.5</td>
<td>V</td>
<td></td>
</tr>
<tr>
<td>$V_{OL}$</td>
<td>Output Low Voltage</td>
<td>0.45</td>
<td></td>
<td>V</td>
<td>$I_{OL} = 2 mA$ (4 mA for TR1 pin)</td>
</tr>
<tr>
<td>$V_{OH}$</td>
<td>Output High Voltage</td>
<td>2.4</td>
<td></td>
<td>V</td>
<td>$I_{OH} = -400 \mu A$ (-150 \mu A for SRO pin)</td>
</tr>
<tr>
<td>$V_{OH-INT}$</td>
<td>Interrupt Output High Voltage</td>
<td>2.4</td>
<td>3.5</td>
<td></td>
<td>$I_{OH} = -400 \mu A$</td>
</tr>
<tr>
<td>$I_{IL}$</td>
<td>Input Leakage</td>
<td>10</td>
<td></td>
<td>\mu A</td>
<td>$V_{IN} = 0V$ to $V_{CC}$</td>
</tr>
<tr>
<td>$I_{OFL}$</td>
<td>Output Leakage Current</td>
<td>±10</td>
<td></td>
<td>\mu A</td>
<td>$V_{OUT} = 0.45V$, $V_{CC}$</td>
</tr>
<tr>
<td>$I_{CC}$</td>
<td>$V_{CC}$ Supply Current</td>
<td>120</td>
<td></td>
<td>mA</td>
<td>$T_A = 0°C$</td>
</tr>
</tbody>
</table>

A.C. CHARACTERISTICS $V_{CC} = 5V \pm 10\%$, $T_A = 0°C$ to 70°C (Commercial)

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Parameter</th>
<th>Min</th>
<th>Max</th>
<th>Unit</th>
<th>Test Conditions</th>
</tr>
</thead>
<tbody>
<tr>
<td>$t_{AR}$</td>
<td>Address Stable Before READ</td>
<td>0</td>
<td></td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>$t_{RA}$</td>
<td>Address Hold After READ</td>
<td>0</td>
<td></td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>$t_{RR}$</td>
<td>READ Width</td>
<td>140</td>
<td></td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>$t_{AD}$</td>
<td>Address Stable to Data Valid</td>
<td>250</td>
<td></td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>$t_{RD}$</td>
<td>READ to Data Valid</td>
<td>100</td>
<td></td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>$t_{RDF}$</td>
<td>Data Float After READ</td>
<td>0</td>
<td>60</td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>$t_{AW}$</td>
<td>Address Stable Before WRITE</td>
<td>0</td>
<td></td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>$t_{WA}$</td>
<td>Address Hold After WRITE</td>
<td>0</td>
<td></td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>$t_{WW}$</td>
<td>WRITE Width</td>
<td>170</td>
<td></td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>$t_{DW}$</td>
<td>Data Set Up Time to the Trailing Edge of WRITE</td>
<td>130</td>
<td></td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>$t_{WD}$</td>
<td>Data Hold Time After WRITE</td>
<td>0</td>
<td></td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>$t_{DKDR4}$</td>
<td>RD↓ or WR↓ to DREQ↓</td>
<td>0</td>
<td>130</td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>$t_{DKDA6}$</td>
<td>RD↓ to Valid Data (D0-D7)</td>
<td>200</td>
<td></td>
<td>ns</td>
<td>DACK↓ to RD↓ 0 ≤ t ≤ 50 ns</td>
</tr>
</tbody>
</table>
WAVEFORMS

READ

WRITE

DMA
A.C. TIMING MEASUREMENT POINTS AND LOAD CONDITIONS

A.C. Testing: Inputs are driven at 2.4V for a Logic "1" and 0.45V for a Logic "0". Timing measurements are made at 2.0V for a Logic "1" and 0.8V for a Logic "0".

GPIB TIMINGS(1)

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Parameter</th>
<th>Max</th>
<th>Units</th>
<th>Test Conditions</th>
</tr>
</thead>
<tbody>
<tr>
<td>TEO13(2)</td>
<td>EOI ↓ to TR1 ↑</td>
<td>135</td>
<td>ns</td>
<td>PPSS, ATN = 0.45V</td>
</tr>
<tr>
<td>TEOD16</td>
<td>EOI ↓ to DIO Valid</td>
<td>155</td>
<td>ns</td>
<td>PPSS, ATN = 0.45V</td>
</tr>
<tr>
<td>TEOT12</td>
<td>EOI ↑ to TR1 ↓</td>
<td>155</td>
<td>ns</td>
<td>PPSS, ATN = 0.45V</td>
</tr>
<tr>
<td>TATND4</td>
<td>ATN ↓ to NDAC ↓</td>
<td>155</td>
<td>ns</td>
<td>TACS, AIDS</td>
</tr>
<tr>
<td>TATT14</td>
<td>ATN ↓ to TR1 ↓</td>
<td>155</td>
<td>ns</td>
<td>TACS, AIDS</td>
</tr>
<tr>
<td>TATT24</td>
<td>ATN ↓ to TR2 ↓</td>
<td>155</td>
<td>ns</td>
<td>TACS, AIDS</td>
</tr>
<tr>
<td>TDVND3-C</td>
<td>DAV ↓ to NDAC ↑</td>
<td>650</td>
<td>ns</td>
<td>AH, CACS</td>
</tr>
<tr>
<td>TNDDV1</td>
<td>NDAC ↑ to DAV ↑</td>
<td>350</td>
<td>ns</td>
<td>SH, STRS</td>
</tr>
<tr>
<td>TNRD1</td>
<td>NRFD ↑ to DREQ ↑</td>
<td>400</td>
<td>ns</td>
<td>SH</td>
</tr>
<tr>
<td>TDVDR3</td>
<td>DAV ↓ to DREQ ↑</td>
<td>600</td>
<td>ns</td>
<td>AH, LACS, ATN = 2.4V</td>
</tr>
<tr>
<td>TDVND2-C</td>
<td>DAV ↑ to NDAC ↓</td>
<td>350</td>
<td>ns</td>
<td>AH, LACS</td>
</tr>
<tr>
<td>TDVNR1-C</td>
<td>DAV ↑ to NRFD ↑</td>
<td>350</td>
<td>ns</td>
<td>AH, LACS, rdy = True</td>
</tr>
<tr>
<td>TRDR3</td>
<td>RD ↓ to NRFD ↑</td>
<td>500</td>
<td>ns</td>
<td>AH, LACS</td>
</tr>
<tr>
<td>TWRD15</td>
<td>WR ↑ to DIO Valid</td>
<td>280</td>
<td>ns</td>
<td>SH, TACS, RS = 0.4V</td>
</tr>
<tr>
<td>TWRE05</td>
<td>WR ↑ to EOI Valid</td>
<td>350</td>
<td>ns</td>
<td>SH, TACS</td>
</tr>
<tr>
<td>TWRDV2</td>
<td>WR ↑ to DAV ↓</td>
<td>830 + tSYNC</td>
<td>ns</td>
<td>High Speed Transfers Enabled, N_F = f_C, tSYNC = 1/2*f_C</td>
</tr>
</tbody>
</table>

NOTES:
1. All GPIB timings are at the pins of the 8291A.
2. The last number in the symbol for any GPIB timing parameter is chosen according to the transition directions of the reference signals. The following table describes the numbering scheme.

<table>
<thead>
<tr>
<th>Transition Direction</th>
<th>Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>↑ to ↑</td>
<td>1</td>
</tr>
<tr>
<td>↑ to ↓</td>
<td>2</td>
</tr>
<tr>
<td>↓ to ↑</td>
<td>3</td>
</tr>
<tr>
<td>↓ to ↓</td>
<td>4</td>
</tr>
<tr>
<td>↑ to VALID</td>
<td>5</td>
</tr>
<tr>
<td>↓ to VALID</td>
<td>6</td>
</tr>
</tbody>
</table>
MODIFIED STATE DIAGRAMS

Figure A-1 presents the interface function state diagrams. It is derived from IEEE Std. state diagrams, with the following changes:

A. The 8291A supports the complete set of IEEE-488 interface functions except for the controller. These include: SH1, AH1, T5, TE5, L3, LE3, SR1, RL1, PP1, DC1, DT1, and C0.

B. Addressing modes included in T, L state diagrams.

Note that in Mode 3, MSA, OSA are generated only after secondary address validity check by the microprocessor (APT interrupt).

C. In these modified state diagrams, the IEEE-488-1978 convention of negative (low true) logic is followed. This should not be confused with the Intel pin- and signal-naming convention based on positive logic. Thus, while the state diagrams below carry low true logic, the signals described elsewhere in this data sheet are consistent with Intel notation and are based on positive logic.

<table>
<thead>
<tr>
<th>Level</th>
<th>Logic</th>
<th>IEEE-488</th>
<th>Intel</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>T</td>
<td>DAV</td>
<td>DAV</td>
</tr>
<tr>
<td>1</td>
<td>F</td>
<td>DAV</td>
<td>DAV</td>
</tr>
<tr>
<td>0</td>
<td>T</td>
<td>NDAC</td>
<td>NDAC</td>
</tr>
<tr>
<td>1</td>
<td>F</td>
<td>NDAC</td>
<td>NDAC</td>
</tr>
<tr>
<td>0</td>
<td>T</td>
<td>NRFD</td>
<td>NRFD</td>
</tr>
<tr>
<td>1</td>
<td>F</td>
<td>NRFD</td>
<td>NRFD</td>
</tr>
</tbody>
</table>

Consider the condition when the Not-Ready-For-Data signal (pin 37) is active. Intel indicates this active low signal with the symbol NRFD (V_{OUT} \leq V_{OL} for AH; \text{V}_{IN} \leq V_{IL} for SH). The IEEE-488-1978 Standard, in its state diagrams, indicates the active state of this signal (True condition) with NRFD.

D. All remote multiline messages decoded are conditioned by ACDS. The multiplication by ACDS is not drawn to simplify the diagrams.

E. The symbol indicates:

1. When event X occurs, the function returns to state S.
2. X overrides any other transition condition in the function.

Statement 2 simplifies the diagram, avoiding the explicit use of X to condition all transitions from S to other states.

Figure A-1. 8291A State Diagrams
**Figure A-1. 8291A State Diagrams (Continued)**

- **F2** = ATN + LACS + LADS
- **F3** = ATN + rdy
- **T3'** = T3 + CPT + APT

---

**Notes:**
- "This transition will never occur under normal operation.
- TDELAY is about 300 ns for debouncing DAV.

---

**Diagram Details:**
- **TIDS:**
  - pon
  - IFC (WITHIN T4)

- **TADS:**
  - F4

- **SPAS:**
  - ATN (WITHIN T2)
  - ATN SPMS

- **TACS:**
  - DAB AVAILABLE TO SH
  - EOI IF DAB = EOS

- **TODAY:**
  - F4 = QTA + (OSA + TPAS + MSA + LPAS) • MODE 1 + MLA • MODE 1
Figure A-1. 8291A State Diagrams (Continued)
Figure A-1. 8291A State Diagrams (Continued)
Figure A-1. 8291A State Diagrams (Continued)
APPENDIX B

Table B-1. IEEE 488 Time Values

<table>
<thead>
<tr>
<th>Time Value Identifier(1)</th>
<th>Function (Applies to)</th>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>T1</td>
<td>SH</td>
<td>Settling Time for Multiline Messages</td>
<td>≥ 2 μs(2)</td>
</tr>
<tr>
<td>t2</td>
<td>LC, IC, SH, AH, T, L</td>
<td>Response to ATN</td>
<td>≤ 200 ns</td>
</tr>
<tr>
<td>T3</td>
<td>AH</td>
<td>Interface Message Accept Time(3)</td>
<td>&gt; 0(4)</td>
</tr>
<tr>
<td>t4</td>
<td>T, TE, L, LE, C, CE</td>
<td>Response to IFC or REN False</td>
<td>&lt; 100 μs</td>
</tr>
<tr>
<td>t5</td>
<td>PP</td>
<td>Response to ATN + EOI</td>
<td>≤ 200 ns</td>
</tr>
<tr>
<td>T6</td>
<td>C</td>
<td>Parallel Poll Execution Time</td>
<td>≥ 2 μs</td>
</tr>
<tr>
<td>T7</td>
<td>C</td>
<td>Controller Delay to Allow Current Talker to see ATN Message</td>
<td>≥ 500 ns</td>
</tr>
<tr>
<td>T8</td>
<td>C</td>
<td>Length of IFC or REN False</td>
<td>&gt; 100 μs</td>
</tr>
<tr>
<td>T9</td>
<td>C</td>
<td>Delay for EOI(5)</td>
<td>≥ 1.5 μs(6)</td>
</tr>
</tbody>
</table>

NOTES:
1. Time values specified by a lower case t indicate the maximum time allowed to make a state transition. Time values specified by an upper case T indicate the minimum time that a function must remain in a state before exiting.
2. If three-state drivers are used on the DIO, DAV, and EOI lines, T1 may be:
   1. ≥ 1100 ns.
   2. Or ≥ 700 ns if it is known that within the controller ATN is driven by a three-state driver.
   3. Or ≥ 500 ns for all subsequent bytes following the first sent after each false transition of ATN (the first byte must be sent in accordance with (1) or (2)).
   4. Or ≥ 350 ns for all subsequent bytes following the first sent after each false transition of ATN under conditions specified in Section 5.2.3 and warning note. See IEEE Standard 488.
   3. Time required for interface functions to accept, not necessarily respond to interface messages.
   4. Implementation dependent.
   5. Delay required for EOI, NDAC, and NRFD signal lines to indicate valid states.
   6. ≥ 600 ns for three-state drivers.
APPENDIX C
THE THREE-WIRE HANDSHAKE

Figure C-1. 3-Wire Handshake Timing at 8291A
The 8292 GPIB Controller is a microprocessor-controlled chip designed to function with the 8291 GPIB Talker/Listener to implement the full IEEE Standard 488 controller function, including transfer control protocol. The 8292 is a preprogrammed Intel® 8041A.
Table 1. Pin Description

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Pin Number</th>
<th>Type</th>
<th>Name and Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>IFCL</td>
<td>1</td>
<td>I</td>
<td>IFC RECEIVED (LATCHED): The 8292 monitors the IFC Line (when not system controller) through this pin.</td>
</tr>
<tr>
<td>X1, X2</td>
<td>2, 3</td>
<td>I</td>
<td>CRYSTAL INPUTS: Inputs for a crystal, LC or an external timing signal to determine the internal oscillator frequency.</td>
</tr>
<tr>
<td>RESET</td>
<td>4</td>
<td>I</td>
<td>RESET: Used to initialize the chip to a known state during power on.</td>
</tr>
<tr>
<td>CS</td>
<td>6</td>
<td>I</td>
<td>CHIP SELECT INPUT: Used to select the 8292 from other devices on the common data bus.</td>
</tr>
<tr>
<td>RD</td>
<td>8</td>
<td>I</td>
<td>READ ENABLE: Allows the master CPU to read from the 8292.</td>
</tr>
<tr>
<td>A0</td>
<td>9</td>
<td>I</td>
<td>ADDRESS LINE: Used to select between the data bus and the status register during read operations and to distinguish between data and commands written into the 8292 during write operations.</td>
</tr>
<tr>
<td>WR</td>
<td>10</td>
<td>I</td>
<td>WRITE ENABLE: Allows the master CPU to write to the 8292.</td>
</tr>
<tr>
<td>SYNC</td>
<td>11</td>
<td>O</td>
<td>SYNC: 8041A instruction cycle synchronization signal; it is an output clock with a frequency of XTAL (\div) 15.</td>
</tr>
<tr>
<td>D0–D7</td>
<td>12–19</td>
<td>I/O</td>
<td>DATA: 8 bidirectional lines used for communication between the central processor and the 8292’s data bus buffers and status register.</td>
</tr>
<tr>
<td>VSS</td>
<td>7, 20</td>
<td>P.S.</td>
<td>GROUND: Circuit ground potential.</td>
</tr>
<tr>
<td>SRQ</td>
<td>21</td>
<td>I</td>
<td>SERVICE REQUEST: One of the IEEE control lines. Sampled by the 8292 when it is controller in charge. If true, SPI interrupt to the master will be generated.</td>
</tr>
<tr>
<td>ATNI</td>
<td>22</td>
<td>I</td>
<td>ATTENTION IN: Used by the 8292 to monitor the GPIB ATN control line. If is used during the transfer control procedure.</td>
</tr>
<tr>
<td>IFC</td>
<td>23</td>
<td>I/O</td>
<td>INTERFACE CLEAR: One of the GPIB management lines, as defined by IEEE Std. 488-1978, places all devices in a known quiescent state.</td>
</tr>
<tr>
<td>SYC</td>
<td>24</td>
<td>I</td>
<td>SYSTEM CONTROLLER: Monitors the system controller switch.</td>
</tr>
<tr>
<td>CLTH</td>
<td>27</td>
<td>O</td>
<td>CLEAR LATCH: Used to clear the IFCR latch after being recognized by the 8292. Usually low (except after hardware Reset), it will be pulsed high when IFCR is recognized by the 8292.</td>
</tr>
<tr>
<td>ATNO</td>
<td>29</td>
<td>O</td>
<td>ATTENTION OUT: Controls the ATN control line of the bus through external logic for tcs and tca procedures. (ATN is a GPIB control line, as defined by IEEE Std. 488-1978.)</td>
</tr>
<tr>
<td>VCC</td>
<td>5, 26, 40</td>
<td>P.S.</td>
<td>VOLTAGE: +5V supply input (\pm) 10%.</td>
</tr>
<tr>
<td>COUNT</td>
<td>39</td>
<td>I</td>
<td>EVENT COUNT: When enabled by the proper command the internal counter will count external events through this pin. High to low transition will increment the internal counter by one. The pin is sampled once per three internal instruction cycles (7.5 (\mu)sec sample period when using 5 MHz XTAL). It can be used for byte counting when connected to NDAC, or for block counting when connected to the EOI.</td>
</tr>
<tr>
<td>REN</td>
<td>38</td>
<td>O</td>
<td>REMOTE ENABLE: The Remote Enable bus signal selects remote or local control of the device on the bus. A GPIB bus signal selects remote or local control of the device on the bus. A GPIB bus management line, as defined by IEEE Std. 488-1978.</td>
</tr>
</tbody>
</table>
Table 1. Pin Description (Continued)

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Pin No.</th>
<th>Type</th>
<th>Name and Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>DAV</td>
<td>37</td>
<td>I/O</td>
<td>DATA VALID: Used during parallel poll to force the 8291 to accept the parallel poll status bit. It is also used during the tcs procedure.</td>
</tr>
<tr>
<td>IBFI</td>
<td>36</td>
<td>O</td>
<td>INPUT BUFFER NOT FULL: Used to interrupt the central processor while the input buffer of the 8292 is empty. This feature is enabled and disabled by the interrupt mask register.</td>
</tr>
<tr>
<td>OBFI</td>
<td>36</td>
<td>O</td>
<td>OUTPUT BUFFER FULL: Used as an interrupt to the central processor while the output buffer of the 8292 is full. The feature can be enabled and disabled by the interrupt mask register.</td>
</tr>
<tr>
<td>EOT2</td>
<td>34</td>
<td>I/O</td>
<td>END OR IDENTIFY: One of the GPIB management lines, as defined by IEEE Std. 488-1978. Used with ATN as Identify Message during parallel poll.</td>
</tr>
<tr>
<td>SPI</td>
<td>33</td>
<td>O</td>
<td>SPECIAL INTERRUPT: Used as an interrupt on events not initiated by the central processor.</td>
</tr>
<tr>
<td>TCI</td>
<td>32</td>
<td>O</td>
<td>TASK COMPLETE INTERRUPT: Interrupt to the control processor used to indicate that the task requested was completed by the 8292 and the information requested is ready in the data bus buffer.</td>
</tr>
<tr>
<td>CIC</td>
<td>31</td>
<td>O</td>
<td>CONTROLLER IN CHARGE: Controls the S/R input of the SRQ bus transceiver. It can also be used to indicate that the 8292 is in charge of the GPIB bus.</td>
</tr>
</tbody>
</table>

**FUNCTIONAL DESCRIPTION**

The 8292 is an Intel 8041A which has been programmed as a GPIB Controller Interface element. It is used with the 8291 GPIB Talker/Listener and two 8293 GPIB Transceivers to form a complete IEEE-488 Bus Interface for a microprocessor. The electrical interface is performed by the transceivers, data transfer is done by the talker/listener, and control of the bus is done by the 8292. Figure 3 is a typical controller interface using Intel's GPIB peripherals.

![Figure 3. Talker/Listener/Controller Configuration](image)
The internal RAM in the 8041A is used as a special purpose register bank for the 8292. Most of these registers (except for the interrupt flag) can be accessed through commands to the 8292. Table 2 identifies the registers used by the 8292 and how they are accessed.

**Interrupt Status Register**

<table>
<thead>
<tr>
<th>SYC</th>
<th>ERR</th>
<th>SRQ</th>
<th>EV</th>
<th>X</th>
<th>IFCR</th>
<th>IBF</th>
<th>OBF</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The 8292 can be configured to interrupt the microprocessor on one of several conditions. Upon receipt of the interrupt the microprocessor must read the 8292 interrupt status register to determine which event caused the interrupt, and then the appropriate subroutine can be performed. The interrupt status register is read with A0 high. With the exception of OBF and IBF, these interrupts are enabled or disabled by the SPI interrupt mask. OBF and IBF have their own bits in the interrupt mask (OBFI and IBFI).

**OBF** Output Buffer Full. A byte is waiting to be read by the microprocessor. This flag is cleared when the output data bus register is read.

**IBF** Input Buffer Full. The byte previously written by the microprocessor has not been read yet by the 8292. If another byte is written to the 8292 before this flag clears, data will be lost. IBF is cleared when the 8292 reads the data byte.

**IFCR** Interface Clear Received. The GPIB system controller has set IFC. The 8292 has become idle and is no longer is charge of the bus. The flag is cleared when the IACK command is issued.

**EV** Event Counter Interrupt. The requested number of blocks of data byte has been transferred. The EV interrupt flag is cleared by the IACK command.

**SRQ** Service Request. Notified the 8292 that a service request (SRQ) message has been received. It is cleared by the IACK command.

**ERR** Error occurred. The type of error can be determined by reading the error status register. This interrupt flag is cleared by the IACK command.

**SYNC** System Controller Switch Change. Notifies the processor that the state of the system controller switch has changed. The actual state is contained in the GPIB Status Register. This flag is cleared by the IACK command.

**Interrupt Mask Register**

<table>
<thead>
<tr>
<th>1</th>
<th>SPI</th>
<th>TCI</th>
<th>SYC</th>
<th>OBFI</th>
<th>IBFI</th>
<th>0</th>
<th>SRQ</th>
</tr>
</thead>
<tbody>
<tr>
<td>D7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The Interrupt Mask Register is used to enable features and to mask the SPI and TCI interrupts. The flags in the Interrupt Status Register will be active even when masked out. The Interrupt Mask Register is written when A0 is low and reset by the RINM command. When the register is read, D1 and D7 are undefined. An interrupt is enabled by setting the corresponding register bit.

**SRQ** Enable interrupts on SRQ received.

**IBFI** Enable interrupts on input buffer empty.

**OBFI** Enable interrupts on output buffer full.

---

**Table 2. 8292 Registers**

<table>
<thead>
<tr>
<th>READ FROM 8292</th>
<th>WRITE TO 8292</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>INTERRUPT STATUS</strong></td>
<td><strong>INTERRUPT MASK</strong></td>
</tr>
<tr>
<td>SYC</td>
<td>ERR</td>
</tr>
<tr>
<td>D7</td>
<td></td>
</tr>
<tr>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td><strong>CONTROLLER STATUS</strong></td>
<td><strong>COMMAND FIELD</strong></td>
</tr>
<tr>
<td>CSBS</td>
<td>CA</td>
</tr>
<tr>
<td>REN</td>
<td>DAV</td>
</tr>
<tr>
<td><strong>EVENT COUNTER STATUS</strong></td>
<td><strong>EVENT COUNTER</strong></td>
</tr>
<tr>
<td>D</td>
<td>D</td>
</tr>
<tr>
<td><strong>TIME OUT STATUS</strong></td>
<td><strong>TIME OUT</strong></td>
</tr>
<tr>
<td>D</td>
<td>D</td>
</tr>
</tbody>
</table>

NOTE: These registers are accessed by a special utility command, see page 7.
SYC  Enable interrupts on a change in the system controller switch.
TCI  Enable interrupts on the task completed.
SPI  Enable interrupts on special events.

NOTE:
The event counter is enabled by the GSEC command, the error interrupt is enabled by the error mask register, and IFC cannot be masked (it will always cause an interrupt).

Controller Status Register

<table>
<thead>
<tr>
<th>CSBS</th>
<th>CA</th>
<th>X</th>
<th>X</th>
<th>SYCS</th>
<th>IFC</th>
<th>REN</th>
<th>SRQ</th>
</tr>
</thead>
<tbody>
<tr>
<td>D7</td>
<td>D6</td>
<td>D5</td>
<td>D4</td>
<td>D3</td>
<td>D2</td>
<td>D1</td>
<td>D0</td>
</tr>
</tbody>
</table>

The Controller Status Register is used to determine the status of the controller function. This register is accessed by the RCST command.

SRQ  Service Request line active (CSRS).
REN  Sending Remote Enable.
IFC  Sending or receiving interface clear.
SYCS System Controller Switch Status (SACS).
CA  Controller Active (CACS + CAWS + CSWS).
CSBS Controller Stand-by State (CSBS, CA) = (0,0)—Controller Idle.

GPIB Bus Status Register

<table>
<thead>
<tr>
<th>REN</th>
<th>DAV</th>
<th>EOI</th>
<th>X</th>
<th>SYC</th>
<th>IFC</th>
<th>ATNI</th>
<th>SRQ</th>
</tr>
</thead>
<tbody>
<tr>
<td>D7</td>
<td>D6</td>
<td>D5</td>
<td>D4</td>
<td>D3</td>
<td>D2</td>
<td>D1</td>
<td>D0</td>
</tr>
</tbody>
</table>

This register contains GPIB bus status information. It can be used by the microprocessor to monitor and manage the bus. The GPIB Bus Register can be read using the RBST command.

Each of these status bits reflect the current status of the corresponding pin on the 8292.

SRQ  Service Request
ATNI Attention In
IFC  Interface Clear
SYC  System Controller Switch
EOI  End or Identify
DAV  Data Valid
REN  Remote Enable

Event Counter Register

<table>
<thead>
<tr>
<th>D7</th>
<th>D6</th>
<th>D5</th>
<th>D4</th>
<th>D3</th>
<th>D2</th>
<th>D1</th>
<th>D0</th>
</tr>
</thead>
</table>

The Event Counter Register contains the initial value for the event counter. The counter can count pulses on pin 39 of the 8292 (COUNT). It can be connected to EOI or NDAC to count blocks or bytes respectively during standby state. A count of zero equals 256. This register cannot be read, and is written using the WEVC command.

Event Counter Status Register

| D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |

This register contains the current value in the event counter. The event counter counts back from the initial value stored in the Event Counter Register to zero and then generates an Event Counter Interrupt. This register cannot be written and can be read using a REVC command.

Time Out Register

| D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |

The Time Out Register is used to store the time used for the time out error function. See the individual timeouts (TOUT1, 2, 3) to determine the units of this counter. This Time Out Register cannot be read, and it is written with the WOUT command.

Time Out Status Register

| D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |

This register contains the current value in the time out counter. The time out counter decrements from the original value stored in the Time Out Register. When zero is reached, the appropriate error interrupt is generated. If the register is read while none of the time out functions are active, the register will contain the last value reached the last time a function was active. The Time Out Status Register cannot be written, and it is read with RTOUT command.

Error Flag Register

| X | X | USER | X | X | TOUT3 | TOUT2 | TOUT1 |

Four errors are flagged by the 8292 with a bit in the Error Flag Register. Each of these errors can be masked by the Error Mask Register. The Error Flag Register cannot be written, and it is read by the IACK command when the error flag in the Interrupt Status Register is set.

TOUT1 Time Out Error 1 occurs when the current controller has not stopped sending ATN after receiving the TCT message for the time period specified by the Time Out Register. Each count in the Time Out Register is at least 1800 tCY. After flagging the error, the 8292 will remain in a loop trying to take control until the current controller stops send-
Commands are performed by the 8292 whenever a command is written by the microprocessor. If a new command is written, the 8292 will return to the loop after executing it.

**TOUT2** Time Out Error 2 occurs when the transmission between the addressed talker and listener has not started for the time period specified by the Time Out Register. Each count in the Time Out Register is at least 45 t_CY. This feature is only enabled when the controller is in the CSBS state.

**TOUT3** Time Out Error 3 occurs when the handshake signals are stuck and the 8292 is not succeeding in taking control synchronously for the time period specified by the Time Out Register. Each count in the Time Out Register is at least 1800 t_CY. The 8292 will continue checking ATNI until it becomes true or a new command is received. After performing the new command, the 8292 will return to the ATNI checking loop.

**USER** User error occurs when request to assert IFC or REN was received and the 8292 was not the system controller.

### Error Mask Register

<table>
<thead>
<tr>
<th>D7</th>
<th>D6</th>
<th>D5</th>
<th>D4</th>
<th>D3</th>
<th>D2</th>
<th>D1</th>
<th>D0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>USER</td>
<td>0</td>
<td>0</td>
<td>TOUT3</td>
<td>TOUT2</td>
<td>TOUT1</td>
</tr>
</tbody>
</table>

The Error Mask Register is used to mask the interrupt from a particular type of error. Each type of error interrupt is enabled by setting the corresponding bit in the Error Mask Register. This register can be read with the RERM command and written with A0 low.

### Command Register

<table>
<thead>
<tr>
<th>D7</th>
<th>D6</th>
<th>D5</th>
<th>D4</th>
<th>D3</th>
<th>D2</th>
<th>D1</th>
<th>D0</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>OP</td>
<td>C</td>
<td>C</td>
<td>C</td>
<td>C</td>
</tr>
</tbody>
</table>

Commands are performed by the 8292 whenever a byte is written with A0 high. There are two categories of commands distinguished by the OP bit (bit 4). The first category is the operation command (OP = 1). These commands initiate some action on the interface bus. The second category is the utility command (OP = 0). These commands are used to aid the communication between the processor and the 8292.

**OPERATION COMMANDS**

Operation commands initiate some action on the GPIB interface bus. It is using these commands that the control functions such as polling, taking and passing control, and system controller functions are performed.

**F0—SPCNI—Stop Counter Interrupts**

This command disables the internal counter interrupt so that the 8292 will stop interrupting the master on event counter underflows. However, the counter will continue counting and its contents can still be used.

**F1—GIDL—Go To Idle**

This command is used during the transfer of control procedure while transferring control to another controller. The 8292 will respond to this command only if it is in the active state. ATNO will go high, and CIC will be high so that this 8292 will no longer be driving the ATN line on the GPIB interface bus. TCI will be set upon completion.

**F2—RST—Reset**

This command has the same effect as asserting the external reset on the 8292. For details, refer to the reset procedure described later.

**F3—RSTI—Reset Intermits**

This command resets any pending interrupts and clears the error flags. The 8292 will not return to any loop it was in (such as from the time out interrupts).

**F4—GSEC—Go To Standby, Enable Counting**

The function causes ATNO to go high and the controller will be enabled. If the 8292 was not the active controller, this command will exit immediately. If the 8292 is the active controller, the controller will be loaded with the value stored in the Event Counter Register, and the internal interrupt will be enabled so that when the counter reaches zero, the SPI interrupt will be generated. SPI will be generated every 256 counts thereafter until the controller exits the standby state or the SPCNI command is written. An initial count of 256 (zero in the Event Counter Register) will be used if the WEVC command is not executed. If the data transmission does not start, a TOUT2 error will be generated.

**F5—EXPP—Execute Parallel Poll**

This command initiates a parallel poll by asserting EOI when ATN is already active. TCI will be set at the end of the command. The 8291 should be previously configured as a listener. Upon detection of DAV true, the 8291 enters ACDS and latches the parallel poll response (PPR) byte into its data in register. The master will be interrupted by the 8291 BI interrupt when the PPR byte is available. No interrupts except the IBFI will be generated by the 8292. The 8292 will respond to this command only when it is the active controller.
F6—GTSB—Go To Standby

If the 8292 is the active controller, ATNO will go high then TCI will be generated. If the data transmission does not start, a TOUT2 error will be generated.

F7—SLOC—Set Local Mode

If the 8292 is the system controller, then REN will be asserted false and TCI will be set true. If it is not the system controller, the User Error bit will be set in the Error Flag Register.

F8—SREM—Set Interface To Remote Control

This command will set REN true and TCI true if this 8292 is the system controller. If not, the User Error bit will be set in the Error Flag Register.

F9—ABORT—Abort All Operation, Clear Interface

This command will cause IFC to be asserted true for at least 100 μsec if this 8292 is the system controller. If it is in CIDS, it will take control over the bus (see the TCNT command).

FA—TCNTR—Take Control

The transfer of control procedure is coordinated by the master with the 8291 and 8292. When the master receives a TCT message from the 8291, it should issue the TCNTR command to the 8292. The following events occur to take control:

1) The 8292 checks to see if it is in CIDS, and if not, it exits.
2) Then ATNIi is checked until it becomes high. If the current controller does not release ATN for the time specified by the Time Out Register, then a TOUT1 error is generated. The 8292 will return to this loop after an error or any command except the RST and RSTI commands.
3) After the current controller releases ATN, the 8292 will assert ATNO and CIC low.
4) Finally, the TCI interrupt is generated to inform the master that it is in control of the bus.

FC—TCASY—Take Control Asynchronously

TCAS transfers the 8292 from CSBS to CACS independent of the handshake lines. If a bus hangup is detected (by an error flag), this command will force the 8292 to take control (asserting ATN) even if the AH function is not in ANRS (Acceptor Not Ready State). This command should be used very carefully since it may cause the loss of a data byte. Normally, control should be taken synchronously. After check-

ing the controller function for being in the CSBS (else it will exit immediately), ATNO will go low, and a TCI interrupt will be generated.

FD—TCSY—Take Control Synchronously

There are two different procedures used to transfer the 8292 from CSBS to CACS depending on the state of the 8291 in the system. If the 8291 is in "continuous AH cycling" mode (Aux. Reg. A0 = A1 = 1), then the following procedures should be followed:

1) The master microprocessor stops the continuous AH cycling mode in the 8291;
2) The master reads the 8291 Interrupt Status 1 Register;
3) If the END bit is set, the master sends the TCSY command to the 8292;
4) If the END bit was not set, the master reads the 8291 Data In Register and then waits for another BI interrupt from the 8291. When it occurs, the master sends the 8292 the TCSY command.

If the 8291 is not in AH cycling mode, then the master just waits for a BI interrupt and then sends the TCSY command. After the TCSY command has been issued, the 8292 checks for CSBS. If CSBS, then it exits the routine. Otherwise, it then checks the DAV bit in the GPIB status. When DAV becomes false, the 8292 will wait for at least 1.5 μsec. (T10) and then ATNO will go low. If DAV does not go low, a TOUT3 error will be generated. If the 8292 successfully takes control, it sets TCI true.

FE—STCNl—Start Counter Interrupts

This command enables the internal counter interrupt. The counter is enabled by the GSEC command.

UTILITY COMMANDS

All these commands are either Read or Write to registers in the 8292. Note that writing to the Error Mask Register and the Interrupt Mask Register are done directly.

E1—WTOU T—Write To Time Out Register

The byte written to the data bus buffer (with A0 = 0) following this command will determine the time used for the time out function. Since this function is implemented in software, this will not be an accurate time measurement. This feature is enable or disable by the Error Mask Register. No interrupts except for the IBFI will be generated upon completion.
E2—WEVC—Write To Event Counter

The byte written to the data bus buffer (with \(A_0 = 0\)) following this command will be loaded into the Event Counter Register and the Event Counter Status for byte counting of EOI counting. Only IBF will indicate completion of this command.

E3—REVC—Read Event Counter Status

This command transfers the contents of the Event Counter into the data bus buffer. A TCI is generated when the data is available in the data bus buffer.

E4—RERF—Read Error Flag Register

This command transfers the contents of the Error Flag Register into the data bus buffer. A TCI is generated when the data is available.

E5—RINM—Read Interrupt Mask Register

This command transfers the contents of the Interrupt Mask Register into the data bus buffer. This register is available to the processor so that it does not need to store this information elsewhere. A TCI is generated when the data is available in the data bus buffer.

E6—RCST—Read Controller Status Register

This command transfers the contents of the Controller Status Register into the data bus buffer and a TCI interrupt is generated.

E7—RBST—Read GPIB Bus Status Register

This command transfers the contents of the GPIB Bus Status Register into the data bus buffer, and a TCI interrupt is generated when the data is available.

E9—RTOUT—Read Time Out Status Register

This command transfers the contents of the Time Out Status Register into the data bus buffer, and a TCI interrupt is generated when the data is available.

EA—RERM—Read Error Mask Register

This command transfers the contents of the Error Mask Register to the data bus buffer so that the processor does not need to store this information elsewhere. A TCI interrupt is generated when the data is available.

Interrupt Acknowledge

<table>
<thead>
<tr>
<th>SYC</th>
<th>ERR</th>
<th>SRQ</th>
<th>EV</th>
<th>IFCR</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Each named bit in an Interrupt Acknowledge (lACK) corresponds to a flag in the Interrupt Status Register. When the 8292 receives this command, it will clear the SPI and the corresponding bits in the Interrupt Status Register. If not all the bits were cleared, then the SPI will be set true again. If the error flag is not acknowledged by the IACK command, then the Error Flag Register will be transferred to the data bus buffer, and a TCI will be generated.

NOTE:

XXXX1X11 is an undefined operation or utility command, so no conflict exists between the IACK operation and utility commands.

SYSTEM OPERATION

8292 To Master Processor Interface

Communication between the 8292 and the Master Processor can be either interrupt based communication or based upon polling the interrupt status register in predetermined intervals.

Interrupt Based Communication

Four different interrupts are available from the 8292:

- OBFI Output Buffer Full Interrupt
- IBFI Input Buffer Not Full Interrupt
- TCI Task Completed Interrupt
- SPI Special Interrupt

Each of the interrupts is enabled or disabled by a bit in the interrupt mask register. Since OBFI and IBFI are directly connected to the OBF and IBF flags, the master can write a new command to the input data bus buffer as soon as the previous command has been read.

The TCI interrupt is useful when the master is sending commands to the 8292. The pending TCI will be cleared with each new command written to the 8292. Commands sent to the 8292 can be divided into two major groups:

1) Commands that require response back from the 8292 to the master, e.g., reading register.
2) Commands that initiate some action or enable features but do not require response back from the 8292, e.g., enable data bus buffer interrupts.
With the first group, the TCI interrupt will be used to indicate that the required response is ready in the data bus buffer and the master may continue and read it. With the second group, the interrupt will be used to indicate completion of the required task, so that the master may send new commands.

The SPI should be used when immediate information or special events is required (see the Interrupt Status Register).

"Polling Status" Based Communication

When interrupt based communication is not desired, all interrupts can be masked by the interrupt mask register. The communication with the 8292 is based upon sequential poll of the interrupt status register. By testing the OBF and IBF flags, the data bus buffer status is determined while special events are determined by testing the other bits.

Receiving IFC

The IFC pulse defined by the IEEE-488 standard is at least 100 μsec. In this time, all operation on the bus should be aborted. Most important, the current controller (the one that is in charge at that time) should stop sending ATN or EOI. Thus, IFC must externally gate CIC (controller in charge) and ATNO to ensure that this occurs.

Reset and Power Up Procedure

After the 8292 has been reset either by the external reset pin, the device being powered on, or a RST command, the following sequential events will take place:

1) All outputs to the GPIB interface will go high (SRQ, ATNI, IFC, SYC, CLTH, ATNO, CIC, TCI, SPI, EOI, OBF, IBF, DAV, REV).

2) The four interrupt outputs (TCI, SPI, OBF, IBF) and CLTH output will go low.

3) The following registers will be cleared:
   - Interrupt Status
   - Interrupt Mask
   - Error Flag
   - Error Mask
   - Time Out
   - Event Counter (= 256), counter is disabled.

4) If the 8292 is the system controller, and ABORT command will be executed, the 8292 will become the controller in charge, and it will enter the CACS state.
   - If it is not the system controller, it will remain in CIDS.

System Configuration

The 8291 and 8292 must be interfaced to an IEEE-488 bus meeting a variety of specifications including drive capability and loading characteristics. To interface the 8291 and the 8292 without the 8293's, several external gates are required, using a configuration similar to that used in Figure 5.
NOTES:
1. Connect to NDAC for byte count or to EOI for block count.
2. Gate ensures open collector operation during parallel poll.

Figure 4. 8291 and 8292 System Configuration
NOTES:
* = GPIB bus transceiver
† = See 8041 data sheet for alternate crystal configurations
‡‡ = Can connect to system reset switch, see 8041A data sheet

Figure 5. 8291, 8292, and 8293 System Configuration
**ABSOLUTE MAXIMUM RATING**

Ambient Temperature Under Bias .............. 0°C to 70°C
Storage Temperature .................... −65°C to +150°C
Voltage to Any Pin with Respect to Ground .............. 0.5V to +7V
Power Dissipation ....................... 1.5 Watt

*Notice: Stresses above those listed under “Absolute Maximum Ratings” may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions above those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

**D.C. CHARACTERISTICS** $T_A = 0°C$ to $70°C$, $V_{SS} = 0V$: 8292, $V_{CC} = \pm 5V \pm 10%$

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Parameter Description</th>
<th>Min</th>
<th>Max</th>
<th>Unit</th>
<th>Test Conditions</th>
</tr>
</thead>
<tbody>
<tr>
<td>$V_{IL1}$</td>
<td>Input Low Voltage (All Except $X_1$, $X_2$, RESET)</td>
<td>$-0.5$</td>
<td>$0.8$</td>
<td>V</td>
<td></td>
</tr>
<tr>
<td>$V_{IL2}$</td>
<td>Input Low Voltage ($X_1$, $X_2$, RESET)</td>
<td>$-0.5$</td>
<td>$0.6$</td>
<td>V</td>
<td></td>
</tr>
<tr>
<td>$V_{IH1}$</td>
<td>Input High Voltage (All Except $X_1$, $X_2$, RESET)</td>
<td>$2.2$</td>
<td>$V_{CC}$</td>
<td>V</td>
<td></td>
</tr>
<tr>
<td>$V_{IH2}$</td>
<td>Input High Voltage ($X_1$, $X_2$, RESET)</td>
<td>$3.8$</td>
<td>$V_{CC}$</td>
<td>V</td>
<td></td>
</tr>
<tr>
<td>$V_{OL1}$</td>
<td>Output Low Voltage ($D_0$–$D_7$)</td>
<td>$0.45$</td>
<td>$V$</td>
<td>$I_{OL} = 2.0\ mA$</td>
<td></td>
</tr>
<tr>
<td>$V_{OL2}$</td>
<td>Output Low Voltage (All Other Outputs)</td>
<td>$0.45$</td>
<td>$V$</td>
<td>$I_{OL} = 1.6\ mA$</td>
<td></td>
</tr>
<tr>
<td>$V_{OH1}$</td>
<td>Output High Voltage ($D_0$–$D_7$)</td>
<td>$2.4$</td>
<td>$V$</td>
<td>$I_{OH} = -400\ \mu A$</td>
<td></td>
</tr>
<tr>
<td>$V_{OH2}$</td>
<td>Output High Voltage (All Other Outputs)</td>
<td>$2.4$</td>
<td>$V$</td>
<td>$I_{OH} = -50\ \mu A$</td>
<td></td>
</tr>
<tr>
<td>$I_{IL}$</td>
<td>Input Leakage Current (COUNT, IFCL, RD, WR, $CS$, $A_0$)</td>
<td>$\pm 10\ \mu A$</td>
<td>$V_{SS} \leq V_{IN} \leq V_{CC}$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$I_{OZ}$</td>
<td>Output Leakage Current ($D_0$–$D_7$, High Z State)</td>
<td>$\pm 10\ \mu A$</td>
<td>$V_{SS} + 0.45 \leq V_{IN} \leq V_{CC}$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$I_{L11}$</td>
<td>Low Input Load Current (Pins 21–24, 27–38)</td>
<td>$0.5\ mA$</td>
<td>$V_{IL} = 0.8V$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$I_{L12}$</td>
<td>Low Input Load Current (RESET)</td>
<td>$0.2\ mA$</td>
<td>$V_{IL} = 0.8V$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$I_{CC}$</td>
<td>Total Supply Current</td>
<td>$125\ mA$</td>
<td>Typical = $65\ mA$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$I_{IH}$</td>
<td>Input High Leakage Current (Pins 21–24, 27–38)</td>
<td>$100\ \mu A$</td>
<td>$V_{IN} = V_{CC}$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$C_{IN}$</td>
<td>Input Capacitance</td>
<td>$10\ pF$</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$C_{I/O}$</td>
<td>I/O Capacitance</td>
<td>$20\ pF$</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**A.C. CHARACTERISTICS** $T_A = 0°C$ to $70°C$, $V_{SS} = 0V$: 8292, $V_{CC} = \pm 5V \pm 10%$

**DBB READ**

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Parameter Description</th>
<th>Min</th>
<th>Max</th>
<th>Unit</th>
<th>Test Conditions</th>
</tr>
</thead>
<tbody>
<tr>
<td>$t_{AR}$</td>
<td>$CS$, $A_0$ Setup to $RD$ ↓</td>
<td>$0$</td>
<td></td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>$t_{RA}$</td>
<td>$CS$, $A_0$ Hold to $RD$ ↑</td>
<td>$0$</td>
<td></td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>$t_{RD}$</td>
<td>$RD$ Pulse Width</td>
<td>$250$</td>
<td></td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>$t_{AD}$</td>
<td>$CS$, $A_0$ to Data Out Delay</td>
<td>$225$</td>
<td></td>
<td>ns</td>
<td>$C_L = 150\ pF$</td>
</tr>
<tr>
<td>$t_{RD}$</td>
<td>$RD$ ↓ to Data Out Delay</td>
<td>$225$</td>
<td></td>
<td>ns</td>
<td>$C_L = 150\ pF$</td>
</tr>
<tr>
<td>$t_{DF}$</td>
<td>$RD$ ↑ to Data Float Delay</td>
<td>$100$</td>
<td></td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>$t_{CY}$</td>
<td>Cycle Time</td>
<td>$2.5$</td>
<td>$15$</td>
<td>$\mu s$</td>
<td></td>
</tr>
</tbody>
</table>
### DBB WRITE

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Parameter</th>
<th>Min</th>
<th>Max</th>
<th>Unit</th>
<th>Test Conditions</th>
</tr>
</thead>
<tbody>
<tr>
<td>t_{AW}</td>
<td>CS, A_0 Setup to WR ↓</td>
<td>0</td>
<td></td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>t_{WA}</td>
<td>CS, A_0 Hold after WR ↑</td>
<td>0</td>
<td></td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>t_{WW}</td>
<td>WR Pulse Width</td>
<td>250</td>
<td></td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>t_{DW}</td>
<td>Data Setup to WR ↑</td>
<td>150</td>
<td></td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>t_{WD}</td>
<td>Data Hold after WR ↓</td>
<td>0</td>
<td></td>
<td>ns</td>
<td></td>
</tr>
</tbody>
</table>

### COMMAND TIMINGS\(^{(1,3)}\)

<table>
<thead>
<tr>
<th>Code</th>
<th>Name</th>
<th>Execution Time</th>
<th>IFI ↑</th>
<th>TCI(2)</th>
<th>SPI</th>
<th>ATNO</th>
<th>CIC</th>
<th>IFIC</th>
<th>REN</th>
<th>EOI</th>
<th>DAV</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>E1</td>
<td>WOUT</td>
<td>63</td>
<td>24</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>E2</td>
<td>WEVC</td>
<td>63</td>
<td>24</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>E3</td>
<td>REVC</td>
<td>71</td>
<td>24</td>
<td>51</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>E4</td>
<td>RERF</td>
<td>67</td>
<td>24</td>
<td>47</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>E5</td>
<td>RINM</td>
<td>69</td>
<td>24</td>
<td>49</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>E6</td>
<td>RCST</td>
<td>97</td>
<td>24</td>
<td>77</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>E7</td>
<td>RBST</td>
<td>92</td>
<td>24</td>
<td>72</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>E8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>E9</td>
<td>RTOUT</td>
<td>69</td>
<td>24</td>
<td>49</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EA</td>
<td>RERM</td>
<td>69</td>
<td>24</td>
<td>49</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F0</td>
<td>SPCNI</td>
<td>53</td>
<td>24</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Count Stops after 39</td>
</tr>
<tr>
<td>F1</td>
<td>GIOL</td>
<td>88</td>
<td>24</td>
<td>70</td>
<td>↑61</td>
<td>↑61</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Not System Controller</td>
</tr>
<tr>
<td>F2</td>
<td>RST</td>
<td>94</td>
<td>24</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>System Controller</td>
</tr>
<tr>
<td>F2</td>
<td>RST</td>
<td>214</td>
<td>24</td>
<td>192</td>
<td>↓52</td>
<td>↓174</td>
<td>↓179</td>
<td>↓101</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F3</td>
<td>RSTI</td>
<td>61</td>
<td>24</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F4</td>
<td>GSEC</td>
<td>125</td>
<td>24</td>
<td>107</td>
<td>↑98</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F5</td>
<td>EXPP</td>
<td>75</td>
<td>24</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F6</td>
<td>GTSB</td>
<td>118</td>
<td>24</td>
<td>100</td>
<td>↑91</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F7</td>
<td>SLOC</td>
<td>73</td>
<td>24</td>
<td>55</td>
<td></td>
<td>↑46</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F8</td>
<td>SREM</td>
<td>91</td>
<td>24</td>
<td>73</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F9</td>
<td>ABORT</td>
<td>155</td>
<td>24</td>
<td>133</td>
<td>↓120</td>
<td>↓115</td>
<td>↓42</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FA</td>
<td>TCNTR</td>
<td>108</td>
<td>24</td>
<td>86</td>
<td>↓71</td>
<td>↓68</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FC</td>
<td>TCAS</td>
<td>92</td>
<td>24</td>
<td>67</td>
<td>↓55</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FD</td>
<td>TCSY</td>
<td>115</td>
<td>24</td>
<td>91</td>
<td>↓80</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FE</td>
<td>STCNI</td>
<td>59</td>
<td>24</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Starts Count after 43</td>
</tr>
<tr>
<td>PIN</td>
<td>RESET</td>
<td>29</td>
<td>—</td>
<td>↓7</td>
<td>↓7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Not System Controller</td>
</tr>
<tr>
<td>X</td>
<td>IACK</td>
<td>116</td>
<td>—</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>If Interrupt Pending</td>
</tr>
</tbody>
</table>

**NOTES:**
1. All times are multiples of t_{CY} from the 8041A command interrupt.
2. TCI clears after 7 t_{CY} on all commands.
3. ↑ indicates a level transition from low to high, ↓ indicates a high to low transition.
A.C. TESTING INPUT, OUTPUT WAVEFORM

A.C. Testing: Inputs are driven at 2.4V for a Logic "1" and 0.45V for a Logic "0". Timing measurements are made at 2.0V for a Logic "1" and 0.8V for a Logic "0".

CLOCK DRIVER CIRCUITS

CRYSTAL OSCILLATOR MODE

NOTE:
Crystal series resistance should be <75Ω at 6 MHz; <180Ω at 3.6 MHz.

LC OSCILLATOR MODE

NOTES:
1. Cpp ≈ 5–10 pF pin-to-pin capacitance
2. Each C should be approximately 20 pF, including stray capacitance.
WAVEFORMS

READ OPERATION—DATA BUS BUFFER REGISTER

WRITE OPERATION—DATA BUS BUFFER REGISTER
APPENDIX A

The following tables and state diagrams were taken from the IEEE Standard Digital Interface for Programmable Instrumentation, IEEE Std. 488-1978. This document is the official standard for the GPIB bus and can be purchased from IEEE, 345 East 47th St., New York, NY 10017.

C MNEMONICS

<table>
<thead>
<tr>
<th>Messages</th>
<th>Interface States</th>
</tr>
</thead>
<tbody>
<tr>
<td>pon = power on</td>
<td>CIDS = controller idle state</td>
</tr>
<tr>
<td>rsc = request system control</td>
<td>CADS = controller addressed state</td>
</tr>
<tr>
<td>rpp = request parallel poll</td>
<td>CTRS = controller transfer state</td>
</tr>
<tr>
<td>gts = go to standby</td>
<td>CACS = controller active state</td>
</tr>
<tr>
<td>tca = take control asynchronously</td>
<td>CPWS = controller parallel poll wait state</td>
</tr>
<tr>
<td>tcs = take control synchronously</td>
<td>CPPS = controller parallel poll state</td>
</tr>
<tr>
<td>sic = send interface clear</td>
<td>CSBS = controller standby state</td>
</tr>
<tr>
<td>sre = send remote enable</td>
<td>CSHS = controller standby hold state</td>
</tr>
<tr>
<td>IFC = interface clear</td>
<td>CAWS = controller active wait state</td>
</tr>
<tr>
<td>ATN = attention</td>
<td>CSWS = controller synchronous wait state</td>
</tr>
<tr>
<td>TCT = take control</td>
<td>CSRS = controller service requested state</td>
</tr>
<tr>
<td></td>
<td>CSNS = controller service not requested state</td>
</tr>
<tr>
<td></td>
<td>SNAS = system control not active state</td>
</tr>
<tr>
<td></td>
<td>SACS = system control active state</td>
</tr>
<tr>
<td></td>
<td>SRIS = system control remote enable idle state</td>
</tr>
<tr>
<td></td>
<td>SRNS = system control remote enable not active state</td>
</tr>
<tr>
<td></td>
<td>SRAS = system control remote enable active state</td>
</tr>
<tr>
<td></td>
<td>SIIS = system control interface clear idle state</td>
</tr>
<tr>
<td></td>
<td>SINS = system control interface clear not active state</td>
</tr>
<tr>
<td></td>
<td>SIAS = system control interface clear active state</td>
</tr>
<tr>
<td></td>
<td>(ACDS) = accept data state (AH function)</td>
</tr>
<tr>
<td></td>
<td>(ANRS) = acceptor not ready state (AH function)</td>
</tr>
<tr>
<td></td>
<td>(SDYS) = source delay state (SH function)</td>
</tr>
<tr>
<td></td>
<td>(STRS) = source transfer state (SH function)</td>
</tr>
<tr>
<td></td>
<td>(TADS) = talker addressed state (T function)</td>
</tr>
</tbody>
</table>

205250-14
NOTES:
* $T_{10} > 1.5 \mu\text{sec}$
† The microprocessor must wait for the 80 interrupt before writing the GTSB or GSEC commands to ensure that $(\text{STRS} \land \text{SDYS})$ is true.
### Remote Message Coding

<table>
<thead>
<tr>
<th>Mnemonic</th>
<th>Message Name</th>
<th>C</th>
<th>T L D</th>
<th>D</th>
<th>NN</th>
<th>Y A I</th>
<th>I</th>
<th>D R D</th>
<th>A E S I R</th>
<th>P S O</th>
<th>O</th>
<th>A F A</th>
<th>T O R F E</th>
<th>E S</th>
<th>8 7 6 5 4 3 2 1</th>
<th>V D C</th>
<th>N I Q C N</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACG</td>
<td>Addressed Command Group</td>
<td>M</td>
<td>AC Y</td>
<td>0 0 0 X X X X X X X X</td>
<td>1 X X X X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ATN</td>
<td>Attention</td>
<td>U</td>
<td>UC X X X X X X X X X X X X</td>
<td>1 X X X X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DAB</td>
<td>Data Byte</td>
<td>(Notes 1, 9)</td>
<td>M DD D D D D D D D D D D</td>
<td>X X X 0 X X X X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DAC</td>
<td>Data Accepted</td>
<td>U</td>
<td>HS X X X X X X X X X X X X</td>
<td>X X 0 X X X X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DAV</td>
<td>Data Valid</td>
<td>U</td>
<td>HS X X X X X X X X X X X X</td>
<td>1 X X X X X X X X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DCL</td>
<td>Device Clear</td>
<td>M</td>
<td>UC Y 0 0 1 0 1 0 0 X X X X</td>
<td>1 X X X X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>END</td>
<td>End</td>
<td>U</td>
<td>ST X X X X X X X X X X X X</td>
<td>0 1 X X X X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EOS</td>
<td>End of String</td>
<td>(Notes 2, 9)</td>
<td>M DD E E E E E E E E X X X 0 X X X X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GET</td>
<td>Group Execute Trigger</td>
<td>M</td>
<td>AC Y 0 0 0 1 0 0 0 0 X X X X</td>
<td>1 X X X X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GTL</td>
<td>Go to Local</td>
<td>M</td>
<td>AC Y 0 0 0 0 0 0 1 X X X X</td>
<td>1 X X X X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>IDY</td>
<td>Identify</td>
<td>U</td>
<td>UC X X X X X X X X X X X X</td>
<td>X X X 1 X X X X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>IFC</td>
<td>Interface Clear</td>
<td>U</td>
<td>UC X X X X X X X X X X X X</td>
<td>X X X X X X 1 X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LAG</td>
<td>Listen Address Group</td>
<td>M</td>
<td>AD Y 0 1 X X X X X X X X X X</td>
<td>1 X X X X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LLO</td>
<td>Local Lock Out</td>
<td>M</td>
<td>UC Y 0 0 1 0 0 0 1 X X X X</td>
<td>1 X X X X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MLA</td>
<td>My Listen Address</td>
<td>(Note 3)</td>
<td>M AD Y 0 1 L L L L L L</td>
<td>X X X X X X X X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MTA</td>
<td>My Talk Address</td>
<td>(Note 4)</td>
<td>M AD Y 1 0 T T T T T X X X X X X X X X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MSA</td>
<td>My Secondary Address</td>
<td>(Note 5)</td>
<td>M SE Y 1 1 S S S S S S S X X X X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NUL</td>
<td>Null Byte</td>
<td>M</td>
<td>SE Y 1 1 X X X X X X X X X X X X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>OSA</td>
<td>Other Secondary Address</td>
<td>M</td>
<td>SE Y 1 1 X X X X X X X X X X X X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>OTA</td>
<td>Other Talk Address</td>
<td>M</td>
<td>AD Y 1 1 X X X X X X X X X X X X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PCG</td>
<td>Primary Command Group</td>
<td>M</td>
<td>PCG = ACG V UCG V LAG V TAG</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PPC</td>
<td>Parallel Poll Configure</td>
<td>M</td>
<td>AC Y 0 0 0 0 1 0 1 X X X X</td>
<td>1 X X X X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PPE</td>
<td>Parallel Poll Enable</td>
<td>(Note 6)</td>
<td>M SE Y 1 1 0 S P P P X X X X X X X X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PPD</td>
<td>Parallel Poll Disable</td>
<td>(Note 7)</td>
<td>M SE Y 1 1 1 0 D D D D X X X 1 X X X X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PPR1</td>
<td>Parallel Poll Response 1</td>
<td>U</td>
<td>ST X X X X X X X X X X X X</td>
<td>1 X X X X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PPR2</td>
<td>Parallel Poll Response 2</td>
<td>U</td>
<td>ST X X X X X X X X X X X X</td>
<td>1 X X X X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PPR3</td>
<td>Parallel Poll Response 3</td>
<td>U</td>
<td>ST X X X X X X X X X X X X</td>
<td>1 X X X X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PPR4</td>
<td>Parallel Poll Response 4</td>
<td>U</td>
<td>ST X X X X X X X X X X X X</td>
<td>1 X X X X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PPR5</td>
<td>Parallel Poll Response 5</td>
<td>U</td>
<td>ST X X X X X X X X X X X X</td>
<td>1 X X X X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PPR6</td>
<td>Parallel Poll Response 6</td>
<td>U</td>
<td>ST X X X X X X X X X X X X</td>
<td>1 X X X X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PPR7</td>
<td>Parallel Poll Response 7</td>
<td>U</td>
<td>ST X X X X X X X X X X X X</td>
<td>1 X X X X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PPR8</td>
<td>Parallel Poll Response 8</td>
<td>U</td>
<td>ST X X X X X X X X X X X X</td>
<td>1 X X X X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PPU</td>
<td>Parallel Poll Unconfigure</td>
<td>M</td>
<td>UC Y 0 0 1 0 1 0 1 X X X X</td>
<td>1 X X X X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>REN</td>
<td>Remote Enable</td>
<td>U</td>
<td>UC X X X X X X X X X X X X</td>
<td>X X X X X X X X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RFD</td>
<td>Ready for Data</td>
<td>U</td>
<td>HS X X X X X X X X X X X X</td>
<td>X 0 X X X X X X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RQS</td>
<td>Request Service</td>
<td>(Notes 9)</td>
<td>U ST X 1 X X X X X X</td>
<td>X X X 0 X X X X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SCG</td>
<td>Secondary Command Group</td>
<td>M</td>
<td>SE Y 1 1 X X X X X X X X X X X X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SDC</td>
<td>Selected Device Clear</td>
<td>M</td>
<td>AG Y 0 0 0 0 1 0 0 X X X X</td>
<td>1 X X X X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SPD</td>
<td>Serial Poll Disable</td>
<td>M</td>
<td>UC Y 0 0 1 1 0 0 1 X X X X X X X X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
REMOTE MESSAGE CODING (Continued)

Bus Signal Line(s) and Coding That Asserts the True Value of the Message

<table>
<thead>
<tr>
<th>Mnemonic</th>
<th>Message Name</th>
<th>C</th>
<th>T</th>
<th>L</th>
<th>D</th>
<th>N</th>
<th>N</th>
<th>I</th>
<th>D</th>
<th>R</th>
<th>D</th>
<th>A</th>
<th>E</th>
<th>S</th>
<th>I</th>
<th>R</th>
</tr>
</thead>
<tbody>
<tr>
<td>SPE</td>
<td>Serial Poll Enable</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>STB</td>
<td>Status Byte (Notes 8, 9)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TCT</td>
<td>Take Control</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TAG</td>
<td>Talk Address Group</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>UCG</td>
<td>Universal Command Group</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>UNL</td>
<td>Unlisten</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>UNT</td>
<td>Untalk (Note 11)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The 1/0 coding on ATN when sent concurrent with multiline messages has been added to this revision for interpretive convenience.

NOTES:
1. D1–D8 specify the device dependent data bits.
2. E1–E8 specify the device dependent code used to indicate the EOS message.
3. L1–L5 specify the device dependent bits of the device's listen address.
4. T1–T5 specify the device dependent bits of the device's talk address.
5. S1–S5 specify the device dependent bits of the device's secondary address.
6. S specifies the sense of the PPR.
7. D1–D4 specify don't-care bits that shall not be decoded by the receiving device. It is recommended that all zeroes be sent.
8. S1–S6, S8 specify the device dependent status (DI07 is used for the RQS message.)
9. The source of the message on the ATN line is always the C function, whereas the messages on the DIO and EOI lines are enabled by the T function.
10. The source of the messages on the ATN and EOI lines is always the C function, whereas the source of the messages on the DIO lines is always the PP function.
11. This code is provided for system use.
The Intel® 8294A Data Encryption Unit (DEU) is a microprocessor peripheral device designed to encrypt and decrypt 64-bit blocks of data using the algorithm specified in the Federal Information Processing Data Encryption Standard. The DEU operates on 64-bit text words using a 56-bit user-specified key to produce 64-bit cipher words. The operation is reversible: if the cipher word is operated upon, the original text word is produced. The algorithm itself is permanently contained in the 8294A; however, the 56-bit key is user-defined and may be changed at any time.

The 56-bit key and 64-bit message data are transferred to and from the 8294A in 8-bit bytes by way of the system data bus. A DMA interface and three interrupt outputs are available to minimize software overhead associated with data transfer. Also, by using the DMA interface two or more DEUs may be operated in parallel to achieve effective system conversion rates which are virtually any multiple of 400 bytes/second. The 8294A also has a 7-bit TTL compatible output port for user-specified functions.

Because the 8294A implements the NBS encryption algorithm it can be used in a variety of Electronic Funds Transfer applications as well as other electronic banking and data handling applications where data must be encrypted.
<table>
<thead>
<tr>
<th>Symbol</th>
<th>Pin No.</th>
<th>Type</th>
<th>Name and Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>NC</td>
<td>1</td>
<td></td>
<td>NO CONNECTION.</td>
</tr>
<tr>
<td>X1</td>
<td>2</td>
<td>I</td>
<td>CRYSTAL: Inputs for crystal, L-C or external timing signal to determine internal oscillator frequency.</td>
</tr>
<tr>
<td>X2</td>
<td>3</td>
<td>I</td>
<td></td>
</tr>
<tr>
<td>RESET</td>
<td>4</td>
<td>I</td>
<td>RESET: A low signal to this pin resets the 8294A.</td>
</tr>
<tr>
<td>VCC</td>
<td>5</td>
<td>I</td>
<td>POWER: Tied high.</td>
</tr>
<tr>
<td>CS</td>
<td>6</td>
<td>I</td>
<td>CHIP SELECT: A low signal to this pin enables reading and writing to the 8294A.</td>
</tr>
<tr>
<td>GND</td>
<td>7</td>
<td>I</td>
<td>GROUND: This pin must be tied to ground.</td>
</tr>
<tr>
<td>RD</td>
<td>8</td>
<td>I</td>
<td>READ: An active low read strobe at this pin enables the CPU to read data and status from the internal DEU registers.</td>
</tr>
<tr>
<td>A0</td>
<td>9</td>
<td>I</td>
<td>ADDRESS: Address input used by the CPU to select DEU registers during read and write operations.</td>
</tr>
<tr>
<td>WR</td>
<td>10</td>
<td>I</td>
<td>WRITE: An active low write strobe at this pin enables the CPU to send data and commands to the DEU.</td>
</tr>
<tr>
<td>SYNC</td>
<td>11</td>
<td>O</td>
<td>SYNC: High frequency (Clock ÷ 15) output. Can be used as a strobe for external circuitry.</td>
</tr>
<tr>
<td>D0</td>
<td>12</td>
<td>I/O</td>
<td>DATA BUS: Three-state, bi-directional data bus lines used to transfer data between the CPU and the 8294A.</td>
</tr>
<tr>
<td>D1</td>
<td>13</td>
<td>I/O</td>
<td></td>
</tr>
<tr>
<td>D2</td>
<td>14</td>
<td>I/O</td>
<td></td>
</tr>
<tr>
<td>D3</td>
<td>15</td>
<td>I/O</td>
<td></td>
</tr>
<tr>
<td>D4</td>
<td>16</td>
<td>I/O</td>
<td></td>
</tr>
<tr>
<td>D5</td>
<td>17</td>
<td>I/O</td>
<td></td>
</tr>
<tr>
<td>D6</td>
<td>18</td>
<td>I/O</td>
<td></td>
</tr>
<tr>
<td>D7</td>
<td>19</td>
<td>I/O</td>
<td></td>
</tr>
<tr>
<td>GND</td>
<td>20</td>
<td>I/O</td>
<td>GROUND: This pin must be tied to ground.</td>
</tr>
<tr>
<td>VCC</td>
<td>40</td>
<td>I/O</td>
<td>POWER: +5V power input: +5V ± 10%.</td>
</tr>
<tr>
<td>NC</td>
<td>39</td>
<td>I/O</td>
<td>NO CONNECTION.</td>
</tr>
<tr>
<td>DACK</td>
<td>38</td>
<td>I</td>
<td>DMA ACKNOWLEDGE: Input signal from the 8257 DMA Controller acknowledging that the requested DMA cycle has been granted.</td>
</tr>
<tr>
<td>DRQ</td>
<td>37</td>
<td>O</td>
<td>DMA REQUEST: Output signal to the 8257 DMA Controller requesting a DMA cycle.</td>
</tr>
<tr>
<td>SRQ</td>
<td>36</td>
<td>O</td>
<td>SERVICE REQUEST: Interrupt to the CPU indicating that the 8294A is awaiting data or commands at the input buffer. SRQ = 1 implies IBF = 0.</td>
</tr>
<tr>
<td>OAV</td>
<td>35</td>
<td>O</td>
<td>OUTPUT AVAILABLE: Interrupt to the CPU indicating that the 8294A has data or status available in its output buffer, OAV = 1 implies OBF = 1.</td>
</tr>
<tr>
<td>NC</td>
<td>34</td>
<td>O</td>
<td>NO CONNECTION.</td>
</tr>
</tbody>
</table>
Table 1. Pin Description (Continued)

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Pin No.</th>
<th>Type</th>
<th>Name and Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>P6</td>
<td>33</td>
<td>O</td>
<td>OUTPUT PORT: User output port lines. Output lines available to the user via a CPU command which can assert selected port lines. These lines have nothing to do with the encryption function. At power-on, each line is in a 1 state.</td>
</tr>
<tr>
<td>P5</td>
<td>32</td>
<td></td>
<td></td>
</tr>
<tr>
<td>P4</td>
<td>31</td>
<td></td>
<td></td>
</tr>
<tr>
<td>P3</td>
<td>30</td>
<td></td>
<td></td>
</tr>
<tr>
<td>P2</td>
<td>29</td>
<td></td>
<td></td>
</tr>
<tr>
<td>P1</td>
<td>28</td>
<td></td>
<td></td>
</tr>
<tr>
<td>P0</td>
<td>27</td>
<td></td>
<td></td>
</tr>
<tr>
<td>VDD</td>
<td>26</td>
<td></td>
<td>POWER: +5V power input. (+5V ± 10%) Low power standby pin.</td>
</tr>
<tr>
<td>VCC</td>
<td>25</td>
<td></td>
<td>POWER: Tied high.</td>
</tr>
<tr>
<td>CCMP</td>
<td>24</td>
<td>O</td>
<td>CONVERSION COMPLETE: Interrupt to the CPU indicating that the encryption/decryption of an 8-byte block is complete.</td>
</tr>
<tr>
<td>NC</td>
<td>23</td>
<td></td>
<td>NO CONNECTION.</td>
</tr>
<tr>
<td>NC</td>
<td>22</td>
<td></td>
<td>NO CONNECTION.</td>
</tr>
<tr>
<td>NC</td>
<td>21</td>
<td></td>
<td>NO CONNECTION.</td>
</tr>
</tbody>
</table>

FUNCTIONAL DESCRIPTION

OPERATION

The data conversion sequence is as follows:

1) A Set Mode command is given, enabling the desired interrupt outputs.

2) An Enter New Key command is issued, followed by 8 data inputs which are retained by the DEU for encryption/decryption. Each byte must have odd parity.

3) An Encrypt Data or Decrypt Data command sets the DEU in the desired mode.

After this, data conversions are made by writing 8 data bytes and then reading back 8 converted data bytes. Any of the above commands may be issued between data conversions to change the basic operation of the DEU; e.g., a Decrypt Data command could be issued to change the DEU from encrypt mode to decrypt mode without changing either the key or the interrupt outputs enabled.

INTERNAL DEU REGISTERS

Four internal registers are addressable by the master processor: 2 for input, and 2 for output. The following table describes how these registers are accessed.

<table>
<thead>
<tr>
<th>RD</th>
<th>WR</th>
<th>CS</th>
<th>A0</th>
<th>Register</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>Data Input Buffer</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>Data Output Buffer</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>Command Input Buffer</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>Status Output Buffer</td>
</tr>
<tr>
<td>X</td>
<td>X</td>
<td>1</td>
<td>X</td>
<td>Don’t Care</td>
</tr>
</tbody>
</table>

The functions of each of these registers are described below.

Data Input Buffer—Data written to this register is interpreted in one of three ways, depending on the preceding command sequence.

1) Part of a key.
2) Data to be encrypted or decrypted.
3) A DMA block count.

Data Output Buffer—Data read from this register is the output of the encryption/decryption operation.

Command Input Buffer—Commands to the DEU are written into this register. (See command summary below.)

Status Output Buffer—DEU status is available in this register at all times. It is used by the processor for poll-driven command and data transfer operations.

STATUS BIT: 7 6 5 4 3 2 1 0
FUNCTION: X X X KPE CF DEC IBF OBF

3-54
OFB Output Buffer Full; OFB = 1 indicates that output from the encryption/decryption function is available in the Data Output Buffer. It is reset when the data is read.

IBF Input Buffer Full; A write to the Data Input Buffer or to the Command Input Buffer sets IBF = 1. The DEU resets this flag when it has accepted the input byte. Nothing should be written when IBF = 1.

DEC Decrypt; indicates whether the DEU is in an encrypt or a decrypt mode. DEC = 1 implies the decrypt mode. DEC = 0 implies the encrypt mode.

After 8294A has accepted a 'Decrypt Data' or 'Encrypt Data' command, 11 cycles are required to update the DEC bit.

CF Completion Flag; This flag may be used to indicate any or all of three events in the data transfer protocol.
1) It may be used in lieu of a counter in the processor routine to flag the end of an 8-byte transfer.
2) It must be used to indicate the validity of the KPE flag.
3) It may be used in lieu of the CCMP interrupt to indicate the completion of a DMA operation.

KPE Key Parity Error; After a new key has been entered, the DEU uses this flag in conjunction with the CF flag to indicate correct or incorrect parity.

COMMAND SUMMARY

1 — Enter New Key
OP CODE: 0 1 0 0 0 0 0 0
This command is followed by 8 data byte inputs which are retained in the key buffer (RAM) to be used in encrypting and decrypting data. These data bytes must have odd parity represented by the LSB.

2 — Encrypt Data
OP CODE: 0 0 1 1 0 0 0 0
This command puts the 8294A into the encrypt mode.

3 — Decrypt Data
OP CODE: 0 0 1 0 0 0 0 0
This command puts the 8294A into the decrypt mode.

4 — Set Mode
OP CODE: 0 0 0 0 A B C D
where:
A is the OAV (Output Available) interrupt enable
B is the SRQ (Service Request) interrupt enable
C is the DMA (Direct Memory Access) transfer enable
D is the CCMP (Conversion Complete) interrupt enable

This command determines which interrupt outputs will be enabled. A "1" in bits A, B, or D will enable the OAV, SRQ, or CCMP interrupts respectively. A "1" in bit C will allow DMA transfers. When bit C is set the OAV and SRQ interrupts should also be enabled (bits A, B = 1). Following the command in which bit C, the DMA bit, is set, the 8294 will expect one data byte to specify the number of 8-byte blocks to be converted using DMA.

5 — Write to Output Port
OP CODE: 1 P6 P5 P4 P3 P2 P1 P0
This command causes the 7 least significant bits of the command byte to be latched as output data on the 8294 output port. The initial output is 1111111. Use of this port is independent of the encryption/decryption function.

PROCESSOR/DEU INTERFACE PROTOCOL

ENTERING A NEW KEY

The timing sequence for entering a new key is shown in Figure 3. A flowchart showing the CPU software to accommodate this sequence is given in Figure 4.

After the Enter New Key command is issued, 8 data bytes representing the new key are written to the data input buffer (most significant byte first). After the eighth byte is entered into the DEU, CF goes true (CF = 1). The CF bit goes false again when KPE is valid. The CPU can then check the KPE flag. If KPE = 1, a parity error has been detected and the DEU has not accepted the key. Each byte is checked for odd parity, where the parity bit is the LSB of each byte.
Since \( CF = 1 \) only for a short period of time after the last byte is accepted, the CPU which polls the CF flag might miss detecting \( CF = 1 \) momentarily. Thus, a counter should be used, as in Figure 4, to flag the end of the new key entry. Then CF is used to indicate a valid KPE flag.

Figure 5 shows the timing sequence for encrypting or decrypting data. The CPU writes 8 data bytes to the DEU's data input buffer for encryption/decryption. CF then goes true (\( CF = 1 \)) to indicate that the DEU has accepted the 8-byte block. Thus, the CPU may test for \( IBF = 0 \) and \( CF = 1 \) to terminate the input mode, or it may use a software counter. When the encryption/decryption is complete, the CCMP and OAV interrupts are asserted and the OBF flag is set true (\( OBF = 1 \)). OAV and OBF are set false again after each of the converted data bytes is read back by the CPU. The CCMP interrupt is set false, and remains false, after the first read. After 8 bytes have been read back by the CPU, CF goes false (\( CF = 0 \)). Thus, the CPU may test for \( CF = 0 \) to terminate the read mode. Also, the CCMP interrupt may be used to initiate a service routine which performs the next series of 8 data reads and 8 data writes.

Figure 6 offers two flowcharts outlining the alternative means of implementing the data conversion protocol. Either the CF flag or a software counter may be used to end the read and write modes.

SRQ = 1 implies \( IBF = 0 \), OAV = 1 implies \( OBF = 1 \). This allows interrupt routines to do data transfers without checking status first. However, the OAV service routine must detect and flag the end of a data conversion.
USING SOFTWARE COUNTER

START

I = 0

NO IBF = 0?

YES

DATA REGISTER ← 1 DATA BYTE

I = I + 1

NO IBF = 0?

YES

READ 1 CODED DATA BYTE

I = I - 1

NO IBF = 0?

YES

END

210465-6

USING CF FLAG

START

NO IBF = 0?

YES

DATA REGISTER ← 1 DATA BYTE

NO CF = 0?

YES

CF = 0?

NO

YES

READ 1 CODED DATA BYTE

210465-7

Figure 6. Data Conversion Flowcharts

USING DMA

The timing sequence for data conversions using DMA is shown in Figure 7. This sequence can be better understood when considered in conjunction with the hardware DMA interface in Figure 8. Note

210465-8

Figure 7. DMA Sequence

Figure 8. DMA Interface
that the use of the DMA feature requires 3 external AND gates and 2 DMA channels (one for input, one for output). Since the DEU has only one DMA request pin, the SRQ and OAV outputs are used in conjunction with two of the AND gates to create separate DMA request outputs for the 2 DMA channels. The third AND gate combines the two active-low DACK inputs.

To initiate a DMA transfer, the CPU must first initialize the two DMA channels as shown in the flowchart in Figure 9. It must then issue a Set Mode command to the DEU enabling the OAV, SRQ, and DMA outputs. The CCMP interrupt may be enabled or disabled, depending on whether that output is desired. Following the Set Mode command, there must be a data byte giving the number of 8-byte blocks of data (n < 256) to be converted. The DEU then generates the required number of DMA requests to the 2 DMA channels with no further CPU intervention. When the requested number of blocks has been converted, the DEU will set CF and assert the CCMP interrupt (if enabled). CCMP then goes false again with the next write to the DEU (command or data). Upon completion of the conversion, the DMA mode is disabled and the DEU returns to the encrypt/decrypt mode. The enabled interrupt outputs, however, will remain enabled until another Set Mode command is issued.

SINGLE BYTE COMMANDS

Figure 10 shows the timing and protocol for single byte commands. Note that any of the commands is effective as a pacify command in that they may be entered at any time, expect during a DMA conversion. The DEU is thus set to a known state. However, if a command is issued out of sequence, an additional protocol is required (Figure 11). The CPU must wait until the command is accepted (IBF = 0). A data read must then be issued to clear anything the preceding command sequence may have left in the Data Output Buffer.

CPU/DEU INTERFACES

Figures 12 through 15 illustrate four interface configurations used in the CPU/DEU data transfers. In all cases SRQ will be true (if enabled) and IBF will be false when the DEU is ready to accept data or commands.
OSCILLATING AND TIMING CIRCUITS

The 8294A's internal timing generation is controlled by a self-contained oscillator and timing circuit. A choice of crystal, L-C or external clock can be used to derive the basic oscillator frequency.

The resident timing circuit consists of an oscillator, a state counter and a cycle counter as illustrated in Figure 16.
**Figure 16. Oscillator Configuration**

**Figure 17. Recommended Crystal**

**Figure 18. Recommended Connection for External Clock Signal**
ABSOLUTE MAXIMUM RATINGS*

Ambient Temperature Under Bias ....... 0°C to +70°C
Storage Temperature ................... -65°C to +150°C
Voltage on Any Pin With
  Respect to Ground ................. -0.5V to +7V
Power Dissipation ..................... 1.5 Watt

*Notice: Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions above those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

D.C. AND OPERATING CHARACTERISTICS

$T_A = 0°C$ to $+70°C$, $V_{CC} = +5V \pm 10\%$, $V_{SS} = 0V$

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Parameter</th>
<th>Limits</th>
<th>Unit</th>
<th>Test Conditions</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Min</td>
<td>Typ</td>
<td>Max</td>
</tr>
<tr>
<td>$V_{IL}$</td>
<td>Input Low Voltage (All Except $X_1$, $X_2$, RESET)</td>
<td>-0.5</td>
<td>0.8</td>
<td>V</td>
</tr>
<tr>
<td>$V_{IL1}$</td>
<td>Input Low Voltage ($X_1$, $X_2$, RESET)</td>
<td>-0.5</td>
<td>0.6</td>
<td>V</td>
</tr>
<tr>
<td>$V_{IH}$</td>
<td>Input High Voltage (All Except $X_1$, RESET)</td>
<td>2.0</td>
<td>$V_{CC}$</td>
<td>V</td>
</tr>
<tr>
<td>$V_{IH1}$</td>
<td>Input High Voltage ($X_1$, RESET)</td>
<td>3.5</td>
<td>$V_{CC}$</td>
<td>V</td>
</tr>
<tr>
<td>$V_{IH2}$</td>
<td>Input High Voltage ($X_2$)</td>
<td>2.2</td>
<td>$V_{CC}$</td>
<td>V</td>
</tr>
<tr>
<td>$V_{OL}$</td>
<td>Output Low Voltage ($D_0$–$D_7$)</td>
<td>0.45</td>
<td>V</td>
<td>$I_{OL} = 2.0 \text{ mA}$</td>
</tr>
<tr>
<td>$V_{OL1}$</td>
<td>Output Low Voltage (All Other Outputs)</td>
<td>0.45</td>
<td>V</td>
<td>$I_{OL} = 1.6 \text{ mA}$</td>
</tr>
<tr>
<td>$V_{OH}$</td>
<td>Output High Voltage ($D_0$–$D_7$)</td>
<td>2.4</td>
<td>V</td>
<td>$I_{OH} = -400 \text{ \mu A}$</td>
</tr>
<tr>
<td>$V_{OH1}$</td>
<td>Output High Voltage (All Other Outputs)</td>
<td>2.4</td>
<td>V</td>
<td>$I_{OH} = -50 \text{ \mu A}$</td>
</tr>
<tr>
<td>$I_{IL}$</td>
<td>Input Leakage Current (RD, WR, CS, $A_0$)</td>
<td>$\pm 10\ \mu A$</td>
<td>$V_{SS} \leq V_{IN} \leq V_{CC}$</td>
<td></td>
</tr>
<tr>
<td>$I_{OFL}$</td>
<td>Output Leakage Current ($D_0$–$D_7$, High Z State)</td>
<td>$\pm 10\ \mu A$</td>
<td>$V_{SS} + 0.45 \leq V_{OUT} \leq V_{CC}$</td>
<td></td>
</tr>
<tr>
<td>$I_{DD}$</td>
<td>$V_{DD}$ Supply Current</td>
<td>5</td>
<td>20</td>
<td>mA</td>
</tr>
<tr>
<td>$I_{DD} + I_{CC}$</td>
<td>Total Supply Current</td>
<td>60</td>
<td>135</td>
<td>mA</td>
</tr>
<tr>
<td>$I_L$</td>
<td>Low Input Load Current (Pins 24, 27–38)</td>
<td>0.3</td>
<td>mA</td>
<td>$V_{IL} = 0.8V$</td>
</tr>
<tr>
<td>$I_{LH}$</td>
<td>Low Input Load Current (RESET)</td>
<td>0.2</td>
<td>mA</td>
<td>$V_{IL} = 0.8V$</td>
</tr>
<tr>
<td>$I_{IH}$</td>
<td>Input High Leakage Current (Pins 24, 27–38)</td>
<td>100</td>
<td>$\mu A$</td>
<td>$V_{IN} = V_{CC}$</td>
</tr>
<tr>
<td>$C_{IN}$</td>
<td>Input Capacitance</td>
<td>10</td>
<td>pF</td>
<td></td>
</tr>
<tr>
<td>$C_{I/O}$</td>
<td>I/O Capacitance</td>
<td>20</td>
<td>pF</td>
<td></td>
</tr>
</tbody>
</table>
A.C. CHARACTERISTICS $T_A = 0^\circ C$ to $+70^\circ C$, $V_{CC} = V_{DD} = +5V \pm 10\%$, $V_{SS} = 0V$

### DBB READ

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Parameter</th>
<th>Min</th>
<th>Max</th>
<th>Unit</th>
<th>Test Conditions</th>
</tr>
</thead>
<tbody>
<tr>
<td>$t_{AR}$</td>
<td>$CS, A_0$ Setup to $RD \downarrow$</td>
<td>0</td>
<td></td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>$t_{RA}$</td>
<td>$CS, A_0$ Hold After $RD \uparrow$</td>
<td>0</td>
<td></td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>$t_{RR}$</td>
<td>$RD$ Pulse Width</td>
<td>160</td>
<td></td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>$t_{AD}$</td>
<td>$CS, A_0$ to Data Out Delay</td>
<td>130</td>
<td></td>
<td>ns</td>
<td>$C_L = 100 \text{ pF}$</td>
</tr>
<tr>
<td>$t_{RD}$</td>
<td>$RD \downarrow$ to Data Out Delay</td>
<td>130</td>
<td></td>
<td>ns</td>
<td>$C_L = 100 \text{ pF}$</td>
</tr>
<tr>
<td>$t_{DF}$</td>
<td>$RD \uparrow$ to Data Float Delay</td>
<td>85</td>
<td></td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>$t_{CY}$</td>
<td>Cycle Time</td>
<td>1.25</td>
<td>15</td>
<td>$\mu$s</td>
<td>1–12 MHz Crystal</td>
</tr>
</tbody>
</table>

### DBB WRITE

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Parameter</th>
<th>Min</th>
<th>Max</th>
<th>Unit</th>
<th>Test Conditions</th>
</tr>
</thead>
<tbody>
<tr>
<td>$t_{AW}$</td>
<td>$CS, A_0$ Setup to $WR \downarrow$</td>
<td>0</td>
<td></td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>$t_{WA}$</td>
<td>$CS, A_0$ Hold After $WR \uparrow$</td>
<td>0</td>
<td></td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>$t_{WW}$</td>
<td>$WR$ Pulse Width</td>
<td>160</td>
<td></td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>$t_{DW}$</td>
<td>Data Setup to $WR \uparrow$</td>
<td>130</td>
<td></td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>$t_{WD}$</td>
<td>Data Hold to $WR \uparrow$</td>
<td>0</td>
<td></td>
<td>ns</td>
<td></td>
</tr>
</tbody>
</table>

### DMA AND INTERRUPT TIMING

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Parameter</th>
<th>Min</th>
<th>Max</th>
<th>Unit</th>
<th>Test Conditions</th>
</tr>
</thead>
<tbody>
<tr>
<td>$t_{ACC}$</td>
<td>DACK Setup to Control</td>
<td>0</td>
<td></td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>$t_{CAC}$</td>
<td>DACK Hold After Control</td>
<td>0</td>
<td></td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>$t_{ACD}$</td>
<td>DACK to Data Valid</td>
<td>130</td>
<td></td>
<td>ns</td>
<td>$C_L = 100 \text{ pF}$</td>
</tr>
<tr>
<td>$t_{CRQ}$</td>
<td>Control L.E. to DRQ T.E.</td>
<td>110</td>
<td></td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>$t_{CI}$</td>
<td>Control T.E. to Interrupt T.E.</td>
<td>400</td>
<td></td>
<td>ns</td>
<td></td>
</tr>
</tbody>
</table>

### CLOCK

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Parameter</th>
<th>Min</th>
<th>Max</th>
<th>Units</th>
</tr>
</thead>
<tbody>
<tr>
<td>$t_{CY}$</td>
<td>Cycle Time</td>
<td>1.25</td>
<td>9.20</td>
<td>$\mu$s$^{(1)}$</td>
</tr>
<tr>
<td>$t_{CYC}$</td>
<td>Clock Period</td>
<td>83.3</td>
<td>613</td>
<td>ns</td>
</tr>
<tr>
<td>$t_{PWH}$</td>
<td>Clock High Time</td>
<td>38</td>
<td></td>
<td>ns</td>
</tr>
<tr>
<td>$t_{PWL}$</td>
<td>Clock Low Time</td>
<td>38</td>
<td></td>
<td>ns</td>
</tr>
<tr>
<td>$t_{R}$</td>
<td>Clock Rise Time</td>
<td>10</td>
<td></td>
<td>ns</td>
</tr>
<tr>
<td>$t_{F}$</td>
<td>Clock Fall Time</td>
<td>10</td>
<td></td>
<td>ns</td>
</tr>
</tbody>
</table>

NOTE:
1. $t_{CY} = 15/\text{f(XTAL)}$
A.C. TESTING INPUT, OUTPUT WAVEFORM

\[ \text{2.4} \]
\[ \text{2.0} \]
\[ \text{TEST POINTS} \]
\[ \text{2.0} \]
\[ \text{0.8} \]
\[ \text{C_L = 150 \text{ pF}} \]

WAVEFORMS

READ OPERATION—OUTPUT BUFFER REGISTER

WRITE OPERATION—INPUT BUFFER REGISTER

210465-21

210465-22

210465-23

3-63
DMA AND INTERRUPT TIMING

CLOCK TIMING
Using the 8291A GPIB Talker/Listener
INTRODUCTION

This application note explains the Intel 8291A GPIB (General Purpose Interface Bus) Talker/Listener as a component, and shows its use in GPIB interface design tasks.

The first section of this note presents an overview of IEEE 488 (GPIB). The second section introduces the Intel GPIB component family. A detailed explanation of the 8291A follows. Finally, some application examples using the component family are presented.

Figure 1. Interface Capabilities and Bus Structure
OVERVIEW OF IEEE 488/GPIB

The GPIB is a parallel interface bus with an asynchronous interlocking data exchange handshake mechanism. It is designed to provide a common communication interface among devices over a maximum distance of 20 meters at a maximum speed of 1 Mbps. Up to 15 devices may be connected together. The asynchronous interlocking handshake dispenses with a common synchronization clock, and allows intercommunication among devices capable of running at different speeds. During any transaction, the data transfer occurs at the speed of the slowest device involved.

The GPIB finds use in a diversity of applications requiring communication among digital devices over short distances. Common examples are: programmable instrumentation systems, computer to peripherals, etc.

The interface is completely defined in the IEEE STD.-488-1978.

A typical implementation consists of logical devices which talk (talker), listen (listeners), and control GPIB activity (controllers).

Interface Functions

The interface between any device and the bus may have a combination of several different capabilities (called 'functions'). Among a total of ten functions defined, the Talker, Listener, Source Handshake, Acceptor Handshake and Controller are the more common examples. The Talker function allows a device to transmit data. The Listener function allows reception. The Source and Acceptor Handshakes, synchronized with the Talker and Listener functions respectively, exchange the handshake signals that coordinate data transfer. The Controller function allows a device to activate the interface functions of the various devices through commands. Other interface functions are: Service request, Remote local, Parallel poll, Device clear and Device trigger. Each interface may not contain all these functions. Further, most of these functions may be implemented to various levels (called 'subsets') of capability. Thus, the overall capability of an interface may be tailored to the needs of the communicating device.

Electrical Signal Lines

As shown in Figure 1, the GPIB is composed of eight data lines (D08–D01), five interface management lines (IFC, ATN, SRQ, REN, EOI), and three transfer control lines (DAV, NRFD, NDAC).

The eight data lines are used to transfer data and commands from one device to another with the help of the management and control lines. Each of the five interface management lines has a specific function.

ATN (attention) is used by the Controller to indicate that it (the controller) has access to the GPIB and that its output on the data lines is to be interpreted as a command. ATN is also used by the controller along with EOI to indicate a parallel poll.

SRQ (service request) is used by a device to request service from the controller.

REN (remote enable) is used by the controller to specify the command source of a device. A device can be issued commands either locally through its front panel or by the controller.

EOI (end or identify) may be used by the controller as well as talker. A controller uses EOI along with ATN to demand a parallel poll. Used by a talker, EOI indicates the last byte of a data block.

IFC (interface clear) forces a complete GPIB interface to the idle state. This could be considered the GPIB's "interface reset." GPIB architecture allows for more than one controller to be connected to the bus simultaneously. Only one of these controllers may be in command at any given time. This device is known as the controller-in-charge. Control can be passed from one controller to another. Only one among all the controllers present on a bus can be the system controller. The system controller is the only device allowed to drive IFC.
Flow diagram outlines sequence of events during transfer of data byte. More than one listener at a time can accept data because of logical connection of NRFD and NDAC lines.

Figure 2. Handshake Flowchart
Transfer Control Lines

The transfer control lines conduct the asynchronous interlocking three-wire handshake.

DAV (data valid) is driven by a talker and indicates that valid data is on the bus.

NRFD (not ready for data) is driven by the listeners and indicates that not all listeners are ready for more data.

NDAC (not data accepted) is used by the listeners to indicate that not all listeners have read the GPIB data lines yet.

The asynchronous 3-wire handshake flowchart is shown in Figure 2. This is a concept fundamental to the asynchronous nature of the GPIB and is reviewed in the following paragraphs.

Assume that a talker is ready to start a data transfer. At the beginning of the handshake, NRFD is false indicating that the listener(s) is ready for data. NDAC is true indicating that the listener(s) has not accepted the data, since no data has been sent yet. The talker places data on the data lines, waits for the required settling time, and then indicates valid data by driving DAV true. All active listeners drive NRFD true indicating that they are not ready for more data. They then read the data and drive NDAC false to indicate acceptance. The talker responds by deasserting DAV and readies itself to transfer the next byte. The listeners respond to DAV false by driving NDAC true. The talker can now drive the data lines with a new data byte and wait for NRFD to be false to start the next handshake cycle.

Bus Commands

When ATN and DAV are true data patterns which have been placed by the controller on the GPIB, they are interpreted as commands by the other devices on the interface. The GPIB standard contains a repertory of commands such as MTA (My Talk Address), MSA (My Secondary Address), SPE (Serial Poll Enable), etc. All other patterns in conjunction with ATN and DAV are classified as undefined commands and their meaning is user-dependent.

Addressing Techniques

To allow the controller to issue commands selectively to specific devices, three types of addressing exist on the GPIB: talk only/listen only (ton/lon), primary, and secondary.

Ton/lon is a method where the ability of the GPIB interface to talk or listen is determined by the device and not by the GPIB controller. With this method, fixed poles can be easily designated in simple systems where reassignment is not necessary. This is appropriate and convenient for certain applications. For example, a logic analyzer might be interfaced via the GPIB to a line printer in order to document some type of failure. In this case, the line printer simply listens to the logic analyzer, which is a talker.

The controller addresses devices through three commands, MTA (my talk address), MLA (my listen address), and MSA (my secondary address). The device address is embedded in the command bit pattern. The device whose address matches the embedded pattern is enabled. Some devices may have the same logical talk and listen addresses. This is allowable since the talker and listener are separate functions. However, two of the same functions cannot have the same address.

In primary addressing, a device is enabled to talk (listen) by receiving the MTA (MLA) message.

Secondary addressing extends the address field from 5 to 10 bits by allowing an additional byte. This additional byte is passed via the MSA message. Secondary addressing can also be used to logically divide devices into various subgroups. The MSA message applies only to the device(s) whose primary address immediately precede it.

INTEL'S® GPIB COMPONENTS

The logic designer implementing a GPIB interface has, in the past, been faced with a difficult and complex discrete logic design. Advances in LSI technology have produced sophisticated microprocessor and peripheral devices which combine to reduce this once complex interface task to a system consisting of a small set of integrated circuits and some software drivers. A microprocessor hardware/software solution and a high-level language source code provide an additional benefit in end-product maintenance. Product changes are a simple matter of revising the product software. Field changes are as easy as exchanging EPROMs.

Intel has provided an LSI solution to GPIB interfacing with a talker/listener device (8291A), a controller device (8292), and a transceiver (8293). An interface with all capabilities except for the controller function can be built with an 8291A and a pair of 8293's. The addition of the 8292 produces a complex interface. Since most devices in a GPIB system will not have the controller function capability, this modular approach provides the least cost to the majority of interface designs.
Overview of the 8291A GPIB Talker/Listener

The Intel 8291A GPIB Talker/Listener operates over a clock range of 1 to 8 MHz and is compatible with the MCS-85, iAPX-86, and 8051 families of microprocessors.

A detailed description of the 8291A is given in the data sheet.

The 8291A implements the following functions: Source Handshake (SH), Acceptor Handshake (AH), Talker Extended (TE), Service Request (SRQ), Listener Extended (LE), Remote/Local (RL), Parallel Poll (PP2), Device Clear (DC), and Device Trigger (DT).

Current states of the 8291A can be determined by examining the device's status read registers. In addition, the 8291A contains 8 write registers. These registers are shown in Figure 3. The three register select pins RS3–RS0 are used to select the desired register.

The data-in register moves data from the GPIB to the microprocessor or to memory when the 8291A is addressed to listen. When the 8291A is addressed to talk, it uses the data-out register to move data onto the GPIB. The serial poll mode and status registers are used to request service and program the serial poll status byte.

A detailed description of each of the registers, along with state diagrams can be found in the 8291A data sheet.

<table>
<thead>
<tr>
<th>Read Registers</th>
<th>Register Select Code</th>
<th>Write Registers</th>
</tr>
</thead>
<tbody>
<tr>
<td>D7  D6  D5  D4  D3  D2  D1  D0</td>
<td>0 0 0 0</td>
<td>DO7  DO6  DO5  DO4  DO3  DO2  DO1  DO0</td>
</tr>
<tr>
<td>CPT  APT  GET  END  DEC  ERR  BO  BI</td>
<td>0 0 1</td>
<td>CPT  APT  GET  END  DEC  ERR  BO  BI</td>
</tr>
<tr>
<td>Interrupt Status 1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>INT  SPAS  LLO  REM  SPC  LLOC  REMC  ADSC</td>
<td>0 1 0</td>
<td>0 0  DMA  DM  SPC  LLOC  REMC  ADSC</td>
</tr>
<tr>
<td>Interrupt Status 2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>S8  S9S  S5  S4  S3  S2  S1</td>
<td>0 1 1</td>
<td>S8  S9S  S5  S4  S3  S2  S1</td>
</tr>
<tr>
<td>Serial Poll Status 2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>tvn  Lon  EOI  LPS  TPS  LA  TA  MJMN</td>
<td>1 0 0</td>
<td>TO  LO  0 0  0 0  AD1  AD0</td>
</tr>
<tr>
<td>Address Status</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CPT7  CPT6  CPT5  CPT4  CPT3  CPT2  CPT1  CPT0</td>
<td>1 0 1</td>
<td>CNT2  CNT1  CNT0  COM4  COM3  COM2  COM1  COM0</td>
</tr>
<tr>
<td>Command Pass Through</td>
<td></td>
<td></td>
</tr>
<tr>
<td>INT  DTO  DLO  AD5–0  AD4–0  AD3–0  AD2–0  AD1–0</td>
<td>1 1 0</td>
<td>ARS  DT  DL  AD5  AD4  AD3  AD2  AD1</td>
</tr>
<tr>
<td>Address 0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>X  DT1  DL1  AD5–1  AD4–1  AD3–1  AD2–1  AD1–1</td>
<td>1 1 1</td>
<td>EC7  EC6  EC5  EC4  EC3  EC2  EC1  EC0</td>
</tr>
<tr>
<td>Address 1</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure 3. 8291A Registers
Address Mode

The address mode and status registers are used to program the addressing modes and track addressing states. The auxiliary mode register is used to select a variety of functions. The command pass through register is used for undefined commands and extended addresses. The address 0/1 register is used to program the addresses to which the 8291A will respond. The address 0 and address 1 registers allow reading of these programmed addresses plus trading of the interrupt bit. The EOS register is used to program the end of sequence character.

Detailed descriptions of the addressing modes available with the 8291A are described in the 8291A data sheet. Examples of how to program these modes are shown below.

1. MODE: Talker has single address of 01H
   Listener has single address of 02H

<table>
<thead>
<tr>
<th>CPU Writes to:</th>
<th>Pattern</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Address Mode Register</td>
<td>0000 0001</td>
<td>Select Mode 1 Addressing</td>
</tr>
<tr>
<td>Address 0/1 Register</td>
<td>0010 0001</td>
<td>Major is Talking. Address = 01H</td>
</tr>
<tr>
<td>Address 0/1 Register</td>
<td>1100 0010</td>
<td>Minor is Listener. Address = 02H</td>
</tr>
</tbody>
</table>

2. MODE: Talker has single address of 01H
   Listener has single address of 02H

<table>
<thead>
<tr>
<th>CPU Writes to:</th>
<th>Pattern</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Address Mode Register</td>
<td>0000 0001</td>
<td>Select Mode 1 Addressing</td>
</tr>
<tr>
<td>Address 0/1 Register</td>
<td>0100 0010</td>
<td>Major is Listener. Address = 02H</td>
</tr>
<tr>
<td>Address 0/1 Register</td>
<td>1010 0001</td>
<td>Minor is Talking. Address = 01H</td>
</tr>
</tbody>
</table>

Note that in both of the above examples, the listener will respond to a MLA message with five least significant bits equal to 02H and the talker to a 01H.

3. MODE: Talker and listener both share a single address of 03H

<table>
<thead>
<tr>
<th>CPU Writes to:</th>
<th>Pattern</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Address Mode Register</td>
<td>0000 0001</td>
<td>Select Mode 1 Addressing</td>
</tr>
<tr>
<td>Address 0/1 Register</td>
<td>0000 0011</td>
<td>Talker and Listener Address = 03</td>
</tr>
<tr>
<td>Address 0/1 Register</td>
<td>1110 0000</td>
<td>Minor Address is disabled</td>
</tr>
</tbody>
</table>

4. MODE: Talker and listener have a primary address of 04H and a secondary address of 05H

<table>
<thead>
<tr>
<th>CPU Writes to:</th>
<th>Pattern</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Address Mode Register</td>
<td>0000 0010</td>
<td>Select Mode 2 Addressing</td>
</tr>
<tr>
<td>Address 0/1 Register</td>
<td>0000 0100</td>
<td>Primary Address = 04H</td>
</tr>
<tr>
<td>Address 0/1 Register</td>
<td>1000 0101</td>
<td>Minor Address is disabled</td>
</tr>
</tbody>
</table>

5. MODE: Talker has a primary address of 06H. Listener has a primary address of 07H

<table>
<thead>
<tr>
<th>CPU Writes to:</th>
<th>Pattern</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Address Mode Register</td>
<td>0000 0011</td>
<td>Select Mode 3</td>
</tr>
<tr>
<td>Address 0/1 Register</td>
<td>0010 0110</td>
<td>Talker Address = 06</td>
</tr>
<tr>
<td>Address 0/1 Register</td>
<td>1100 0111</td>
<td>Listener Primary = 07</td>
</tr>
</tbody>
</table>

The CPU will verify the secondary addresses which could be the same or different.
APPLICATION OF THE 8291A

This phase of the application note will examine programming of the 8291A, corresponding bus commands and responses, CPU interruption, etc. for a variety of GPIB activities. This should provide the reader with a clear understanding of the role of the 8291A performs in a GPIB system. The talker function, listener function, remote message handling, and remote/local operations including local lockout, are discussed.

Talker Functions

TALK-ONLY (ton). In talk only mode the 8291A will not respond to the MTA message from a controller. Generally, ton is used in an environment which does not have a controller. Ton is also employed in an interface that includes the controller function.

When the 8291A is used with the 8292, the sequence of events for initialization are as follows:

1) The Interrupt/Enable registers are programmed.
2) Ton is selected.
3) Settling time is selected.
4) EOS character is loaded.
5) "Pon" local message is sent.
6) CPU waits for Byte Out (BO) and sends a byte to the data out register.

Addressed Talker (via MTA Message)

The GPIB controller will direct the 8291A to talk by sending a My Talk Address (MTA) message containing the 8291A's talk address. The sequence of events is as follows:

1) The interrupt enable and serial poll mode registers are programmed.
2) Mode 1 is selected.
3) Settling time is selected.
4) Talker and listener addresses are programmed.
5) Power on (pon) local message is sent.
6) CPU waits for an interrupt. When the controller has sent the MTA message for the 8291A, the ADSC bit will be set.
7) CPU reads the Address Status register to determine if the 8291A has been addressed to talk (TA = 1).
8) CPU waits for an interrupt from either BO or ADSC
9) When BO is set, the CPU writes the data byte to the data out register.
10) CPU continues to poll the status registers.
11) When unaddressed ADSC, will be set and TA reset.

LISTENER FUNCTIONS

LISTEN-ONLY (lon). In listen-only mode the 8291 will not respond to the My Listen Address (MLA) message from the controller. The sequence of events is as follows:

1) The Interrupt Enable registers are programmed.
2) Lon is selected.
3) EOS character is programmed.
4) "Pon" local message is sent.
5) CPU waits for BI and reads the byte from the data-in register.

Note that enabling both ton and lon can create an internal loopback as long as another listener exists.

Addressed Listening (via the MLA Message)

The GPIB controller will direct the 8291A to listen by sending a MLA message containing the 8291A's listen address. The sequence of events is as follows:

1) The interrupt enable and serial poll mode registers are programmed.
2) The serial poll mode register is loaded as desired.
3) Talker and listener addresses are loaded.
4) "Pon" local message is sent.
5) The CPU waits for an interrupt. When the controller has sent the MLA message for the 8291A, the ADSC bit will be set.
6) The CPU reads the Address Status Register to determine if the 8291A has been addressed to listen (LA = 1).
7) CPU waits for an interrupt for BI or ADSC.
8) When BI is set, the CPU reads the data byte from the data-in register.
9) The CPU continues to poll the status registers.
10) When unaddressed, ADSC will be set and LA reset.

Remote/Local and Lockout

Remote and local refer to the source of control of a device connected to the GPIB. Remote refers to control from the GPIB controller-in-charge. Local refers to control from the device's own system. Reference should be made to the RL state diagram in the 8291A data sheet.

Upon "pon" the 8291A is in the local state. In this state the REM bit in Interrupt Status 1 Register is reset. When the GPIB controller takes control of the bus it will drive the REM (remote enable) line true. This will cause the REM bit and REMC (remote/local change) bit to be set. The distinction between remote and local modes is necessary in that some types of devices will have local controls which have functions which are also controlled by remote messages.
In the local state the device is allowed to store, but not respond to, remote messages which control functions which are also controlled by local messages. A device which has been addressed to listen will exit the local state and go to the remote state if the REN message is true and the local rtl (return to local) message is false. The state of the "rtl" local message is ignored and the device is "locked" into the local state if the LLO remote message is true. In the Remote state the device is not allowed to respond to local messages which control function that are also controlled by remote messages. A device will exit the remote state and enter the local state when REN goes false. It will also enter the local state if the GTL (go to local) remote message is true and the device has been addressed to listen. It will also enter the local state if the rtl message is true and the LLO message is false or ACDS is inactive.

A device will exit the remote state and enter RWLS (remote with lockout state) if the LLO (local lockout) message is true and ACDS is active. In this mode, those local messages which control functions which are also controlled by remote messages are ignored. In other words, the "rtl" message is ignored. A device will exit RWLS and go to the local state if REN goes false. The device will exit RWLS and go to LWLS if the GTL message is true and the device is addressed to listen.

### Polling

The IEEE-488 standard specifies two methods for a slave device to let the controller know that it needs service.

These two methods are called Serial and Parallel Poll. The controller performs one of these two polling methods after a slave device requests service. As implied in the name, a Serial Poll is when the controller sequentially asks each device if it requested service. In a Parallel Poll the controller asks all of the devices on the GPIB if they requested service, and they reply in parallel.

#### Serial Poll

When the controller performs a Serial Poll, each slave device sends back to the controller a Serial Poll Status Byte. One of the bits in the Serial Poll Status Byte indicates whether this device requested service or not. The remaining 7 bits are used defined, and they are used to indicate what type of service is required. The IEEE-488 spec only defines the service request bit, however HP has defined a few more bits in the Serial Poll Status Byte. This can be seen in Figure 4.

When a slave device needs service it drives the SRQ line on the GPIB bus true (low). For the 8291A this is done by setting bit 7 in the Serial Poll Status Byte. The CPU in the controller may be interrupted by SRQ or it may poll a register to determine the state of SRQ. Using the 8292 one could either poll the interrupt status register for the SRQ interrupt status bit, or enables SRQ to interrupt the CPU. After the controller recognizes a service request, it goes into the serial poll routine.

The first thing the controller does in the serial poll routine is assert ATN. When ATN is asserted true the controller takes control of the GPIB, and all slave de-

---

**Figure 4. The Serial Poll Status Byte**
vices on the bus must listen. All bytes sent over the bus while ATN is true are commands. After the controller takes control, it sends out a Universal Unlisten (UNL), which tells all previously addressed listeners to stop listening. The controller then sends out a byte called SPE (Serial Poll Enable). This command notifies all of the slaves on the bus that the controller has put the GPIB in the Serial Poll Mode State (SPMS). Now the controller addresses the first slave device to TALK and puts itself in the listen mode. When the controller resets ATN the device addressed to talk transmits to the controller its Serial Poll Status Byte. If the device just polled was the one requesting service, the SRQ line on the GPIB goes false, and bit 7 in the serial poll status byte of the 8291A is reset. If more than one device is requesting service, SRQ remains low until all of the devices requesting service have been polled, since SRQ is wire-ored. To continue the Serial Poll, the controller asserts ATN, addresses the next device to talk then releases ATN. When the controller finishes polling it asserts ATN, sends the universal untalk command (UNT), then sends the Serial Poll Disable command (SPD). The flow of the serial poll can be seen from the example in Figure 5.

| 0) DEVICE A REQUESTS SERVICE (SRQ) |
| 1) ASSERT ATN |
| 2) UNIVERSAL UNLISTEN (UNL) |
| 3) SERIAL POLL ENABLE (SPE) |
| 4) DEVICE A TALK ADDRESS (MTA) |
| 5) RELEASE ATN |
| 6) DEVICE A STATUS BYTE (STD) (RQS SET) |
| 7) ASSERT ATN |
| 8) DEVICE B TALK ADDRESS (MTA) |
| 9) RELEASE ATN |
| 10) DEVICE B STATUS BYTE (STB) (RQS CLEAR) |
| 11) ASSERT ATN |
| 12) DEVICE C TALK ADDRESS (MTA) |
| 13) RELEASE ATN |
| 14) DEVICE C STATUS BYTE (STB) (RQS CLEAR) |
| 15) ASSERT ATN |
| 16) UNIVERSAL UNTALK (UNT) |
| 17) SERIAL POLL DISABLE (SPD) |
| 18) GO PROCESS SERVICE REQUEST |

Figure 5. Serial Polling

The following section describes the events which happen in a serial poll when 8291A and 8292 are the controller, and another 8291A is the slave device. While going through this section the reader should refer to the register diagrams for the 8291A and 8292.

A. DEVICE A REQUESTS SERVICE (SRQ BECOMES TRUE)

The slave devices rsv bit in the 2819A's serial poll mode register is set.

B. CONTROLLER RECOGNIZES SRQ AND ASSERTS ATN

The 8292's SPI pin 33 interrupts the CPU. The CPU reads the 8292's Interrupt status register and finds the SRQ bit set. The CPU tells the 8292 to 'Take Control Synchronously' by writing a OFDH to the 8292's command register.

C. THE CONTROLLER SENDS OUT THE FOLLOWING COMMANDS: UNIVERSAL UNLISTEN (UNL), SERIAL POLL ENABLE (SPE), MY TALK ADDRESS (MTA)

(MTA is a command which tells one of the devices on the bus to talk.)

The CPU in the controller waits for a BO (byte out) interrupts in the 8291A's interrupt status 1 register before it writes to the Data Out register a 3FH (UNL), 18H (SPE), 010XXXXX (MTA). The X represents the programmable address of a device on the GPIB. When the 8291A in the slave device receives its talk address, the ADSC bit in the Interrupt Status register 2 is set, and in the Address Status Register TA and TPAS bits are set.

D. CONTROLLER RECONFIGURES ITSELF TO LISTEN AND RESETS ATN

The CPU in the controller puts the 8291A in the listen only mode by writing a 40H to the Address Mode register of the 8291A, and then a 00H to the Aux Mode register. The second write is an 'Immediate Execute pon' which must be used when switching addressing modes such as talk only to listen only. To reset ATN the CPU tells the 8292 to 'Go To Standby' by writing a 0F6H to the command register. The moment ATN is reset, the 8291A in the slave device sets SPAS in Interrupt Status 2 register, and transmits the serial poll status byte. SRQS in the Serial Poll Status byte of the 8291A slave device is reset, and the SRQ line on the GPIB bus becomes false.

E. THE CONTROLLER READS THE SERIAL POLL STATUS BYTE, SETS ATN, THEN RECONFIGURES ITSELF TO TALK

The CPU in the controller waits for the Byte In bit (BI) in the 8291A's Interrupt Status 1 register. When this bit is set the CPU reads the Data In register to receive the Serial Poll Status Byte. Since bit 7 is set, this was the device which requested service. The CPU in the controller tells the 8292 to 'Take Control Synchronously' which asserts ATN. The moment ATN is asserted true the 8291A in the slave device resets SPAS, and sets the

3-74
Serial Poll Complete (SPC) bit in the Interrupt Status 2 register. The controller reconfigures itself to talk by setting the TO bit in the Address Mode register and then writing a OOH to the Aux Mode register.

F. THE CONTROLLER SENDS THE COMMANDS UNIVERSAL UNTALK (UNT), AND SERIAL POLL DISABLE (SPD) THEN RESETS THE SRQ BIT IN THE 8292 INTERRUPT STATUS REGISTER

The CPU in the controller waits for the BO Interrupt status bit to be set in the Interrupt Status 1 register of the 8291A before it writes 5FH (UNT) and 19H (SPD) to the Data Out register. The CPU then writes a 2BH to the 8292's command register to reset the SRQ status bit in the Interrupt Status register. When the 8291A in the slave device receives the UNT command the ADSC bit in the Interrupt Status 2 register is set, and the TA and TPAS bits in the Address Status register will be reset. At this point the controller can service the slave device's request.

Note that in the software listing of AP-66 (USING THE 8292 GPIB CONTROLLER) there is a bug in the serial poll routines. In the 'SRQ ROUTINE' when the CPU finds that the SRQ bit in the interrupt status register is set, it immediately writes the interrupt Acknowledge command to the 8292 to reset this bit. However, the SRQ GPIB line will still be driven true until the slave device driving SRQ has been polled. Therefore, the SRQ status bit in the 8292 will become set and latched again, and as a result the SRQ status bit in the 8292 will still be set after the serial poll. The proper time to reset the SRQ bit in the 8292 is after SRQ on the GPIB becomes false.

Parallel Poll

The 8291A supports an additional method for obtaining status from devices known as parallel poll (PPOL). This method limits the controller to a maximum of 8 devices at a time since each device will produce a single bit response on the GPIB data lines. As shown in the state diagrams, there are three basic parallel poll states: PPIS (parallel poll idle state), PPSS (parallel poll standby state), and PPAS (parallel poll active state).

In PPIS, the device's parallel poll function is in the idle state and will not respond to a parallel poll. PPSS is the standby state, a state in which the device will respond to a parallel poll from the controller. The response is initiated by the controller driving both ATN and EOI true simultaneously.

The 8291A state diagram shows a transition from PPIS to PPSS with the "lpe" message. This is a PP2 implementation for a parallel poll. This "lpe" (local poll enable) local message is achieved by writing 011USP3P2P1 to the Aux Mode Register with U=0. The S bit is the sense bit. If the "ist" (individual status) local message value matches the sense bit, then the 8291A will give a true response to a parallel poll. Bits P3−P1 identify which data line is used for a response.

For example, assume the programmer decides that the system containing the 8291A shall participate in parallel poll. The programmer, upon system initialization would write to the Aux Mode Register and reset the U bit and set the S bit plus identify a data line (P3−P1 bits). At "pon," the 8291A would not respond true to a parallel poll unless the parallel poll flag is set (via Aux Mode Register command).

When a status condition in the user system occurs and the programmer decides that this condition warrants a true response, then programmers software should set the parallel poll flag. Since the S bit value matches the "ist" (set) condition a true response will be given to all parallel polls.

An additional method of parallel polling reading exists known as a PP1 implementation. In this case the controller sends a PPE (parallel poll enable) message. PPE contains a bit pattern similar to the bit pattern used to program the "lpe" local message. The 8291A will receive this as an undefined command and use it to generate an "lpe" message. Thus the controller is specifying the sense bits and data lines for a response. A PPD (parallel poll disable) message exists which clears the bits SP3P2P1 and sets the U bit. This also will be received by the 8291A and used to generate an "lpe" false local message.

The actual sequence of events is as follows. The controller sends a PPC (parallel poll configure) message. This is an undefined command which is received in the CPT register and the handshake is held off. The local CPU reads this bit pattern, decodes it, and sends a VSCMD message to the Aux Mode Register. The controller then sends a ppe message which is also received as an undefined command in the CPT register. The local CPU reads this, decodes it clears the MSB, and writes this to the Aux Mode Register generating the "lpe" message.

The controller then sends ATN and EOI true and the 8291A drives the appropriate data line if the "ist" (parallel poll flag) is true. The controller will then send a PPD (parallel poll disable) message (again, an undefined command). The CPU reads this from the CPT register and uses it to write new "lpe" message (this "lpe" message will be false). The controller then sends a PPU (parallel poll unconfigure) message. Since this is also an undefined command, it goes into the CPT register. When the local CPU decodes this, the CPU should clear the "ist" (parallel poll flag).
APPLICATION EXAMPLES

In the course of developing this application note, two complete and identical; GPIB systems were built. The schematics and block diagrams are contained in Appendix 1. These systems feature an 8088 CPU, 8237 DMA controller, serial I/O (8215a and 8253), RAM, EPROM, and a complete GPIB talker/listener controller. Jumper switches were provided to select between a controller function and a talker/listener function. This system design is based on the design of Intel's SDK-86 prototyping kit and thus shares the same I/O and memory addresses. This system uses the same download software to transfer object files from Intel development systems.

Two Software Drivers

Two software drivers were developed to demonstrate a ton/lon environment. These two programs (BOARD 1 and BOARD 2) are contained in Appendix 2.

In this example, one of the systems (BOARD 1) initially is programmed in talk-only mode and synchronization is achieved by waiting for the listening board to become active. This is sensed by the lack of a GPIB error since a condition of no active listener produces an ERR status condition. Board 1 upon detecting the presence of an active listener transmits a block of 100 bytes from a PROM memory across the bus. The second system (BOARD 2) receives this data and stores it in a buffer, E01 is sent true by the talker (BOARD 1) with the last byte of data. Upon detection of E01, BOARD 2 switches to the talk only mode while BOARD 1 upon terminal count switches to the listen only mode. BOARD 2 then detects the presence of an active listener and transmits the contents of its buffer back to BOARD 1 which stores this data in the buffer. E01 again is sent with the last byte and BOARD 2 switches back to listen-only. BOARD 1 upon detecting E01 then compares the contents of its buffer with the contents of its PROM to ensure that no data transmission errors occurred. The process then repeats itself.

8291A with HP 9835A

An example of the 8291A used in conjunction with a bus controller is also included in this application note. In this example, the 8291A system used in previous experiments was connected via the GPIB to a Hewlett-Packard 9835A desktop computer. This computer contains, in addition to a GPIB interface, a black and white CRT, keyboard, tape drive for high quality data cassettes, and a calculator type printer. The software for the HP9835S is shown in Appendix 3. The user should refer to the operation manuals for the HP 9835A for information on the features and programming methods for the HP 9835A.

In this example, the 8292 was removed from its socket and the OPTA and OPTB pins of the two 8293 transceiver reconfigured to modes 0 and 1. Optionally, the mode pins could have been left wired for modes 2 and 3 and the 8292 left in its socket with its SYC pin wired to ground. This would have produced the same effect.

The first action performed is sending IFC. Generally, this is done when a controller first comes on line. This pulse is at least 100 \( \mu \)s in duration as specified by the IEEE-488 standard.

The software checks to see if active listeners are on line. For demonstration purposes, the HP 9835A will flag the operator to indicate that listeners are on line.

The HP 9835A then configures and performs a parallel poll (PPOL). The parallel poll indicates 1 bit of status of each device in a group of up to 8 devices. Such information could be used by an application program to determine whether optional devices are part of a system configuration. Such optional devices might include mass storage devices, printers, etc., where the application software for the controller might need to format data to match each type of device. Once the PPOL sequence is finished, the HP 9835A offers the user the opportunity to execute user commands from the keyboard. At this time the HP 9835A sits in a loop waiting for an SRQ condition. When the operator hits a key on the keyboard, the HP 9835A processor is interrupted and vectors to a service routine where the key is read and the appropriate routine is executed. The HP 9835A will then return to the loop checking for the SRQ true.

For this application, the valid keys are G, D, R, H, and X. Pressing the “G” key causes the GET command to be sent across the bus. A message to this effect is printed in the CRT and the HP 9835A returns. The “D” key causes the SDC message to be sent with the 8291A being the addressed device. Again, an appropriate message is output on the HP 9835A CRT. The “R” key causes the GTL message to be sent. The CRT displays “REMOTE MESSAGE SENT.” The “H” key causes a menu to be displayed on the HP 9835A CRT screen. This menu lists the allowed commands and their functions. NO GPIB commands are sent. The “X” key allows the operator to send one line of data across the bus. The line of data is terminated by a carriage return and line feed produced by pressing the “CONTINUE” key on the HP 9835A.

The characters are stored in the sequence entered into a buffer whose maximum size is 80 characters. Pressing the “CONTINUE” key terminates storing characters in the array and all characters including the carriage return and line feed are sent. E01 is then sent true with a false byte of 0OH. This false byte is due to the 1975 standard which allows asynchronous sending and reception of E01. (The 8291A supports the later 1978 standard which eliminates this false byte.)
After any key command is serviced control returns to the loop which checks for SRQ active. Should SRQ be active, then the keyboard interrupt is disabled and a message printed to indicate that SRQ has been received true.

The controller then performs a parallel poll.

This is an example of how parallel poll may be used to quickly check which group of devices contains a device sending SRQ. The eight devices in a group would, of course, have software drivers which allow a true response to a PPOL if that device is currently driving SRQ true. This would be a valuable method of isolation of the SRQ source in a system with a large number of devices. In this application program, only the response from the 8291A is of concern and only the 8291A’s response is considered. It does, however, demonstrate the technique employed. If a true response from the 8291A is detected, then a message to this effect is printed on the HP 9835A CRT screen. From this process, the controller has identified the device requesting service and will use a serial poll (SPOL) to determine the reason for the service request. This method of using PPOL is not specifically defined by the IEEE-488 standard but is a use of the resources provided.

The controller software then prints a message to indicate that it is about to perform a serial poll. This serial poll will return to the controller the current status of the 2819A and clear the service request. The status byte received is then printed on the CRT screen of the HP 9835A. One of the 8291A status bits indicates that the 8291A system has a field (on line or less) of data to transfer to the HP 9835A. If this bit is set, then the HP 9835A addresses the 8291A system to talk. The data is sent by the 8291A system is then printed on the CRT screen of the HP 9835A. The HP 9835 then enables the keyboard interrupts and goes into its SRQ checking loop.

Appendix 4 contains the software for the 8291A system which is connected to the HP 9835A via the GPIB. This software throws away the first byte of data it receives since this transfer was used by the HP 9835A to test when the 8291A system came on line.

Next, both status registers are read and stored in the two variable STAT 1 and STAT 2. It is necessary to store the status since reading the status registers clears the status bits.

Initially, six status bits are evaluated (END, GET, CPT, DEC, REMC, ADSC). Some of these conditions require that additional status bits be evaluated.

If END is true, then the 8291A system has received a block from the HP 9835A and the contents of a buffer is printed on the CRT screen. Next, the CPT bit is checked. PPC and PPE are only valid undefined commands in this example.

Next, the GET bit is examined and if true, the CRT screen connected to the serial channel on the 8291A system prints a message to indicate that the trigger command has been received. A similar process occurs with the DEC and REMC status bits.

Address Status Chagne (ADSC) is checked to see if the 8291A has been addressed or unaddressed by the controller. If ADSC is false, then the software checks the keyboard at the CRT terminal. If ADSC is set, then the TA and LA bits are read and evaluated to determine whether the 8291A has been addressed to talk or listen. The DMA controller is set to start transfers at the start of the character buffer and the type of transfer is determined by whether the 8291A in in TADS or LADS. We only need to set up the DMA controller since the transfers will be transparent to the system processor. The keyboard from the CRT terminal is then checked. If a key has been hit, then this character is stored in the character buffer and the buffer printer set to the next character location. This process repeats until the received character is a line feed. The line feed is echoed to the CRT, the serial poll status byte updated and the SRQ line driven true. This allows the 8291A system to store up to one line of characters before requesting a transfer to the controller. Recall that upon receiving an SRQ, the controller will perform a serial poll and subsequently address the 8291A to talk. The 8291A system then goes back to reading the status register thus repeating the process.

CONCLUSION

This application note has shown a basic method to view the IEEE 488 bus, when used in conjunction with Intel's 8291A.

The main reference for GPIB questions is the IEEE Standard 488—1978. Reference 8291A's data sheet for detailed information on it.

Additional Intel GPIB products include iSBX-488, which is a multimode board consisting of the 8291A, 8292, and 8293.

REFERENCES

8291A Data Sheet
8292 Data Sheet
8293 Data Sheet
Application Note #66 “Using the 8292 GPIB Controller”
PLM-86 User Manual
HP 9835A User's Manual
IEEE—488—1978 Standard
APPENDIX A
SYSTEM BLOCK DIAGRAM WITH 8088
APPENDIX B
SOFTWARE DRIVERS FOR BLOCK DATA TRANSFER

PL/M-86 COMPILER BOARD 1
ISIS-II PL/M-86 VI 1 COMPILATION OF MODULE BOARD 1
OBJECT MODULE PLACED IN F1 BRD1 OBJ
COMPILER INVOKED BY PLMSF F1 BRD1 SRC SYMBOLS MEDIUM

/* BOARD 1 TPT PROGRAM */
/* THIS BOARD TALKS TO THE OTHER BOARD BY */
/* TRANSFERRING A BLOCK OF DATA VIA THE 8237 */
/* COUPLED WITH THE 8291A THE 8291A IS PROGRAM- */
/* MED TO SEND EOI WHEN RECOGNIZING THE LAST */
/* DATA BYTE'S BIT PATTERN WHILE DATA IS BEING */
/* TRANSFERRED, THE PROCESSOR PERFORMS I/O READS */
/* OF THE 8237 COUNT REGISTERS TO SIMULATE BUS */
/* ACTIVITY, AND TO DETERMINE WHEN TO TURN THE */
/* LINE AROUND. AFTER THE 8237 HAS REACHED */
/* TERMINAL COUNT, THE 8291A IS PROGRAMMED TO */
/* THE LISTENER STATE AND WAITS FOR THE BLOCK */
/* TO BE TRANSMITTED BACK FROM THE SECOND BOARD. */
/* THIS DATA IS PLACED IN A SECOND BUFFER AND */
/* ITS CONTENTS COMPARED WITH THE ORIGINAL DATA */
/* TO CHECK FOR INTERFACE INTEGRITY */

1 BOARD1.

DO:

/* PROCEDURES */

2 1 PROCEDURE (XXX):
3 2 DECLARE XXX BYTE,
4 2 SER*STAT LITERALLY 'OFFF2H',
5 2 SER*DATA LITERALLY 'OFFFOH',
6 2 TXRDY LITERALLY '01H',
7 2 DO WHILE (INPUT (SER*STAT) AND TXRDY) <> TXRDY:
8 3 END:
9 2 OUTPUT (SER*DATA) = XXX;
10 2 END

11 /* SETUP BUFFERS */
12 1 DECLARE BUFSF (100) BYTE, /* RAM STORAGE AREA */
13 1 DECLARE BUFSF (100) BYTE DATA
14 (1,2,3,4,5,6,7,8,9,10H,
17 31H, 32H, 33H, 34H, 35H, 36H, 37H, 38H, 39H, 40H,
18 41H, 42H, 43H, 44H, 45H, 46H, 47H, 48H, 49H, 50H,
19 51H, 52H, 53H, 54H, 55H, 56H, 57H, 58H, 59H, 60H,
20 61H, 62H, 63H, 64H, 65H, 66H, 67H, 68H, 69H, 70H,
21 71H, 72H, 73H, 74H, 75H, 76H, 77H, 78H, 79H, 80H,
22 81H, 82H, 83H, 84H, 85H, 86H, 87H, 88H, 89H, 90H,

230832-5
DECLARE BUFF3(17) BYTE DATA
(ODH, OA, 'COMPARE ERROR', ODH, CAH), /* ROM STORAGE AREA */

/* 8237 PORT ADDRESSES */

DECLARE

CLEAR#FF LITERALLY 'OFFDDH', /* MASTER CLEAR */
START#O#LO LITERALLY 'OFFDDH',
START#O#HI LITERALLY 'OFFDDH',
O#COUNT#LO LITERALLY 'OFFDDH',
O#COUNT#HI LITERALLY 'OFFDDH',
SET#MODE LITERALLY 'OFFDDH',
CMD#37 LITERALLY 'OFFDDH',
SET#MASK LITERALLY 'OFFDDH',

/* 8237 COMMAND - DATA BYTES */
DECLARE DMA#ADR#TALK POINTER,
DECLARE DMA#ADR#LSTN POINTER,
DECLARE

RD$TRANSFER LITERALLY '48H',
WR$TRANSFER LITERALLY '44H',
NORM$TIME LITERALLY '20H',
TC#LO1 LITERALLY 'OFFH',
TC#HI LITERALLY '00H',
TC#LO2 LITERALLY '9DH', /* 100 XFERS */
TC LITERALLY '01H',
I BYTE;

DECLARE

DMA#WR#TALK (2) WORD AT (@DMA#ADR#TALK),
DMA#WR#LSTN(2) WORD AT (@DMA#ADR#LSTN),

/* 8291A PORT ADDRESSES */

DECLARE

PORT#OUT LITERALLY 'OFFCOH', /* DATA OUT */
PORT#IN LITERALLY 'OFFCOH',
STATUS$1 LITERALLY 'OFFCH', /* INTR STAT 2 */
STATUS$2 LITERALLY 'OFFCH', /* INTR STAT 2 */
ADDR$STATUS LITERALLY 'OFFC4H',
COMMAND$MOD LITERALLY 'OFFC5H', /* CMD PASS THRU */
ADDR$#O LITERALLY 'OFFC6H',
EOS$REG LITERALLY 'OFF7H', /* EOS REGISTER */
AP-166

PL/M-86 COMPILER BOARD

DECLARE
END$EO LITERALLY '88H',
DNE LITERALLY '10H',
PON LITERALLY '00H',
RESET LITERALLY '02H',
CLEAR LITERALLY '00H',
DMA$REG$ LITERALLY '10H',
DMA$REG$T LITERALLY '20H',
MOD1$T LITERALLY '80H',
MOD1$LO LITERALLY '40H',
EOS LITERALLY '0DH',
PRESCALER LITERALLY '23H',
HIGH$SPEED LITERALLY 'OA4H',
OKAY LITERALLY '0FFFFH',
XYZ BYTE MATCH WORD,
BO LITERALLY '02H',
BI LITERALLY '01H',
ERR LITERALLY '04H'.

/* CODE BEGINS */

START91:

OUTPUT (STATUS$2) =CLEAR, /* SHUT-OFF DMA REG BITS TO */
/* PREVENT EXTRA DMA REGS */
/* FROM 8291A */

/* MANIPULATE DMA ADDRESS VARIABLES */

DMA$DRD$TALK = (@BUFF1),
DMA$ADR$ = (@BUFF2),
DMA$WRD$TALK$ = SHL (DMA$WRD$TALK$ + DMA$WRD$TALK$ + DMA$WRD$TALK$),
DMA$WRD$TALK$ = DMA$WRD$TALK$ + DMA$WRD$TALK$,
DMA$WRD$TALK$ = DMA$WRD$TALK$ + DMA$WRD$TALK$ + DMA$WRD$TALK$.

/* INIT B237 FOR TALKER FUNCTIONS */

OUTPUT (CLEAR$FF) =CLEAR, /* TGOOLE MASTER CLEAR */
OUTPUT (CMD$37) =NORM$TIME,
OUTPUT (SET$MODE) =RD$TRANSFER,
OUTPUT (SET$MASK) =CLEAR,
OUTPUT (START$O$LO) =DMA$WRD$TALK$,
OUTPUT (START$O$HI) =DMA$WRD$TALK$,
OUTPUT (O$COUNT$LO) =TC$LO$,
OUTPUT (O$COUNT$HI) =TC$HI$.

/* INIT B291A FOR TALKER FUNCTIONS */

/* 3-81 */
AP-166

34 1  OUTPUT (EOS*REG)  =EOS.
35 1  OUTPUT (COMMAND*MOD)  =END*EOI. /* EOI ON EOS SENT */
36 1  OUTPUT (ADDR*STATUS)  =MOD1*TO. /* TALK ONLY */
37 1  OUTPUT (COMMAND*MOD)  =PRESCALER.
38 1  OUTPUT (COMMAND*MOD)  =HIGH*SPEED.
39 1  OUTPUT (COMMAND*MOD)  =PON;

40 1  DO WHILE (INPUT (STATUS*1) AND BO) =0;
41 2  END. /* WAIT FOR BO INTR */
42 1  OUTPUT (PORT*OUT)  =GAH.

43 1  DO WHILE (INPUT (STATUS*1) AND ERR) = ERR;
44 2  DO WHILE (INPUT (STATUS*1) AND BO) = 0;
45 3  END. /* WAIT FOR BO INTR */
46 2  OUTPUT (PORT*OUT)  =GAH.
47 2  END;

48 1  OUTPUT (SATUS*2)  =DMA*REG*T. /* ENABLE DMA REGS */
49 1  DO WHILE (INPUT (CMD*37) AND TC) < TC;
50 2  /* WAIT FOR TC = 0 */
51 1  END;

52 1  INIT37L;
53 2  OUTPUT (STATUS*2)  =CLEAR. /* DISABLE DMA REGS */
54 1  /* INIT 8237 FOR LISTENER FUNCTIONS */
55 1  OUTPUT (CLEAR*FF)  =CLEAR. /* TOGGLE MASTER RESET */
56 1  OUTPUT (CMD*37)  =NORM*TIME.
57 1  OUTPUT (SET*MODE)  =WR*TRANSFER.
58 1  OUTPUT (SET*MASK)  =CLEAR.
59 1  OUTPUT (START*LO)  =DMA*WRD*LSTN (0),
60 1  DMA*WRD*LSTN (0)  =SHK (DMA*WRD*LSTN (0), B),
61 1  OUTPUT (START *HI)  =DMA*WRD*LSTN (0),
62 1  OUTPUT (COUNT*LO)  =TC*LO1,
63 1  OUTPUT (COUNT*HI)  =TC*HI1.
64 1  /* INIT 8291A FOR LISTENER FUNCTIONS */
65 1  OUTPUT (COMMAND*MOD)  =RESET.
66 1  OUTPUT (ADDR*STATUS)  =MOD1*LO. /* LISTEN ONLY */
67 1  OUTPUT (COMMAND*MOD)  =PON;
68 1  DO WHILE (INPUT (STATUS*1) AND BI) = 0,
69 2  END. /* WAIT FOR BI INTR */
70 1  XYZ  = INPUT (PORT*IN),
71 1  OUTPUT (STATUS*2)  =DMA*REGSL. /* ENABLE DMA REGS */
72 1  OUTPUT (STATUS*2)  =DMA*REGSL. /* ENABLE DMA REGS */
73 1  DO WHILE (INPUT (STATUS*1) AND DNE) = DNE,
74 2  /* WAIT FOR EOI RECEIVED */

230832-8
PL/M-86 COMPILATION  BOARD 1

70 1 CMPBLKS
   /* COMPARE THE TWO BUFFERS CONTENTS */
   MATCH=CMPB (@BUFF1, @BUFF2, 100);
71 1 IF MATCH = OKAY THEN GOTO START91,
   /* SEND ERROR MESSAGE IN BUFFER 3 */
73 1 DO I=0 TO 16,
74 2 CALL CO (BUFF 3 (I)),
75 2 END,
76 1 GOTO START91,
77 1 END,

MODULE INFORMATION:
CODE AREA SIZE =01DBH  475D
CONSTANT AREA SIZE =0075H  117D
VARIABLE AREA SIZE =0070H  112D
MAXIMUM STACK SIZE =0006H  6D
243 LINES READ
0 PROGRAM ERROR (5)

END OF PL/M-86 COMPILATION
PL/M-8t. COMPILER  BOARD2

ISIS-II PL/M-8t. VI 1 COMPILED OF MODULE BOARD2
OBJECT MODULE PLACED IN FI BRD2. OBJ
COMPILER INVOKED BY  PLM86  FI  BRD2. SRC

/* BOARD 2 TTP PROGRAM */
/* THIS BOARD LISTENS TO THE OTHERWISE BOARD (1) */
/* AND DMA'S DATA INTO A BUFFER, WHILE WAITING */
/* FOR THE END INTERRUPT BIT TO BECOME ACTIVE */
/* UPON END ACTIVE, THE DATA IN THE BUFFER IS */
/* SENT BACK TO THE FIRST BOARD VIA THE GPD */
/* WHEN THE BLOCK IS FINISHED THE B291A IS */
/* PROGRAMMED BACK INTO THE LISTENER MODE */

/
BOARD2

DO.

/* 8237 PORT ADDRESSES */

DECLARE
CLEAR*OFF  LITERALLY 'OFFDDH', /*MASTER CLEAR*/
START$O$LO  LITERALLY 'OFFDOH',
START$O$HI  LITERALLY 'OFFDOH',
O$COUNT$LO  LITERALLY 'OFFDIH',
O$COUNT$HI  LITERALLY 'OFFDIH',
SET$MODE  LITERALLY 'OFFDBH',
CMD$37  LITERALLY 'OFFDBH',
SET$MASK  LITERALLY 'OFFDFH',

/* 8237 COMMAND - DATA BYTES */

DECLARE
RD$TRANSFER  LITERALLY '48H',
WR$TRANSFER  LITERALLY '44H',
ADDR$1A  LITERALLY '00H',
ADDR$1B  LITERALLY '01H',
NORM$TIME  LITERALLY '20H',
TC$LO1  LITERALLY 'OFFH',
TC$HI1  LITERALLY '00H',
TC$LO2  LITERALLY '990',
TC$HI2  LITERALLY '00H',
TC  LITERALLY '01H',
/* 8291A PORT ADDRESSES */

DECLARE
PORT$OUT  LITERALLY 'OFFCOH',
PORT$IN  LITERALLY 'OFFCOH', /* DATA IN */
STATUS$1  LITERALLY 'OFFCH', /* INTR STAT 1 */
STATUS$2  LITERALLY 'OFFC2H', /* INTR STAT 2 */
ADDR$STATUS  LITERALLY 'OFFCAH', /* ADDR STAT */
COMMAND$MOD  LITERALLY 'OFFC5H', /* CMD PASS THRU */
DECLARE
END*EOI LITERALLY 'BBH',
DNE LITERALLY '10H',
FON LITERALLY '00H',
RESET LITERALLY '02H',
CLEAR LITERALLY '00H',
DMA*REG*0 LITERALLY '10H',
DMA*REG*1 LITERALLY '20H',
MODI*TO LITERALLY '00H',
MODI*LO LITERALLY '40',
EOS LITERALLY '0DH',
PRESCALER LITERALLY '23H',
HIGH*SPEED LITERALLY 'A4H',
XYZ BYTE,
BO LITERALLY '02H',
BI LITERALLY '01H',
ERR LITERALLY '04H',

61

START91,

OUTPUT (STATUS$2) =CLEAR; /* END INITIALIZATION STATE */

/* INIT 9237 FOR LISTENER FUNCTION */

71

INIT37L,

OUTPUT (CLEAR$FF) =CLEAR; /* TOGGLE MASTER RESET */
81
OUTPUT (CMD$37) =NORM$TIME,
91
OUTPUT (SET$MODE) =WR$TRANSFER, /* BLOCK XFER MODE */
101
OUTPUT (SET$MASK) =CLEAR,
111
OUTPUT (START$0$LO) =ADDR$1$A;
121
OUTPUT (START$0$HI) =ADDR$1$B;
131
OUTPUT (0$COUNT$LO) =TC$LO$1,
141
OUTPUT (0$COUNT$HI) =TC$HI$1,

/* INIT 8291A FOR LISTENER FUNCTIONS */
151
OUTPUT (COMMAND$MOD) =RESET;
161
OUTPUT (ADDR$STATUS) =MODI$LO,
171
OUTPUT (COMMAND$MOD) =PON,
181
DO WHILE (INPUT (STATUS$1) AND BI) =0;
192
END. /* WAIT FOR BI INTR */
201
XYZ= INPUT (PORT$IN),
211
OUTPUT (STATUS$2) =DMA*REG$0L;

/* WAIT UNTIL EOI RCVD AND END INTR-BIT SET */
221
DO WHILE (INPUT (STATUS$1) AND DNE ) <> DNE;
PL/M-86 COMPILER BOARD2

23 1 END.

24 1 INIT37T.
/* INIT 6237 FOR TALKER FUNCTION */

25 1 OUTPUT (STATUS$2) =CLEAR; /* CLEAR 8291A DRQ */
26 1 OUTPUT (CLEAR$FF) =CLEAR.
27 1 OUTPUT (CMD$37) =NORM$TIME.
28 1 OUTPUT (SET$MOD$) =RD$TRANSFER, /* BLOCK XFER MODE */
29 1 OUTPUT (SET$MASK) =CLEAR.
30 1 OUTPUT (START$OS$LO) =ADDR$1A.
31 1 OUTPUT (O$COUNT$LO) =TC$LO2.
32 1 OUTPUT (O$COUNT$HI) =TC$HI2.

/* INIT 8291A FOR TALKER FUNCTION */

33 1 OUTPUT (EOS$REG) =EOS.
34 1 OUTPUT (COMMAND$MOD) =END$EDI; /* EDI ON EOS SENT */
35 1 OUTPUT (ADDR$STATUS) =MOD$ID. /* TALK ONLY */
36 1 OUTPUT (COMMAND$MOD) =PRES$CALER.
37 1 OUTPUT (COMMAND$MOD) =HIG$H$SPEED.
38 1 OUTPUT (COMMAND$MOD) =P.ON.

39 1 DO WHILE (INPUT (STATUS$1) AND BO) =0;
40 2 END. /* WAIT FOR BO INTR */
41 1 OUTPUT (PORT$OUT) =OAAH.

42 1 DO WHILE (INPUT (STATUS$1) AND ERR) =ERR;
43 2 DO WHILE (INPUT (STATUS$1) AND BO) =0;
44 3 END. /* WAIT FOR BO INTR */
45 2 OUTPUT (PORT$OUT) =OAAH.
46 2 END;

47 1 OUTPUT (STATUS$2) =DMA$REG$T.
/* WAIT FOR TC=0 */

48 1 DO WHILE (INPUT (CMD$37) AND TC) <> TC;
49 2 END;
50 1 GOTO START91.
51 1 END;

MODULE INFORMATION

| CODE AREA SIZE | =0122H 290D |
| CONSTANT AREA SIZE | =0000H 0D |
| VARIABLE AREA SIZE | =0001H 1D |
| MAXIMUM STACK SIZE | =0000H 0D |
| 152 LINES READ |
| 0 PROGRAM ERROR (S) |

230832-12
APPENDIX C
SOFTWARE FOR HP 9835A

```
10 REM SEND IN
20 ARBITRO ?
30 REM FORCE E
40 REM UNTIL KEV
50 REM INTERFACE CLEAR
60 PUT 784 USING "*E*" 
70 IF Err=1 THEN GOTO ?err
80 PRINT CHRS
90 REM CLEAR
100 PRINT CHRS
110 PRINT CHRS
120 PRINT CHRS
130 PRINT CHRS
140 PRINT CHRS
150 PRINT CHRS
160 PRINT CHRS
170 PRINT CHRS
180 PRINT CHRS
190 PRINT CHRS
200 PRINT CHRS
210 PRINT CHRS
220 PRINT CHRS
230 PRINT CHRS
240 PRINT CHRS
250 PRINT CHRS
260 PRINT CHRS
270 PRINT CHRS
280 PRINT CHRS
290 PRINT CHRS
300 PRINT CHRS
310 PRINT CHRS
320 PRINT CHRS
330 PRINT CHRS
340 PRINT CHRS
350 PRINT CHRS
360 PRINT CHRS
370 PRINT CHRS
380 PRINT CHRS
390 PRINT CHRS
400 PRINT CHRS
410 PRINT CHRS
420 PRINT CHRS
430 PRINT CHRS
440 PRINT CHRS
450 PRINT CHRS
460 PRINT CHRS
470 PRINT CHRS
480 PRINT CHRS
490 PRINT CHRS
500 PRINT CHRS
510 PRINT CHRS
520 PRINT CHRS
530 PRINT CHRS
540 PRINT CHRS
550 PRINT CHRS
560 PRINT CHRS
570 PRINT CHRS
580 PRINT CHRS
590 PRINT CHRS
600 PRINT CHRS
610 PRINT CHRS
620 PRINT CHRS
630 PRINT CHRS
640 PRINT CHRS
650 PRINT CHRS
660 PRINT CHRS
670 PRINT CHRS
680 PRINT CHRS
690 PRINT CHRS
700 PRINT CHRS
710 PRINT CHRS
720 PRINT CHRS
730 PRINT CHRS
740 PRINT CHRS
750 PRINT CHRS
760 PRINT CHRS
770 PRINT CHRS
780 PRINT CHRS
790 PRINT CHRS
800 PRINT CHRS
810 PRINT CHRS
820 PRINT CHRS
830 PRINT CHRS
840 PRINT CHRS
850 PRINT CHRS
860 PRINT CHRS
870 PRINT CHRS
880 PRINT CHRS
890 PRINT CHRS
900 PRINT CHRS
910 PRINT CHRS
920 PRINT CHRS
930 PRINT CHRS
940 PRINT CHRS
950 PRINT CHRS
960 PRINT CHRS
970 PRINT CHRS
980 PRINT CHRS
990 PRINT CHRS
```

```
00 PRINT "*P"
01 PRINT "*E"
02 PRINT "*E"
03 PRINT "*E"
04 PRINT "*E"
05 PRINT "*E"
06 PRINT "*E"
07 PRINT "*E"
08 PRINT "*E"
09 PRINT "*E"
10 PRINT "*E"
11 PRINT "*E"
12 PRINT "*E"
13 PRINT "*E"
14 PRINT "*E"
15 PRINT "*E"
16 PRINT "*E"
17 PRINT "*E"
18 PRINT "*E"
19 PRINT "*E"
20 PRINT "*E"
21 PRINT "*E"
22 PRINT "*E"
23 PRINT "*E"
24 PRINT "*E"
25 PRINT "*E"
26 PRINT "*E"
27 PRINT "*E"
28 PRINT "*E"
29 PRINT "*E"
30 PRINT "*E"
31 PRINT "*E"
32 PRINT "*E"
33 PRINT "*E"
34 PRINT "*E"
35 PRINT "*E"
36 PRINT "*E"
37 PRINT "*E"
38 PRINT "*E"
39 PRINT "*E"
40 PRINT "*E"
41 PRINT "*E"
42 PRINT "*E"
43 PRINT "*E"
44 PRINT "*E"
45 PRINT "*E"
46 PRINT "*E"
47 PRINT "*E"
48 PRINT "*E"
49 PRINT "*E"
50 PRINT "*E"
51 PRINT "*E"
52 PRINT "*E"
53 PRINT "*E"
54 PRINT "*E"
55 PRINT "*E"
56 PRINT "*E"
57 PRINT "*E"
58 PRINT "*E"
59 PRINT "*E"
60 PRINT "*E"
61 PRINT "*E"
62 PRINT "*E"
63 PRINT "*E"
64 PRINT "*E"
65 PRINT "*E"
66 PRINT "*E"
67 PRINT "*E"
68 PRINT "*E"
69 PRINT "*E"
70 PRINT "*E"
71 PRINT "*E"
72 PRINT "*E"
73 PRINT "*E"
74 PRINT "*E"
75 PRINT "*E"
76 PRINT "*E"
77 PRINT "*E"
78 PRINT "*E"
79 PRINT "*E"
80 PRINT "*E"
81 PRINT "*E"
82 PRINT "*E"
83 PRINT "*E"
84 PRINT "*E"
85 PRINT "*E"
86 PRINT "*E"
87 PRINT "*E"
88 PRINT "*E"
89 PRINT "*E"
90 PRINT "*E"
91 PRINT "*E"
92 PRINT "*E"
93 PRINT "*E"
94 PRINT "*E"
95 PRINT "*E"
96 PRINT "*E"
97 PRINT "*E"
98 PRINT "*E"
99 PRINT "*E"
```

```
```
```
APPENDIX D
SOFTWARE FOR HP 8088/HP 9835A VIA GPIB

PL/M-86 COMPILER  HPIB

ISIS-II PL/M-86 V1.1 COMPILATION OF MODULE HPIB
OBJECT MODULE PLACED IN :F1:HPIB.OBJ
COMPILER INVOKED BY: PLMB6 :F1:HPIB.SRC LARGE

1  HPIB:
   /**
    * PARAMETER DECLARATIONS
   */
   DO;

2  DECLARE
    ADDR#1 LITERALLY '01H',
    ADDR#0 LITERALLY '00H',
    ADSC LITERALLY '01H',
    BI LITERALLY '01H',
    BO LITERALLY '02H',
    CHAR#COUNT BYTE,
    CHAR# BYTE,
    CHAR#BO BYTE,
    CLEAR LITERALLY '00H',
    CPT LITERALLY '00H',
    CRLF LITERALLY '0AH',
    DEC LITERALLY '08H',
    DMS#ADR#LSTN POINTER,
    DMS#ADR#TALK POINTER,
    DMS#RD#LSTN(2) WORD AT (DMS#ADR#LSTN),
    DMS#RD#TALK(2) WORD AT (DMS#ADR#TALK),
    DMS#REQ#L LITERALLY '10H',
    DMS#REQ#T LITERALLY '20H',
    DNE LITERALLY '10H',
    END#EDI LITERALLY '80H',
    EOS LITERALLY '00H',
    ERR LITERALLY '04H',
    GET LITERALLY '20H',
    I BYTE,
    LISTEN LITERALLY '04H',
    MLA LITERALLY '04H',
    MODE#1 LITERALLY '01H',
    NO#DMA LITERALLY '00H',
    NO#RSV LITERALLY '00H',
    NORM#TIME LITERALLY '20H',
    PON LITERALLY '00H',
    PPC LITERALLY '05H',
    PPE#MASK LITERALLY '40H',
    PPO#L#CN#F#O#FLAG LITERALLY '01H',
    PPO#L#E#N#BYTE BYTE,
    PRI#BUF#BO BYTE AT (@CHARS),
    R#XFER LITERALLY '40H',
    RESET LITERALLY '02H',
    REMC LITERALLY '02H',
    RSV LITERALLY '02H',
    RXRDY LITERALLY '02H',

   230832-16
PL/M-86 COMPILER

DECLARE

ADDRESS LITERALLY 'OFFC%H',
ADDRESSSTATUS LITERALLY 'OFFC4H',
CLEAR#FF LITERALLY 'OFFFDDH',
COM#37 LITERALLY 'OFFD8H',
COMMAND#MOD LITERALLY 'OFFC5H',
COUNT#HI LITERALLY 'OFFD1H',
COUNT#LO LITERALLY 'OFFD1H',
CPT#REG LITERALLY 'OFFC5H',
EDS#REG LITERALLY 'OFFC7H',
PORT#IN LITERALLY 'OFFC9H',
PORT#OUT LITERALLY 'OFFCOH',
SER#DATA LITERALLY 'OFFFOH',
SER#STAT LITERALLY 'OFFF2H',
SET#MASK LITERALLY 'OFFDFH',
SET#MODE LITERALLY 'OFFDBH',
SPOLL#STAT LITERALLY 'OFFC3H',
START#HI LITERALLY 'OFFDDH',
START#LO LITERALLY 'OFFDDH',
STATUS#1 LITERALLY 'OFFC1H',
STATUS#2 LITERALLY 'OFFC2H'

DECLARE GET#MSG(11) BYTE DATA (ODH,OAH,'TRIGGER',OAH,ODH);
DECLARE DECS#MSG(16) BYTE DATA (ODH,OAH,'DEVC CLEAR',OAH,ODH);
DECLARE REM#MSG(10) BYTE DATA (ODH,OAH,'REMOTE',OAH,ODH);
DECLARE CPT#MSG(22) BYTE DATA (ODH,OAH,'Undef CMD RECEIVED',OAH,ODH);
DECLARE HUH#MSG(11) BYTE DATA (ODH,OAH,'HUH ???',ODH,OAH);

/* called procedures */

REGSER: PROCEDURE;

230832-17
AP·166

PL/M-86 COMPILER HPB

10 2 OUTPUT (SPOLL*STAT)=TRG;
11 2 DO WHILE (INPUT (SPOLL*STAT) AND SRGS)=SRGS;
12 3 END;
13 2 OUTPUT (SPOLL*STAT)=NO*RSV;
14 2 END REGSER;
15 1 CO: PROCEDURE(XXX);
16 2 DECLARE XXX BYTE;
17 2 DO WHILE (INPUT (SER*STAT) AND TXRDY)<TXRDY;
18 3 END;
19 2 OUTPUT (SER*DATA)=XXX;
20 2 END CO;
21 1 HUH: PROCEDURE;
22 2 DO I=0 TO 10;
23 3 CALL CO (HUH*MSG(I));
24 3 END;
25 2 END HUH;
26 1 CI: PROCEDURE;
27 2 IF (INPUT (SER*STAT) AND RXRDY)=RXRDY THEN
28 2 DO;
29 3 I=0;
30 3 CHAR*COUNT=0;
31 3 STORE*CHAR: CHAR=(INPUT (SER*DATA) AND 7FH));
32 3 CHAR*COUNT=CHAR*COUNT+1;
33 3 CALL CO (CHAR);
34 3 CHAR*(I)=CHAR;
35 3 I=I+1;
36 3 IF CHAR <> CR/LF THEN
37 3 DO;
38 4 DO WHILE (INPUT (SER*STAT) AND RXRDY)<RXRDY;
39 5 END;
40 4 QOTO STORE*CHAR;
41 4 END;
42 3 END;
43 3 END;
44 2 END CI;
45 1 TALK*EXEC: PROCEDURE;
46 2 OUTPUT (STATUS*2)=CLEAR;
47 2 /*
48 2 manipulate address bits for DMA controller
49 2 */
50 2 DMA*ADR*TALK=((CHAR));
51 2 DMA*WRD*TALK(1)=SHL((DMA*WRD*TALK(1),4));
52 2 DMA*WRD*TALK(0)=DMA*WRD*TALK(0)+DMA*WRD*TALK(1);
53 2 OUTPUT (CLEAR*FF)=CLEAR;

230832-18

3-90
PL/M-86 COMPILER

AP-166

51 2  OUTPUT (CMD37)=NORMSTIME;
52 2  OUTPUT (SETMODE)=RDXFER;
53 2  OUTPUT (SETMASK)=CLEAR;
54 2  OUTPUT (STARTLO)=DMAWRDSTALK(0);
55 2  DMAWRDSTALK(0)=SHR(DMAWRDSTALK(0),8);
56 2  OUTPUT (STARTHI)=DMAWRDSTALK(0);
57 2  OUTPUT (COUNTLO)=CHARSCOUNT;
58 2  OUTPUT (COUNTHI)=0;
59 2  OUTPUT (EOSREQ)=EOS;
60 2  OUTPUT (COMMANDMOD)=ENDEOI;
61 2  DO WHILE (INPUT (STATUS1) AND BO)=0;
62 3  END;
63 2  OUTPUT (PORTOUT)=OAAH;
64 2  DO WHILE (INPUT (STATUS1) AND ERR)=ERR;
65 3  DO WHILE (INPUT (STATUS1) AND BO)=0;
66 4  END;
67 3  OUTPUT (PORTOUT)=OAAH;
68 3  END;
69 2  OUTPUT (STATUS2)=DMAREGST;
70 2  END TALKEXEC;

LISTENEXEC:  PROCEDURE;
71 1  OUTPUT (STATUS2)=CLEAR;
72 2  OUTPUT (CLEARFF)=CLEAR;
73 2  OUTPUT (CMD37)=NORMSTIME;
74 2  OUTPUT (SETMODE)=RDXFER;
75 2  OUTPUT (SETMASK)=CLEAR;
76 2  DMAADRSTN=@CHARS);
77 2  DMAWRDSLSTN(1)=SHL(DMAWRDSLSTN(1),4);
78 2  DMAWRDSLSTN(0)=DMAWRDSLSTN(0)+DMAWRDSLSTN(1);
79 2  OUTPUT (STARTLO)=DMAWRDSLSTN(0);
80 2  DMAWRDSLSTN(0)=SHR(DMAWRDSLSTN(0),8);
81 2  OUTPUT (STARTHI)=DMAWRDSLSTN(0);
82 2  OUTPUT (COUNTLO)=TCLO;
83 2  OUTPUT (COUNTHI)=TCHI;
84 2  OUTPUT (STATUS2)=DMAREGST;
85 2  END LISTENEXEC;
86 2  END PRINTER;  PROCEDURE;
87 1  I=0;
88 2  DO WHILE PRISBUF(1)<>CRLF;
89 2  CALL CO (PRISBUF(1));
90 3  I=I+1;
91 3  END;
92 3  END;
93 2  CALL CO (PRISBUF(1));
94 2  END PRINTER;

230832-19
PL/M-86 COMPILER HPIB

95 1 ADSCEXEC:  PROCEDURE;

96 2 TA0R:LA=INPUT (ADDR:STATUS);
97 2 IF (TA0R:LA AND TALK)=TALK THEN
98 2 CALL TALK:EXEC;
99 2 IF (TA0R:LA AND LISTEN)=LISTEN THEN
100 2 CALL LISTEN:EXEC;
101 2 END ADSCEXEC;

102 1 GETEXEC:  PROCEDURE;
103 2 DO I=0 TO 10;
104 3 CALL CO (GETMSG(I));
105 3 END;
106 2 END GETEXEC;

107 1 DECEXEC:  PROCEDURE;
108 2 DO I=0 TO 15;
109 3 CALL CO (DECMG(I));
110 3 END;
111 2 END DECEXEC;

112 1 REMEXEC:  PROCEDURE;
113 2 DO I=0 TO 9;
114 3 CALL CO (REMSG(I));
115 3 END;
116 2 END REMEXEC;

117 1 PPOLL:CON:  PROCEDURE;
118 2 OUTPUT (COMMAND:MOD)=PPOLL:CNFG:FLAG;
119 2 PPOLL:CON;
120 1 PPOLL:EN:  PROCEDURE;
121 2 PPOLL:EN:BYTE=(UDC AND $AF);
122 2 OUTPUT (COMMAND:MOD)=PPOLL:EN:BYTE;
123 2 END PPOLL:EN;

124 1 CPTEXEC:  PROCEDURE;
125 2 DO I=0 TO 21;
126 3 CALL CO (CPTMSG(I));
127 3 END;
128 2 UDC=INPUT (CPT:REG);
129 2 UDC=(UDC AND $7F);
130 2 IF (UDC AND PPC)=PPC THEN
131 2 CALL PPOLL:CON;
132 2 IF (UDC AND PPE:MASK)=PPE:MASK THEN
133 2 CALL PPOLL:EN;

230892-20
AP-166

PL/M-B6 COMPILER  HPIB

134 2  END CPT*EXEC:

/*
BEGIN CODE
*/

135 1  INIT:

   OUTPUT (CLEAR*FF) =CLEAR;
   OUTPUT (COMMAND*MOD) =RESET;
   OUTPUT (ADDR*STATUS) =MODE*1;
   OUTPUT (ADDR*0) =MLA;
   OUTPUT (STATUS*2) =NO*DMA;
   OUTPUT (COMMAND*MOD) =PON;

141 1  LISTENERS:

   /* response to listeners check */

142 2  DO WHILE (INPUT (STATUS*1) AND BI)=0;
   END;

143 1  XYZ=INPUT (PORT*IN);
144 1  XYZ=INPUT (STATUS*2);

145 1  CMD:

   RDSTAT:
   /* read status registers and interpret command */

146 1  STAT1=INPUT (STATUS*1);
147 1  STAT2=INPUT (STATUS*2);

148 1  IF (STAT1 AND DNE)=DNE THEN
149 1  CALL PRINTER;
150 1  IF (STAT1 AND CPT)=CPT THEN
151 2  CALL CPT*EXEC;
152 2  STAT2=(STAT2 AND OFEH);
153 2  END;
154 1  IF (STAT1 AND GET)=GET THEN
155 1  DO;
156 2  CALL GET*EXEC;
157 2  STAT2=(STAT2 AND OFEH);
158 2  END;
159 1  IF (STAT1 AND DEC)=DEC THEN
160 1  DO;
161 2  CALL DEC*EXEC;
162 2  STAT2=(STAT2 AND OFEH);
163 2  END;
164 1  IF (STAT2 AND REMC)=REM THEN
165 1  DO;
166 2  CALL REMC*EXEC;
167 2  STAT2=(STAT2 AND OFEH);
168 2  END;
169 1  IF (STAT2 AND ADSC)=ADSC THEN

230832-21
PL/M-B6 COMPILER HPIB

170  1        DO;
171  2       CALL ADSC$EXEC;
172  2      STAT2=(STAT2 AND OFEH);
173  2       END;

174  1     CALL CI;
175  1   GOTO CMD;

176  1    END;

MODULE INFORMATION:

CODE AREA SIZE = 0475H  1141D
CONSTANT AREA SIZE = 0000H  0D
VARIABLE AREA SIZE = 0061H  97D
MAXIMUM STACK SIZE = 000AH  10D
349 LINES READ
0 PROGRAM ERROR(S)

END OF PL/M-B6 Compilation
Using the 8292 GPIB Controller
INTRODUCTION

The Intel® 8292 is a preprogrammed UPI™-41A that implements the Controller function of the IEEE Std 488-1978 (GPIB, HP-IB, IEC Bus, etc.). In order to function the 8292 must be used with the 8291 Talker/Listener and suitable interface and transceiver logic such as a pair of Intel 8293s. In this configuration the system has the potential to be a complete GPIB Controller when driven by the appropriate software. It has the following capabilities: System Controller, send IFC and Take Charge, send REN, Respond to SRQ, send Interface messages, Receive Control, Pass Control, Parallel Poll and Take Control Synchronously.

This application note will explain the 8292 only in the system context of an 8292, 8291, two 8293s and the driver software. If the reader wishes to learn more about the UPI-41A aspects of the 8292, Intel's Application Note AP-41 describes the hardware features and programming characteristics of the device. Additional information on the 8291 may be obtained in the data sheet. The 2893 is detailed in its data sheet. Both chips will be covered here in the details that relate to the GPIB controller.

The next section of this application note presents an overview of the GPIB in a tutorial, but comprehensive nature. The knowledgeable reader may wish to skip this section; however, certain basic semantic concepts introduced there will be used throughout this note.

Additional sections cover the view of the 8292 from the CPU's data bus, the interaction of the 3 chip types (8291, 8292, 8293), the 8292's software protocol and the system level hardware/software protocol. A brief description of interrupts and DMA will be followed by an application example. Appendix A contains the source code for the system driver software.

GPIB/IEEE 488 OVERVIEW

Design Objectives

WHAT IS THE IEEE 488 (GPIB)?

The experience of designing systems for a variety of applications in the early 1970's caused Hewlett-Packard to define a standard intercommunication mechanism which would allow them to easily assemble instrumentation systems of varying degrees of complexity. In a typical situation each instrument designer designed his/her own interface from scratch. Each one was inconsistent in terms of electrical levels, pin-outs on a connector, and types of connectors. Every time they built a system they had to invent new cables and new documentation just to specify the cabling and interconnection procedures.

Based on this experience, Hewlett-Packard began to define a new interconnection scheme. They went further than that, however, for they wanted to specify the typical communication protocol for systems of instruments. So in 1972, Hewlett-Packard came out with the first version of the bus which since has been modified and standardized by a committee of several manufacturers, coordinated through the IEEE, to perfect what is now known as the IEEE 488 Interface Bus (also known as the HPIB, the GPIB and the IEC bus). While this bus specification may not be perfect, it is a good compromise of the various desires and goals of instrumentation and computer peripheral manufacturers to produce a common interconnection mechanism. It fits most instrumentation systems in use today and also fits very well the microcomputer I/O bus requirements. The basic design objectives for the GPIB were to:

1) Specify a system that is easy to use, but has all of the terminology and the definitions related to that system precisely spelled out so that everyone uses the same language when discussing the GPIB.

2) Define all of the mechanical, electrical, and functional interface requirements of a system, yet not define any of the device aspects (they are left up to the instrument designer).

3) Permit a wide range of capabilities of instruments and computer peripherals to use a system simultaneously and not degrade each other's performance.

4) Allow different manufacturers' equipment to be connected together and work together on the same bus.

5) Define a system that is good for limited distance interconnections.

6) Define a system with minimum restrictions on performance of the devices.

7) Define a bus that allows asynchronous communication with a wide range of data rates.

8) Define a low cost system that does not require extensive and elaborate interface logic for the low cost instruments, yet provides higher capability for the higher cost instruments if desired.

9) Allow systems to exist that do not need a central controller; that is, communication directly from one instrument to another is possible.

Although the GPIB was originally designed for instrumentation systems, it became obvious that most of these systems would be controlled by a calculator or computer. With this in mind several modifications were made to the original proposal before its final adoption as an international standard. Figure I lists the salient characteristics of the GPIB as both an instrumentation bus and as a computer I/O bus.
Data Rate

1M bytes/s, max
250k bytes/s, typ

Multiple Devices

15 devices, max (electrical limit)
8 devices, typ (interrupt flexibility)

Bus Length

20 m, max
2 m/device, typ

Byte Oriented

8-bit commands
8-bit data

Block Multiplexed

Optimum strategy on GPIB due to
setup overhead for commands

Interrupt Driven

Serial poll (slower devices)
Parallel poll (faster devices)

Direct Memory Access

One DMA facility at controller
serves all devices on bus

Asynchronous

One talker
Multiple listeners } 3-wire handshake
I/O to I/O Transfers
Talker and listeners need not
include microcomputer/controller

Figure 1. Major Characteristics of GPIB as Microcomputer I/O Bus

The bus can be best understood by examining each of these characteristics from the viewpoint of a general microcomputer I/O bus.

Data Rate—Most microcomputer systems utilize peripherals of differing operational rates, such as floppy discs at 31k or 62k bytes/s (single or double density), tape cassettes at 5k to 10k bytes/s, and cartridge tapes at 40k to 80k bytes/s. In general, the only devices that need high speed I/O are 0.5" (1.3-cm) magnetic tapes and hard discs, operational at 30k to 781k bytes/s, respectively. Certainly, the 250k-bytes/s data rate that can be easily achieved by the IEEE 488 bus is sufficient for microcomputers and their peripherals, and is more than needed for typical analog instruments that take only a few readings per second. The 1M-byte/s maximum data rate is not easily achieved on the GPIB and requires special attention to considerations beyond the scope of this note. Although not required, data buffering in each device will improve the overall bus performance and allow utilization of more of the bus bandwidth.

Multiple Devices—Many microcomputer systems used as computers (not as components) service from three to seven peripherals. With the GPIB, up to 8 devices can be handled easily by 1 controller; with some slowdown in interrupt handling, up to 15 devices can work together. The limit of 8 is imposed by the number of unique parallel poll responses available; the limit of 15 is set by the electrical drive characteristics of the bus. Logically, the IEEE 488 Standard is capable of accommodating more device addresses (31 primary, each potentially with 31 secondaries).

Bus Length—Physically, the majority of microcomputer systems fit easily on a desk top or in a standard 19" (48-cm) rack, eliminating the need for extra long cables. The GPIB is designed typically to have 2m of length per device, which accommodates most systems. A line printer might require greater cable lengths, but this can be handled at the lower speeds involved by using extra dummy terminations.

Byte Oriented—The 8-bit byte is almost universal in I/O applications; even 16-bit and 32-bit computers use byte transfers for most peripherals. The 8-bit byte matches the ASCII code for characters and is an integral submultiple of most computer word sizes. The GPIB has an 8-bit wide data path that may be used to transfer ASCII or binary data, as well as the necessary status and control bytes.

Block Multiplexed—Many peripherals are block oriented or are used in a block mode. Bytes are transferred in a fixed or variable length group; then there is a wait before another group is sent to that device, e.g., one sector of a floppy disc, one line on a printer or type punch, etc. The GPIB is, by nature, a block multiplexed bus due to the overhead involved in addressing various devices to talk and listen. This overhead is less bothersome if it only occurs once for a large number of data bytes (once per block). This mode of operation matches the needs of microcomputers and most of their peripherals. Because of block multiplexing, the bus works best with buffered memory devices.

Interrupt Driven—Many types of interrupt systems exist, ranging from complex, fast, vectored/priority networks to simple polling schemes. The main tradeoff is usually cost versus speed of response. The GPIB has two interrupt protocols to help span the range of applications. The first is a single service request (SRQ) line that may be asserted by all interrupting devices. The controller then polls all devices to find out which wants service. The polling mechanism is well defined and can
be easily automated. For higher performance, the parallel poll capability in the IEEE 488 allows up to eight devices to be polled at once—each device is assigned to one bit of the data bus. This mechanism provides fast recognition of an interrupting device. A drawback is the frequent need for the controller to explicitly conduct a parallel poll, since there is no equivalent of the SRQ line for this mode.

Direct Memory Access (DMA)—In many applications, no immediate processing of I/O data on a byte-by-byte basis is needed or wanted. In fact, programmed transfers slow down the data transfer rate unnecessarily in these cases, and higher speed can be obtained using DMA. With the GPIB, one DMA facility at the controller serves all devices. There is no need to incorporate complex logic in each device.

Asynchronous Transfers—An asynchronous bus is desirable so that each device can transfer at its own rate. However, there is still a strong motivation to buffer the data at each device when used in large systems in order to speed up the aggregate data rate on the bus by allowing each device to transfer at top speed. The GPIB is asynchronous and uses a special 3-wire handshake that allows data transfers from one talker to many listeners.

I/O to I/O Transfers—In practice, I/O to I/O transfers are seldom done due to the need for processing data and changing formats or due to mismatched data rates. However, the GPIB can support this mode of operation where the microcomputer is neither the talker nor one of the listeners.

Figure 2. Interface Capabilities and Bus Structure
GPIB Signal Lines

DATA BUS

The lines DIO1 through DIO8 are used to transfer addresses, control information and data. The formats for addresses and control bytes are defined by the IEEE 488 standard (see Appendix C). Data formats are undefined and may be ASCII (with or without parity) or binary. DIO1 is the Least Significant bit (note that this will correspond to bit 0 on most computers).

MANAGEMENT BUS

ATN—Attention. This signal is asserted by the Controller to indicate that it is placing an address or control byte on the Data Bus. ATN is de-asserted to allow the assigned Talker to place status or data on the Data Bus. The Controller regains control by reasserting ATN; this is normally done synchronously with the handshake to avoid confusion between control and data bytes.

EOI—End or Identify. This signal has two uses as its name implies. A talker may assert EOI simultaneously with the last byte of data to indicate end of data. The Controller may assert EOI along with ATN to initiate a Parallel Poll. Although many devices do not use Parallel Poll, all devices should use EOI to end transfers (many currently available ones do not).

SRQ—Service Request. This line is like an interrupt: it may be asserted by any device to request the Controller to take some action. The Controller must determine which device is asserting SRQ by conducting a Serial Poll at its earliest convenience. The device deasserts SRQ when polled.

IFC—Interface Clear. This signal is asserted only by the System Controller in order to initialize all device interfaces to a known state. After deasserting IFC, the System Controller is the active controller of the system.

REN—Remote Enable. This signal is asserted only by the System Controller. Its assertion does not place devices into Remote Control mode; REN only enables a device to go remote when addressed to listen. When in Remote, a device should ignore its front panel controls.

TRANSFER BUS

NRFD—Not Ready For Data. This handshake line is asserted by a listener to indicate it has not yet accepted the data or control byte on the DIO lines. Note that the Controller will not see NRFD deasserted (i.e., data accepted) until all devices have deasserted NRFD.

NDAC—Not Data Accepted. This handshake line is asserted by a Listener to indicate it has not yet accepted the data or control byte on the DIO lines. Note that the Controller will not see NDAC deasserted (i.e., data accepted) until all devices have deasserted NDAC.

DAV—Data Valid. This handshake line is asserted by the Talker to indicate that a data or control byte has been placed on the DIO lines and has had the minimum specified settling time.

Figure 3. GPIB Handshake Sequence

GPIB Interface Functions

There are ten (10) interface functions specified by the IEEE 488 standard. Not all devices will have all functions and some may only have partial subsets. The ten functions are summarized below with the relevant section number from the IEEE document given at the beginning of each paragraph. For further information please see the IEEE standard.

1) SH—Source Handshake (section 2.3). This function provides a device with the ability to properly transfer data from a Talker to one or more Listeners using the three handshake lines.

2) AH—Accepter Handshake (section 2.4). This function provides a device with the ability to properly receive data from the Talker using the three handshake lines. The AH function may also delay the beginning (NRFD) or end (NDAC) of any transfer.

3) T—Talker (section 2.5). This function allows a device to send status and data bytes when addressed to talk. An address consists of one (Primary) or two (Primary and Secondary) bytes. The latter is called an extended Talker.
4) \textit{L}—Listener (section 2.6). This function allows a device to receive data when addressed to listen. There can be extended Listeners (analogous to extended Talkers above).

5) \textit{SR}—Service Request (section 2.7). This function allows a device to request service (interrupt) the Controller. The SRQ line may be asserted asynchronously.

6) \textit{RL}—Remote Local (section 2.8). This function allows a device to be operated in two modes: Remote via the GPIB or Local via the manual front panel controls.

7) \textit{PP}—Parallel Poll (section 2.9). This function allows a device to present one bit of status to the Controller-in-charge. The device need not be addressed to talk and no handshake is required.

8) \textit{DC}—Device Clear (section 2.10). This function allows a device to be cleared (initialized) by the Controller. Note that there is a difference between DC (device clear) and the IFC line (interface clear).

9) \textit{DT}—Device Trigger (section 2.11). This function allows a device to have its basic operation started either individually or as part of a group. This capability is often used to synchronize several instruments.

10) \textit{C}—Controller (section 2.12). This function allows a device to send addresses, as well as universal and addressed commands to other devices. There may be more than one controller on a system, but only one may be the controller-in-charge at any one time.

At power-on time the controller that is hardwired to be the System Controller becomes the active controller-in-charge. The System Controller has several unique capabilities including the ability to send Interface Clear (IFC—clears all device interfaces and returns control to the System Controller) and to send Remote Enable (REN—allows devices to respond to bus data once they are addressed to listen). The System Controller may optionally Pass Control to another controller, if the system software has the capability to do so.

**GPIB Connector**

The GPIB connector is a standard 24-pin industrial connector such as Cinch or Amphenol series 57 Micro-Ribbon. The IEEE standard specifies this connector, as well as the signal connections and the mounting hardware.

The cable has 16 signal lines and 8 ground lines. The maximum length is 20 meters with no more than two meters per device.

**GPIB Signal Levels**

The GPIB signals are all TTL compatible, low true signals. A signal is asserted (true) when its electrical voltage is less than 0.5 volts and is deasserted (false) when it is greater than 2.4 volts. Be careful not to become confused with the two handshake signals, NRFD and NDAC which are also low true (i.e. > 0.5 volts implies the device is Not Ready For Data).

The Intel 8293 GPIB transceiver chips ensure that all relevant bus driver/receiver specifications are met. Detailed bus electrical specifications may be found in Section 3 of the IEEE Std 488-1978. The Standard is the ultimate reference for all GPIB questions.

**GPIB Message Protocols**

The GPIB is a very flexible communications medium and as such has many possible variations of protocols. To bring some order to the situation, this section will discuss a protocol similar to the one used by Ziatech's ZT80 GPIB controller for Intel's MULTIBUS\textsuperscript{TM} computers. The ZT80 is a complete high-level interface processor that executes a set of high level instructions that map directly into GPIB actions. The sequences of commands, addresses and data for these instructions provide a good example of how to use the GPIB (additional information is available in the ZT80 Instruction Manual). The 'null' at the end of each instruction is for cosmetic use to remove previous information from the DIO lines.
**DATA**—Transfer a block of data from device A to devices B, C . . .

1) Device A Primary (Talk) Address
   Device A Secondary Address (if any)
2) Universal Unlisten
3) Device B Primary (Listen) Address
   Device B Secondary Address (if any)
   Device C Primary (Listen) Address
   etc.
4) First Data Byte
   Second Data Byte
   .
   .
   Last Data Byte (EOI)
5) Null

**TRIGR**—Trigger devices A, B . . . to take action

1) Universal Unlisten
2) Device A Primary (Listen) Address
   Device A Secondary Address (if any)
   Device B Primary (Listen) Address
   Device B Secondary Address (if any)
   etc.
3) Group Execute Trigger
4) Null

**PSCTL**—Pass control to device A

1) Device A Primary (Talk) Address
   Device A Secondary Address (if any)
2) Talk Control
3) Null

**CLEAR**—Clear all devices

1) Device Clear
2) Null

**REMAL**—Remote Enable

1) Assert REN continuously

**GOREM**—Put devices A, B, . . . into Remote

1) Assert REN continuously
2) Device A Primary (Listen) Address
   Device A Secondary Address (if any)
   Device B Primary (Listen) Address
   Device B Secondary Address (if any)
   etc.
3) Null

**GOLOC**—Put devices A, B, . . . into Local

1) Device A Primary (Listen) Address
   Device A Secondary Address (if any)
   Device B Primary (Listen) Address
   Device B Secondary Address (if any)
   etc.
2) Go To Local
3) Null

**LOCAL**—Reset all devices to Local

1) Stop asserting REN

**LLKAL**—Prevent all devices from returning to Local

1) Local Lock Out
2) Null

**SPOLL**—Conducts a serial poll of devices A, B, . . .

1) Serial Poll Enable
2) Universal Unlisten
3) ZT 80 Primary (Listen) Address
   ZT 80 Secondary Address
4) Device Primary (Talk) Address
   Device Secondary Address (if any)
5) Status byte from device
6) Go to Step 4 until all devices on list have been polled
7) Serial Poll Disable
8) Null

**PPUAL**—Unconfigure and disable Parallel Poll response from all devices

1) Parallel Poll Unconfigure
2) Null

**ENAPP**—Enable Parallel Poll response in devices A, B, . . .

1) Universal Unlisten
2) Device Primary (Listen) Address
   Device Secondary Address (if any)
3) Parallel Poll Configure
4) Parallel Poll Enable
5) Go to Step 2 until all devices on list have been configured.
6) Null

**DISPP**—Disable Parallel Poll response from devices A, B, . . .

1) Universal Unlisten
2) Device Primary (Listen) Address
   Device Secondary Address (if any)
   Device B Primary (Listen) Address
   Device B Secondary Address (if any)
   etc.
3) Disable Parallel Poll
4) Null

This Ap Note will detail how to implement a useful subset of these controller instructions.
HARDWARE ASPECTS OF THE SYSTEM

8291 GPIB Talker/Listener

The 8291 is a custom designed chip that implements many of the non-controller GPIB functions. It provides hooks so the user's software can implement additional features to complete the set. This chip is discussed in detail in its data sheet. The major features are summarized here:

- Designed to interface microprocessors to the GPIB
- Complete Source and Acceptor Handshake
- Complete Talker and Listener Functions with extended addressing
- Service Request, Parallel Poll, Device Clear, Device Trigger, Remote/Local functions
- Programmable data transfer rate
- Maskable interrupts
- On-chip primary and secondary address recognition
- 1–8 MHz clock range
- 16 registers (8 read, 8 write) for CPU interface
- DMA handshake provision
- Trigger output pin
- On-chip EOS (End of Sequence)

The pinouts and block diagram are shown in Figure 5. One of eight read registers is for data transfer to the CPU; the other seven allow the microprocessor to monitor the GPIB states and various bus and device conditions. One of the eight write registers is for data transfer from the CPU; the other seven control various features of the 8291.

The 8291 interface functions will be software configured in this application example to the following subsets for use with the 8292 as a controller that does not pass control. The 8291 is used only to provide the handshake logic and to send and receive data bytes. It is not acting as a normal device in this mode, as it never sees ATN asserted.

SH1 Source Handshake
AH1 Acceptor Handshake
T3 Basic Talk-Only
L1 Basic Listen-Only
SR0 No Service Requests
RL0 No Remote/Local
PP0 No Parallel Poll Response
DC0 No Device Clear
DT0 No Device Trigger

If control is passed to another controller, the 8291 must be reconfigured to act as a talker/listener with the following subsets:

SH1 Source Handshake
AH1 Acceptor Handshake
T5 Basic Talker and Serial Poll
L3 Basic Listener
SR1 Service Requests
RL1 Remote/Local with Lockout
PP2 Reconfigured Parallel Poll
DC1 Device Clear
DT1 Device Trigger
C0 Not a Controller

![Figure 5. 8291 Pin Configuration and Block Diagram](image-url)
Most applications do not pass control and the controller is always the system controller (see 8292 commands below).

**8292 GPIB Controller**

The 8292 is a preprogrammed Intel 8051A that provides the additional functions necessary to implement a GPIB controller when used with an 8291 Talker/Listener. The 8041A is documented in both a user's manual and in AP-41. The following description will serve only as an outline to guide the later discussion.

The 8292 acts as an intelligent slave processor to the main system CPU. It contains a processor, memory, I/O and is programmed to perform a variety of tasks associated with GPIB controller operation. The on-chip RAM is used to store information about the state of the Controller function, as well as a variety of local variables, the stack and certain user status information. The timer/counter may be optionally used for several time-out functions or for counting data bytes transferred. The I/O ports provide the GPIB control signals, as well as the ancillary lines necessary to make the 8291, 2, 3 work together.

The 8292 is closely coupled to the main CPU through three on-chip registers that may be independently accessed by both the master and the 8292 (UPI-41A). Figure 6 shows this Register Interface. Also refer to Figure 12.

The status register is used to pass Interrupt Status information to the master CPU (A0 = 1 on a read).

The DBBOUT register is used to pass one of five other status words to the master based on the last command written into DBBIN. DBBOUT is accessed when A0 = 0 on a Read. The five status words are Error Flag, Controller Status, GPIB Status, Event Counter Status or Time Out Status.

DBBIN receives either commands (A0 = 1 on a Write) or command related data (A0 = 0 on a write) from the master. These command related data are Interrupt Mask, Error Mask, Event Counter or Time Out.

**8293 GPIB Transceivers**

The 8293 is a multi-use HMOS chip that implements the IEEE 488 bus transceivers and contains the additional logic required to make the 8291 and 8292 work together. The two option strapping pins are used to internally configure the chip to perform the specialized gating required for use with 8291 as a device or with 8291/92 as a controller.

In this application example the two configurations used are shown in Figure 7a and 7b. The drivers are set to open collector or three state mode as required and the special logic is enabled as required in the two modes.
Figure 7

8291/2/3 Chip Set

Figure 8 shows the four chips interconnected with the special logic explicitly shown.

The 8291 acts only as the mechanism to put commands and addresses on the bus while the 8292 is asserting ATN. The 8291 is tricked into believing that the ATN line is not asserted by the ATN2 output of the ATN transceiver and is placed in Talk-only mode by the CPU. The 8291 then acts as though it is sending data, when in reality it is sending addresses and/or commands. When the 8292 deasserts ATN, the CPU software must place the 8291 in Listen-only, Talk-only or Idle based on the implicit knowledge of how the controller is going to participate in the data transfer. In other words, the 8291 does not respond directly to addresses or commands that it sends on the bus on behalf of the Controller. The user software, through the use of Listen-only or Talk-only, makes the 8291 behave as though it were addressed.

Although it is not a common occurrence, the GPIB specification allows the Controller to set up a data transfer between two devices and not directly participate in the exchange. The controller must know when to go active again and regain control. The chip set accomplishes this through use of the “Continuous Accept Handler Handshake cycling mode” and the ability to detect EOI or EOS at the end of the transfer. See XFER in the Software Driver Outline below.

If the 8292 is not the System Controller as determined by the signal on its SYC pin, then it must be able to respond to an IFC within 100 μsec. This is accomplished by the cross-coupled NORs in Figure 7a which deassert the 8293’s internal version of CIC (Not Controller-in-Charge). This condition is latched until the 8292’s firmware has received the IFCL (interface clear received latch) signal by testing the IFCL input. The firmware then sets its signals to reflect the inactive condition and clears the 8293’s latch.
Figure 8. Talker/Listener/Controller
In order for the 8292 to conduct a Parallel Poll the 8291 must be able to capture the PP response on the DIO lines. The only way to do this is to fool the 8291 by putting it into Listen-only mode and generating a DAV condition. However, the bus spec does not allow a DAV during Parallel Poll, so the back-to-back 3-state buffers (see Figure 7b) in the 8293 isolate the bus and allow the 8292 to generate a local DAV for this purpose. Note that the 8291 cannot assert a Parallel Poll response. When the 8292 is not the controller-in-charge the 8291 may respond to PPs and the 8293 guarantees that the DIO drivers are in “open collector” mode through the OR gate (Figure 7b).

ZT7488/18 GPIB Controller

Ziatech’s GPIB Controller, the ZT7488/18 will be used as the controller hardware in this Application Note. The controller consists of an 8291, 8292, an 8 bit input port and TTL logic equivalent to that shown in Figure 8. Figure 9 shows the card’s block diagram. The ZT7488/18 plugs into the STD bus, a 56 pin 8 bit microprocessor oriented bus. An 8085 CPU card is also available on the STD bus and will be used to execute the driver software.

The 8291 uses I/O Ports 60H to 67H and the 8292 uses I/O Ports 68H and 69H. The five interrupt lines are connected to a three-state buffer at I/O Port 6FH to facilitate polling operation. This is required for the TCI, as it cannot be read internally in the 8292. The other three 8229 lines (SPI, IBF, OBF) and the 8291’s INT line are also connected to minimize the number of I/O reads necessary to poll the devices.

NDAC is connected to COUNT on the 8292 to allow byte counting on data transfers. The example driver software will not use this feature, as the software is simpler and faster if an internal 8085 register is used for counting in software.
### READ REGISTERS

<table>
<thead>
<tr>
<th>B10</th>
<th>B9</th>
<th>B8</th>
<th>B7</th>
<th>B6</th>
<th>B5</th>
<th>B4</th>
<th>B3</th>
<th>B2</th>
<th>B1</th>
<th>B0</th>
<th>B11</th>
<th>B12</th>
<th>B13</th>
<th>B14</th>
<th>B15</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPT</td>
<td>APT</td>
<td>GET</td>
<td>END</td>
<td>DEC</td>
<td>ERR</td>
<td>BO</td>
<td>BI</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### WRITE REGISTERS

<table>
<thead>
<tr>
<th>B10</th>
<th>B9</th>
<th>B8</th>
<th>B7</th>
<th>B6</th>
<th>B5</th>
<th>B4</th>
<th>B3</th>
<th>B2</th>
<th>B1</th>
<th>B0</th>
<th>B11</th>
<th>B12</th>
<th>B13</th>
<th>B14</th>
<th>B15</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPT</td>
<td>APT</td>
<td>GET</td>
<td>END</td>
<td>DEC</td>
<td>ERR</td>
<td>BO</td>
<td>BI</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### INTERRUPT STATUS 1

| INT | SPAS | LLO | REM | SPASC | LLOC | REMC | ADSC |

### INTERRUPT STATUS 2

| S8 | SRQS | S6 | S5 | S4 | S3 | S2 | S1 |

### SERIAL POLL STATUS

| ION | ION | EOI | LPAS | TPAS | LA | TA | MJMN |

### ADDRESS STATUS

| CPT7 | CPT6 | CPT5 | CPT4 | CPT3 | CPT2 | CPT1 | CPT0 |

### COMMAND PASS THROUGH

| X | DT0 | DL0 | AD5-0 | AD4-0 | AD3-0 | AD2-0 | AD1-0 |

### ADDRESS 0

| X | DT1 | DL1 | AD5-1 | AD4-1 | AD3-1 | AD2-1 | AD1-1 |

### ADDRESS 1

| X | DT0 | DL0 | AD5-0 | AD4-0 | AD3-0 | AD2-0 | AD1-0 |

### INTERRUPT MASK 1

| 0 | 0 | DMA0 | DMA1 | SPASC | LLOC | REMC | ADSC |

### INTERRUPT MASK 2

| S8 | rsV | S6 | S5 | S4 | S3 | S2 | S1 |

### SERIAL POLL MODE

| TO | LO | 0 | 0 | 0 | 0 | ADM1 | ADM0 |

### ADDRESS MODE

| CNT2 | CNT1 | CNT0 | COM4 | COM3 | COM2 | COM1 | COM0 |

### AUX MODE

| ARS | DT | DL | AD5 | AD4 | AD3 | AD2 | AD1 |

### ADDRESS 0/1

| EC7 | EC6 | EC5 | EC4 | EC3 | EC2 | EC1 | EC0 |

---

**Figure 10. 8291 Registers**

---

**Figure 11. DMA/Interrupt GPIB Controller Block Diagram**

3-107
The application example will not use DMA or interrupts; however, the Figure 11 block diagram includes these features for completeness.

The 8257-5 DMA chip can be used to transfer data between the RAM and the 8291 Talker/Listener. This mode allows a faster data rate on the GPIB and typically will depend on the 8291’s EOS or EOI detection to terminate the transfer. The 8259-5 interrupt controller is used to vector the five possible interrupts for rapid software handling of the various conditions.

### 8292 COMMAND DESCRIPTION

This section discusses each command in detail and relates them to a particular GPIB activity. Recall that although the 8041A has only two read registers and one write register, through the magic of on-chip firmware the 8292 appears to have six read registers and five write registers. These are listed in Figure 12. Please see the 8292 data sheet for detailed definitions of each register. Note the two letter mnemonics to be used in later discussions. The CPU must not write into the 8292 while IBF (Input Buffer Full) is a one, as information will be lost.

**Direct Commands**

Both the Interrupt Mask (IM) and the Error Mask (EM) register may be directly written with the LSB of the address bus (A0) a “0”. The firmware uses the MSB of the data written to differentiate between IM and EM.

**LOAD INTERRUPT MASK**

This command loads the Interrupt Mask with D7–D0. Note that D7 must be a “1” and that interrupts are enabled by a corresponding “1” bit in this register. IFC interrupt cannot be masked off; however, when the 8292 is the System Controller, sending an ABORT command will not cause an IFC interrupt.

<table>
<thead>
<tr>
<th>READ FROM 8292 INTERRUPT STATUS</th>
<th>PORT #</th>
<th>WRITE TO 8292 COMMAND FIELD</th>
</tr>
</thead>
<tbody>
<tr>
<td>SYC</td>
<td>ERR</td>
<td>SRQ</td>
</tr>
<tr>
<td>D7</td>
<td>ERROR FLAG*</td>
<td>D0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CSBS</td>
<td>CA</td>
<td>X</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>REN</td>
<td>DAV</td>
<td>EOI</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D</td>
<td>D</td>
<td>D</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D</td>
<td>D</td>
<td>D</td>
</tr>
</tbody>
</table>

*Note: These registers are accessed by a special utility command.

Figure 12. 8292 Registers
LOAD ERROR MASK

This command loads the Error Mask with D7–D0. Note that D7 must be a zero and that interrupts are enabled by a corresponding “1” bit in this register.

Utility Commands

These commands are used to read or write the 8292 registers that are not directly accessible. All utility commands are written with A0 = 1, D7 = D6 = D5 = 1, D4 = 0. D3–D0 specify the particular command. For writing into registers the general sequence is:
1) wait for IBF = 0 in Interrupt Status Register
2) write the appropriate command to the 8292,
3) write the desired register value to the 8292 with A0 = 1 with no other writes intervening,
4) wait for indication of completion from 8292 (IBF = 0).

For reading a register the general sequence is:
1) wait for IBF = 0 in Interrupt Status Register
2) write the appropriate command to the 8292
3) wait for a TCI (Task Complete Interrupt)
4) Read the value of the accessed register from the 8292 with A0 = 0.

*WEVC*—Write to Event Counter
(Command = 0E2H)

The byte written following this command will be loaded into the event counter register and event counter status for byte counting. The internal counter is incremented on a high to low transition of the COUNT (T1) input. In this application example NDAC is connected to count. The counter is an 8 bit register and therefore can count up to 256 bytes (writing 0 to the EC implies a count of 256). If longer blocks are desired, the main CPU must handle the interrupts every 256 counts and carefully observe the timing constraints.

Because the counter has a frequency range from 0 to 133 kHz when using a 6 MHz crystal, this feature may not be usable with all devices on the GPIB. The 8291 can easily transfer data at rates up to 250 kHz and even faster with some tuning of the system. There is also a 500 ns minimum high time requirement for COUNT which can potentially be violated by the 8291 in continuous acceptor handshake mode (i.e., TDVND1 + TDVND2 – C = 350 + 350 = 700 max). When cable delays are taken into consideration, this problem will probably never occur.

When the 8292 has completed the command, IBF will become a “0” and will cause an interrupt if masked on.

*WTOUT*—Write to Time Out Register
(Command = 0E1H)

The byte written following this command will be used to determine the number of increments used for the time out functions. Because the register is 8 bits, the maximum time out is 256 time increments. This is probably enough for most instruments on the GPIB but is not enough for a manually stepped operation using a GPIB logic analyzer like Ziatech’s ZT488. Also, the 488 Standard does not set a lower limit on how long a device may take to do each action. Therefore, any use of a time out must be able to be overridden (this is a good general design rule for service and debugging considerations).

The time out function is implemented in the 8292’s firmware and will not be an accurate time. The counter counts backwards to zero from its initial value. The function may be enabled/disabled by a bit in the Error mask register. When the command is complete IBF will be set to a “0” and will cause an interrupt if masked on.

*REVC*—Read Event Counter Status
(Command = 0E3H)

This command transfers the content of the Event Counter to the DBBOUT register. The firmware then sets TCI = 1 and will cause an interrupt if masked on. The CPU may then read the value from the 8292 with A0 = 0.

*RINM*—Read Interrupt Mask Register
(Command = 0E5H)

This command transfers the content of the Interrupt Mask register to the DBBOUT register. The firmware sets TCI = 1 and will cause an interrupt if masked on. The CPU may then read the value.

*RERM*—Read Error Mask Register
(Command = 0EAH)

This command transfers the content of the Error Mask register to the DBBOUT register. The firmware sets TCI = 1 and will cause an interrupt if masked on. The CPU may then read the value.

*RCST*—Read Controller Status Register
(Command = 0E6H)

This command transfers the content of the Error Mask register to the DBBOUT register. The firmware sets TCI = 1 and will cause an interrupt if masked on. The CPU may then read the value.
This command transfers the content of the Controller Status register to the DBBOUT register. The firmware sets TCI = 1 and will cause an interrupt if masked on. The CPU may then read the value.

**RTOU**—Read Time Out Status Register (Command = 0E9H)

This command transfers the content of the Time Out Status register to the DBBOUT register. The firmware sets TCI = 1 and will cause an interrupt if masked on. The CPU may then read the value.

If this register is read while a time-out function is in process, the value will be the time remaining before time-out occurs. If it is read after a time-out, it will be zero. If it is read when no time-out is in process, it will be the last value reached when the previous timing occurred.

**RBST**—Read Bus Status Register (Command = 0E7H)

This command causes the firmware to read the GPIB management lines, DA\textsuperscript{V} and the SYC pin and place a copy in DBBOUT. TCI is set to "1" and will cause an interrupt if masked on. The CPU may then read the value.

**RERF**—Read Error Flag Register (Command = 0E4H)

This command transfers the content of the Error Flag register to the DBBOUT register. The firmware sets TCI = 1 and will cause an interrupt if masked on. The CPU may then read the value.

This register is also placed in DBBOUT by an IACK command if ERR remains set. TCI is set to "1" in this case also.

**IACK**—Interrupt Acknowledge (Command = A1 A2 A3 A4 1 A5 1 1)

This command is used to acknowledge any combinations of the five SPI interrupts (A1–A5): SYC, ERR, SRQ, EV, and IFCR. Each bit A1–A5 is an individual acknowledgement to the corresponding bit in the Interrupt Status Register. The command clears SPI but it will be set again if all of the pending interrupts were not acknowledged.

If A2 (ERR) is "1", the Error Flag register is placed in DBBOUT and TCI is set. The CPU may then read the Error Flag without issuing an RERF command.

**Operation Commands**

The following diagram (Figure 13) is an attempt to show the interrelationships among the various 8292 Operation Commands. It is not meant to replace the complete controller state diagram in the IEEE Standard.

**RST**—Reset (Command = 0F2H)

This command has the same effect as an external reset applied to the chip's pin #4. The 8292's actions are:

1) All outputs go to their electrical high state. This means that SPI, TCI, OBFI, IBFI, CLTH will be TRUE and all other GPIB signals will be FALSE.
2) The 8292's firmware will cause the above mentioned five signals to go FALSE after approximately 17.5 \(\mu s\) (at 6 MHz).
3) These registers will be cleared: Interrupt Status, Interrupt Mask, Error Mask, Time Out, Event Counter, Error Flag.
4) If the 8292 is the System Controller (SYC is TRUE), then IFC will be sent TRUE for approximately 100 \(\mu s\) and the Controller function will end up in charge of the bus. If the 8292 is not the System Controller then it will end up in an Idle state.
5) TCI will not be set.

**RSTI**—Reset Interrupts (Command = 0F3)

This command clears all pending interrupts and error flags. The 8292 will stop waiting for actions to occur (e.g., waiting for ATN to go FALSE in a TCNTR command or waiting for the proper handshake state in a TCSY command). TCI will not be set.

**ABORT**—Abort all operations and Clear Interface (Command = 0F9H)

If the 8292 is not the System Controller this command acts like a NOP and flags a USER ERROR in the Error Flag Register. No TCI will occur.
If the 8292 is the system Controller then IFC is set TRUE for approximately 100 µs and the 8292 becomes the Controller-in-Charge and asserts ATN. TCI will be set, only if the 8292 was NOT the CIC.

**STCN**—Start Counter Interrupts  
(Command = 0FEH)

Enables the EV Counter Interrupt. TCI will not be set. Note that the counter must be enabled by a GSEC command.

**SPCN**—Stop Counter Interrupts  
(Command = 0FOH)

The 8292 will not generate an EV interrupt when the counter reaches 0. Note that the counter will continue counting. TCI will not be set.

**SREM**—Set Interface to Remote Control  
(Command = 0F8H)

If the 8292 is the System Controller, it will set REN and TCI TRUE. Otherwise it only sets the User Error Flag.

**SLOC**—Set Interface to Local Mode  
(Command = 0F7H)

If the 8292 is the System Controller, it will set REN FALSE and TCI TRUE. Otherwise, it only sets the User Error Flag.

**EXPP**—Execute Parallel Poll  
(Command = 0F5H)

If not Controller-in-Charge, the 8292 will treat this as a NOP and does not set TCI. If it is the Controller-in-Charge then it sets IDY (EOI & ATN) TRUE and generates a local DAV pulse (that never reaches the GPIB because of gates in the 8293). If the 8291 is configured as a listener, it will capture the Parallel Poll Response byte in its data register. TCI is not generated, the CPU must detect the BI (Byte In) from the 8291. The 8292 will be ready to accept another command before the BI occurs; therefore the 8291’s BI serves as a task complete indication.

**GTSB**—Go To Standby (Command = 0F6H)

If the 8292 is not the Controller-in-Charge, it will treat this command as a NOP and does not set TCI TRUE. Otherwise, it goes to Controller Standby State (CSBS), sets ATN FALSE and TCI TRUE. This command is used as part of the Send, Receive, Transfer and Serial Poll System commands (see next section) to allow the addressed talker to send data/status.

If the data transfer does not start within the specified Time-Out, the 8292 sets TOUT2 TRUE in the Error Flag Register and sets SPI (if enabled). The controller continues waiting for a new command. The CPU must decide to wait longer or to regain control and take corrective action.

**GSEC**—Go To Standby and Enable Counting  
(Command = 0F4H)

This command does the same things as GTSB but also initializes the event counter to the value previously stored in the Event Counter Register (default value is 256) and enables the counter. One may wire the count input to NDAC to count bytes. When the counter reaches zero, it sets EV (and SPI if enabled) in Interrupt Status and will set EV every 256 bytes thereafter. Note that there is a potential loss of count information if the CPU does not respond to the EV/SPI before another 256 bytes have been transferred. TCI will be set at the end of the command.

**TCSY**—Take Control Synchronously  
(Command = 0FDH)

If the 8292 is not in Standby, it treats this command as a NOP and does not set TCI. Otherwise, it waits for the proper handshake state and sets ATN TRUE. The 8292 will set TOUT3 if the handshake never assumes the correct state and will remain in this command until the handshake is proper or a RSTI command is issued. If the 8292 successfully takes control, it sets TCI TRUE.

This is the normal way to regain control at the end of a Send, Receive, Transfer or Serial Poll System Command. If TCSY is not successful, then the controller must try TCAS (see warning below).

**TCAS**—Take Control Asynchronously  
(Command = 0FCH)

If the 8292 is not in Standby, it treats this command as a NOP and does not set TCI. Otherwise, it waits for the proper handshake state and sets ATN TRUE. The 8292 will set TOUT3 if the handshake never assumes the correct state and will remain in this command until the handshake is proper or a RSTI command is issued. If the 8292 successfully takes control, it sets TCI TRUE.

This command should be used only in emergencies. If TCAS fails, then the System Controller will have to issue an ABORT to clean things up.
**GIDL**—Go to Idle (Command = 0F1H)

If the 8292 is not the Controller in Charge and Active, then it treats this command as a NOP and does not set TCI. Otherwise, it sets ATN FALSE, becomes Not Controller in Charge, and sets TCI TRUE. This command is used as part of the Pass Control System Command.

**TCNTR**—Take (Receive) Control
(Command = 0FAH)

If the 8292 is not Idle, then it treats this command as a NOP and does not set TCI. Otherwise, it waits for the current Controller-in-Charge to set ATN FALSE. If this does not occur within the specified Time Out, the 8292 sets TOUT1 in the Error Flag Register and sets SPI (if enabled). It will not proceed until ATN goes false or it receives an RSTI command. Note that the Controller in Charge must previously have sent this controller (via the 8291’s command pass through register) a Pass Control message. When ATN goes FALSE, the 8292 sets CIC, ATN and TCI TRUE and becomes Active.

**SOFTWARE DRIVER OUTLINE**

The set of system commands discussed below is shown in Figure 14. These commands are implemented in software routines executed by the main CPU.

The following section assumes that the Controller is the System Controller and will not Pass Control. This is a valid assumption for 99 + % of all controllers. It also assumes that no DMA or Interrupts will be used. SYC (System Control Input) should not be changed after Power-on in any system—it adds unnecessary complexity to the CPU’s software.

In order to use polling with the 8292 one must enable TCI but not connect the pin to the CPU’s interrupt pin. TCI must be readable by some means. In this application example it is connected to bit 1 port 6FH on the ZT7488/18. In addition, the other three 8292 interrupt lines and the 8291 interrupt are also on that port (SPI-Bit 2, IBFI-Bit 4, OBFI-Bit 3, 8291 INT-Bit 0).

These drivers assume that only primary addresses will be used on the GPIB. To use secondary addresses, one must modify the test for valid talk/listen addresses (range macro) to include secondaries.

<table>
<thead>
<tr>
<th>INIT</th>
<th>INITIALIZE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Talker/Listener</td>
<td></td>
</tr>
<tr>
<td>SEND</td>
<td>SEND DATA</td>
</tr>
<tr>
<td>RECV</td>
<td>RECEIVE DATA</td>
</tr>
<tr>
<td>XFER</td>
<td>TRANSFER DATA</td>
</tr>
<tr>
<td>Controller</td>
<td></td>
</tr>
<tr>
<td>TRIG</td>
<td>GROUP EXECUTE</td>
</tr>
<tr>
<td>DCLR</td>
<td>DEVICE CLEAR</td>
</tr>
<tr>
<td>SPOL</td>
<td>SERIAL POLL</td>
</tr>
<tr>
<td>PPEN</td>
<td>PARALLEL POLL</td>
</tr>
<tr>
<td>PPDS</td>
<td>PARALLEL POLL</td>
</tr>
<tr>
<td>PPUN</td>
<td>PARALLEL POLL</td>
</tr>
<tr>
<td>PPOL</td>
<td>PARALLEL POLL</td>
</tr>
<tr>
<td>PCTL</td>
<td>PASS CONTROL</td>
</tr>
<tr>
<td>RCTL</td>
<td>RECEIVE CONTROL</td>
</tr>
<tr>
<td>SRQD</td>
<td>SERVICE REQUESTED</td>
</tr>
<tr>
<td>System Controller</td>
<td></td>
</tr>
<tr>
<td>REME</td>
<td>REMOTE ENABLE</td>
</tr>
<tr>
<td>LOCL</td>
<td>LOCAL</td>
</tr>
<tr>
<td>IFCL</td>
<td>ABORT/INTERFACE CLEAR</td>
</tr>
</tbody>
</table>

Figure 14. Software Drive Routines
**Initialization**

8292—Comes up in Controller Active State when SYC is TRUE. The only initialization needed is to enable the TCI interrupt mask. This is done by writing 0A0H to Port 68H.

8291—Disable both the major and minor addresses because the 8291 will never see the 8292’s commands/addresses (refer to earlier hardware discussion). This is done by writing 60H and 0E0H to Port 66H.

Set Address Mode to Talk-only by writing 80H to Port 64H.

---

**INIT:**

| Enable-8292 | ;Set up In. pins for Port 6FH |
| Enable TCI  | ;Task complete must be on |
| Disable major address | ;In controller usage, the 8291 |
| Disable minor address | ;Is set to talk only and/or listen only |
| ton         | ;Talk only is our rest state |
| Clock frequency | ;3 MHz in this ap note example |
| All interrupts off | ;Releases 8291 from init. state |
| Immediate execute pon | |

---

**Talker/Listener Routines**

**SEND DATA**

*SEND* *<listener list pointer>* *<count>* *<EOS>* *<data buffer pointer>*

This system command sends data from the CPU to one or more devices. The data is usually a string of ASCII characters, but may be binary or other forms as well. The data is device-specific.

My Talk Address (MTA) must be output to satisfy the GPIB requirement of only one talker at a time (any other talker will stop when MTA goes out). The MTA is not needed as far as the 8291 is concerned—it will be put into talk-only mode (ton).

Set internal counter to 3 MHz to match the clock input coming from the 8085 by writing 23H to Port 65H. High speed mode for the handshakes will not be used here even though the hardware uses three-state drivers.

No interrupts will be enabled now. Each routine will enable the ones it needs for ease of polling operation. The INT bit may be read through Port 6FH. Clear both interrupt mask registers.

Release the chip’s initialization state by writing 0 to Port 65H.

---

This routine assumes a non-null listener list in that it always sends Univeral Unlisten. If it is desired to send data to the listeners previously addressed, one could add a check for a null list and not send UNL. Count must be 255 or less due to an 8 bit register. This routine also always uses an EOS character to terminate the string output; this could easily be eliminated and rely on the count. Items in brackets () are optional and will not be included in the actual code in Appendix A.

---

3-113
**SEND:**

Output-to-8291 MTA, UNL
Put EOS into 8291
While 2OH ≤ listener ≤ 3EH
  output-to-8291 listener
  Increment listen list pointer
Output-to-8292 GTSB
Enable-8291
Output EOI on EOS sent
If count < > 0 then
  While not (end or count = 0)
    (could check tout 2 here)
    Output-to-8291 data
    Increment data buffer pointer
    Decrement count
Output-to-8292 TCSY
(If tout3 then take control async)
Enable 8291
No output EOI on EOS sent
Return

;We will talk, nobody listen
;End of string compare character
;GPIB listen addresses are
;"space" thru ">" ASCII
;Address all listeners
;8292 stops asserting ATN, go to standby
;Send EOI along with EOS character
;Wait for EOS or end of count
;Optionally check for stuck bus-tout 2
;Output all data, one byte at a time
;8085 CREG will count for us
;8292 asserts ATN, take control sync.
;If unable to take control sync.
;Restore 8291 to standard condition

---

**Figure 15. Flowchart for Receive Ending Conditions**
**RECEIVE DATA**

`RECV <talker> <count> <EOS> <data buffer pointer>`

This system command is used to input data from a device. The data is typically a string of ASCII characters.

This routine is the dual of SEND. It assumes a new talker will be specified, a count of less than 257, and an EOS character to terminate the input. EOI received will also terminate the input. Figure 15 shows the flow chart for the RECV ending conditions. My Listen Address (MLA) is sent to keep the GPIB transactions totally regular to facilitate analysis by a GPIB logic analyzer like the Ziatech ZT488. Otherwise, the bus would appear to have no listener even though the 8291 will be listening.

Note that although the count may go to zero before the transmission ends, the talker will probably be left in a strange state and may have to be cleared by the controller. The count ending of RECV is therefore used as an error condition in most situations.
**RECV:**

Put EOS into 8291
If \( 40H \leq \text{talker} \leq 5EH \) then
Output-to-8291 talker
Increment talker pointer
Output-to-8291 UNL, MLA
Enable-8291
Holdoff on end
End on EOS received
lon, reset ton
Immediate execute pon
Output-to-8292 GTSB
While not (end or count = 0 (or tout2))
Input-from-8291 data
Increment data buffer pointer
Decrement count
(If count = 0 then error)
Output-to-8292 TCSY
(If Tout3 then take control async.)
Enable-8291
No holdoff on end
No end on EOS received
ton, reset lon
Finish handshake
Immediate execute pon
Return error-indicator

;End of string compare character
;GPIB talk addresses are
;"@" thru "^" ASCII
;Do this for consistency's sake
;Everyone except us stop listening
;Stop when EOS character is
;Detected by 8291
;Listen only (no talk)
;8292 stops asserting ATN, go to standby
;wait for EOS or EOI or end of count
;optionally check for stuck bus-tout2
;input data, one byte at a time
;Use 8085 C register as counter
;Count should not occur before end
;8292 asserts ATN take control
;If unable to take control sync.
;Put 8291 back as needed for
;Controller activity and
;Clear holdoff due to end
;Complete holdoff due to end, if any
;Needed to reset lon

---

**Figure 17. RECV from "R"; EOS = 0DH**

**Figure 18. XFER from "^" to "1", "2", "$"; EOS = 0DH**
TRANSFER DATA

XFER <Talker> <Listener list> <EOS>

This system command is used to transfer data from a talker to one or more listeners where the controller does not participate in the transfer of the ASCII data.

This is accomplished through the use of the 8291's continuous acceptor handshake mode while in listen-only.

This routine assumes a device list that has the ASCII talker address as the first byte and the string (one or more) or ASCII listener addresses following. The EOS character or an EOI will cause the controller to take control synchronously and thereby terminate the transfer.

```
XFER:
Output-to-8291: Talker, UNL
While 20H ≤ listen ≤ 3EH
   Output-to-8291: Listener
   Increment listen list pointer
Enable-8291
   lon, no ton
   Continuous AH mode
   End on EOS received
   Immediate execute PON
Put EOS into 8291
Output-to-8292: GTSB
Upon end (or tout2) then
   Take control synchronously
Enable-8291
   Finish handshake
   Not continuous AH mode
   Not END on EOS received
   ton
   Immediate execute pon
Return

;Send talk address and unlisten
;Send listen address
;Controller is pseudo listener
;Handshake but don't capture data
;Capture EOS as well as EOI
;Initialize the 8291
;Set up EOS character
;Go to standby
;8292 waits for EOS or EOI and then
;Regains control
;Go to Ready for Data
```

Controller

GROUP EXECUTE TRIGGER

TRIG <Listener list>

This system command causes a group execute trigger (GET) to be sent to all devices on the listener list. The intended use is to synchronize a number of instruments.

```
TRIG:
Output-to-8291 UNL
While 20H ≤ listen ≤ 3EH
   Output-to-8291 Listener
   Increment listen list pointer
Output-to-8291 GET
Return

;Everybody stop listening
;Check for valid listen address
;Address each listener
;Terminate on any non-valid character
;Issue group execute trigger
```
**DEVICE CLEAR**

*DCLR <Listener list>*

This system command causes a device clear (SDC) to be sent to all devices on the listener list. Note that this is not intended to clear the GPIB interface of the device, but should clear the device-specific logic.

```
DCLR:
Output-to-8291 UNL ;Everybody stop listening
While 20H ≤ listener ≤ 3EH ;Check for valid listen address
Output-to-8291 Listener ;Address each listener
Increment listen list pointer ;Terminate on any non-valid character
Output-to-8291 SDC ;Selective device clear
Return
```

**SERIAL POLL**

*SPOL <Talker list> <status buffer pointer>*

This system command sequentially addresses the designated devices and receives one byte of status from each. The bytes are stored in the buffer in the same order as the devices appear on the talker list. MLA is output for completeness.


**SPOL:**  
Output-to-8291 UNL, MLA, SPE  
While 40H ≤ talker ≤ 5 EH  
Output-to-8291 talker  
Increment talker list pointer  
Enable-8291  
lon, reset ton  
Immediate execute pon  
Output-to-8292 GTSB  
Wait for data in (BI)  
Output-to-8292 TCSY  
Input-from-8291 data  
Increment buffer pointer  
Enable 8291  
ton, reset lon  
Immediate execute pon  
Output-to-8291 SPD  
Return

;Unlisten, we listen, serial poll enable  
;Only one byte of serial poll  
;Status wanted from each talker  
;Check for valid transfer  
;Address each device to talk  
;One at a time  
;Listen only to get status  
;This resets ton  
;Go to standby  
;Serial poll status byte into 8291  
;Take control synchronously  
;Actually get data from 8291  
;Reset lon  
;Send serial poll disable after all devices polled

---

**Figure 21. SPOL “Q”, “R”, “K”, “^”**

**Figure 22. PPEN “2”; iP3P2P1 = 0111B**

**PARALLEL POLL ENABLE**

*PPEN*< Listener list > < Configuration Buffer pointer >

This system command configures one or more devices to respond to Parallel Poll, assuming they implement subset PP1. The configuration information is stored in a buffer with one byte per device in the same order as
devices appear on the listener list. The configuration byte has the format XXXXIP3P2P1 as defined by the IEEE Std. P3P2P1 indicates the bit # to be used for a response and I indicates the assertion value. See Sec. 2.9.3.3 of the Std. for more details.

**PPEN:**

<table>
<thead>
<tr>
<th>Output-to-8291 UNL</th>
<th>;Universal unlisten</th>
</tr>
</thead>
<tbody>
<tr>
<td>While 20H ≤ Listener ≤ 3EH</td>
<td>;Check for valid listener</td>
</tr>
<tr>
<td>Output-to-8291 listener</td>
<td>;Stop old listener, address new</td>
</tr>
<tr>
<td>Output-to-8291 PPC, (PPE or data)</td>
<td>;Send parallel poll info</td>
</tr>
<tr>
<td>Increment listener list pointer</td>
<td>;Point to next listener</td>
</tr>
<tr>
<td>Increment buffer pointer</td>
<td>;One configuration byte per listener</td>
</tr>
<tr>
<td>Return</td>
<td></td>
</tr>
</tbody>
</table>

**PARALLEL POLL DISABLE**

**PPDS<listener list>**

This system command disables one or more devices from responding to a Parallel Poll by issuing a Parallel Poll Disable (PPD). It does not deconfigure the devices.

**PPDS:**

<table>
<thead>
<tr>
<th>Output-to-8291 UNL</th>
<th>;Universal Unlisten</th>
</tr>
</thead>
<tbody>
<tr>
<td>While 20H ≤ Listener ≤ 3EH</td>
<td>;Check for valid listener</td>
</tr>
<tr>
<td>Output-to-8291 listener</td>
<td>;Address listener</td>
</tr>
<tr>
<td>Increment listener list pointer</td>
<td></td>
</tr>
<tr>
<td>Output-to-8291 PPC, PPD</td>
<td>;Disable PP on all listeners</td>
</tr>
<tr>
<td>Return</td>
<td></td>
</tr>
</tbody>
</table>

---

**Figure 23. PPDS “÷”, “+”, “>”**

**Figure 24. PPUN**
PARALLEL POLL UNCONFIGURE

**PPUN**

This system command deconfigures the Parallel Poll response of all devices by issuing a Parallel Poll Unconfigure message.

<table>
<thead>
<tr>
<th>PPUN:</th>
<th>Output-to-8291 PPUn</th>
<th>Unconfigure all parallel poll</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Return</td>
<td></td>
</tr>
</tbody>
</table>

CONDUCT A PARALLEL POLL

**PPOL**

This system command causes the controller to conduct a Parallel Poll on the GPIB for approximately 12.5 μsec (at 6 MHz). Note that a parallel poll does not use the handshake; therefore, to ensure that the device knows whether or not its positive response was observed by the controller, the CPU should explicitly acknowledge each device by a device-dependent data string. Otherwise, the response bit will still be set when the next Parallel Poll occurs. This command returns one byte of status.

<table>
<thead>
<tr>
<th>PPOL:</th>
<th>Enable-8291</th>
<th>Listen only</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>lon</td>
<td>This resets ton</td>
</tr>
<tr>
<td></td>
<td>Immediate execute pon</td>
<td>Execute parallel poll</td>
</tr>
<tr>
<td></td>
<td>Output-to-8292 EXPP</td>
<td>When byte is input</td>
</tr>
<tr>
<td></td>
<td>Upon BI</td>
<td>Read it</td>
</tr>
<tr>
<td></td>
<td>Input-from-8291 data</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Enable-8291 ton</td>
<td>Talk only</td>
</tr>
<tr>
<td></td>
<td>Immediate execute pon</td>
<td>This resets lon</td>
</tr>
<tr>
<td></td>
<td>Return Data (status byte)</td>
<td></td>
</tr>
</tbody>
</table>

PASS CONTROL

**PCTL<talker>**

This system command allows the controller to relinquish active control of the GPIB to another controller. Normally some software protocol should already have informed the controller to expect this, and under what conditions to return control. The 8291 must be set up to become a normal device and the CPU must handle all commands passed through, otherwise control cannot be returned (see Receive Control below). The controller will go idle.

<table>
<thead>
<tr>
<th>PCTL:</th>
<th>If 40H ≤ talker ≤ 5EH then</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>if talker &lt; &gt; MTA then</td>
</tr>
<tr>
<td></td>
<td>output-to-8291 talker, TCT</td>
</tr>
<tr>
<td></td>
<td>Enable-8291</td>
</tr>
<tr>
<td></td>
<td>not ton, not lon</td>
</tr>
<tr>
<td></td>
<td>Immediate execute pon</td>
</tr>
<tr>
<td></td>
<td>My device address, mode 1</td>
</tr>
<tr>
<td></td>
<td>Undefined command pass through</td>
</tr>
<tr>
<td></td>
<td>(Parallel Poll Configuration)</td>
</tr>
<tr>
<td></td>
<td>Output-to-8292 GIDL</td>
</tr>
<tr>
<td></td>
<td>Return</td>
</tr>
</tbody>
</table>

3-121
RECEIVE CONTROL

RCTL

This system command is used to get control back from the current controller-in-charge if it has passed control to this inactive controller. Most GPIB systems do not use more than one controller and therefore would not need this routine.

To make passing and receiving control a manageable event, the system designer should specify a protocol whereby the controller-in-charge sends a data message to the soon-to-be-active controller. This message should give the current state of the system, why control is being passed, what to do, and when to pass control back. Most of these issues are beyond the scope of this Ap Note.
RCTL:
Upon CPT
If (command=TCT) then
  If TA then
    Enable-8291
    Disable major device number
    ton
    Mask off interrupts
    Immediate execute pon
    Output-to-8292 TCNTR
    Enable-8291
    Valid command
    Return valid
  Else
    Enable-8291
    Invalid command
Else
  Enable-8291
  Invalid command
Return invalid

;Wait for command pass through bit in 8291
;if command is take control and
;We are talker addressed
;Controller will use ton and lon
;Talk only mode
;Take (receive) control
;Release handshake
;Not talker addr. so TCT not for us
;Not TCT, so we don't care

Figure 27. RCTL

Figure 28. REME
SERVICE REQUEST

SRQD

This system command is used to detect the occurrence of a Service Request on the GPIB. One or more devices may assert SRQ simultaneously, and the CPU would normally conduct a Serial Poll after calling this routine to determine which devices are SRQing.

\[
\text{SRQD:}\ \\
\text{If SRQ then}\ \\
\text{Output-to-8292 IACK.SRQ}\ \\
\text{Return SRQ}\ \\
\text{Else return no SRQ}
\]

System Controller

REMOTE ENABLE

REME

This system command asserts the Remote Enable line (REN) on the GPIB. The devices will not go remote until they are later addressed to listen by some other system command.

\[
\text{REME:}\ \\
\text{Output-to-8292 SREM}\ \\
\text{Return}
\]

LOCAL

LOCL

This system command deasserts the REN line on the GPIB. The devices will go local immediately.

\[
\text{LOCL:}\ \\
\text{Output-to-8292 SLOC}\ \\
\text{Return}
\]
INTERFACES CLEAR/ABORT

IFCL

This system command asserts the GPIB's Interface Clear (IFC) line for at least 100 microseconds. This causes all interface logic in all devices to go to a known state. Note that the device itself may or may not be reset, too. Most instruments do totally reset upon IFC. Some devices may require a DCLR as well as an IFCL to be completely reset. The (system) controller becomes Controller-in-Charge.

IFCL:
Output-to-8292 ABORT
Return

;8292 asserts Interface Clear
;For 100 microseconds

INTERRUPTS AND DMA CONSIDERATIONS

The previous sections have discussed in detail how to use the 8291, 8292, 8293 chip set as a GPIB controller with the software operating in a polling mode and using programmed transfer of the data. This is the simplest mode of use, but it ties up the microprocessor for the duration of a GPIB transaction. If system design constraints do not allow this, then either Interrupts and/or DMA may be used to free up processor cycles.

The 8291 and 8292 provide sufficient interrupts that one may return to do other work while waiting for such things as 8292 Task Completion, 8291 Next Byte In, 8291 Last Byte Out, 8292 Service Request In, etc. The only difficulty lies in integrating these various interrupt sources and their matching routines into the overall system’s interrupt structure. This is highly situation-specific and is beyond the scope of this Ap Note.

The strategy to follow is to replace each of the WAIT routines (see Appendix A) with a return to the main code and provide for the corresponding interrupt to bring the control back to the next section of GPIB.
code. For example WAITO (Wait for Byte Out of 8291) would be replaced by having the BO interrupt enabled and storing the (return) address of the next instruction in a known place. This co-routine structure will then be activated by a BO interrupt. Figure 31 shows an example of the flow of control.

DMA is also useful in relieving the processor if the average length of a data buffer is long enough to overcome the extra time used to set up a DMA chip. This decision will also be a function of a data rate of the instrument. The best strategy is to use the DMA to handle only the data buffer transfers on SEND and RECV and to do all the addressing and control just as shown in the driver descriptions.

Another major reason for using a DMA chip is to increase the data rate and therefore increase the overall transaction rate. In this case the limiting factor becomes the time used to do the addressing and control of the GPIB using software like that in Appendix A. The data transmission time becomes insignificant at DMA speeds unless extremely long buffers are used.

Refer to Figure 11 for a typical DMA and interrupt based design using the 8291, 8292, 8293. A system like this can achieve a 250K byte transfer rate while under DMA control.

APPLICATION EXAMPLE

This section will present the code required to operate a typical GPIB instrument set up as shown in Figure 32. The HP5328A universal counter and the HP3325 function generator are typical of many GPIB devices; however, there are a wide variety of software protocols to be found on the GPIB. The Ziatech ZT488 GPIB analyzer is used to single step the bus to facilitate debugging the system. It also serves as a training/familiarization aid for newcomers to the bus.

This example will set up the function generator to output a specific waveform, frequency and amplitude. It will then tell the counter to measure the frequency and Request Service (SRQ) when complete. The program will then read in the data. The assembled source code will be found at the end of Appendix A.

![Figure 31. GPIB Interrupt and Co-Routine Flow of Control](image_url)

![Figure 32. GPIB Example Configuration](image_url)
CONCLUSION

This Application Note has shown a structured way to view the IEEE 488 bus and has given typical code sequences to make the Intel 8291, 8292, and 8293’s behave as a controller of the GPIB. There are other ways to use the chip set, but whatever solution is chosen, it must be integrated into the overall system software.

The ultimate reference for GPIB questions is the IEEE Std 488 -1978 which is available from IEEE, 345 East 47th St., New York, NY, 10017. The ultimate reference for the 8292 is the source listing for it (remember it’s a pre-programmed UPI-41A) which is available from IN-SITE, Intel Corp., 3065 Bowers Ave., Santa Clara, CA 95051.
AP-66

APPENDIX A

ISIS-II 8086/8085 MACHO ASSEMBLER, V3.0
GPIB CONTROLLER SUBROUTINES

LOC OJH LINE SOURCE STATEMENT

1  TITLE ('GPIB CONTROLLER SUBROUTINES')
2  ;
3  ; GPIB CONTROLLER SUBROUTINES
4  ;
5  ; for Intel 8291, 8292 on ZT 748A/18
6  ; Bert Forbes, Ziatech Corporation
7  ; 2410 Broad Street
8  ; San Luis Obispo, CA, USA 93401
9  ;
10  ;
11  ; General Definitions & Equates
12  ; R291 Control Values
13  ;
14  ;
15  ; ORG 1000H ; For ZT748A/18 w/8085
16  ;
17  ; PRT91 EQU 6AH ; R291 base port #
18  ;
19  ;
20  ; Req #0 Data in & Data out
21  ;
22  ;
23  ; Req #1 Interrupt 1 Constants
24  ;
25  ;
26  ; Reg 14
27  ;
28  ; Reg IS
29  ;
30  ; Reg #4 Address Mode Constants
31  ;
32  ;
33  ;
34  ;
35  ;
36  ;
37  ;
38  ;
39  ;
40  ;
41  ;
42  ;
43  ;
44  ;
45  ;
46  ;
47  ;
48  ;
49  ;

GPIB CONTROLLER SUBROUTINES

for Intel 8291, 8292 on ZT 748A/18

Ernst Forbes, Ziatech Corporation

2410 Broad Street
San Luis Obispo, CA, USA 93401

General Definitions & Equates
R291 Control Values

ORG 1000H ; For ZT748A/18 w/8085

PRT91 EQU 6AH ; R291 base port #

Req #0 Data in & Data out

Req #1 Interrupt 1 Constants

Reg 14

Reg IS

Reg #4 Address Mode Constants

(Read) Address Status Register

(Write) Auxiliary Mode Register

auxiliary mode register #

3 MHz clock input

231324-30
0083 58 PHSK EQU 03 ; 91 finish handshake command
0084 51 SDEOI EQU 05 ; 91 send EOI with next byte
0085 52 AXRA EQU 05H ; 91 aux. req & pattern
0086 53 HNHSK EQU 01 ; 91 hold off handshake on all bytes
0087 54 HOEND EQU 02 ; 91 hold off handshake on end
0088 55 CANCY EQU 03 ; 91 continuous AH cycling
0089 56 IDEOS EQU 04 ; 91 end on EOS received
008A 57 EDSI EQU 08 ; 91 output EDSI on EOS sent
008B 58 VSCMD EQU 0FH ; 91 valid command pass through
008C 59 VCMD EQU 07H ; 91 invalid command pass through
008D 48 AXRH EQU 0AH ; Aux. req. & pattern
008E 41 CPTEN EQU 01H ; command pass thru enable
62
63 ;
65 ;
66 ;
67 ;
68 ;
69 ;
70 ;
71 ;
72 ;
73 ;
74 ;
75 ;
8292 CONTROL VALUES
76 ;
77 ;
78 ;
79 PRT92 EQU PRT91+8 ; 8292 Base Port 1 (CS7)
80 ;
81 INTMH EQU PRT92+0 ; 92 INTRP Mask Req
82 INTMH EQU 0AH ; TCI
83 ;
84 ERMH EQU PRT92+0 ; 92 Error Mask Req
85 TOUT1 EQU 01 ; 92 Time Out for Pass Control
86 TOUT2 EQU 02 ; 92 Time Out for Standby
87 TOUT3 EQU 04 ; 92 Time Out for Take Control Sync
88 EVREG EQU PRT92+0 ; 92 Event Counter Pseudo Req
89 TOREG EQU PRT92+0 ; 92 Time Out Pseudo Req
90 ;
91 CMD92 EQU PRT92+1 ; 92 Command Register
92 ;
93 INTS T EQU PRT92+1 ; 92 Interrupt Status Reg
94 EVB I T EQU 1AH ; Event Counter Bit
95 IBFBT EQU #2 ; Input Ruffer Full Bit
96 SQBGT EQU 2AH ; Seq bit
97 ;
98 ERL FG EQU PRT92+0 ; 92 Error Flag Pseudo Req
99 CLS T EQU PRT92+0 ; 92 Controller Status Pseudo Reg
9A BUSST EQU PRT92+0 ; 92 GPIB (Bus) Status Pseudo Reg
9B EVGST EQU PRT92+0 ; 92 Event Counter Status Pseudo Reg
9C TOGST EQU PRT92+0 ; 92 Time Out Status Pseudo Reg
9D ;
9E ;
9F ;
A0 ;
A1 ;
A2 ;
A3 ;
A4 ;
A5 ;
A6 ;
A7 ;
A8 ;
A9 ;
AA ;
AB ;
AC ;
AD ;
AE ;
AF ;
B0 ;
B1 ;
B2 ;
B3 ;
B4 ;
B5 ;
B6 ;
B7 ;
B8 ;
B9 ;
BA ;
BB ;
BC ;
BD ;
BE ;
BF ;
C0 ;
C1 ;
C2 ;
C3 ;
C4 ;
C5 ;
C6 ;
C7 ;
C8 ;
C9 ;
CA ;
CB ;
CC ;
CD ;
CE ;
CF ;
D0 ;
D1 ;
D2 ;
D3 ;
D4 ;
D5 ;
D6 ;
D7 ;
D8 ;
D9 ;
DA ;
DB ;
DC ;
DD ;
DE ;
DF ;
E0 ;
E1 ;
E2 ;
E3 ;
E4 ;
E5 ;
E6 ;
E7 ;
E8 ;
E9 ;
EA ;
EB ;
EC ;
ED ;
EE ;
EF ;
F0 ;
F1 ;
F2 ;
F3 ;
F4 ;
F5 ;
F6 ;
F7 ;
F8 ;
F9 ;
FA ;
FB ;
FC ;
FD ;
FE ;
00 ;
01 ;
02 ;
03 ;
04 ;
05 ;
06 ;
07 ;
08 ;
09 ;
0A ;
0B ;
0C ;
0D ;
0E ;
0F ;
10 ;
11 ;
12 ;
13 ;
14 ;
15 ;
16 ;
17 ;
18 ;
19 ;
1A ;
1B ;
1C ;
1D ;
1E ;
1F ;
20 ;
21 ;
22 ;
23 ;
24 ;
25 ;
26 ;
27 ;
28 ;
29 ;
2A ;
2B ;
2C ;
2D ;
2E ;
2F ;
123 ; ; 0292 \text{UTILITY COMMANDS} \\
124 ; \\
125 ; \\
126 \text{WOUT EQU 0E1H} ; Write to timeout reg \\
127 \text{WEVC EQU 0E2H} ; Write to event counter \\
128 \text{REV EQU 0E3H} ; Read event counter status \\
129 \text{RENF EQU 0E4H} ; Read error flag reg \\
130 \text{RIMK EQU 0E5H} ; Read interrupt mask reg \\
131 \text{RCST EQU 0E6H} ; Read controller status reg \\
132 \text{RBST EQU 0E7H} ; Read GPIB bus status reg \\
133 \text{RTOUT EQU 0E8H} ; Read timeout status reg \\
134 \text{RERM EQU 0E9H} ; Read error mask reg \\
135 \text{IACK EQU 0EH} ; Interrupt Acknowledge \\
136 ; \\
137 ; \\
138 ; \\
139 ; \\
140 ; \\
141 ; \\
142 \text{PORT F BIT ASSIGNMENTS} \\
143 ; \\
144 ; \\
145 ; \\
146 ; \\
147 ; \\
148 ; \\
149 ; \\
150 ; \\
151 \text{MDA EQU 1} ; My device address is 1 \\
152 \text{MTA EQU MDA+4} ; My talk address is 1 ("A") \\
153 \text{MLA EQU MDA+28} ; My listen address is 1 ("T") \\
154 \text{UML EQU 3FH} ; Universal unlisten \\
155 \text{GET EQU 0} ; Group Execute Triger \\
156 \text{SDE EQU 04H} ; Device Clear \\
157 \text{SPE EQU 18H} ; Serial poll enable \\
158 \text{SPD EQU 19H} ; Serial poll disable \\
159 \text{PPC EQU 05H} ; Parallel poll configure \\
160 \text{PPD EQU 78H} ; Parallel poll disable \\
161 \text{PPE EQU 58H} ; Parallel poll disable \\
162 \text{PPU EQU 15H} ; Parallel poll unconfigured \\
163 \text{TCT EQU 09} ; Take control (pass control) \\
164 ; \\
165 ; \\
166 ; \\
167 ; \\
168 ; \\
169 \text{SETF MACRO} \\
170 \text{ORA A} \\
171 \text{ENDM} \\
172 ; \\
173 \text{WAITO MACRO} \text{; Wait for last 91 byte to be done} \\
174 \text{LOCAL WAITL} \\
175 \text{WAITL: IN INT1} ; Get INT1 status \\
176 \text{ANI 00H} ; Check for byte out \\
177 \text{JZ WAITL} ; If not, try again \\
178 \text{ENDM} ; Until it is \\
179 ; \\
180 ; \\
181 \text{WAITI MACRO} \text{; Wait for 91 byte to be input} \\
182 \text{LOCAL WAITL} \\
183 \text{WAITL: IN INT1} ; Get INT1 status \\
184 \text{MOV B,A} ; Save status in B \\
185 \text{ANI 00H} ; Check for byte in \\
186 \text{JZ WAITL} ; If not, just try again \\
187 \text{ENDM} ; Until it is \\
188 ; \\
189 \text{WAITX MACRO} \text{; Wait for 92's TCI to go false} \\
190 \text{LOCAL WAITL} \\
191 \text{WAITL: IN PRTF} \\
192 \text{ANI TCF} \\
193 \text{JZ WAITL} \\
194 \text{ENDM} \\
195 ;
AP-66

<table>
<thead>
<tr>
<th>196 WAITT</th>
<th>MACRO</th>
<th>WAITT</th>
</tr>
</thead>
<tbody>
<tr>
<td>197 LOCAL</td>
<td>WAITL:</td>
<td>IN PRTF ;Get task complete int, etc.</td>
</tr>
<tr>
<td>198 ANI</td>
<td>TCIF   ;Mask it</td>
<td></td>
</tr>
<tr>
<td>200 JZ</td>
<td>WAITL  ;Wait for task to be complete</td>
<td></td>
</tr>
<tr>
<td>201 ENDM</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>202 RANGE</th>
<th>MACRO</th>
<th>LOWER, UPPER, LABEL</th>
</tr>
</thead>
<tbody>
<tr>
<td>204 ;Checks for value in range</td>
<td></td>
<td></td>
</tr>
<tr>
<td>205 ;Branches to label if not</td>
<td></td>
<td></td>
</tr>
<tr>
<td>206 ;in range. Falls through if</td>
<td></td>
<td></td>
</tr>
<tr>
<td>207 ;lower &lt;= (L) &lt;= upper.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>208 ;Get next byte.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>209 MOV</th>
<th>A,M</th>
</tr>
</thead>
<tbody>
<tr>
<td>210 CPI</td>
<td>LOWER</td>
</tr>
<tr>
<td>211 JM</td>
<td>LABEL</td>
</tr>
<tr>
<td>212 CPI</td>
<td>UPPER+1</td>
</tr>
<tr>
<td>213 JP</td>
<td>LABEL</td>
</tr>
<tr>
<td>214 ENDM</td>
<td></td>
</tr>
</tbody>
</table>

| 215 ;      |
| 216 CLRA   | MACRO  | A ;A XOR A =0 |
| 217 XRA    | ENDM   |

| 219 ;      |
| 220 ;      |
| 221 ;      |
| 222 ;      |
| 223 ;      |
| 224 ;      |
| 225 ;      |
| 226 ;      |
| 227 ;      |
| 228 ;      |
| 229 ;      |
| 230 ;      |
| 231 ;      |
| 232 ;      |
| 233 ;      |
| 234 ;      |
| 235 ;      |
| 236 ;      |
| 237 ;      |
| 238 ;      |

| 239 ;      |
| 240 ;      |

| 240 ;      |
| 241 ;      |
| 242 ;      |
| 243 ;      |
| 244 ;      |
| 245 ;      |
| 246 ;      |
| 247 ;      |

| 1003 IEA0  | INIT   | MVI A, INTM ;Enable TCI |
| 1002 D348  | OUTPUT | MVI A, DTDI1 ;Disable major talker/listener |
| 1004 364A  | MVI    | A, ADR81 |
| 1050 D36A  | MVI    | A, DTDI2 ;Disable minor talker/listener |
| 103A D35A  | MVI    | A, ADR81 |
| 103C 3E80  | MVI    | A, TON ;Talk only mode |
| 106C D344  | OUT    | ADRWDM |
| 1018 D323  | MVI    | A, CLART ;3 kHz for delay timer |
| 1012 D355  | OUT    | AUXWDM |
| 258 CLRA   |         |
| 1014 AF    |         |
| 1015 D361  |         |
| 1017 D362  |         |
| 1019 D345  |         |
| 1018 C9    |         |

| 259 ;      |
| 260 ;      |
| 261 ;      |
| 262 ;      |
| 263 ;      |

| 264 ;      |
| 265 ;      |

| 266 ;      |
| 267 ;      |
| 268 ;      |

| SEND ROUTINE |

| 269 ;      |
INPUTS:  HL listener list pointer
DE data buffer pointer,
C count-- if will cause no data to be sent
b EOS character-- software detected

OUTPUTS: none

CALLS: none

DESTROYS: A, C, DE, HL, F
184D CA7F10 343 JZ SEND5 ; If char = EOS, go finish
1870 DB61 345+ ??90006: IN INT1 ; Get Int1 status
1872 EA42 346+ ANI BOM ; Check for byte out
1874 CA7B10 347+ JZ ??9006 ; If not, try again
1877 13 348 INX D ; Increment buffer pointer
1878 8D 349 DCR C ; Decrement count
1879 C26910 350 JMP SEND3 ; If count < 0, go send
187C C38810 351 JMP SEND6 ; Else go finish
187F 13 352 SEND5: INX D ; For consistency
1888 0D 353 DCR C ; Send
1889 54 354 WAITO ; This ensures that the standard entry
1881 DB61 355+ ??9007: IN INT1 ; Get Int1 status
1883 EA52 356+ ANI BOM ; Check for byte out
1885 CA8110 357+ JZ ??9997 ; If not, try again
1888 39FD 358 SEND5: MVI A,TCIF ; Take control synchronously
188A D349 359 OUT CMD92
188C 38B8 360 MVI A,XXRA ; Reset send EOI on EOS
188E D365 361 OUT AUX4D
1890 54 362 DCR C ; Wait for TCI false
1899 DB6F 363+ ??9008: IN PRTF
1892 EA52 364+ ANI TCIF
1894 C29310 365+ JNZ ??9008 ; Wait for TCI
1897 DB6F 366+ ??9009: IN PRTF ; Get task complete int, etc.
1899 EA92 367+ ANI TCIF ; Mask it
189B CA9710 368+ JZ ??9009 ; Wait for task to be complete
189E C9 369+ RTC
371 ; ***********************************************
373 ; RECEIVE ROUTINE
375 ;
376 ; 377: INPUT: HL talker pointer
378 ; C data buffer pointer
379 ; C count (max buffer size) B implies 256
380 ; B EOS character
381; OUTPUT: Fills buffer pointed at by ME
382; CALLS: None
383; DESTROYS: A, BC, DE, HL, F
384 ;
385; RETURNS: A= normal termination--EOS detected
386 ; A= error--count overrun
387 ; A= # or A>15EH error---bad talk address
388 ;
389 ;
18A9 78 390 MOV A, B ; Get EOS character
18A8 D367 391 OUT EORS ; Output it to 91
18A6 2F6E 392 MOV A, M
18A5 FA3910 393+ CPI 4RH
18A8 FA5F 394+ CPI 5EH
18AA FA2910 395+ JP RECV5
18AD D368 396+ OUT DOUT ; Output byte
18AF 23 397+ INX H ; Increment pointer for consistency
18B0 DB61 398+ ??90010: IN INT1 ; Get Int1 status
18B2 EA52 399+ ANI BOM ; Check for byte out
18B4 CA8110 400+ JZ ??9910 ; If not, try again
18B7 38B8 401 MVI A,UNL ; Stop other listeners
18B9 D349 402 OUT DOUT
403 ;
18BA DB61 404+ ??9011: IN INT1 ; Get Int1 status
18BD EA92 405+ ANI BOM ; Check for byte out
18BF CA9710 406+ JZ ??9911 ; If not, try again

231324-35

3-133
WAITT  

;Wait for TCI=1

XOR A =21

;For completeness

;End when

;EOD or EDI & Holdoff

;Immediate XDR PON

;A XOR A =0

;Get completion character

;Pattern to clear 91 END conditions

;This bit pattern already in "A"

;Finish handshake

;Output TON

;Input data

;Else set error indicator

;Get int status (EVD & or BI)

;Check for byte out

;Get int status

;Check for EOS or EDI

;Check for BI

;Increment buffer pointer

;Decrement counter

;If either the above conditions

;or wait for last BI

;Set normal completion indicators

;Take control synchronously

;Take task complete int, etc.

;Get task complete int, etc.

;Task complete int

;Set normal completion indicators

;Wait for TCI=0 (7 tcy)

;Wait for task to be complete

;Immediate execute PON-Reset LON

;Output AUXMD

;Output AUXMD

;Immediate execute PON A XOR ~A

;A XOR A =0

;Get completion character

;If timeout is to be checked, the above WAITT should

;Be omitted & the appropriate code to look for TCI or

;JNZ inserted here.

;If count > 0 go back & wait

;Got to standby

;Increment buffer pointer

;Store it in buffer

;Wait for last BI

;Else wait for last BI

;Take control

;Wait for BI

;Check for BI

;Get data byte

;Wait for last BI

;Else set error indicator

;Get 91 Int status

;Check for 81

;YES, NO, or wait for BI

;Retrieved status

;Save it in B for BI check later

;Get int status

;Try again

;Go back & wait

;Wait for last BI

;Else set error indicator

;Take control

;If BI then no input data

;Take control

;Wait for TCI=0

;Then wait for last BI

;Else set error indicator

;Wait for TCI=0

;Take control

;Immed

;Wait for TCI=0

;Return

;Wait for TCI=0

;If not, try again

;Get int status

;Check for byte out

;Get int status

;Check for 81

;YES, NO, or wait for BI

;Save it in B for BI check later

;Get int status

;Check for byte out

;Get int status

;Check for EOS or EDI

;Check for BI

;Increment buffer pointer

;Decrement counter

;If count > 0 go back & wait

;Got to standby

;Increment buffer pointer

;Store it in buffer

;Wait for last BI

;Else wait for last BI

;Take control

;If BI then no input data

;Take control

;Wait for TCI=0

;Then wait for last BI

;Else set error indicator

;Take control

;If BI then no input data

;Take control

;Wait for TCI=0

;Then wait for last BI

;Else set error indicator

;Take control

;If BI then no input data

;Take control

;Wait for TCI=0

;Then wait for last BI

;Else set error indicator

;Take control

;If BI then no input data

;Take control

;Wait for TCI=0

;Then wait for last BI

;Else set error indicator

;Take control

;If BI then no input data

;Take control

;Wait for TCI=0

;Then wait for last BI

;Else set error indicator

;Take control

;If BI then no input data

;Take control

;Wait for TCI=0

;Then wait for last BI

;Else set error indicator

;Take control

;If BI then no input data

;Take control

;Wait for TCI=0

;Then wait for last BI

;Else set error indicator
491 ; **********************************************************************
492 ; XFER ROUTINE
493 ;
494 ; INPUTS: 
495 ; HL device list pointer
496 ; EOG character
497 ; OUTPUTS: None
498 ; CALLS: None
499 ; DESTROYS: A, HL, F
500 ; RETURNS: A=0 normal, A<>0 bad talker
501 ;
502 ;
503 ; NOTE: XFER will not work if the talker
504 ; uses EOI to terminate the transfer.
505 ; Intel will be making hardware
506 ; modifications to the 8291 that will
507 ; correct this problem. Until that time,
508 ; only EOS may be used without possible
509 ; loss of the last data byte transferred.
510 ; XFER: RANGE 48H,5EH,XFER4 ;Check for valid talker
511 ; RANGE 2CH,3EH,XFER2 ;Check for valid listener
512 ; Checks for value in range
513 ; branches to label if not
514 ; in range. Falls through if
515 ; lower <= (H)(L) <= upper.
516 ; Get next byte.
517 ;
518 ; MOV A,H
519 ;
520 ; CPI 58H
521 ;
522 ; JP XFER4
523 ; Send it to GPIO
524 ;; Send the last byte.
525 ;
526 ; OUT DOUT ;Send it to GPIB
527 ;
528 ; INX ;Incr pointer
529 ;
530 ; WAITO
531 ;
532 ; INTR ;Get INT status
533 ;
534 ; CPI 78H
535 ;
536 ; XI ;Check for byte out
537 ;
538 ; If not, try again
539 ;
540 ; JMP XFER1 ;Check for valid listener
541 ;
542 ; Checks for value in range
543 ; branches to label if not
544 ; in range. Falls through if
545 ; lower <= (H)(L) <= upper.
546 ; Get next byte.
547 ;
548 ; MOV A,H
549 ;
550 ; CPI 28H
551 ;
552 ; JM XFER2
553 ;
554 ;; Get the next byte.
555 ;
556 ; OUT DOUT ;Send it to GPIO
557 ;
558 ; INX ;Incr pointer
559 ;
560 ; WAITO
561 ;
562 ; INTR ;Get INT status
563 ;
564 ; ANI 70H
565 ;
566 ; JZ DOUT ;Check for byte out
567 ;
568 ; If not, try again
569 ;
570 ; OUT DOUT ;Send it to GPIO
571 ;
572 ; JM XFER1 ;Check for valid listener
573 ;
574 ; Checks for value in range
575 ; branches to label if not
576 ; in range. Falls through if
577 ; lower <= (H)(L) <= upper.
578 ; Get next byte.
579 ;
580 ; MOV A,H
581 ;
582 ; CPI 28H
583 ;
584 ; JM XFER2
585 ;
586 ;; Get the next byte.
587 ;
588 ; OUT DOUT ;Send it to GPIO
589 ;
590 ; INX ;Incr pointer
591 ;
592 ; WAITO
593 ;
594 ; INTR ;Get INT status
595 ;
596 ; ANI 59H
597 ;
598 ; JZ DOUT ;Check for byte out
599 ;
600 ; If not, try again
601 ;
602 ; OUT DOUT ;Send it to GPIO
603 ;
604 ; JM XFER1 ;Check for valid listener
605 ;
606 ; Checks for value in range
607 ; branches to label if not
608 ; in range. Falls through if
609 ; lower <= (H)(L) <= upper.
610 ; Get next byte.
611 ;
612 ; MOV A,H
613 ;
614 ; CPI 28H
615 ;
616 ; JM XFER2
617 ;
618 ;; Get the next byte.
619 ;
620 ; OUT DOUT ;Send it to GPIO
621 ;
622 ; INX ;Incr pointer
623 ;
624 ; WAITO
625 ;
626 ; INTR ;Get INT status
627 ;
628 ; ANI 67H
629 ;
630 ; JZ DOUT ;Check for byte out
631 ;
632 ; If not, try again
633 ;
634 ; OUT DOUT ;Send it to GPIO
635 ;
636 ; JM XFER1 ;Check for valid listener
637 ;
638 ; Checks for value in range
639 ; branches to label if not
640 ; in range. Falls through if
641 ; lower <= (H)(L) <= upper.
642 ; Get next byte.
643 ;
644 ; MOV A,H
645 ;
646 ; CPI 28H
647 ;
648 ; JM XFER2
649 ;
650 ;; Get the next byte.
651 ;
652 ; OUT DOUT ;Send it to GPIO
653 ;
654 ; INX ;Incr pointer
655 ;
656 ; WAITO
657 ;
658 ; INTR ;Get INT status
659 ;
660 ; ANI 68H
661 ;
662 ; JZ DOUT ;Check for byte out
663 ;
664 ; If not, try again
665 ;
666 ; OUT DOUT ;Send it to GPIO
667 ;
668 ; JM XFER1 ;Check for valid listener
669 ;
670 ; Checks for value in range
671 ; branches to label if not
672 ; in range. Falls through if
673 ; lower <= (H)(L) <= upper.
674 ; Get next byte.
675 ;
676 ; MOV A,H
677 ;
678 ; CPI 28H
679 ;
680 ; JM XFER2
681 ;
682 ;; Get the next byte.
683 ;
684 ; OUT DOUT ;Send it to GPIO
685 ;
686 ; INX ;Incr pointer
687 ;
688 ; WAITO
689 ;
690 ; INTR ;Get INT status
691 ;
692 ; ANI 69H
693 ;
694 ; JZ DOUT ;Check for byte out
695 ;
696 ; If not, try again
697 ;
698 ; OUT DOUT ;Send it to GPIO
699 ;
700 ; JM XFER1 ;Check for valid listener
701 ;
702 ; Checks for value in range
703 ; branches to label if not
704 ; in range. Falls through if
705 ; lower <= (H)(L) <= upper.
706 ; Get next byte.
707 ;
708 ; MOV A,H
709 ;
710 ; CPI 28H
711 ;
712 ; JM XFER2
713 ;
714 ;; Get the next byte.
**TRIGGER ROUTINE**

568 ;
581 ;
689 ;
698 ;
799 ;
*

118C DB6F
118F E682
1198 C28511
11BC DB6F
11B6 E682
1198 CABC11
1193 DB61
1195 F610
1197 CA311
119A 32FD
119C D349
119E DB6F
11A8 E682
11A2 C29E11
11A5 DB6F
11A7 E682
11A9 CAA511
11AC 3E80
11B4 D365
11B8 3E83
11B8 3E80
11B6 D364
11BA AF
11B8 D365
11BB C9

564 , WAITX
565+770428: IN PRTF
566+ IN TCIF
567+ JNZ 770428
568+ WAITT ;Wait for TCS
569+770421: IN PRTF ;Get task complete int,etc.
570+ ANI TCIF ;Mask it
571+ JZ 770421 ;Wait for task to be complete
572 XFER3: IN INTI ;Get END status hit
573 ANI ENDMK ;Mask it
574 JZ XFER3
575 MVI A,TCIF ;Take control synchronously
576 OUT CMD92
577 WAITX
578+770422: IN PRTF
579+ ANI TCIF
580+ JNZ 770422
581+ WAITT ;Wait for TCI
582+770423: IN PRTF ;Get task complete int,etc.
583+ ANI TCIF ;Mask it
584+ JZ 770423 ;Wait for task to be complete
585 MVI A,AXRA ;Not cont AH or END on EOS
586 OUT AUXMD
587 MVI A,PNHSK ;Finish handshake
588 OUT AUXMD
589 MVI A,TOM ;Talk only
590 OUT ADRMK
591 CLR A ;Normal return
592+ XRA A 00000000
593 OUT AUXMD ;Immediate XEO PON
594 XFER4: RET

**************************************************
11E8 CAE411 638+ JZ ????26 ;If not, try again
11E8 C9 639 RET
11E8 640 ;
11E8 641 ;-----------------------------------------------
11E8 642 ;
11E8 643 ;DEVICE CLEAR ROUTINE
11E8 644 ;
11E8 645 ;
11E8 646 ;
11E8 647 ;INPUTS: HL listener pointer
11E8 648 ;OUTPUT: None
11E8 649 ;CALLS: None
11E8 650 ;DESTROYS: A, HL, P
11E8 651 ;
11EC 3E3F 652 DCLR: MVI A,UNL
11EE D360 653 OUT DOUT
11D4 DCLR1: RANGE 2BH,3EH,DCLR2
11F8 655+ ;Checks for value in range
11F8 656+ ;branches to label if not
11F8 657+ ;in range. Falls through if
11F8 658+ ;lower <= ( (R)(L) ) <= upper.
11F8 659+ ;Get next byte.
11F8 660+ DCLR2: WAITO 661+ MOV A,M
11F8 662+ CPI 28H
11F8 663+ CMP DCLR2
11F8 664+ JP DCLR2
11F8 665+ WAITO
11FD D66 666+???27: IN INT1 ;Get IntI status
11FD D68 667+ AMI BOM ;Check for byte out
11FD CAF011 668+ JZ ???27 ;If not, try again
11FF 669 MOV A,M
11FE D360 670 OUT DOUT ;Send listener to GPIB
11FF D361 671 INX H
11FF D362 672 JMP DCLR1
11FF D66 673 DCLR2: WAITO
11FF D68 674+???28: IN INT1 ;Get IntI status
11FF D69 675+ AMI BOM ;Check for byte out
11FF D6A 676+ JZ ???.28 ;If not, try again
11FF D6B 677 MVI A,DCD ;Send device clear
11FF D6C 678 OUT DOUT ;To all addressed listeners
11FF D6D 679 WAITO
11FF D6E 680+???.29: IN INT1 ;Get IntI status
11FF D6F 681+ AMI BOM ;Check for byte out
11FF D70 682+ JZ ???.29 ;If not, try again
11FF D71 683 RET
11FF 684 ;
11FF 685 ;-----------------------------------------------
11FF 686 ;
11FF 687 ;SERIAL POLL ROUTINE
11FF 688 ;
11FF 689 ;INPUTS: HL talker list pointer
11FF 690 ;DE status buffer pointer
11FF 691 ;OUTPUTS: Fills buffer pointed to by DE
11FF 692 ;CALLS: None
11FF 693 ;DESTROYS: A, BC, DE, HL, P
11FF 694 ;
11FC 3E3F 695 SPOI: MVI A,UNL ;Universal unlisten
11FE D360 696 OUT DOUT
11FE D67 697 WAITO
11FE D68 698+???.30: IN INT1 ;Get IntI status
11FE D69 699+ AMI BOM ;Check for byte out
11FE D6A 700+ JZ ???.30 ;If not, try again
11FE D6B 701 MVI A,MLA ;My listen address
11FE D6C 702 OUT DOUT
11FE D6D 703 WAITO
11FE D6E 704+???.31: IN INT1 ;Get IntI status
11FE D6F 705+ AMI BOM ;Check for byte out
11FF D70 706+ JZ ???.31 ;If not, try again
11FF D71 707 MVI A,SPE ;Serial poll enable
11FF D72 D360 708 OUT DOUT ;To be formal about it
11FF D73 709 WAITO
11FF D74 710+???.32: IN INT1 ;Get IntI status
231324-39
AP-66

1238 6482  711+   ANI   ROM   ;Check for byte out
123A CA3412  712+   JS    ???#32  ;If not, try again
123D 7E     713+   MOV   A,M   ;Check for valid talker
123E FE48  714+   CPI   48H   ;Checks for value in range
1240 FA48  715+   JMP   SPOL2   ;Branches to label if not
1241 FE5F  716+   CPI   5EH1   ;in range. Falls through if
1243 F29412  717+   JP    SPOL2   ;lower <= (M)(L) <= upper.
1247 8E     718+   MOV   A,M   ;Get talker
1249 D36A  719+   OUT   DOUT   ;Send to GPIB
124B 23     720+   INX   H   ;Incr talker list pointer
124C 4684  721+   MVI   A,L0W   ;Listen only
124E D364  722+   OUT   AD11MD   ;Wait for talk address to complete
1258 DB61  729+   WAIT0   ;Get IntI status
125E E502  730+   IN    INT1   ;Check for byte out
125F C5012  731+   ANI   ROM   ;Pattern for immediate XEQ PON
125F AF     732+   XRA   A   ;A XOR A =#V
125F 0360  733+   MOV   B,A   ;Get talker
1262 C25612  734+   INJ   ??#54   ;Take, control sync
1265 DB6F  735+   WAIT0   ;Wait for TCI false
126B E502  736+   IN    TC1F   ;Get task complete int, etc.
126C C25612  737+   JMP   ??#55   ;Wait for TCI
126E 47     738+   MOV   B,A   ;Get task complete int, etc.
126F E491  739+   IN    INT1   ;Wait for TCI false
1271 C4C12  740+   AND   BIM   ;Mask it
1274 36FD  741+   CJNZ  ??#56   ;Wait for task to be complete
1276 D359  742+   OUT   CM092   ;Wait for TCI false
1278 DB6F  743+   WAIT0   ;Get task complete int, etc.
127A E502  744+   IN    TC1F   ;Wait for task to be complete
127C C27612  745+   JMP   ??#57   ;Wait for byte input
127F DB6F  746+   IN    INT1   ;Save status in B
1281 E502  747+   AND   BIM   ;Check for byte in
1283 C7F12  748+   INJ   ??#58   ;Check for immediate XEQ PON
1286 DB60  749+   JMP   ??#59   ;Take control sync
1288 12     750+   IN    DIN   ;A XOR A =#V
1289 13     751+   MOV   B,A   ;Wait for TCI false
128A 3880  752+   IN    A,TCSY   ;Get control sync
128C D344  753+   OUT   AD11MD   ;Go to next device on list
128E AF     754+   JMP   ??#5A   ;Immediate XEQ PON
1291 C3D12  755+   XRA   A   ;Go to next device on list
1294 3E19  756+   MVI   A,SP0L   ;Immediate XEQ PON to clear LA
1295 D360  757+   JMP   SPOl1   ;Get next byte.
1298 DB61  758+   OUT   DOUT   ;We know 80 was set (WA1T0 above)
129A E502  759+   IN    INT1   ;Get IntI status
129C C9812  760+   AND   BIM   ;Wait for byte out
129F AF     761+   INJ   77#69   ;Check for byte out
12A0 D365  762+   SJMP  ??#99   ;If not, try again
12A2 C9     763+   OUT   AUXMD   ;Immediate XEQ PON to clear LA
12A7 7E     764+   IN    INT1   ;Immediate XEQ PON to clear LA

3-138

231324-40
785 ; PARALLEL POLL ENABLE ROUTINE
786 ;
787 ; INPUTS: HL listener list pointer
788 ; DE configuration byte pointer
789 ; OUTPUTS: None
790 ; CALLS: None
791 ; DESTRYOS: A, DE, HL, F
792 ;
793 ;
12A3 3E3F 12A4 D350
12A5 PPEN: MOV A,HL ;Universal unlisten
12A6 7E OUT DOUT
12A7 PPEN1: RANGE 2EH,3EH,PPEN2 ;Check for valid listener
12A8 7E ;Checks for value in range
12A9 80+ ;branches to label if not
12AA 7E ;in range. Falls through if
12AB 80+ ;lower <= (H)(L) <= upper.
12AC 80+ ;Get next byte.
12AD 82+ MOV A,M
12AE FE20 ; unconditional branch
12AF 80+ CPI 2NH
12B0 FAD812 804+ JMP PPEN2
12B1 FE3F 805+ CPI 3EH+1
12B2 FDD812 806+ JP PPEN2
12B3 807 WAITO ;Valid wait 91 data out req
12B4 D861 808+770044: IN INT1 ;Get IntI status
12B5 E82 809+ ANI ROM ;Check for byte out
12B6 CAB212 810+ JZ 770044 ;If not, try again
12B7 7E 811 MOV A,M ;Get listener
12B8 D350 812 OUT DOUT
12B9 813 WAITO
12BA D861 814+7700441: IN INT1 ;Get IntI status
12BB E82 815+ ANI ROM ;Check for byte out
12BC CAB212 816+ JZ 7700441 ;If not, try again
12BD 3E85 817 MOV A,PPC ;Parallel poll configure
12BE D350 818 OUT DOUT
12BF 819 WAITO
12C0 D861 820+7700442: IN INT1 ;Get IntI status
12C1 E82 821+ ANI ROM ;Check for byte out
12C2 CAC712 822+ JZ 770042 ;If not, try again
12C3 1A 823 LDAX D ;Get matching configuration byte
12C4 F660 824 ORI PPE ;Merge with parallel poll enable
12C5 D350 825 OUT DOUT
12C6 23 826 INX H ;Incr pointers
12C7 13 827 INX D
12C8 3A712 828 JMP PPEN1 ;Loop until invalid listener char
12C9 D861 829 PPEN2: WAITO
12CA E82 830+7700443: IN INT1 ;Get IntI status
12CB CAD812 831+ ANI ROM ;Check for byte out
12CC D861 832+ JZ 770043 ;If not, try again
12CD 833 CMP
12CE 3E3F 834 ;
12CF D350 835 ; PARALLEL POLL DISABLE ROUTINE
12D0 7E 836 ;
12D1 82+ MOV A,M
12D2 FE20 ; unconditional branch
12D3 80+ CPI 2NH
12D4 FAD812 804+ JMP PPDS2
12D5 FE3F 805+ CPI 3EH+1
12D6 FDD812 806+ JP PPDS2
12D7 805 WAITO
12D8 D861 806+770044: IN INT1 ;Get IntI status
12D9 D861 855+7700441: IN INT1 ;Get IntI status
12DA FE3F 857+ ANI ROM ;Check for byte out
12DB CAEF12 858+ JZ 7700441 ;If not, try again

231324-41

3-139
12F6 7E 859 MOV A,M ;Get listener
12F7 D348 860 OUT DOUT ;Inc pointer
12F9 23 861 INX H ;Loop until invalid listener
12FA C3E12 862 JMP PPOS1
12FD 7E 863 PPOS2: WAITO
12FF E042 854+7E845: INT1 ;Get INT1 status
1301 CAPD12 856+ JZ ?78045 ;If not, try again
1304 3E85 847 MVI A,PPC ;Parallel poll configure
1306 D358 858 OUT DOUT
1308 DB61 859 WAITO
130A E042 855+ ANI ROM ;Check for byte out
130C CA8013 856+ JZ ?78045 ;If not, try again
130F E770 873 MVI A,PPD ;Parallel poll disable
1311 D358 874 OUT DOUT
1313 DB61 875 WAITO
1315 E042 876+ ANI ROM ;Check for byte out
1317 CA1313 877+ JZ ?78047 ;If not, try again
131A C9 879 RET

880 ; PARALLEL POLL UNCONFIGURE ALL ROUTINE
882 ;
884 ;INPUTS: None
885 ;OUTPUTS: None
886 ;CALLS: None
887 ;DESTROYS: A, F
888 ;
131B 3E15 889 PPU: MVI A,PPU ;Parallel poll unconfigure
131D D358 890 OUT DOUT
131F DB61 891 WAITO
1321 E042 892+?78048: IN INT1 ;Get INT1 status
1323 CA1F13 893+ ANI ROM ;Check for byte out
1326 C9 894+ JZ ?78047 ;If not, try again
1327 3E48 895 RET

897 ;****************************************************
898 ; I Conduct A PARALLEL POLL
899 ;
89A ;
89B ;INPUTS: None
89C ;OUTPUTS: None
89D ;CALLS: None
89E ;DESTROYS: A, B, F
89F ;RETURNS: A= parallel poll status byte
900 ;
1327 3E48 900 PPU: MVI A,OLON ;Listen only
1329 D364 901 OUT ADRMD
132B AF 902 CLRA ;Immediate XEQ PON
132C D365 903 XRA A ;A XOR A =
132E 3E55 904 OUT AUXMD ;Reset TON
1330 D349 905 MVI A,EXP ;Execute parallel poll
1332 D861 906 OUT CMD92
1334 47 907 WAITI ;Wait for completion= BI on 91
1335 E681 908 MOV B,A ;Save status in B
1337 CA2213 909+ ANI RIM ;Check for byte in
133A 3E80 910+ JZ ?78049 ;If not, just try again
133C D364 911 MVI A,TON ;Talk only
133E AF 912 CLRA ;Immediate XEQ PON
133F D355 913 XRA A ;A XOR A =
1341 D848 914 OUT AUXMD ;Reset LON
1343 C9 915 IN D1N ;Get PP byte

927 ;****************************************************
928 ; PASS CONTROL ROUTINE
930 ;
931 ;INPUTS: HL pointer to talker
932 ;OUTPUTS: None

231324-42
933 \textbf{CALLS:} None
934 \textbf{DESTROYS:} A, HL, F
935 \textbf{PCTL: RANGE} 40H, SEH, PCTL1 \textbf{Is it a valid talker?}
936+ \textbf{Checks for value in range}
937+ \textbf{Branches to label if not}
938+ \textbf{In range, falls through if}
939+ \textbf{Lower <= (H)(L) <= upper.}
940+ \textbf{Get next byte.}

1344 \textbf{7E} 941+ MOV A, M
1345 \textbf{FE40} 942+ CPI 48H
1347 \textbf{FABA13} 943+ J' PCTL1
134A \textbf{FE5F} 944+ CPI SEH+1
134C \textbf{F2A13} 945+ JP PCTL1
134F \textbf{FE41} 946 CPI MTA \textbf{Is it my talker address}
1351 \textbf{CABA13} 947 JZ PCTL1 \textbf{Yes, just return}
1354 \textbf{D360} 948 OUT DOUT \textbf{Send on GPIB}
1356 \textbf{D861} 949 \textbf{WAITO}
1358 \textbf{EGF7} 950+ IN INTI \textbf{Get Int status}
135A \textbf{CA56} 951+ ANI BOM \textbf{Check for byte out}
135D \textbf{EB81} 952+ JZ 778055 \textbf{If not, try again}
135F \textbf{D360} 953 MVI A, TCT \textbf{Take control message}
1361 \textbf{D861} 954 OUT DOUT \textbf{Send on GPIB}
1363 \textbf{E6F2} 955+ \textbf{WAITO}
1365 \textbf{CA61} 956+ IN INTI \textbf{Get Int status}
1368 \textbf{EB81} 957+ ANI BOM \textbf{Check for byte out}
136A \textbf{E381} 958+ JZ 778055 \textbf{If not, try again}
136C \textbf{AF} 959 MVI A, MODE1 \textbf{Not talk only or listen only}
136D \textbf{D360} 960 OUT ADRM1 \textbf{Enable 91 address mode 1}
136F \textbf{D861} 961 CLR A \textbf{A XOR A = 0}
1370 \textbf{E365} 962 OUT AUXMD \textbf{Immediate X80 PON}
1372 \textbf{E881} 963 MVI A, MDA \textbf{My device address}
1374 \textbf{E366} 964 OUT ADRM1 \textbf{Enabled to talk and listen}
1376 \textbf{E361} 965 MVI A, AXR8+PCTE1 \textbf{Command pass thru enable}
1378 \textbf{E345} 966 OUT AUXMD \textbf{Enable}
137A \textbf{E345} 967 \textbf{WAITO}
137C \textbf{AF} 968 \textbf{******* Optional PP configuration goes here*******}
137E \textbf{3F1} 969 MVI A, GIDL \textbf{92 go idle command}
1380 \textbf{D369} 970 OUT CMD92
1382 \textbf{D86F} 971 \textbf{WAITX}
1384 \textbf{E682} 972+?778052: IN PRTF
1386 \textbf{C781} 973+ ANI TCF1
1388 \textbf{C781} 974+ JNZ ?778052
138A \textbf{CA82} 975 \textbf{WAITT} \textbf{Wait for TCI}
138C \textbf{E86F} 976+?778053: IN PRTF \textbf{Get task complete int, etc.}
138E \textbf{E361} 977 ANI TCF1 \textbf{Mask it}
1390 \textbf{E361} 978+ JZ ?778053 \textbf{Wait for task to be complete}
1392 \textbf{E361} 979 INX H
1394 \textbf{E361} 980 PCTL1: RET
981+ ;
983 \textbf{***********************************************}
984+ ;
985 \textbf{RECEIVE CONTROL ROUTINE}
986+ ;
987 \textbf{INPUTS:} None
988 \textbf{OUTPUTS:} None
989 \textbf{CALLS:} None
990 \textbf{DESTROYS:} A, F
991 \textbf{RETURNS:} R= invalid (not take control to us or CPT bit not on)
992+ \textbf{< R = valid take control -- 92 will now be in control}
993 \textbf{NOTE:} THIS CODE MUST BE TIGHTLY INTEGRATED INTO ANY USER
994+ ; SOFTWARE THAT FUNCTIONS WITH THE 8291 AS A DEVICE.
995+ ; NORMALLY SOME ADVANCE WARNING OF IMPENDING PASS
996+ ; CONTROL SHOULD BE GIVEN TO US BY THE CONTROLLER
997+ ; WITH OTHER USEFUL INFO, THIS PROTOCOL IS SITUATION
998+ ; SPECIFIC AND WILL NOT BE COVERED HERE,
999+ ;
1000+ ;
1388 \textbf{D851} 1001 HCTL: IN INTI \textbf{Get INTI reg (i.e. CPT etc.)}
138D \textbf{E688} 1002 ANI CPT \textbf{Is command pass thru on?}
138F \textbf{CACA13} 1003 JZ RCTL2 \textbf{No, invalid -- go return}
1392 \textbf{D856} 1004 IN CPTRO \textbf{Get command}
1394 \textbf{F859} 1005 CPI TCT \textbf{Is it take control?}
1395  C2CA13  1085  JNZ  RCTL1  ;No, go return invalid
1399  DB64  1087  IN    ADRI1  ;Get address status
1399  DB42  1088  ANI   TA  ;Is TA on?
139D  CACA1J  1089  JZ    RCTL1  ;No -- go return invalid
13A8  3E88  108A  MVI   A,TDTL1 ;Disable talker listener
13A7  D355  108B  OUT   ADRT1
13A4  3E88  108C  MVI   A,TOL  ;Talk only
13A6  0354  108D  OUT   ADRTD
13A8  AP   108E  XRA   A  ;A XOR A =0
13A9  D361  108F  OUT   INT1  ;Mask off INT 1 hits
13A8  D362  1090  OUT   INT2
13AD  D355  1091  OUT   AUXMD
13AF  3EFA  1092  MVI   A,TCT  ;Take (receive) control 92 command
13B1  D349  1093  OUT   CMD92
13B3  3E8F  1094  MVI   A,VSCMD ;Valid command pattern for 91
13B5  D345  1095  OUT   AUXMD
1096  ;******* OPTIONAL TRUE1 check could be put here ******
1097  WAITX
1098  1025+??0054: IN  PHTF
1099  1806  1026+??0054: IN  PHTF
1100  197813  1027+??0054: JNZ  ??0054
1101  1B28  1028  WAITT ;Wait for TCI
1102  13C0  E692  1029  IN    PHTF  ;Get task complete int, etc.
1103  13C2  CAFE13  1030+??0055: IN  PHTF  ;Mask it
1104  13C5  3E89  1031+??0055: JZ  ??0055  ;Wait for task to be complete
1105  13C7  C3CF13  1032  MVI   A,TCT  ;Valid return pattern
1106  13CA  3E8F  1033  JMP   RCTL2  ;Only one return per routine
1107  13CC  D365  1034  RCTL1: MVI   A,VSCMD ;Acknowledgment CPT
1108  13CF  0354  1035  OUT   AUXMD
1109  13CC  0365  1036  CLRA  ;Error return pattern
1110  13CE  AF   1037+??0054: XRA   A  ;A XOR A =0
1111  13CF  C9   1038+??0054: RCTL2: RET
1112  1039  ;
1113  1040+;**************************************************
1114  1041  ;
1115  1042+;SRO ROUTINE
1116  1043+;
1117  1044+;INPUTS: None
1118  1045+;OUTPUTS: None
1119  1046+;CALLS: None
1120  1047+;RETURNS: A=0 no SRO
1121  1048+;A<>0 SRO occurred
1122  1049  ;
1123  1050+;**************************************************
1124  1051+;SRQD: IN  INTST ;Get 92's INTRO status
1125  1052+;SRQD: IN  SHRD  ;Mask off SRQ
1126  1053+;JZ    SHRD  ;Not set--- go return
1127  1054+;ORI   IACK  ;Set--- must clear it with IACK
1128  1055+;OUT   CMD92
1129  1056+;SRQD1: IN  INTST  ;Get IRF
1130  1057+;ANI   IRFRT  ;Mask it
1131  1058+;JZ    SRQD1  ;Wait if not set
1132  13E2  C9   1059+;SRQD2: RET
1133  1060  ;
1134  1061+;**************************************************
1135  1062  ;
1136  1063+;REMOTE ENABLE ROUTINE
1137  1064  ;
1138  1065+;INPUTS: None
1139  1066+;OUTPUTS: None
1140  1067+;CALLS: NONE
1141  1068+;GETNOVRS: A, F
1142  1069+;**************************************************
1143  13E3  3E89  1070+;REME: MVI   A,SREM  ;92 asserts remote enable
1144  13E5  D369  1071+;OUT   CMD92
1145  1072+;WAITX ;Wait for TCI = #
1146  13E7  3E8F  1073+??0054: IN  PHTF
1147  13E9  E922  1074+??0054: ANI   TCIF
1148  13EB  C2713  1075+??0054: JNZ  ??0054
1149  13E7  D34F  1076+??0054: WAITT ;Wait for TCI
1150  13E8  E922  1077+??0054: IN  PHTF
1151  13F0  E922  1078+??0054: ANI   TCIF
1152  13F2  CAE13  1079+??0054: JNZ  ??0054

231324-44

3-142
13F5 C9
1088 RET
1088 ;***************************************************************
1089 ;LOCAL ROUTINE
1089 ;***************************************************************
1089 ;INPUTS: None
1089 ;OUTPUTS: None
1089 ;CALLS: None
1089 ;DESTROYS: A, F
1091 ;
13F8 3EF7
1092 LOCAL: MVI A,560C
13F8 D359
1093 OUT CMD92 ;92 stops asserting remote enable
13FA DB6F
1094 WAITX ;Wait for TCI =
13FC E602
1096+ ANI TCIF
13FE C2FA13
1097+ JNZ 77059
1098 WAITT ;Wait for TCI
1401 DB6F
1099+ IN PRTF ;Get task complete int, etc.
1403 E602
1100+ ANI TCIF ;Mask it
1405 CA0114
1101+ JZ 77059 ;Wait for task to be complete
1408 C9
1102 RET
1103 ;
1104 ;***************************************************************
1105 ;INTERFACE CLEAR / ABORT ROUTINE
1107 ;
1108 ;
1109 ;INPUTS: None
1109 ;OUTPUTS: None
1110 ;CALLS: None
1111 ;DESTROYS: A, F
1112 ;
1113 ;
1114 ;
1409 3EF9
1115 IFCL: MVI A,ABORT
140B D349
1116 OUT CMD92 ;Send IFC
140D DB6F
1117 WAITX ;Wait for TCI =
140F E602
1118+ IN PRTF ;Get task complete int, etc.
1411 C20D14
1119+ ANI TCIF
1411 220D14
1120+ JNZ 77059
1411 1121 WAITT ;Wait for TCI
1414 DB6F
1122+ IN PRTF ;Get task complete int, etc.
1416 E602
1123+ ANI TCIF ;Mask it
1418 CA1414
1124+ JZ 77059+1 ;Wait for task to be complete
1418 1125 ;Delete both WAITX & WAITT if this routine
141B C9
1130 RET
1132 ;
1133 ; APPLICATION EXAMPLE CODE FOR 8985
1134 ;
1135 FCDNL EQU '2' ; Func gen device num "2" ASCII,latin
1136 FCDNL EQU '1' ; Func gen device num "1" ASCII,latin
1137 FCDNT EQU 'Q' ; Freq ctr talk address
1138 CR EQU 3DH ; ASCII carriage return
1139 LF EQU 3AH ; ASCII line feed
1140 LEND EQU 0FFH ; List end for Talk/Listen lists
1141 SRQM EQU 4AH ; Bit indicating device sent SRO
1142 CR EQU "0t" ; ASCII carriage return
1143 LFD EQU 9FH ; List end for Talk lists
1144 FCDNL, LEND ; Listen list for func. gen
1145 FCDNT, LEND ; Talk list for freq ctr
1146 LXI D, FCDATA ; Data pointer
1147 LXI H, LL1 ; Listen list pointer
1148 LXI D, FCDNL, LEND ; Listen list for func. gen
1149 LXI D, FCDNT, LEND ; Talk list for freq ctr
1150 ;
1151 ; SETUP FUNCTION GENERATOR
1152 MVI B, CR ; EOS
1153 MVI C, LIM1 ; Count
1155 F CD1C10 ; CALL SEND
1156 CALL SEND
1157 ; SETUP FREQ COUNTER
1158 ;
1159 ;
1160 ;
1161 MVI C, LIM2 ; Count
1162 LXI D, FCDATA ; Data pointer
1163 LXI H, LLI ; Listen list pointer
1164 CALL SEND
1165 ;
1166 ; WAIT FOR SRO FROM FREQ CTR
1167 ;
1168 LOOP: CALL SRQM ; Has SRO occurred ?
1169 JZ LOOP ; No, wait for it
1170 ;
1171 ; SERIAL POLL TO CLEAR SRO
1172 ;
1173 LXI D, SBSYTE ; Buffer pointer
1174 LXI H, TLI ; Talk list pointer
1175 ;
1176 D CX D ; Backup buffer pointer to ctr byte
1177 LDAX D ; Get status byte
1178 ANI SRQM ; Did ctr assert SRO ?
1179 JZ ERROR ; Ctr should have said yes
1180 ;
1181 ; RECEIVE READING FROM COUNTER
1182 ;
1183 MVI B, LF ; EOS
1184 MVI C, LIM3 ; Count
1185 LXI H, TLI ; Talk list pointer
1186 LXI D, FCDATI ; Data in buffer pointer
1187 CALL RECV
1188 JNZ ERRNR ; User dependent error handling
1189 ;
1190 ;
1191 ;
1192 ;
1193 ERROR: NOP ;
1194 ; ETC.
1195 ORG 3C00H
1196 SPBYTE: DS 1 ; Location for serial poll byte
1197 LIM3 EQU 017 ; Max freq counter input

231324-46
PUBLIC SYMBOLS

EXTERNAL SYMBOLS

USEN SYMBOLS

ASSEMBLY COMPLETE, NO ERRORS 231324-47
APPENDIX B

Test Cases for the Software Drivers

The following test cases were used to exercise the software routines and to check their action. To provide another device/controller on the GPIB a ZT488 GPIB Analyzer was used. This analyzer acted as a talker, listener or another controller as needed to execute the tests. The sequence of outputs are shown with each test. All numbers are hexadecimal.

Send Test Cases

<table>
<thead>
<tr>
<th>B</th>
<th>44</th>
<th>44</th>
<th>44</th>
</tr>
</thead>
<tbody>
<tr>
<td>C</td>
<td>30</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>DE</td>
<td>3E80</td>
<td>3E80</td>
<td>3E80</td>
</tr>
<tr>
<td>HL</td>
<td>3E70</td>
<td>3E70</td>
<td>3E70</td>
</tr>
<tr>
<td>3E70: 20 30 3E 3F</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3E80: 11 44</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GPIB output: 41 ATN 41 ATN 41 ATN</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3F ATN 3F ATN 3F ATN</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>20 ATN 20 ATN 20 ATN</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>30 ATN 30 ATN 30 ATN</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3E ATN 3E ATN 3E ATN</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11 11</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>44 EOI 44 EOI</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Ending B = 44 44 44 44 44 44 44 44 44 0 = 256
Ending C = 2E 0 0
Ending DE = 3E82 3E82 3E80
Ending HL = 3E73 3E73 3E73

Receive Test Cases

<table>
<thead>
<tr>
<th>B</th>
<th>44</th>
<th>44</th>
<th>44</th>
<th>44</th>
<th>44</th>
<th>44</th>
<th>44</th>
<th>44</th>
</tr>
</thead>
<tbody>
<tr>
<td>C</td>
<td>30</td>
<td>30</td>
<td>30</td>
<td>30</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>DE</td>
<td>3E80</td>
<td>3E80</td>
<td>3E80</td>
<td>3E80</td>
<td>3E80</td>
<td>3E80</td>
<td>3E80</td>
<td>3E80</td>
</tr>
<tr>
<td>HL</td>
<td>3E70</td>
<td>3E70</td>
<td>3E70</td>
<td>3E70</td>
<td>3E70</td>
<td>3E70</td>
<td>3E70</td>
<td>3E70</td>
</tr>
<tr>
<td>3E70: 40 50 5E 5F</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GPIB output: 40 ATN 50 ATN 5E ATN 40 ATN 40 ATN 40 ATN</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3F ATN 3F ATN 3F ATN 3F ATN 3F ATN 3F ATN</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>21 ATN 21 ATN 21 ATN 21 ATN 21 ATN 21 ATN</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ZT488 Data 1 1 1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>In 2 2 2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3 3 3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4 4 4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>44 5,EOI</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Ending A = 0 0 0 5F 40 0 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Ending B = 0 0 44 40 0 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Ending C = 2B 2B 2C 30 0 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Ending DE = 3E85 3E85 3E84 3E80 3E84 3E84</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Ending HL = 3E71 3E71 3E71 3E70 3E71 3E71</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Serial Poll Test Cases

<table>
<thead>
<tr>
<th>C</th>
<th>DE</th>
<th>HL</th>
</tr>
</thead>
<tbody>
<tr>
<td>30</td>
<td>3E80</td>
<td>3E70</td>
</tr>
</tbody>
</table>

3E70: 40 50
5E
5F
GPIB output: 3F ATN 21 ATN 18 ATN

input*: 00
output: 50 ATN
input*: 41
output: 5E ATN
input*: 7F
output: 19 ATN

*NOTE: leave ZT488 in single step mode even on input
Ending C = 30
Ending DE = 3E80
Ending HL = 3E73
Ending 3E80: 00 41 7F

Pass Control Test Cases

<table>
<thead>
<tr>
<th>HL</th>
<th>GPIB output:</th>
</tr>
</thead>
<tbody>
<tr>
<td>3E70</td>
<td>40</td>
</tr>
</tbody>
</table>

3E70: 41(MTA) 5F

GPIB output: 09 ATN

Ending HL = 3E71 3E70 3E70
Ending A = 02 41(MTA) 5F

Receive Control Test Cases

<table>
<thead>
<tr>
<th>GPIB input</th>
<th>10 ATN</th>
<th>40 ATN</th>
<th>41 ATN</th>
</tr>
</thead>
<tbody>
<tr>
<td>ATN</td>
<td>09 ATN</td>
<td>09 ATN</td>
<td></td>
</tr>
</tbody>
</table>

Run Receive Control
GPIB Input ATN ATN
Ending A = 0 0 09
Parallel Poll Enable Test Cases

<table>
<thead>
<tr>
<th>DE</th>
<th>HL</th>
<th>3E70</th>
<th>3E80</th>
</tr>
</thead>
<tbody>
<tr>
<td>3E80</td>
<td>3E70</td>
<td>20 30 3E 3F</td>
<td>01 02 03</td>
</tr>
</tbody>
</table>

GPIB output: 3F ATN 20 ATN 05 ATN 61 ATN 30 ATN 05 ATN 62 ATN 3E ATN 05 ATN 63 ATN

Ending DE = 3E83
Ending HL = 3E73

Parallel Poll Disable Test Cases

<table>
<thead>
<tr>
<th>HL</th>
<th>3E70</th>
<th>3E70</th>
</tr>
</thead>
<tbody>
<tr>
<td>3E70</td>
<td>20 30 3E 3F</td>
<td>3F</td>
</tr>
</tbody>
</table>

GPIB output: 3F ATN 20 ATN 05 ATN 30 ATN 3E ATN 05 ATN 70 ATN

Ending HL = 3E73

Parallel Poll Unconfigure Test Case

GPIB output: 15 ATN

Parallel Poll Test Cases

Set DIO# 1 2 3 4 5 6 7 8 None
Ending A 1 2 4 8 10 20 40 80 0

SRQ Test

Set SRQ momentarily
Ending A = 02
Reset SRQ
Ending A = 00
Trigger Test

\[ \begin{align*}
\text{HL} &= 3E70 \\
\text{DE} &= 3E80 \\
\text{BC} &= 4430 \\
3E70: & \quad 20 \ 30 \ 3E \ 3F \\
\text{GPIB output:} & \quad 3F \text{ ATN} \\
& \quad 20 \text{ ATN} \\
& \quad 30 \text{ ATN} \\
& \quad 3E \text{ ATN} \\
& \quad 08 \text{ ATN} \\
\text{Ending HL} &= 3E73 \\
\text{DE} &= 3E80 \\
\text{BC} &= 4430
\end{align*} \]

Device Clear Test

\[ \begin{align*}
\text{HL} &= 3E70 \\
\text{DE} &= 3E80 \\
\text{BC} &= 4430 \\
3E70: & \quad 20 \ 30 \ 3E \ 3F \\
\text{GPIB output:} & \quad 3F \text{ ATN} \\
& \quad 20 \text{ ATN} \\
& \quad 30 \text{ ATN} \\
& \quad 3E \text{ ATN} \\
& \quad 14 \text{ ATN} \\
\text{Ending HL} &= 3E73 \\
\text{DE} &= 3E80 \\
\text{RC} &= 4430
\end{align*} \]

XFER Test

\[ \begin{align*}
\text{B} &= 44 \\
\text{HL} &= 3E70: \\
3E70: & \quad 40 \ 20 \ 30 \ 3E \ 3F \\
\text{GPIB output:} & \quad 40 \text{ ATN} \\
& \quad 3F \text{ ATN} \\
& \quad 20 \text{ ATN} \\
& \quad 30 \text{ ATN} \\
& \quad 3E \text{ ATN} \\
\text{GPIB input:} & \quad 0 \\
& \quad 1 \\
& \quad 2 \\
& \quad 3 \\
& \quad 44 \\
\text{Ending A} &= 0 \\
\text{B} &= 44 \\
\text{HL} &= 3E74
\end{align*} \]
Application Example
GPIB Output/Input

GPIB output:
41 ATN
3F ATN
32 ATN
46
55
31
46
52
33
37
4B
48
41
4D
32
56
4F
0D EOI
41 ATN
3F ATN
31 ATN
50
46
34
47
37
54 EOI
SRQ
GPIB input:
GPIB output:
3F ATN
21 ATN
18 ATN
51 ATN

GPIB input:
GPIB output:
40 SRQ
19 ATN
51 ATN
3F ATN
21 ATN
<table>
<thead>
<tr>
<th>GPIB input:</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>20</td>
</tr>
<tr>
<td></td>
<td>2B</td>
</tr>
<tr>
<td></td>
<td>20</td>
</tr>
<tr>
<td></td>
<td>20</td>
</tr>
<tr>
<td></td>
<td>33</td>
</tr>
<tr>
<td></td>
<td>37</td>
</tr>
<tr>
<td></td>
<td>30</td>
</tr>
<tr>
<td></td>
<td>30</td>
</tr>
<tr>
<td></td>
<td>30</td>
</tr>
<tr>
<td></td>
<td>30</td>
</tr>
<tr>
<td></td>
<td>2E</td>
</tr>
<tr>
<td></td>
<td>30</td>
</tr>
<tr>
<td></td>
<td>45</td>
</tr>
<tr>
<td></td>
<td>2B</td>
</tr>
<tr>
<td></td>
<td>30</td>
</tr>
<tr>
<td></td>
<td>0D</td>
</tr>
<tr>
<td></td>
<td>0A</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>GPIB output:</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>XX ATN</td>
</tr>
</tbody>
</table>
### APPENDIX C

#### REMOTE MESSAGE CODING

<table>
<thead>
<tr>
<th>Mnemonic</th>
<th>Message Name</th>
<th>Bus Signal Line(s) and Coding That Asserts the True Value of the Message</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACG</td>
<td>addressed command group</td>
<td>T I D</td>
</tr>
<tr>
<td>ATN</td>
<td>attention</td>
<td>UC</td>
</tr>
<tr>
<td>DAB</td>
<td>data byte</td>
<td>UC</td>
</tr>
<tr>
<td>DAC</td>
<td>data accepted</td>
<td>HS</td>
</tr>
<tr>
<td>DAV</td>
<td>data valid</td>
<td>HS</td>
</tr>
<tr>
<td>DCL</td>
<td>device clear</td>
<td>UC</td>
</tr>
<tr>
<td>END</td>
<td>end</td>
<td>ST</td>
</tr>
<tr>
<td>EOS</td>
<td>end of string</td>
<td>ST</td>
</tr>
<tr>
<td>GET</td>
<td>group execute trigger</td>
<td>AC</td>
</tr>
<tr>
<td>GTL</td>
<td>go to local</td>
<td>AC</td>
</tr>
<tr>
<td>IDY</td>
<td>identify</td>
<td>UC</td>
</tr>
<tr>
<td>IFC</td>
<td>interface clear</td>
<td>UC</td>
</tr>
<tr>
<td>LAG</td>
<td>listen address group</td>
<td>AD</td>
</tr>
<tr>
<td>LLO</td>
<td>local lock out</td>
<td>AD</td>
</tr>
<tr>
<td>MLA</td>
<td>my listen address</td>
<td>AC</td>
</tr>
<tr>
<td>MTA</td>
<td>my talk address</td>
<td>AD</td>
</tr>
<tr>
<td>MSA</td>
<td>my secondary address</td>
<td>SE</td>
</tr>
<tr>
<td>NUL</td>
<td>null byte</td>
<td>DD</td>
</tr>
<tr>
<td>OSA</td>
<td>other secondary address</td>
<td>SE</td>
</tr>
<tr>
<td>OTA</td>
<td>other talk address</td>
<td>AD</td>
</tr>
<tr>
<td>PCG</td>
<td>primary command group</td>
<td>AC</td>
</tr>
<tr>
<td>PPC</td>
<td>parallel poll configure</td>
<td>AC</td>
</tr>
<tr>
<td>PPE</td>
<td>parallel poll enable</td>
<td>AC</td>
</tr>
<tr>
<td>PPD</td>
<td>parallel poll disable</td>
<td>AC</td>
</tr>
<tr>
<td>PPR1</td>
<td>parallel poll response 1</td>
<td>AC</td>
</tr>
<tr>
<td>PPR2</td>
<td>parallel poll response 2</td>
<td>AC</td>
</tr>
</tbody>
</table>

Notes:
1. U = Universal
2. E = End of String
3. L = Local
4. H = My Talk
5. S = My Secondary
6. P = Parallel Poll Enable
7. F = Parallel Poll Disable
8. N = NUL
9. V = \( \text{ACG \lor UCG \lor LAG \lor TAG} \)
10. D = Device Clear
11. I = Identify
12. O = Other Talk
13. C = Command Group
14. A = Address Group
15. S = Secondary Address
16. T = Talk Address
### Bus Signal Line(s) and Coding That Asserts the True Value of the Message

| C | T | I | D | N | y | a | l | D | R | D | A | E | S | I | R | p | s | o | O | A | F | A | T | O | R | F | E |
| 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | VDC | N | Q | C | N | PPR3 | parallel poll response 3 | U | ST | X | X | X | X | X | X | XXX | 1 | 1 | X | X | X |
| 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | VDC | N | Q | C | N | PPR4 | parallel poll response 4 | U | ST | X | X | X | X | X | X | XXX | 1 | 1 | X | X | X |
| 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | VDC | N | Q | C | N | PPR5 | parallel poll response 5 | U | ST | X | X | X | X | X | X | XXX | 1 | 1 | X | X | X |
| 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | VDC | N | Q | C | N | PPR6 | parallel poll response 6 | U | ST | X | X | X | X | X | X | XXX | 1 | 1 | X | X | X |
| 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | VDC | N | Q | C | N | PPR7 | parallel poll response 7 | U | ST | X | X | X | X | X | X | XXX | 1 | 1 | X | X | X |
| 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | VDC | N | Q | C | N | PPR8 | parallel poll response 8 | U | ST | X | X | X | X | X | X | XXX | 1 | 1 | X | X | X |
| 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | VDC | N | Q | C | N | PPU | parallel poll unconfigure | M | UC | Y | 0 | 0 | 1 | 0 | 1 | 0 | 1 | XXX | 1 | X | X | X |
| 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | VDC | N | Q | C | N | REN | remote enable | U | UC | X | X | X | X | X | X | XXX | X | X | X | 1 |
| 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | VDC | N | Q | C | N | RFD | ready for data | U | HS | X | X | X | X | X | X | XXX | X | X | X | X |
| 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | VDC | N | Q | C | N | RQS | request service | U | ST | X | X | X | X | X | X | XXX | X | X | X | 1 |
| 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | VDC | N | Q | C | N | SCG | secondary command group | M | SE | Y | 1 | 1 | X | X | X | X | XXX | 1 | X | X | X |
| 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | VDC | N | Q | C | N | SDC | selected device clear | M | AC | Y | 0 | 0 | 0 | 1 | 0 | 0 | 1 | XXX | 1 | X | X | X |
| 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | VDC | N | Q | C | N | SPD | serial poll disable | M | UC | Y | 0 | 0 | 1 | 1 | 0 | 0 | 1 | XXX | 1 | X | X | X |
| 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | VDC | N | Q | C | N | SPE | serial poll enable | M | UC | Y | 0 | 0 | 1 | 1 | 0 | 0 | 0 | XXX | 1 | X | X | X |
| 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | VDC | N | Q | C | N | SRQ | service request | U | ST | X | X | X | X | X | X | XXX | X | X | X | 1 |
| 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | VDC | N | Q | C | N | STB | status byte | (Note 9) | M | ST | S | S | S | S | S | S | S | S | S | X | X | X | X |
| 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | VDC | N | Q | C | N | TCT | take control | M | AC | Y | 0 | 0 | 0 | 1 | 0 | 0 | 1 | XXX | 1 | X | X | X |
| 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | VDC | N | Q | C | N | TAG | talk address group | M | AD | Y | 1 | 0 | X | X | X | X | XXX | 1 | X | X | X |
| 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | VDC | N | Q | C | N | UCG | universal command group | M | UC | Y | 0 | 0 | 1 | X | X | X | XXX | 1 | X | X | X |
| 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | VDC | N | Q | C | N | UNL | unlisten | M | AD | Y | 0 | 1 | 1 | 1 | 1 | 1 | XXX | 1 | X | X | X |
| 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | VDC | N | Q | C | N | UNT | untalk | (Note 11) | M | AD | Y | 1 | 0 | 1 | 1 | 1 | 1 | 1 | XXX | 1 | X | X | X |

The 1/0 coding on ATN when sent concurrent with multiline messages has been added to this revision for interpretive convenience.

### NOTES:
1. D1–D8 specify the device dependent data bits.
2. E1–E8 specify the device dependent code used to indicate the EOS message.
3. L1–L5 specify the device dependent bits of the device's listen address.
4. T1–T5 specify the device dependent bits of the device's talk address.
5. S1–S5 specify the device dependent bits of the device's secondary address.
6. S specifies the sense of the PPR.

<table>
<thead>
<tr>
<th>S</th>
<th>Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

P1–P3 specify the PPR message to be sent when a parallel poll is executed.

<table>
<thead>
<tr>
<th>P3</th>
<th>P2</th>
<th>P1</th>
<th>PPR Message</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>PPR1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>PPR8</td>
</tr>
</tbody>
</table>

7. D1–D4 specify don't-care bits that shall not be decoded by the receiving device. It is recommended that all zeroes be sent.
8. S1–S6, S8 specify the device dependent status. (DIO7 is used for the RQS message.)
9. The source of the message on the ATN line is always the C function, whereas the messages on the DIO and EOI lines are enabled by the T function.
10. The source of the messages on the ATN and EOI lines is always the C function, whereas the source of the messages on the DIO lines is always the PP function.
11. This code is provided for system use, see 6.3.
89024
2400 BPS INTELLIGENT MODEM CHIP SET

- For Public Switched Telephone Network and Unconditioned Leased Lines Applications
- V.22 bis, V.22 A/B, V.21, Bell 212A, and Bell 103 Compatible
- Serial Command Set Compatible with Hayes Smartmodem 2400
- Automatically Adapts to Remote Modem Type with Recognition of Data Rates
- DTMF and Pulse Dialing, with Automatic Selection of Dial Signaling
- On-Chip Hybrid and Billing Delay Timer
- On-Chip Serial Port and Handshake Signals for RS-232/V.24 Interface
- Telephone Line Audio Monitor Output
- Analog/Digital Loopback Diagnostics with Mark/Space Pattern Generation and Error Detection
- Simple Serial Interface to External NVRAM

- Easily Customized Command Set and Features
- Two Chip Intelligent Modem Solution with Minimal External Components
- No External µC Required
- Output Level Programmable over 16 dB Range
- Dial and Re-dial Capability
- Full Set of Control Signals for DAA Interface
- Local, External, or Slave Timing Options in Synchronous Mode
- Adaptive Equalization
- Capable of Detecting Dial, Busy, Ringback and Modem Answer Tones of Most International Networks
- Auxiliary Relay Control Output

---

**Figure 1. 89024 System Block Diagram**
GENERAL DESCRIPTION

The Intel 89024 chip set is a highly integrated, high performance, intelligent modem, providing a complete system in two chips. The system conforms to the following CCITT and Bell standards:

- CCITT V.22 bis
  2400 bps sync and async
  1200 bps sync and async (fall-back)
- CCITT V.22 A & B
  1200 bps sync and async
  600 bps sync and async (fall-back)
- CCITT V.21
  0 to 300 bps anisochronous
- BELL 212A
  1200 bps sync and async
  300 bps fall-back mode
- BELL 103
  0 to 300 bps anisochronous

The 89024 system consists of a 16 bit application specific processor (89026) and an analog front end device (89027). The 89026 processor performs all "Digital Signal Processing" algorithm execution for processing the modem signals, as well as providing all modem control functions typically performed by an external processor. The analog front end provides for 2 wire and 4 wire telephone line interface, D/A conversion, and most of the complex filtering functions required in QAM/DPSK/FSK modems. Refer to Figure 1 for a simplified block diagram of the system.

In stand-alone modem applications, the 89024 chip set along with a Data Access Arrangement (DAA), a serial NVRAM, and RS-232 driver/receivers, represent the circuitry required for implementing an auto-dial, auto-answer, 300 to 2400 bps, full duplex Hayes compatible intelligent modem.

A complete set of industry standard AT commands is provided for modem configuration and user interface. Virtually all PC software written for the Hayes Smartmodem 2400 can also be used with this chip set. Alternatively, in applications where user proprietary modem control commands and features are desired, the user can replace the 89024 internal command module with custom proprietary software resident in the 89026 microcontroller's on-chip ROM or an external memory device.

The 89024 has a set of default features. Upon power up, the modem configuration will be in accordance with these default options, unless a different configuration has been saved in the external NVRAM with the &W command.

The 89024 modem has built in auto-dialing and auto-answering capabilities. It can be configured to the proper line signaling mode (Tone or Pulse), and to the type (CCITT or Bell) and speed of the calling or answering modem. It can also detect and identify call set-up signals of telephone networks, allowing unattended data call operation.

A full set of diagnostic loop-test features compatible with CCITT V.54 is supported. The chip set also provides a line signal for audio monitoring of call progress, a comprehensive set of DAA control lines for a simple interface to the telephone network, and a full complement of TTL level RS-232/ V.24 handshake signals.

NOTICE:
Hayes is a registered trademark of Hayes Microcomputer Products, Inc.
Smartmodem 2400 is a trademark of Hayes Microcomputer Products, Inc.
Smartcom II is a registered trademark of Hayes Microcomputer Products, Inc.
PACKAGING

Both devices are available in standard DIP packages as well as PLCC packages for surface mount applications.

Figure 2. Device Packages
CALL ESTABLISHMENT, TERMINATION AND RETRAIN

The 89024 modem system incorporates all protocols and functions required for automatic (or manual) establishment, progress and termination of a data call.

The modem chip-set has a built in auto-dialer, both DTMF and Pulse type, and is capable of automatically adapting to the telephone dial type. The dialing sequence on the telephone link conforms to the CCITT V.25 recommendations. An exception to the V.25 is that the interrupted calling tone will not be transmitted by the calling modem, as is suggested in V.22 bis.

The modem can detect the dial, busy and ringback signals at remote end, and will provide call progress messages to the user. The modem is capable of redialing the last number dialed, by one command.

The modem when configured for auto-answer, will answer an incoming call, remain silent for the two second billing delay interval, before transmitting the answer tones. Afterwards modem to modem identification and handshaking will proceed at a speed and operating mode acceptable to both ends of the link.

The data call can also be setup by manual dialing with the modems set to data mode, or by voice to data transfer by means of mechanical switch (exclusion key), using the SH pin. Once set to data mode, the modem handshaking will proceed before the modems will be ready to accept and exchange data.

During data transmission, if one of the modems finds that the received data is likely to have a high bit error rate (indicated by a large mean square error in the adaptive equalizer), it initiates a retrain sequence. This automatic retrain feature is only available at 2400 bps, and conforms to CCITT V.22 bis recommendations.

Disconnection of the data call can be initiated by the DTE at the local end, or by the remote DTE, (if the modem is configured to accept it). Whether DTR will initiate a disconnect, depends on the last &D command. Receiving a long space from a remote modem will initiate a disconnect only after a Y1 command. The optional disconnect requests originated by the remote modem, are of two types, (1) disconnect when receiving long-space, and (2) disconnect when received carrier is dropped. The modem chip-set can also be configured to transmit 'long-space' just before disconnection, in each of the aforementioned cases.

Because the CCITT and Bell modem connection protocols are quite different from each other and do not provide recognition of remote modem type (i.e. V.22 bis to 212A), the Intel chip-set provides the additional capability of identifying the remote modem type. This feature is beneficial during the migration phase of the technology from the 1200 bps to 2400 bps. In North America, where the installed base of 1200 bps modems is mostly made-up of 212A type, this feature allows a "Data Base Service Provider" to easily upgrade the existing 212A modems to 2400 bps V.22 bis standard, transparently, to 212A users. Similarly, a user with a 89024 based modem system can automatically call data bases with either 212A or V.22 bis modems, without concern over the difference. This feature's benefits are realized in smooth upgrading of data links, with minimum cost and reduced disruption in services. Refer to Table 1 for a detailed description of remote modem compatibility.

SOFTWARE CONFIGURATION COMMANDS

This section lists the 89024 commands and registers that may be used while configuring the modem. Commands instruct the modem to perform an action, the value in the associated registers determine how the commands are performed, and the result codes returned by the modem tell the user about the execution of the commands.

The commands may be entered in a string, with or without spaces in between. Any spaces within or between commands will be ignored by the modem. During the entry of any command, the 'backspace' key (CNTRL H) can be used to correct any error. Upper case or lower case characters can be used in the commands. Commands described in the following paragraphs refer to asynchronous terminals using ASCII codes.
### Table 1. Remote Modem Compatibility

<table>
<thead>
<tr>
<th>Originating Modem</th>
<th>Bell Modem</th>
<th>Answer Modem</th>
</tr>
</thead>
<tbody>
<tr>
<td>89024</td>
<td>300 Bell</td>
<td>CCITT CCITT</td>
</tr>
<tr>
<td></td>
<td>1200</td>
<td>300 300</td>
</tr>
<tr>
<td>CCITT</td>
<td>600</td>
<td>300 600 1200</td>
</tr>
<tr>
<td></td>
<td>1200</td>
<td>1200 1200 1200</td>
</tr>
<tr>
<td></td>
<td>2400</td>
<td>1200 1200 2400</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Answering Modem</th>
<th>Bell Modem</th>
<th>Originating Modem</th>
</tr>
</thead>
<tbody>
<tr>
<td>89024</td>
<td>300 Bell</td>
<td>CCITT CCITT CCITT</td>
</tr>
<tr>
<td></td>
<td>1200</td>
<td>300 300 300 600</td>
</tr>
<tr>
<td>CCITT</td>
<td>600</td>
<td>300 600 1200 1200</td>
</tr>
<tr>
<td></td>
<td>1200</td>
<td>300 1200 1200 1200</td>
</tr>
<tr>
<td></td>
<td>2400</td>
<td>300 1200 1200 2400</td>
</tr>
</tbody>
</table>

* These connection data rates are obtained when connecting 89024 based modems end to end. The same results may not be obtained when a 89024 based modem is connected to other modems.

---

### Command Set

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AT</td>
<td>Attention code.</td>
</tr>
<tr>
<td>A</td>
<td>Go off-hook in answer mode</td>
</tr>
<tr>
<td>A/</td>
<td>Repeat previous command string</td>
</tr>
<tr>
<td>Bn</td>
<td>BELL/CCITT Protocol Compatibility at 1200 bps</td>
</tr>
<tr>
<td>Ds</td>
<td>The dialing commands (0-9 A B C D * # P R T S W ; : @)</td>
</tr>
<tr>
<td>En</td>
<td>Echo command (En)</td>
</tr>
<tr>
<td>Hn</td>
<td>Switch-Hook Control</td>
</tr>
<tr>
<td>In</td>
<td>Request Product Code and Checksum</td>
</tr>
<tr>
<td>Ln</td>
<td>Speaker Volume</td>
</tr>
<tr>
<td>Mn</td>
<td>Monitor On/Off</td>
</tr>
<tr>
<td>O</td>
<td>On-Line</td>
</tr>
<tr>
<td>Qn</td>
<td>Result Codes</td>
</tr>
<tr>
<td>Sn=x</td>
<td>Write S Register</td>
</tr>
<tr>
<td>Sn</td>
<td>Read S Register</td>
</tr>
<tr>
<td>Vn</td>
<td>Enable Short-Form Result Codes</td>
</tr>
<tr>
<td>Xn</td>
<td>Enable Extended Result Code</td>
</tr>
<tr>
<td>Yn</td>
<td>Enable Long Space Disconnect</td>
</tr>
<tr>
<td>Z</td>
<td>Fetch Configuration Profile</td>
</tr>
<tr>
<td>+++</td>
<td>The Default Escape Code</td>
</tr>
</tbody>
</table>

### & Command Set (Continued)

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&amp;C</td>
<td>DCD Options</td>
</tr>
<tr>
<td>&amp;D</td>
<td>DTR Options</td>
</tr>
<tr>
<td>&amp;F</td>
<td>Fetch Factory Configuration Profile</td>
</tr>
<tr>
<td>&amp;G</td>
<td>Guard Tone</td>
</tr>
<tr>
<td>&amp;J</td>
<td>Telephone Jack Selection</td>
</tr>
<tr>
<td>&amp;L</td>
<td>Leased/Dial-up Line Selection</td>
</tr>
<tr>
<td>&amp;M</td>
<td>Async/Sync Mode Selection</td>
</tr>
<tr>
<td>&amp;P</td>
<td>Make/Break Pulse Ratio</td>
</tr>
<tr>
<td>&amp;R</td>
<td>RTS/CTS Options</td>
</tr>
<tr>
<td>&amp;S</td>
<td>DSR Options</td>
</tr>
<tr>
<td>&amp;T</td>
<td>Test Commands</td>
</tr>
<tr>
<td>&amp;W</td>
<td>Write Configuration to Non Volatile Memory</td>
</tr>
<tr>
<td>&amp;X</td>
<td>Sync Clock Source</td>
</tr>
<tr>
<td>&amp;Z</td>
<td>Store Telephone Number</td>
</tr>
</tbody>
</table>
CONFIGURATION REGISTERS

The modem stores all the configuration information in a set of registers. Some registers are dedicated to special command and function, and others are bit-mapped, with different commands sharing the register space to store the command status.

S  *  Ring to Answer
S1  Ring Count. (Read Only)
S2  Escape Code Character
S3  Carriage Return Character
S4  Line Feed Character
S5  Back Space Character
S6  Wait for Dial Tone
S7  Wait for Data Carrier
S8  Pause Time for the Comma Dial Modifier
S9  Carrier Detect Response Time
S10 Lost Carrier to Hang Up Delay
S11 Not Used
S12 Escape Code Guard Time
S13 Not Used
S14  *  Bit Mapped Option Register
S15 Not Used
S16 Modem Test Options
S17 Not Used
S18  *  Test Timer
S19 Not Used
S20 Not Used
S21  *  Bit Mapped Options Register
S22  *  Bit Mapped Options Register
S23  *  Bit Mapped Options Register
S24 Not Used
S25  *  Delay to DTR (Sync Only)
S26  *  RTS to CTS Delay (Half Dup.)
S27  *  Bit Mapped Options Register

NOTE:
* These S registers can be stored in the NVRAM.

### Dial Modifiers

<table>
<thead>
<tr>
<th></th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>P</td>
<td>Pulse Dial</td>
</tr>
<tr>
<td>R</td>
<td>Originate call in Answer Mode</td>
</tr>
<tr>
<td>T</td>
<td>Tone Dial</td>
</tr>
<tr>
<td>S</td>
<td>Dial a stored number</td>
</tr>
<tr>
<td>W</td>
<td>Wait for dial tone</td>
</tr>
<tr>
<td>;</td>
<td>Delay a dial sequence</td>
</tr>
<tr>
<td>@</td>
<td>Return to command state</td>
</tr>
<tr>
<td></td>
<td>Initiate a flash</td>
</tr>
<tr>
<td></td>
<td>Wait for quit</td>
</tr>
</tbody>
</table>

Example:

Terminal: AT &Z T 1 (602) 555-1212
Modem: OK
Result: Modem stores T16025551212 in the external NVRAM.

The number can be dialed from asynchronous mode by issuing the following command:

Terminal: AT DS
Modem: T16025551212

or by turning on DTR when in Synchronous Mode 2. Up to 33 symbols (dial digits and dial modifiers) may be stored. Spaces and other delimiters are ignored and do not need to be included in the count. If more than 33 symbols are supplied, the dial string will be truncated to 33.

### APPLICATIONS OVERVIEW

The block diagram of a stand-alone 300 to 2400 bps Hayes compatible modem is depicted in Figure 3. The DAA section shown in this diagram may be obtained with FCC registration, or implemented using the suggested diagram in Figure 4.
NOTE:
Pin #22 and pin #19 are NO CONNECT for the 89024 modem application.

Figure 3. Typical Modem Configuration

Figure 4. Typical Telephone Line Interface with Built In Hybrid
### SYSTEM COMPATIBILITY SPECIFICATIONS

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Specification</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Synchronous</strong></td>
<td><strong>Specification</strong></td>
</tr>
<tr>
<td></td>
<td>2400 bps ± 0.01% V.22 bis</td>
</tr>
<tr>
<td></td>
<td>1200 bps ± 0.01% V.22 and BELL 212A</td>
</tr>
<tr>
<td></td>
<td>600 bps ± 0.01% V.22 A,B</td>
</tr>
<tr>
<td>Asynchronous</td>
<td>2400, 1200, 600 bps, character asynchronous.</td>
</tr>
<tr>
<td>Asynchronous Speed Range</td>
<td>+1% − 2.5% default. Extended +2.3% −2.5% range of CCITT standards optional</td>
</tr>
<tr>
<td>Asynchronous Format</td>
<td>8,9,10,11 bits, including start, stop, parity. Bits 8, 9, 11 optional</td>
</tr>
<tr>
<td>Asynchronous Timing Source</td>
<td>Internal, derived from the local oscillator.</td>
</tr>
<tr>
<td></td>
<td>External, provided by DTE through XTCLK.</td>
</tr>
<tr>
<td></td>
<td>Slave, derived from the received clock.</td>
</tr>
<tr>
<td>Telephone Line Interface</td>
<td>Two wire full duplex over public switched network or 4 wire leased lines.</td>
</tr>
<tr>
<td></td>
<td>On-chip hybrid and billing delay timers.</td>
</tr>
<tr>
<td></td>
<td>Output level −1 to −16 dBm</td>
</tr>
<tr>
<td>Modulation</td>
<td>V.22 bis, 16 point QAM at 600 baud.</td>
</tr>
<tr>
<td></td>
<td>V.22 and 212A, 4 point DPSK at 600 baud.</td>
</tr>
<tr>
<td></td>
<td>V.21 and 103, binary phase coherent FSK</td>
</tr>
<tr>
<td>Output Spectral Shaping</td>
<td>Square root of 75% raised cosine, QAM/PSK.</td>
</tr>
<tr>
<td>Transmit Carrier Frequencies</td>
<td><strong>Specify the carrier frequencies</strong></td>
</tr>
<tr>
<td>V.22 bis, V.22, 212A</td>
<td><strong>Origin</strong> 1200 Hz ± 0.01%</td>
</tr>
<tr>
<td></td>
<td><strong>Answer</strong> 2400 Hz ± 0.01%</td>
</tr>
<tr>
<td>V.21</td>
<td><strong>Origin 'space'</strong> 1180 Hz ± 0.01%</td>
</tr>
<tr>
<td></td>
<td><strong>Origin 'mark'</strong> 980 Hz ± 0.01%</td>
</tr>
<tr>
<td></td>
<td><strong>Answer 'space'</strong> 1850 Hz ± 0.01%</td>
</tr>
<tr>
<td></td>
<td><strong>Answer 'mark'</strong> 1650 Hz ± 0.01%</td>
</tr>
<tr>
<td>Bell 103 mode</td>
<td><strong>Origin 'space'</strong> 1070 Hz ± 0.01%</td>
</tr>
<tr>
<td></td>
<td><strong>Origin 'mark'</strong> 1270 Hz ± 0.01%</td>
</tr>
<tr>
<td></td>
<td><strong>Answer 'space'</strong> 2020 Hz ± 0.01%</td>
</tr>
<tr>
<td></td>
<td><strong>Answer 'mark'</strong> 2225 Hz ± 0.01%</td>
</tr>
<tr>
<td>Receive Carrier Frequency Limits</td>
<td><strong>Specify the receive carrier frequency limits</strong></td>
</tr>
<tr>
<td>V.22 bis, V.22, 212A</td>
<td><strong>Origin</strong> 2400 Hz ± 7 Hz</td>
</tr>
<tr>
<td></td>
<td><strong>Answer</strong> 1200 Hz ± 7 Hz</td>
</tr>
<tr>
<td>V.21</td>
<td><strong>Origin 'space'</strong> 1850 Hz ± 12 Hz</td>
</tr>
<tr>
<td></td>
<td><strong>Origin 'mark'</strong> 1650 Hz ± 12 Hz</td>
</tr>
<tr>
<td></td>
<td><strong>Answer 'space'</strong> 1180 Hz ± 12 Hz</td>
</tr>
<tr>
<td></td>
<td><strong>Answer 'mark'</strong> 960 Hz ± 12 Hz</td>
</tr>
<tr>
<td>Bell 103</td>
<td><strong>Origin 'space'</strong> 2020 Hz ± 12 Hz</td>
</tr>
<tr>
<td></td>
<td><strong>Origin 'mark'</strong> 2225 Hz ± 12 Hz</td>
</tr>
<tr>
<td></td>
<td><strong>Answer 'space'</strong> 1070 Hz ± 12 Hz</td>
</tr>
<tr>
<td></td>
<td><strong>Answer 'mark'</strong> 1270 Hz ± 12 Hz</td>
</tr>
<tr>
<td>Energy Detect Sensitivity</td>
<td>Greater than −43 dBm ED is ON. Less than −48 dBm ED is OFF. Signal in dBm</td>
</tr>
<tr>
<td></td>
<td>measured at A02.</td>
</tr>
<tr>
<td>Line Equalization</td>
<td><strong>Specify line equalization</strong></td>
</tr>
<tr>
<td></td>
<td>Fixed compromise equalization, transmit.</td>
</tr>
<tr>
<td></td>
<td>Adaptive equalizer for DPSK/QAM, receive.</td>
</tr>
<tr>
<td>Diagnostics Available</td>
<td><strong>Specify diagnostics available</strong></td>
</tr>
<tr>
<td></td>
<td>Local analog loopback.</td>
</tr>
<tr>
<td></td>
<td>Local digital loopback.</td>
</tr>
<tr>
<td></td>
<td>Remote digital loopback.</td>
</tr>
<tr>
<td></td>
<td>Local interface loopback.</td>
</tr>
<tr>
<td>Self Test Pattern Generator</td>
<td><strong>Specify self test pattern generator</strong></td>
</tr>
<tr>
<td></td>
<td>Alternate 'ones' and 'zeros' and error detector, to be used along with most</td>
</tr>
<tr>
<td></td>
<td>loopbacks.</td>
</tr>
<tr>
<td></td>
<td>A number indicating the bit errors detected is sent to DTE.</td>
</tr>
</tbody>
</table>

4-8
**RECEIVER PERFORMANCE SPECIFICATIONS**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Specification</th>
</tr>
</thead>
<tbody>
<tr>
<td>Test condition: Unconditioned 3002 line, across the full dynamic range. The noise bandwidth is 3 KHz flat.</td>
<td></td>
</tr>
<tr>
<td>Random Noise</td>
<td>Typical Bit Error rate of 1 in 100000 or better at 12 dB SNR at 300 bps, 5 dB SNR at 600 bps, 8 dB SNR at 1200 bps and 16 dB SNR at 2400 bps.</td>
</tr>
<tr>
<td>Frequency Offsets(1)</td>
<td>± 7 Hz.</td>
</tr>
<tr>
<td>Phase Jitter(1)</td>
<td>2400 bps - 15° peak to peak, at up to 300 Hz. 600, 1200 bps - 45° peak to peak, at up to 300 Hz.</td>
</tr>
</tbody>
</table>

**NOTE:**
1. There are no observable data errors for the received signals, for the above limits of line impairments. These impairments are applied one at a time in absence of noise.

---

**PERFORMANCE SPECIFICATIONS**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Min</th>
<th>Typ</th>
<th>Max</th>
<th>Units</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>DTMF Level</td>
<td>1.0</td>
<td></td>
<td></td>
<td>dBm</td>
<td>at AO1</td>
</tr>
<tr>
<td>DTMF Second Harmonic</td>
<td></td>
<td></td>
<td>-35</td>
<td>dB</td>
<td>HYB enabled into 600Ω</td>
</tr>
<tr>
<td>DTMF Twist (Balance)</td>
<td>3</td>
<td></td>
<td></td>
<td>dB</td>
<td></td>
</tr>
<tr>
<td>DTMF Duration</td>
<td>100</td>
<td></td>
<td></td>
<td>ms</td>
<td>Software Controlled</td>
</tr>
<tr>
<td>Pulse Dialing Rate</td>
<td>10</td>
<td></td>
<td></td>
<td>pps</td>
<td></td>
</tr>
<tr>
<td>Pulse Dialing Make/Break</td>
<td>39/61</td>
<td>%</td>
<td></td>
<td>%</td>
<td>US, UK, Hong Kong</td>
</tr>
<tr>
<td>Pulse Interdigit Interval</td>
<td>785</td>
<td></td>
<td></td>
<td>ms</td>
<td></td>
</tr>
<tr>
<td>Billing Delay Interval</td>
<td>2.1</td>
<td></td>
<td></td>
<td>sec</td>
<td></td>
</tr>
<tr>
<td>Guard Tone Frequency Amplitude</td>
<td>540</td>
<td>Hz</td>
<td></td>
<td>reference to High channel transmit. QAM/DPSK Modes Only</td>
<td></td>
</tr>
<tr>
<td>Guard Tone Frequency Amplitude</td>
<td>-3</td>
<td>dB</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Guard Tone Frequency Amplitude</td>
<td>1800</td>
<td>Hz</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Guard Tone Frequency Amplitude</td>
<td>-6</td>
<td>dB</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Dial Tone Detect Duration</td>
<td>3.0</td>
<td>sec</td>
<td></td>
<td>Off/On Ratio</td>
<td></td>
</tr>
<tr>
<td>Ringback Tone Detect</td>
<td>0.75</td>
<td>sec</td>
<td></td>
<td>Off/On Ratio</td>
<td></td>
</tr>
<tr>
<td>Ringback Tone Detect</td>
<td>1.5</td>
<td>sec</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Busy Tone Detect Duration</td>
<td>0.67</td>
<td>sec</td>
<td>1.5</td>
<td>Off/On Ratio</td>
<td></td>
</tr>
<tr>
<td>Busy Tone Detect Cadence</td>
<td>0.2</td>
<td>sec</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
89026 OVERVIEW

The 89026 processor performs data manipulation, signal processing and user interface functions. It supports an external ROM, for user designed software. This option allows customer designed code to control the signal processing algorithms resident in the 89026. For example proprietary modem control and call progress management applications can be implemented using EPROMs or alternatively by having it burnt in the processor ROM (done so by Intel factory contracting). On-chip ROM is 8 Kbytes. A block diagram of 89026 is in Figure 5.

89026 contains a TTL compatible serial link to DTE/DCE equipment, along with a full complement of V.24/RS-232-C control signals. Alternatively, UART or USART may be used to directly transfer data to and from a microcomputer bus. The industry standard AT command set is supported by the 89026, facilitating communications compatibility between 89024 and most PC software written for the AT command set.

In the transmit operation, the 89026 synthesizes DTMF tones and the 300 bps FSK modem signal prior to transmitting them to the 89027 as digitized amplitude samples. During 1200 and 2400 bps operation, DPSK and QAM is used to send 2 or 4 bits of information respectively at 800 baud to 89027. Since the QAM coding technique is an inherently synchronous transmission mechanism, during asynchronous QAM transmission, the asynchronous data is synchronized by adding or deleting stop bits. Following the synchronization process, the 89026 transmits digitized phase and amplitude samples to 89027 over a high speed serial link.

In the receive operation, the information is received by 89026 from 89027 as two signals which are 90 degrees phase shifted from each other. These analog signals are then digitized by the 89026's on-board A/D converter, and using DSP software algorithms the signals are gain adjusted, adaptively equalized for telephone line delay and amplitude distortion, and demodulated. Following the demodulation process by the 89026, the data is unscrambled, and if necessary, returned to asynchronous format.

---

Figure 5. 89026 Block Diagram
### 89026 PINOUT

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Function (89026)</th>
<th>Direction</th>
<th>Pin No.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>68 pin</td>
<td></td>
</tr>
<tr>
<td>CLKN</td>
<td>12.96 MHz master clock from 89027</td>
<td>In</td>
<td>67</td>
</tr>
<tr>
<td>RST</td>
<td>Chip reset (active low)</td>
<td>In</td>
<td>16</td>
</tr>
<tr>
<td>I</td>
<td>In-phase received signal</td>
<td>In</td>
<td>11</td>
</tr>
<tr>
<td>Q</td>
<td>Quadrature-phase received signal</td>
<td>In</td>
<td>10</td>
</tr>
<tr>
<td>STR</td>
<td>Symbol Timing from 89027</td>
<td>In</td>
<td>24</td>
</tr>
<tr>
<td>ED</td>
<td>Energy Detect input</td>
<td>In</td>
<td>9</td>
</tr>
<tr>
<td>TSYNC</td>
<td>Transmitter sync pulse to 89027</td>
<td>Out</td>
<td>35</td>
</tr>
<tr>
<td>SDATA</td>
<td>Serial Data to 89027</td>
<td>Out</td>
<td>17</td>
</tr>
<tr>
<td>SCLK</td>
<td>Serial Clock to 89027</td>
<td>Out</td>
<td>18</td>
</tr>
<tr>
<td>OH</td>
<td>Off-Hook control to DAA</td>
<td>Out</td>
<td>33</td>
</tr>
<tr>
<td>SH</td>
<td>Switch-Hook from dataphone</td>
<td>In</td>
<td>44</td>
</tr>
<tr>
<td>RI</td>
<td>Ring Indicator from DAA</td>
<td>In</td>
<td>42</td>
</tr>
<tr>
<td>AR</td>
<td>Aux Relay control to DAA</td>
<td>Out</td>
<td>38</td>
</tr>
<tr>
<td>TCL1</td>
<td>NVRAM Data I/O</td>
<td>I/O</td>
<td>20</td>
</tr>
<tr>
<td>TCL0</td>
<td>NVRAM CLK</td>
<td>Out</td>
<td>19</td>
</tr>
<tr>
<td>B/C</td>
<td>103/V.21 default option</td>
<td>In</td>
<td>15</td>
</tr>
<tr>
<td>S/A</td>
<td>NVRAM CE</td>
<td>Out</td>
<td>21</td>
</tr>
<tr>
<td>D/S</td>
<td>Dumb/Smart mode select</td>
<td>In</td>
<td>6</td>
</tr>
<tr>
<td>CONFIG</td>
<td>Custom Firmware Disable</td>
<td>In</td>
<td>8</td>
</tr>
<tr>
<td>TM</td>
<td>Test Mode Indicator</td>
<td>Out</td>
<td>39</td>
</tr>
<tr>
<td>TXD</td>
<td>Transmitted data from DTE</td>
<td>In</td>
<td>27</td>
</tr>
<tr>
<td>RXD</td>
<td>Received data to DTE</td>
<td>Out</td>
<td>29</td>
</tr>
<tr>
<td>RTS</td>
<td>Request to send from DTE</td>
<td>In</td>
<td>22</td>
</tr>
<tr>
<td>CTS</td>
<td>Clear to Send to DTE</td>
<td>Out</td>
<td>23</td>
</tr>
<tr>
<td>TSR</td>
<td>Data Set Ready to DTE</td>
<td>Out</td>
<td>30</td>
</tr>
<tr>
<td>DCD</td>
<td>Data Carrier Detect to DTE</td>
<td>Out</td>
<td>31</td>
</tr>
<tr>
<td>DTR</td>
<td>Data Terminal Ready from DTE</td>
<td>In</td>
<td>25</td>
</tr>
<tr>
<td>RCLK</td>
<td>Received clock to DTE</td>
<td>Out</td>
<td>34</td>
</tr>
<tr>
<td>TCLK</td>
<td>Transmit clock to DTE</td>
<td>Out</td>
<td>28</td>
</tr>
<tr>
<td>XTCLK</td>
<td>External timing clock from DTE</td>
<td>In</td>
<td>26</td>
</tr>
<tr>
<td>SI</td>
<td>Speed Indicator to DTE</td>
<td>Out</td>
<td>32</td>
</tr>
<tr>
<td>SS</td>
<td>Speed select from DTE(4)</td>
<td>In</td>
<td>5</td>
</tr>
<tr>
<td>RELMB</td>
<td>Remote Loopback Command from DTE</td>
<td>In</td>
<td>7</td>
</tr>
<tr>
<td>LCLLB</td>
<td>Local Loopback Command from DTE</td>
<td>In</td>
<td>4</td>
</tr>
<tr>
<td>VCC</td>
<td>Positive power supply (+5V)</td>
<td>+5V</td>
<td>1</td>
</tr>
<tr>
<td>VPD</td>
<td>Ram back-up power</td>
<td>+5V</td>
<td>14</td>
</tr>
<tr>
<td>VREF</td>
<td>A/D converter reference</td>
<td>+5V</td>
<td>13</td>
</tr>
<tr>
<td>VSS1</td>
<td>Digital ground</td>
<td>GND</td>
<td>36</td>
</tr>
<tr>
<td>VSS2</td>
<td>Digital ground</td>
<td>GND</td>
<td>68</td>
</tr>
<tr>
<td>AGND</td>
<td>Analog ground</td>
<td>AGND</td>
<td>12</td>
</tr>
<tr>
<td>VBB</td>
<td>Back-bias generator output</td>
<td>Out</td>
<td>37</td>
</tr>
<tr>
<td>EA</td>
<td>External Memory enable</td>
<td>In</td>
<td>2</td>
</tr>
<tr>
<td>AD0-AD15</td>
<td>External memory access address/data(5)</td>
<td>I/O</td>
<td>60-45</td>
</tr>
<tr>
<td>AA</td>
<td>Auto Answer</td>
<td>Out</td>
<td>60</td>
</tr>
<tr>
<td>JS</td>
<td>Jack Select</td>
<td>Out</td>
<td>59</td>
</tr>
</tbody>
</table>
### 89026 PINOUT (Continued)

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Function (89026)</th>
<th>Direction</th>
<th>Pin No.</th>
</tr>
</thead>
<tbody>
<tr>
<td>NMI</td>
<td>No-maskable Interrupt(VSS)(^{(1)})</td>
<td>In</td>
<td>3</td>
</tr>
<tr>
<td>X2</td>
<td>Crystal output(NC)(^{(2)})</td>
<td>Out</td>
<td>66</td>
</tr>
<tr>
<td>CLKOUT</td>
<td>Clk output (NC)(^{(2)})</td>
<td>Out</td>
<td>65</td>
</tr>
<tr>
<td>TEST</td>
<td>Factory test(VCC)(^{(3)})</td>
<td>In</td>
<td>64</td>
</tr>
<tr>
<td>INST</td>
<td>External memory instruction fetch</td>
<td>Out</td>
<td>63</td>
</tr>
<tr>
<td>ALE</td>
<td>Address latch enable</td>
<td>Out</td>
<td>62</td>
</tr>
<tr>
<td>RD</td>
<td>External memory read</td>
<td>Out</td>
<td>61</td>
</tr>
<tr>
<td>READY</td>
<td>External memory ready(VCC)(^{(3)})</td>
<td>In</td>
<td>43</td>
</tr>
<tr>
<td>BHE</td>
<td>External memory bus high enable</td>
<td>Out</td>
<td>41</td>
</tr>
<tr>
<td>WR</td>
<td>External memory write</td>
<td>Out</td>
<td>40</td>
</tr>
</tbody>
</table>

**NOTES:**
1. Pins marked with (VSS) must be connected to VSS.
2. Pins marked with (NC) are to be left unconnected.
3. Pins marked with (VCC) must be connected to VCC.
4. SS pin reserved for future use.
5. With internal ROM enabled, AD0-AD1 are used as AX and JS.

### 89026 PIN DESCRIPTION

**XTCLK**
Transmitter timing from DTE, when external clock option is selected.

**TXD**
The serial data from DTE to be transmitted on the line. A logic 'high' is mark. In synchronous mode, 89026 samples this data on the rising edges of TCLK.

**TCLK**
Clock output from 89026 as timing source for data exchange from DTE to modem, Serial data is read on the rising edges of the TCLK. This output is High in asynchronous mode.

**RXD**
The serial data to DTE. 'Mark' is a logic High. In synchronous mode, the rising edge of RCLK occurs in the middle of RXD.

**RCLK**
Synchronous clock output. Rising edge of RCLK occurs in the middle of each RXD bit. This pin remains High in asynchronous mode.

**VBBS**
This pin to be connected to AGND through a 0.01 μF capacitor.

**TM**
A Low indicates maintenance condition in the modem.

**DCD**
In async operation, DCD remains Low regardless of data carrier (default), or it can be programmed to indicate received carrier signal is within the required timing and amplitude limits. In sync operation Low indicates the received carrier signal is within the required timing and amplitude limits.

**DSR**
Low indicates modem is off-hook, and it is in data transmission mode, and the answer tone is being exchanged. CTS Low indicates modem is prepared to accept data.

**RTS**
In async mode RTS is ignored. Under command control, in sync mode RTS can be ignored, or the modem can respond with a Low on CTS.

**DTR**
&D0 command will cause the modem to ignore DTR. For &D1 the modem assumes the asynchronous command state on a Low to High transition of the DTR circuit. The &D2 command does the same as &D1 except the state of DTR will enable/disable auto answer. A Low to High transition of DTR after the &D3 command will cause the modem to assume the initialization state.

**B/C**
Low configures the modem to CCITT V.21. High will configure the modem to Bell 103, when at 300 bps speed. This pin only affects the modem in 300 bps operation.
TCL1, TCL0
These pins are used as the serial clock and data for interface to an NVRAM. Refer to Figure 3. TCL0 is used to output a clock and serial data is read in on TCL1.

AR
This Auxiliary relay control is for switching a relay for voice or data calls. High is voice, Low is data.

RI
A Low signal from DAA indicates line ringing. This input is ignored when the modem is configured for leased line. This signal should follow the ring cadence.

OH
Low controls off hook. High indicates on hook. When dialing, this control is used to pulse dial the line.

SH
Used as a telephone voice to data switch or vice versa. Any logic level transition will toggle the modem state. This input is ignored, if a software command attempts to switch the modem between voice and data.

AA
Used as an indicator for Auto Answer status and Ring indicator. Active low.

LCLLB
A Low will set the modem in the local analog loopback test mode. Logic Low levels applied simultaneously to REMLB and LCLLB pins, sets the modem to the local digital loopback.

REMLB
A logic Low on this pin initiates a remote loopback condition.

S1
Selects one of the two data rates or ranges of rates in the DTE to correspond to the rate in modem. Low selects the higher rate (2400 CCITT/1200 BELL) or range of rates. High selects the Low rate or range of rates.

D/S
A Low on this pin will indicate the smart mode which will respond to all commands. A High will ignore all commands.

VREF
Voltage reference for the analog to digital converter should be connected to the 89027 AVcc.

VPD
The internal RAM power down supply voltage to be connected to 5 Volts during normal operation.

S/A
The function of this pin is re-defined as external NVRAM CE.

CONFIG
Low indicates availability of custom software modules in off-chip memory.

EA
When High, memory access from address 2000H to 4000H1 are directed to on-chip ROM. When Low, all memory access is directed to off-chip memory.

JS
Low is used to pulse A and A1 leads to control a 1A2 Key System jack.

89026 ABSOLUTE MAXIMUM RATINGS*
Temperature Under Bias .............. −10 to +80° C
Storage Temperature .............. −40 to +125° C
Voltage from Any Pin to
Vss or AGND .............. −0.3V to +7.0V
Average Output Current from Any Pin ....... 10 mA
Power Dissipation .............. 1.5 Watts

*Notice: Stresses above those listed under “Absolute Maximum Ratings” may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions above those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

NOTICE: Specifications contained within the following tables are subject to change.
# OPERATING CONDITIONS

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Parameter</th>
<th>Min</th>
<th>Max</th>
<th>Units</th>
</tr>
</thead>
<tbody>
<tr>
<td>TA</td>
<td>Ambient Temperature Under Bias</td>
<td>0</td>
<td>+70</td>
<td>°C</td>
</tr>
<tr>
<td>VCC</td>
<td>Digital Supply Voltage</td>
<td>4.75</td>
<td>5.25</td>
<td>V</td>
</tr>
<tr>
<td>VREF</td>
<td>Analog Supply Voltage</td>
<td>4.75</td>
<td>5.25</td>
<td>V</td>
</tr>
<tr>
<td></td>
<td></td>
<td>VCC -.3</td>
<td>VCC +.3</td>
<td>V</td>
</tr>
<tr>
<td>FREQ</td>
<td>CLkin Frequency 12.96 Mhz</td>
<td>-0.01%</td>
<td>+0.01%</td>
<td></td>
</tr>
<tr>
<td>VPDL</td>
<td>Power-Down Supply Voltage</td>
<td>4.75</td>
<td>5.25</td>
<td>V</td>
</tr>
</tbody>
</table>

**NOTE:**
*VBB* should be connected to AGND through a 0.01 μF capacitor. AGND, Vss and the 89027 Vss, AGND must be nominally at the same potential.

# DC CHARACTERISTICS

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Parameter</th>
<th>Min</th>
<th>Max</th>
<th>Units</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>VIL</td>
<td>Input Low Voltage</td>
<td>-0.3</td>
<td>+0.8</td>
<td>V</td>
<td>Except RST</td>
</tr>
<tr>
<td>VIL1</td>
<td>Input Low Voltage, RST</td>
<td>-0.3</td>
<td>+0.7</td>
<td>V</td>
<td></td>
</tr>
<tr>
<td>VIH</td>
<td>Input High Voltage</td>
<td>2.0</td>
<td>VCC +.5</td>
<td>V</td>
<td>Except RST, NMI, CLKin</td>
</tr>
<tr>
<td>VIH1</td>
<td>Input High Voltage, RST Rising</td>
<td>2.4</td>
<td>VCC +.5</td>
<td>V</td>
<td></td>
</tr>
<tr>
<td>VIH2</td>
<td>Input High Voltage, RST Falling</td>
<td>2.1</td>
<td>VCC +.5</td>
<td>V</td>
<td></td>
</tr>
<tr>
<td>VIH3</td>
<td>Input High Voltage, NMI, CLKin</td>
<td>2.4</td>
<td>VCC +.5</td>
<td>V</td>
<td></td>
</tr>
<tr>
<td>VOL</td>
<td>Output Low Voltage</td>
<td>0.45</td>
<td>V</td>
<td></td>
<td>See Note 1.</td>
</tr>
<tr>
<td>VOH</td>
<td>Output High Voltage</td>
<td>2.4</td>
<td>V</td>
<td></td>
<td>See Note 2.</td>
</tr>
<tr>
<td>ICC</td>
<td>Vcc Supply Current</td>
<td>200</td>
<td>mA</td>
<td></td>
<td>All outputs disconnected</td>
</tr>
<tr>
<td>IPD</td>
<td>VPDL Supply Current</td>
<td>1</td>
<td>mA</td>
<td></td>
<td>Normal operation and Power-Down</td>
</tr>
<tr>
<td>IREF</td>
<td>VREF Supply Current</td>
<td>15</td>
<td>mA</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ILI</td>
<td>Input Leakage Current</td>
<td>±10</td>
<td>μA</td>
<td></td>
<td>VIL = 0 to VCC See Note 3</td>
</tr>
<tr>
<td>IIL</td>
<td>Input High Current to EA</td>
<td>100</td>
<td>μA</td>
<td></td>
<td>VIL = 2.4V</td>
</tr>
<tr>
<td>IIH</td>
<td>Input Low Current</td>
<td>-100</td>
<td>μA</td>
<td></td>
<td>VIL = 0.45V See Note 4</td>
</tr>
<tr>
<td>IIL1</td>
<td>Input Low Current to RST</td>
<td>-2</td>
<td>mA</td>
<td></td>
<td>VIL = 0.45V</td>
</tr>
<tr>
<td>IIL2</td>
<td>Input Low Current S/A, SH, RI, READY</td>
<td>-50</td>
<td>μA</td>
<td></td>
<td>VIL = 0.45V</td>
</tr>
<tr>
<td>Cs</td>
<td>Pin Capacitance (Any Pin to Vss)</td>
<td>10</td>
<td>pF</td>
<td>1 MHz</td>
<td></td>
</tr>
</tbody>
</table>

**NOTES:**
1. IOL = 0.36 mA for pins TLC0, TCL1, B/C, RTS, CTS, DSr, DCD, SI, AR, and OH. Also if AD0 - AD15 are configured as I/O ports.
2. IOL = 2.0 mA for TM, CLKOUT, ALE, BHE, RD, WR, RXD, TCLK, and AD0 - AD15 when used as external memory bus.
3. IOL = -20 μA for pins TLC0, TLC1, B/C, RTS, CTS, DSr, DCD, SI, AR, and OH.
4. IOL = -200 μA for TM, CKOUT, ALE, BHE, RD, WR, RXD, TCLK, and AD0 - AD15 when used as external memory bus.
5. Power must be applied to the device in the following sequence: Vss first, then Vcc.
AC CHARACTERISTICS \((V_{CC}, V_{PD} = 4.75 \text{ to } 5.25 \text{ Volts}; T_A = 0^\circ \text{C to } 70^\circ \text{C}; \text{CLKIN} = 12.96 \text{ MHz})\)

Test Conditions: Load capacitance on output pins = 80 pF
Frequency = 12.96 MHz

### Timing Requirements

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Parameter</th>
<th>Min</th>
<th>Max</th>
<th>Units</th>
</tr>
</thead>
<tbody>
<tr>
<td>TAVDV</td>
<td>Address Valid to Input Data Valid</td>
<td></td>
<td>5Tosc – 90</td>
<td>ns</td>
</tr>
<tr>
<td>TRLDV</td>
<td>RD Active to Input Data Valid</td>
<td></td>
<td>3Tosc – 60</td>
<td>ns</td>
</tr>
<tr>
<td>TRXDX</td>
<td>Data Hold after RD Inactive (1)</td>
<td>0</td>
<td></td>
<td>ns</td>
</tr>
<tr>
<td>TRXHZ</td>
<td>RD Inactive to Input Data Float (1)</td>
<td></td>
<td>Tosc – 20</td>
<td>ns</td>
</tr>
</tbody>
</table>

### Timing Responses

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Parameter</th>
<th>Min</th>
<th>Max</th>
<th>Units</th>
</tr>
</thead>
<tbody>
<tr>
<td>FXTAL</td>
<td>CLKin Frequency</td>
<td>-0.01%</td>
<td>+0.01%</td>
<td>ns</td>
</tr>
<tr>
<td>Tosc</td>
<td>CLKin Period</td>
<td>77</td>
<td></td>
<td>ns</td>
</tr>
<tr>
<td>TCHCH</td>
<td>CLKOUT Period (1)</td>
<td>3Tosc (2)</td>
<td>3Tosc (2)</td>
<td>ns</td>
</tr>
<tr>
<td>TCHCL</td>
<td>CLKOUT High Time</td>
<td>Tosc – 20</td>
<td>Tosc + 20</td>
<td>ns</td>
</tr>
<tr>
<td>TCLDH</td>
<td>CLKOUT Low to ALE High</td>
<td>-5</td>
<td>20</td>
<td>ns</td>
</tr>
<tr>
<td>TLLCH</td>
<td>ALE Low to CLKOUT High</td>
<td>Tosc – 20</td>
<td>Tosc + 40</td>
<td>ns</td>
</tr>
<tr>
<td>TLHLL</td>
<td>ALE Pulse Width</td>
<td>Tosc – 25</td>
<td>Tosc + 15</td>
<td>ns</td>
</tr>
<tr>
<td>TAVLL</td>
<td>Address Setup to End of ALE</td>
<td>Tosc – 50</td>
<td></td>
<td>ns</td>
</tr>
<tr>
<td>TLLRL</td>
<td>End of ALE to RD or WR active</td>
<td>Tosc – 20</td>
<td></td>
<td>ns</td>
</tr>
<tr>
<td>TLLAX</td>
<td>Address Hold after End of ALE</td>
<td>Tosc – 20</td>
<td></td>
<td>ns</td>
</tr>
<tr>
<td>TWLWH</td>
<td>WR Pulse Width</td>
<td>2Tosc – 35</td>
<td></td>
<td>ns</td>
</tr>
<tr>
<td>TQVWX</td>
<td>Output Data Setup to End of WR</td>
<td>2Tosc – 60</td>
<td></td>
<td>ns</td>
</tr>
<tr>
<td>TXWQX</td>
<td>Output Data Hold after End of WR</td>
<td>Tosc – 25</td>
<td></td>
<td>ns</td>
</tr>
<tr>
<td>TWXLS</td>
<td>End of WR to next ALE</td>
<td>2Tosc – 30</td>
<td></td>
<td>ns</td>
</tr>
<tr>
<td>TRLWH</td>
<td>RD Pulse Width</td>
<td>3Tosc – 30</td>
<td></td>
<td>ns</td>
</tr>
<tr>
<td>TRHLH</td>
<td>End of RD to next ALE</td>
<td>Tosc – 25</td>
<td></td>
<td>ns</td>
</tr>
</tbody>
</table>

**NOTES:**
(1) This specification is not tested, but is verified by design analysis and/or derived from other tested parameters.
(2) CLKOUT is directly generated as a divide by 3 of the oscillator. The period will be 3Tosc ± 10 nsec if Tosc is constant and the rise and fall times on XTAL are less than 10 nsec.
Figure 6. Bus Signal Timings
89027 OVERVIEW

The 89027 is a 28 pin CMOS analog front end device, which performs most of the complex filtering functions required in modern transmitters and receivers. A general block diagram of this chip is provided in Figure 7. Most of the analog signal processing functions in this chip are implemented with CMOS switched capacitor technology. The 89027 functions are controlled by 89026, through a high speed serial data link.

During FSK transmit operation, the 89027 receives digitally synthesized mark and space sinusoid amplitude information from the 89026. The 89027 converts the signal to its analog equivalent, filters it, and transmits it to the telephone line. For QAM transmission, the signal constellation points are transferred to the 89027. This information is modulated into an analog signal, passed through spectral shaping filters, combined with the necessary guard tone, smoothed by a low pass filter, and transmitted to the line. Prior to transmitting either FSK or QAM signals to the telephone line, the 89027 adjusts the signal gain through an on-board programmable gain amplifier.

During the receive operation, the received FSK and QAM signals are passed through anti-alias filters, bandsplit filters, automatic gain control and carrier detect circuits, a Hilbert transform filter, and the output sent to the 89026 processor as analog signals.

Other functions provided by the 89027 are: an on-board two wire to four wire circuit with disable capability, an audio monitor output with software configurable gain, and a programmable gain transmit signal.

The 89027 is available in 28 pin plastic DIP and PLCC packages.

![Figure 7. 89027 Block Diagram](image-url)
# 89027 PINOUT

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Function (89027)</th>
<th>Direction</th>
<th>Pin No.</th>
</tr>
</thead>
<tbody>
<tr>
<td>VCC</td>
<td>Positive Power Supply (Digital)</td>
<td>+5V</td>
<td>28</td>
</tr>
<tr>
<td>VB</td>
<td>Negative Power Supply</td>
<td>−5V</td>
<td>15</td>
</tr>
<tr>
<td>VSS</td>
<td>Digital Ground</td>
<td>DGND</td>
<td>24</td>
</tr>
<tr>
<td>AGND</td>
<td>Analog Ground</td>
<td>AGND</td>
<td>21</td>
</tr>
<tr>
<td>AVCC</td>
<td>Positive Power Supply (Analog)</td>
<td>+5</td>
<td>7</td>
</tr>
<tr>
<td>X1</td>
<td>Xtal Oscillator</td>
<td>In</td>
<td>23</td>
</tr>
<tr>
<td>X2</td>
<td>Xtal Oscillator</td>
<td>Out</td>
<td>25</td>
</tr>
<tr>
<td>CLKOUT</td>
<td>12.96 MHz Clock Output to Microcontroller</td>
<td>Out</td>
<td>26</td>
</tr>
<tr>
<td>RST</td>
<td>Chip reset (active low)</td>
<td>In</td>
<td>20</td>
</tr>
<tr>
<td>HYB</td>
<td>Enable on-chip hybrid(1)</td>
<td>In</td>
<td>10</td>
</tr>
<tr>
<td>AZ1</td>
<td>Auto-zero capacitor</td>
<td>Out</td>
<td>16</td>
</tr>
<tr>
<td>AZ2</td>
<td>Auto-zero capacitor</td>
<td>In</td>
<td>17</td>
</tr>
<tr>
<td>SDATA</td>
<td>Serial data from 89026</td>
<td>In</td>
<td>2</td>
</tr>
<tr>
<td>SCLK</td>
<td>Serial clock from 89026</td>
<td>In</td>
<td>1</td>
</tr>
<tr>
<td>TSYNC</td>
<td>Transmitter sync from 89026</td>
<td>In</td>
<td>3</td>
</tr>
<tr>
<td>STR</td>
<td>Symbol timing to 89026</td>
<td>Out</td>
<td>27</td>
</tr>
<tr>
<td>ED</td>
<td>Receiver energy detect to 89026</td>
<td>Out</td>
<td>18</td>
</tr>
<tr>
<td>I</td>
<td>In phase received signal to 89026</td>
<td>Out</td>
<td>13</td>
</tr>
<tr>
<td>Q</td>
<td>Quadrature-phase received signal to 89026</td>
<td>Out</td>
<td>14</td>
</tr>
<tr>
<td>AO1</td>
<td>Transmitter output</td>
<td>Out</td>
<td>6</td>
</tr>
<tr>
<td>AO2</td>
<td>Receiver input</td>
<td>In</td>
<td>12</td>
</tr>
<tr>
<td>AMP</td>
<td>Output to monitor speaker</td>
<td>Out</td>
<td>11</td>
</tr>
<tr>
<td>TX0</td>
<td>Transmitter level control (LSB)(1)</td>
<td>In</td>
<td>9</td>
</tr>
<tr>
<td>TX1</td>
<td>Transmitter level control(1)</td>
<td>In</td>
<td>8</td>
</tr>
<tr>
<td>TX2</td>
<td>Transmitter level control(1)</td>
<td>In</td>
<td>5</td>
</tr>
<tr>
<td>TX3</td>
<td>Transmitter level control (MSB)(1)</td>
<td>In</td>
<td>4</td>
</tr>
</tbody>
</table>

**NOTES:**
1. When held high, these pins should be connected through 10K resistors to AVCC.
2. Pins #19 and #22 are No Connect.

## 89027 Pinout Description

### TXO-3
These four pins control the transmitted signal level.

### HYB
This pin enables the on-chip hybrid. A line impedance matching network must be connected between AO1 and AO2 when HYB is enabled. If HYB is disabled and an external 4W/2W hybrid is used, the hybrid receive path must be amplified by 6 dB.

*AO1*
Transmitter output.

*AO2*
Receiver input.

*AMP*
This output can be used to monitor the call progress tones and operation of the line.
ABSOLUTE MAXIMUM RATINGS
Temperature Under Bias .......... -10 to +80° C
Storage Temperature ............ -40 to +125° C
All Input and Output Voltages 
with Respect to VBB ............ -0.3V to +13.0V
All Input and Output Voltages 
with Respect to VCC & AVCC .......... -13.0V to 0.3V

Power Dissipation .................. 1.35W
Voltage with Respect 
VSS(1) .................... -0.3V to 6.5V

NOTE:
1. Applies to pins SCLK, SDATA, TSYNC, RST, HYB, TX0–TX3 only.

POWER DISSIPATION Ambient Temp = 0°C to 70°C, VCC = AVCC = 5V ± 5%, VSS = AGND = 0V.

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Parameter</th>
<th>Min</th>
<th>Typ</th>
<th>Max</th>
<th>Units</th>
</tr>
</thead>
<tbody>
<tr>
<td>AlcC1</td>
<td>AVCC Operating Current</td>
<td>19</td>
<td>25</td>
<td>mA</td>
<td></td>
</tr>
<tr>
<td>Icc1</td>
<td>VCC Operating Current</td>
<td>7</td>
<td>10</td>
<td>mA</td>
<td></td>
</tr>
<tr>
<td>Ibb1</td>
<td>VBB Operating Current</td>
<td>-19</td>
<td>-25</td>
<td>mA</td>
<td></td>
</tr>
<tr>
<td>Alccs</td>
<td>VCC Standby Current</td>
<td>0.2</td>
<td>1</td>
<td>mA</td>
<td></td>
</tr>
<tr>
<td>Iccs</td>
<td>VCC Standby Current</td>
<td>7</td>
<td>10</td>
<td>mA</td>
<td></td>
</tr>
<tr>
<td>Ibbs</td>
<td>VBB Standby</td>
<td>-0.6</td>
<td>-2</td>
<td>mA</td>
<td></td>
</tr>
<tr>
<td>Pdo</td>
<td>Operating Power Dissipation</td>
<td>225</td>
<td>300</td>
<td>mW</td>
<td></td>
</tr>
<tr>
<td>Pds</td>
<td>Standby Power Dissipation</td>
<td>40</td>
<td>70</td>
<td>mW</td>
<td></td>
</tr>
</tbody>
</table>

DC CHARACTERISTICS (Ta = 0°C to 70°C, AVCC = VCC = 5V ± 5%, VBB = 5V ± 5%, AGND = VSS = 0V), supply voltage must be at the same potential as the 89026 power supply. Typical Values are for Ta = 25°C and nominal power supply values. Power must be applied in the following sequence: VSS, AGND, VBB, VCC, and AVCC. VCC, AVCC and 89026 VREF must be nominally at the same potential.

Inputs: TX0, TX1, TX2, TX3, HYB, RST
Outputs: CLKOUT

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Parameter</th>
<th>Min</th>
<th>Max</th>
<th>Units</th>
<th>Test Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>il</td>
<td>Input Leakage Current</td>
<td>-10</td>
<td>+10</td>
<td>μA</td>
<td>VSS ≤ Vin ≤ VCC</td>
</tr>
<tr>
<td>vil</td>
<td>Input Low Voltage</td>
<td>VSS</td>
<td>0.8</td>
<td>V</td>
<td></td>
</tr>
<tr>
<td>vih</td>
<td>Input High Voltage</td>
<td>3.0</td>
<td>VCC</td>
<td>V</td>
<td></td>
</tr>
<tr>
<td>vol</td>
<td>Output Low Voltage</td>
<td>0.4</td>
<td>V</td>
<td>V</td>
<td>lio ≥ -1.6mA, 1 TTL load</td>
</tr>
<tr>
<td>voh</td>
<td>Output High Voltage</td>
<td>2.4</td>
<td>V</td>
<td>V</td>
<td>ich ≤ 50μA, 1 TTL load</td>
</tr>
<tr>
<td>Vco1</td>
<td>CLKOUT Low Voltage</td>
<td>0.4</td>
<td>V</td>
<td>V</td>
<td>C1 = 60 pF</td>
</tr>
<tr>
<td>vcoh</td>
<td>CLKOUT High Voltage</td>
<td>2.4</td>
<td>V</td>
<td>V</td>
<td>C1 = 60 pF</td>
</tr>
</tbody>
</table>
AC CHARACTERISTICS (Ta = 25°C, V<sub>CC</sub> = AV<sub>CC</sub> = 5 V, V<sub>SS</sub> = AGND = 0, V<sub>BB</sub> = −5 V)

ANALOG INPUTS: AO2

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Min</th>
<th>Typ</th>
<th>Max</th>
<th>Units</th>
<th>Test Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>AO2 Input Voltage Range</td>
<td></td>
<td></td>
<td>−9</td>
<td>dBm</td>
<td></td>
</tr>
<tr>
<td>AO2 Input Resistance</td>
<td>10</td>
<td></td>
<td></td>
<td>Mohms</td>
<td>−3.5V &lt; Vin &lt; +3.5V</td>
</tr>
<tr>
<td>AO2 Allowed DC offset</td>
<td>−30</td>
<td></td>
<td>+30</td>
<td>mV</td>
<td>Relative to AGND</td>
</tr>
</tbody>
</table>

AUTO ZERO CAPACITANCE

Capacitance = 0.015 μF  
Tolerance = ±10 %  
Voltage Rating = 10V  
Type = Non-Electrolytic, low leakage.

CRYSTAL REQUIREMENTS

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Min</th>
<th>Typ</th>
<th>Max</th>
<th>Unit</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Frequency Accuracy (0°C–70°C)</td>
<td>−0.0035%</td>
<td></td>
<td>+0.0035%</td>
<td>12.960 Mhz</td>
<td>Refer to Figure 8</td>
</tr>
<tr>
<td>Rx</td>
<td>10</td>
<td>0.024</td>
<td>16</td>
<td>ohms</td>
<td></td>
</tr>
<tr>
<td>Cx</td>
<td>5.6</td>
<td></td>
<td>6.1</td>
<td>pF</td>
<td></td>
</tr>
<tr>
<td>Co</td>
<td>−5%</td>
<td></td>
<td>+5%</td>
<td>33 pF</td>
<td>2 Load Capacitors</td>
</tr>
<tr>
<td>CL</td>
<td></td>
<td></td>
<td></td>
<td>33 pF</td>
<td></td>
</tr>
</tbody>
</table>

Crystal Type: Series Resonant

ANALOG OUTPUTS: A01, AMP

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Min</th>
<th>Typ</th>
<th>Max</th>
<th>Units</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Load Resistance</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>AO1</td>
<td>600</td>
<td></td>
<td></td>
<td>ohms</td>
<td></td>
</tr>
<tr>
<td>AMP</td>
<td>10</td>
<td></td>
<td></td>
<td>Kohms</td>
<td></td>
</tr>
<tr>
<td>Load Capacitance</td>
<td></td>
<td></td>
<td>100</td>
<td>pF</td>
<td></td>
</tr>
<tr>
<td>AMP</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Audio Amp Gain</td>
<td>−4</td>
<td></td>
<td></td>
<td>dB</td>
<td>Software</td>
</tr>
<tr>
<td>AO1 to Amp</td>
<td>−13</td>
<td></td>
<td></td>
<td>dB</td>
<td>Selectable</td>
</tr>
<tr>
<td></td>
<td>−20</td>
<td></td>
<td></td>
<td>dB</td>
<td></td>
</tr>
<tr>
<td></td>
<td>−60</td>
<td></td>
<td></td>
<td>dB</td>
<td></td>
</tr>
<tr>
<td>Audio Amp Gain (1)</td>
<td></td>
<td></td>
<td></td>
<td>dB</td>
<td></td>
</tr>
<tr>
<td>AO2 to Amp</td>
<td>+18</td>
<td></td>
<td></td>
<td>dB</td>
<td>Software</td>
</tr>
<tr>
<td></td>
<td>+9</td>
<td></td>
<td></td>
<td>dB</td>
<td>Selectable</td>
</tr>
<tr>
<td></td>
<td>+2</td>
<td></td>
<td></td>
<td>dB</td>
<td></td>
</tr>
<tr>
<td></td>
<td>−60</td>
<td></td>
<td></td>
<td>dB</td>
<td></td>
</tr>
</tbody>
</table>

NOTE:
1. Assumes on-chip hybrid is enabled. When on-chip hybrid is disabled, gain with respect to AO2 is reduced by 6 dB.
TRANSMIT LEVEL

<table>
<thead>
<tr>
<th>OUTPUT TRANSMIT LEVEL (1)</th>
<th>TX 3,2,1,0</th>
<th>Typ</th>
<th>Units</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 0 0 0</td>
<td></td>
<td>+5</td>
<td>dBm</td>
</tr>
<tr>
<td>0 0 0 1</td>
<td></td>
<td>+4</td>
<td>dBm</td>
</tr>
<tr>
<td>•</td>
<td></td>
<td>•</td>
<td>•</td>
</tr>
<tr>
<td>•</td>
<td></td>
<td>•</td>
<td>•</td>
</tr>
<tr>
<td>1 1 1 0</td>
<td></td>
<td>-9</td>
<td>dBm</td>
</tr>
<tr>
<td>1 1 1 1</td>
<td></td>
<td>-10</td>
<td>dBm</td>
</tr>
</tbody>
</table>

NOTE:
1. For FSK, PSK, QAM xmit signal. Measured at A01.

89024 REFERENCE MANUAL

Overview

The 89024 Reference Manual details design information for the 89024 Modem Chip Set. It provides descriptions and specifications of the two chips comprising the 89024, the 89026 and the 89027. In addition, it describes the control interface between the two chips.

ORDERING INFORMATION

Intel literature number: 296235-001

The reference manual also provides a full description of all the AT commands and S-registers supported by the 89024 Modem Chip Set.
89C024XE
HIGH PERFORMANCE 2400 BPS
INTELLIGENT MODEM CHIP SET

- Support for Error Correction
- Optional MNP Class 4/5
- CHMOS
- For Public Switched Telephone Network and Unconditioned Leased Lines Applications
- V.22 bis, V.22 A/B, V.21, Bell 212A, and Bell 103 Compatible
- Serial Command Set Compatible with Hayes Smartmodem 2400
- Automatically Adapts to Remote Modem Type with Recognition of Data Rates
- DTMF and Pulse Dialing, with Automatic Selection of Dial Signaling
- On-Chip Hybrid and Billing Delay Timer
- On-Chip Serial Port and Handshake Signals for RS-232/V.24 Interface
- Telephone Line Audio Monitor Output
- Analog/Digital Loopback Diagnostics with Mark/Space Pattern Generation and Error Detection

- Simple Serial Interface to External NVRAM
- Easily Customized Command Set and Features
- Two Chip Intelligent Modem Solution with Minimal External Components
- No External μC Required
- Output Programmable over 16 dB Range
- Dial and Re-dial Capability
- Full Set of Control Signals for DAA Interface
- Local, External, or Slave Timing Options in Synchronous Mode
- Adaptive Equalization
- Capable of Detecting Dial, Busy, Ringback and Modem Answer Tones of Most International Networks
- Auxiliary Relay Control Output

Figure 1. 89C024XE System Block Diagram
GENERAL DESCRIPTION

Intel 89C024XE is a highly integrated, low power, high performance, intelligent modem chip set. This two chip solution is composed of the 89027 Analog Front End and 89C026XE microcontroller. At 12.96 MHz the microcontroller is capable of executing error correction and data compression routines. The system conforms to the following CCITT and BELL standards.

- CCITT V.22 bis
  2400 bps sync and async
  1200 bps sync and async (fall-back)
- CCITT V.22 A & B
  1200 bps sync and async
  600 bps sync and async (fall-back)
- CCITT V.21
  0 to 300 bps anisochronous
- BELL 212A
  1200 bps sync and async
  300 bps fall-back mode
- BELL 103
  0 to 300 bps anisochronous

The 89C024XE system consists of a 16 bit application specific processor (89C026XE) and an analog front end device (89027). The 89C026XE processor performs all "Digital Signal Processing" algorithm execution for processing the modem signals, as well as providing all modem control functions typically performed by an external processor. The analog front end provides for 2 wire and 4 wire telephone line interface, D/A conversion, and most of the complex filtering functions required in QAM/DPSK/FSK modems. Refer to Figure 1 for a simplified block diagram of the system.

In stand-alone modem applications, the 89C024XE chip set along with a Data Access Arrangement (DAA), a serial NVRAM, and RS-232 driver/receivers and EPROM represent the circuitry required for implementing an auto-dial, auto-answer, 300 to 2400 bps, full duplex Hayes compatible intelligent modem.

A complete set of Industry Standard AT commands is provided for modem configuration and user interface. Virtually all PC software written for the Hayes Smartmodem 2400 can also be used with this chip set. Alternatively, in applications where user proprietary modem control commands and features are desired, the user can replace the 89C024XE internal command module with custom proprietary software.

The 89C024XE has a set of default features. Upon power up, the modem configuration will be in accordance with these default options, unless a different configuration has been saved in the external NVRAM with the &W command.

The 89C024XE modem has built in auto-dialing and auto-answering capabilities. It can be configured to the proper line signaling mode (Tone or Pulse), and to the type (CCITT or Bell) and speed of the calling or answering modem. It can also detect and identify call set-up signals of telephone networks, allowing unattended data call operation.

A full set compatible with CCITT V.54 diagnostic loop-test features is supported. The chip set also provides a line signal for audio monitoring of call progress, a comprehensive set of DAA control lines for a simple interface to the telephone network, and a full complement of TTL level RS-232/ V.24 hand-shake signals.

NOTICE:
Hayes is a registered trademark of Hayes Microcomputer Products, Inc.
Smartmodem 2400 is a trademark of Hayes Microcomputer Products, Inc.
Smartcom II is a registered trademark of Hayes Microcomputer Products, Inc.
PACKAGING

89027 is available in PLCC and standard plastic DIP packages. The 89C026XE is available in PLCC package.

Figure 2. Device Packages
CALL ESTABLISHMENT, TERMINATION AND RETRAIN

The 89C024XE modem system incorporates all protocols and functions required for automatic (or manual) establishment, progress and termination of a data call.

The modem chip-set has a built in auto-dialer, both DTMF and Pulse type, and is capable of automatically adapting to the telephone dial type. The dialing sequence on the telephone link conforms to the CCITT V.25 recommendations. An exception to the V.25 is that the interrupted calling tone will not be transmitted by the calling modem, as is suggested in V.22 bis.

The modem can detect the dial, busy and ringback signals at remote end, and will provide call progress messages to the user. The modem is capable of redialing the last number dialed, by one command.

The modem when configured for auto-answer, will answer an incoming call, remain silent for the two second billing delay interval, before transmitting the answer tones. Afterwards modem to modem identification and handshaking will proceed at a speed and operating mode acceptable to both ends of the link.

The data call can also be setup by manual dialing with the modems set to data mode, or by voice to data transfer by means of mechanical switch (exclusion key), using the SH pin. Once set to data mode, the modem handshaking will proceed before the modems will be ready to accept and exchange data.

During data transmission, if one of the modems finds that the received data is likely to have a high bit error rate (indicated by a large mean square error in the adaptive equalizer), it initiates a retrain sequence. This automatic retrain feature is only available at 2400 bps, and conforms to CCITT V.22 bis recommendations.

Disconnection of the data call can be initiated by the DTE at the local end, or by the remote DTE, (if the modem is configured to accept it). Whether DTR will initiate a disconnect, depends on the last &D command. Receiving a long space from a remote modem will initiate a disconnect only after a Y1 command. The optional disconnect requests originated by the remote modem, are of two types, (1) disconnect when receiving long-space, and (2) disconnect when received carrier is dropped. The modem chip-set can also be configured to transmit 'long-space' just before disconnection, in each of the aforementioned cases.

Because the CCITT and Bell modem connection protocols are quite different from each other and do not provide recognition of remote modem type (i.e. V.22 bis to 212A), the Intel chip-set provides the additional capability of identifying the remote modem type. This feature is beneficial during the migration phase of the technology from the 1200 bps to 2400 bps. In North America, where the installed base of 1200 bps modems is mostly made-up of 212A type, this feature allows a "Data Base Service Provider" to easily upgrade the existing 212A modems to 2400 bps V.22 bis standard, transparently, to 212A users. Similarly, a user with a 89C024XE based modem system can automatically call data bases with either 212A or V.22 bis modems, without concern over the difference. This feature's benefits are realized in smooth upgrading of data links, with minimum cost and reduced disruption in services. Refer to Table 1 for a detailed description of remote modem compatibility.

SOFTWARE CONFIGURATION COMMANDS

This section lists the 89C024XE commands and registers that may be used while configuring the modem. Commands instruct the modem to perform an action, the value in the associated registers determine how the commands are performed, and the result codes returned by the modem tell the user about the execution of the commands.

The commands may be entered in a string, with or without spaces in between. Any spaces within or between commands will be ignored by the modem. During the entry of any command, the 'backspace' key (CNTRL H) can be used to correct any error. Upper case or lower case characters can be used in the commands. Commands described in the following paragraphs refer to asynchronous terminals using ASCII codes.
Table 1. Remote Modem Compatibility

<table>
<thead>
<tr>
<th>Originating 89C024XE Modem</th>
<th>Answering Modem</th>
<th>BELL</th>
<th>BELL</th>
<th>CCITT</th>
<th>CCITT</th>
<th>CCITT</th>
<th>CCITT</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Bell 300</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bell</td>
<td>300</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1200</td>
<td>1200</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CCITT</td>
<td>300</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>600</td>
<td>-</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1200</td>
<td>1200*</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2400</td>
<td>1200*</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>300</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>300</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>300</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>600</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>1200</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>1200*</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>1200</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>2400</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Answering 89C024XE Modem

<table>
<thead>
<tr>
<th>Answering 89C024XE Modem</th>
<th>BELL</th>
<th>BELL</th>
<th>CCITT</th>
<th>CCITT</th>
<th>CCITT</th>
<th>CCITT</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Bell 300</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bell</td>
<td>300</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1200</td>
<td>300</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CCITT</td>
<td>300</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>600</td>
<td>-</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1200</td>
<td>300*</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2400</td>
<td>300*</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

* These connection data rates are obtained when connecting 89C024XE based modems end to end. The same results may not be obtained when a 89C024XE based modem is connected to other modems.

Command Set

<table>
<thead>
<tr>
<th>AT</th>
<th>Attention code.</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>Go off-hook in answer mode</td>
</tr>
<tr>
<td>A/</td>
<td>Repeat previous command string</td>
</tr>
<tr>
<td>Bn</td>
<td>BELL/CCITT Protocol Compatibility at 1200 bps</td>
</tr>
<tr>
<td>Ds</td>
<td>The dialing commands (0-9 A B C D * # P R T S W ; @)</td>
</tr>
<tr>
<td>En</td>
<td>Echo command (En)</td>
</tr>
<tr>
<td>Hn</td>
<td>Switch-Hook Control</td>
</tr>
<tr>
<td>If &amp;J1 option is selected, H1 will also switch the auxiliary relay</td>
<td></td>
</tr>
<tr>
<td>In</td>
<td>Request Product Code and Checksum</td>
</tr>
<tr>
<td>Ln</td>
<td>Speaker Volume</td>
</tr>
<tr>
<td>Mn</td>
<td>Monitor On/Off</td>
</tr>
<tr>
<td>O</td>
<td>On-Line</td>
</tr>
<tr>
<td>Qn</td>
<td>Result Codes</td>
</tr>
<tr>
<td>Sn=x</td>
<td>Write S Register</td>
</tr>
<tr>
<td>Sn</td>
<td>Read S Register</td>
</tr>
<tr>
<td>Vn</td>
<td>Enable Short-Form Result Codes</td>
</tr>
<tr>
<td>Xn</td>
<td>Enable Extended Result Code</td>
</tr>
<tr>
<td>Yn</td>
<td>Enable Long Space Disconnect</td>
</tr>
<tr>
<td>Z</td>
<td>Fetch Configuration Profile</td>
</tr>
<tr>
<td>++</td>
<td>The Default Escape Code</td>
</tr>
</tbody>
</table>

& Command Set (Continued)

| &C | DCD Options |
| &D | DTR Options |
| &F | Fetch Factory Configuration Profile |
| &G | Guard Tone |
| &J | Telephone Jack Selection |
| &L | Leased/Dial-up Line Selection |
| &M | Async/Sync Mode Selection |
| &P | Make/Break Pulse Ratio |
| &R | RTS/CTS Options |
| &S | DSR Options |
| &T | Test Commands |
| &W | Write Configuration to Non Volatile Memory |
| &X | Sync Clock Source |
| &Z | Store Telephone Number |
CONFIGURATION REGISTERS

The modem stores all the configuration information in a set of registers. Some registers are dedicated to special command and function, and others are bit-mapped, with different commands sharing the register space to store the command status.

<table>
<thead>
<tr>
<th>S</th>
<th>Ring to Answer</th>
</tr>
</thead>
<tbody>
<tr>
<td>S1</td>
<td>Ring Count. (Read Only)</td>
</tr>
<tr>
<td>S2</td>
<td>Escape Code Character</td>
</tr>
<tr>
<td>S3</td>
<td>Carriage Return Character</td>
</tr>
<tr>
<td>S4</td>
<td>Line Feed Character</td>
</tr>
<tr>
<td>S5</td>
<td>Back Space Character</td>
</tr>
<tr>
<td>S6</td>
<td>Wait for Dial Tone</td>
</tr>
<tr>
<td>S7</td>
<td>Wait for Data Carrier</td>
</tr>
<tr>
<td>S8</td>
<td>Pause Time for the Comma Dial Modifier</td>
</tr>
<tr>
<td>S9</td>
<td>Carrier Detect Response Time</td>
</tr>
<tr>
<td>S10</td>
<td>Lost Carrier to Hang Up Delay</td>
</tr>
<tr>
<td>S11</td>
<td>Not Used</td>
</tr>
<tr>
<td>S12</td>
<td>Escape Code Guard Time</td>
</tr>
<tr>
<td>S13</td>
<td>Not Used</td>
</tr>
<tr>
<td>S14*</td>
<td>Bit Mapped Option Register</td>
</tr>
<tr>
<td>S15</td>
<td>Not Used</td>
</tr>
<tr>
<td>S16</td>
<td>Modern Test Options</td>
</tr>
<tr>
<td>S17</td>
<td>Not Used</td>
</tr>
<tr>
<td>S18*</td>
<td>Test Timer</td>
</tr>
<tr>
<td>S19</td>
<td>Not Used</td>
</tr>
<tr>
<td>S20</td>
<td>Not Used</td>
</tr>
<tr>
<td>S21*</td>
<td>Bit Mapped Options Register</td>
</tr>
<tr>
<td>S22*</td>
<td>Bit Mapped Options Register</td>
</tr>
<tr>
<td>S23*</td>
<td>Bit Mapped Options Register</td>
</tr>
<tr>
<td>S24</td>
<td>Not Used</td>
</tr>
<tr>
<td>S25*</td>
<td>Delay to DTR (Sync Only)</td>
</tr>
<tr>
<td>S26*</td>
<td>RTS to CTS Delay (Half Dup.)</td>
</tr>
<tr>
<td>S27*</td>
<td>Bit Mapped Options Register</td>
</tr>
</tbody>
</table>

NOTE:
* These S registers can be stored in the NVRAM.

Dial Modifiers

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>P</td>
<td>Pulse Dial</td>
</tr>
<tr>
<td>R</td>
<td>Originate call in Answer Mode</td>
</tr>
<tr>
<td>T</td>
<td>Tone Dial</td>
</tr>
<tr>
<td>S</td>
<td>Dial a stored number</td>
</tr>
<tr>
<td>W</td>
<td>Wait for dial tone</td>
</tr>
<tr>
<td>;</td>
<td>Delay a dial sequence</td>
</tr>
<tr>
<td>@</td>
<td>Return to command state</td>
</tr>
<tr>
<td></td>
<td>Initiate a flash</td>
</tr>
<tr>
<td></td>
<td>Wait for quit</td>
</tr>
</tbody>
</table>

If neither P or T is specified in the command string, the modem automatically selects the proper dial mode.

Example:

Terminal: AT &Z T 1 (602) 555-1212
Modem: OK
Result: Modem stores T16025551212 in the external NVRAM.

The number can be dialed from asynchronous mode by issuing the following command:

Terminal: AT DS
Modem: T16025551212

or by turning on DTR when in Synchronous Mode 2. Up to 33 symbols (dial digits and dial modifiers) may be stored. Spaces and other delimiters are ignored and do not need to be included in the count. If more than 33 symbols are supplied, the dial string will be truncated to 33.

APPLICATIONS OVERVIEW

The block diagram of a stand-alone 300 to 2400 bps Hayes compatible modem is depicted in Figure 3. The DAA section shown in this diagram may be obtained with FCC registration, or implemented using the suggested diagram in Figure 4.
Figure 3. Typical Modem Configuration

Figure 4. Typical Telephone Line Interface with Built In Hybrid
## SYSTEM COMPATIBILITY SPECIFICATIONS

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Specification</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Synchronous</strong></td>
<td>2400 bps ± 0.01% V.22 bis</td>
</tr>
<tr>
<td></td>
<td>1200 bps ± 0.01% V.22 and BELL 212A</td>
</tr>
<tr>
<td></td>
<td>600 bps ± 0.01% V.22 A,B</td>
</tr>
<tr>
<td><strong>Asynchronous</strong></td>
<td>2400, 1200, 600 bps, character asynchronous.</td>
</tr>
<tr>
<td></td>
<td>0 - 300 bps anisochronous.</td>
</tr>
<tr>
<td><strong>Asynchronous Speed Range</strong></td>
<td>+1% - 2.5% default. Extended +2.3% -2.5% range of CCITT standards optional via software customization.</td>
</tr>
<tr>
<td><strong>Asynchronous Format</strong></td>
<td>8,9,10,11 bits, including start, stop, parity. Bits 8, 9, 11 optional via S/W customization.</td>
</tr>
<tr>
<td><strong>Synchronous Timing Source</strong></td>
<td>Internal, derived from the local oscillator.</td>
</tr>
<tr>
<td></td>
<td>External, provided by DTE through XTCLK.</td>
</tr>
<tr>
<td></td>
<td>Slave, derived from the received clock.</td>
</tr>
<tr>
<td><strong>Telephone Line Interface</strong></td>
<td>Two wire full duplex over public switched network or 4 wire leased lines. On-chip hybrid and billing delay timers. Output level −1 to −16 dBm</td>
</tr>
<tr>
<td><strong>Modulation</strong></td>
<td>V.22 bis, 16 point QAM at 600 baud.</td>
</tr>
<tr>
<td></td>
<td>V.22 and 212A, 4 point DPSK at 600 baud.</td>
</tr>
<tr>
<td></td>
<td>V.21 and 103, binary phase coherent FSK</td>
</tr>
<tr>
<td><strong>Output Spectral Shaping</strong></td>
<td>Square root of 75% raised cosine, QAM/PSK.</td>
</tr>
<tr>
<td><strong>Transmit Carrier Frequencies</strong></td>
<td></td>
</tr>
<tr>
<td>V.22 bis, V.22, 212A</td>
<td>Originate 1200 Hz ± .01%</td>
</tr>
<tr>
<td></td>
<td>Answer 2400 Hz ± .01%</td>
</tr>
<tr>
<td>V.21</td>
<td>Originate 'space' 1180 Hz ± .01%</td>
</tr>
<tr>
<td></td>
<td>Originate 'mark' 980 Hz ± .01%</td>
</tr>
<tr>
<td></td>
<td>Answer 'space' 1850 Hz ± .01%</td>
</tr>
<tr>
<td></td>
<td>Answer 'mark' 1650 Hz ± .01%</td>
</tr>
<tr>
<td>Bell 103 mode</td>
<td>Originate 'space' 1070 Hz ± .01%</td>
</tr>
<tr>
<td></td>
<td>Originate 'mark' 1270 Hz ± .01%</td>
</tr>
<tr>
<td></td>
<td>Answer 'space' 2020 Hz ± .01%</td>
</tr>
<tr>
<td></td>
<td>Answer 'mark' 2225 Hz ± .01%</td>
</tr>
<tr>
<td><strong>Receive Carrier Frequency Limits</strong></td>
<td></td>
</tr>
<tr>
<td>V.22 bis, V.22, 212A</td>
<td>Originate 2400 Hz ± 7 Hz</td>
</tr>
<tr>
<td></td>
<td>Answer 1200 Hz ± 7 Hz</td>
</tr>
<tr>
<td>V.21</td>
<td>Originate 'space' 1850 Hz ± 12 Hz</td>
</tr>
<tr>
<td></td>
<td>Originate 'mark' 1650 Hz ± 12 Hz</td>
</tr>
<tr>
<td></td>
<td>Answer 'space' 1180 Hz ± 12 Hz</td>
</tr>
<tr>
<td></td>
<td>Answer 'mark' 980 Hz ± 12 Hz</td>
</tr>
<tr>
<td>Bell 103</td>
<td>Originate 'space' 2020 Hz ± 12 Hz</td>
</tr>
<tr>
<td></td>
<td>Originate 'mark' 2225 Hz ± 12 Hz</td>
</tr>
<tr>
<td></td>
<td>Answer 'space' 1070 Hz ± 12 Hz</td>
</tr>
<tr>
<td></td>
<td>Answer 'mark' 1270 Hz ± 12 Hz</td>
</tr>
<tr>
<td><strong>Energy Detect Sensitivity</strong></td>
<td>Greater than −43 dBm ED is ON. Less than −48 dBm ED is OFF. Signal in dBm measured at A02.</td>
</tr>
<tr>
<td><strong>Line Equalization</strong></td>
<td>Fixed compromise equalization, transmit.</td>
</tr>
<tr>
<td></td>
<td>Adaptive equalizer for DPSK/QAM, receive.</td>
</tr>
<tr>
<td><strong>Diagnostics Available</strong></td>
<td>Local analog loopback.</td>
</tr>
<tr>
<td></td>
<td>Local digital loopback.</td>
</tr>
<tr>
<td></td>
<td>Remote digital loopback.</td>
</tr>
<tr>
<td></td>
<td>Local interface loopback.</td>
</tr>
<tr>
<td><strong>Self Test Pattern Generator</strong></td>
<td>Alternate 'ones' and 'zeros' and error detector, to be used along with most loopbacks. A number indicating the bit errors detected is sent to DTE.</td>
</tr>
</tbody>
</table>

4-29
# RECEIVER PERFORMANCE SPECIFICATIONS

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Specification</th>
</tr>
</thead>
<tbody>
<tr>
<td>Test condition:</td>
<td>Unconditioned 3002 line, across the full dynamic range. The noise bandwidth is 3 KHz flat.</td>
</tr>
<tr>
<td>Random Noise</td>
<td>Typical Bit Error rate of 1 in 100000 or better at 12 dB SNR at 300 bps, 5 dB SNR at 600 bps, 8 dB SNR at 1200 bps and 16 dB SNR at 2400 bps.</td>
</tr>
<tr>
<td>Frequency Offsets(1)</td>
<td>± 7 Hz.</td>
</tr>
<tr>
<td>Phase Jitter(1)</td>
<td>2400 bps - 15° peak to peak, at up to 300 Hz. 600, 1200 bps - 45° peak to peak, at up to 300 Hz.</td>
</tr>
</tbody>
</table>

**NOTE:**
1. There are no observable data errors for the received signals, for the above limits of line impairments.
2. These impairments are applied one at a time in absence of noise.

# PERFORMANCE SPECIFICATIONS

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Min</th>
<th>Typ</th>
<th>Max</th>
<th>Units</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>DTMF Level</td>
<td>5.0</td>
<td></td>
<td></td>
<td>dBm</td>
<td>at AO1</td>
</tr>
<tr>
<td>DTMF Second Harmonic</td>
<td></td>
<td></td>
<td>-35</td>
<td>dB</td>
<td>HYB enabled into 600Ω</td>
</tr>
<tr>
<td>DTMF Twist (Balance)</td>
<td>3</td>
<td></td>
<td></td>
<td>dB</td>
<td></td>
</tr>
<tr>
<td>Default DTMF Duration</td>
<td>100</td>
<td></td>
<td></td>
<td>ms</td>
<td>Software Controlled</td>
</tr>
<tr>
<td>Pulse Dialing Rate</td>
<td>10</td>
<td></td>
<td></td>
<td>pps</td>
<td></td>
</tr>
<tr>
<td>Pulse Dialing Make/Break</td>
<td>39/61</td>
<td></td>
<td>33/67</td>
<td>%</td>
<td>US</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>%</td>
<td>UK, Hong Kong</td>
</tr>
<tr>
<td>Pulse Interdigit Interval</td>
<td>785</td>
<td></td>
<td></td>
<td>ms</td>
<td></td>
</tr>
<tr>
<td>Billing Delay Interval</td>
<td></td>
<td></td>
<td>2.1</td>
<td>sec</td>
<td></td>
</tr>
<tr>
<td>Guard Tone Frequency</td>
<td>540</td>
<td></td>
<td></td>
<td>Hz</td>
<td>referenced to High channel transmit.</td>
</tr>
<tr>
<td>Amplitude</td>
<td>-3</td>
<td></td>
<td></td>
<td>dB</td>
<td></td>
</tr>
<tr>
<td>Frequency</td>
<td>1800</td>
<td></td>
<td></td>
<td>Hz</td>
<td>QAM/DPSK Modes Only</td>
</tr>
<tr>
<td>Amplitude</td>
<td>-6</td>
<td></td>
<td></td>
<td>dB</td>
<td></td>
</tr>
<tr>
<td>Dial Tone Detect Duration</td>
<td>3.0</td>
<td></td>
<td></td>
<td>sec</td>
<td></td>
</tr>
<tr>
<td>Ringback Tone Detect</td>
<td>0.75</td>
<td></td>
<td>1.5</td>
<td>sec</td>
<td>Off/On Ratio</td>
</tr>
<tr>
<td>Duration</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Cadence</td>
<td>1.5</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Busy Tone Detect Duration</td>
<td>0.67</td>
<td></td>
<td>0.2</td>
<td>sec</td>
<td>Off/On Ratio</td>
</tr>
<tr>
<td>Cadence</td>
<td></td>
<td></td>
<td>1.5</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
89C026XE OVERVIEW

The 89C026XE processor performs data manipulation, signal processing and user interface functions. It supports external ROM and RAM to perform asynchronous, synchronous, and/or custom code with or without high level protocol functions. These options will allow proprietary modem control, functions, call progress management applications to be implemented. A block diagram of the 89C026XE is provided in Figure 5.

89C026XE contains a TTL compatible serial link to DTE equipment, along with a full complement of V.24/RS-232-C control signals. Alternatively, a UART or USART may be used directly to transfer data to and from a microcomputer bus. The industry standard AT command set is supported by the 89C026XE, facilitating compatibility between 89C024XE and most PC software written for the AT command set.

During transmit operation, the 89C026XE synthesizes DTMF tones and the 300 BPS FSK modem signal and transmits them to the 89027 as digitized amplitude samples. During 1200 and 2400 BPS operation, DPSK and QAM is used to send 2 to 4 bits of information respectively at 600 baud to the AFE. Because the QAM coding technique is an inherently synchronous transmission mechanism, in the case of asynchronous QAM transmission, the asynchronous data is synchronized by adding or deleting stop bits. Following the synchronization process, the 89C026XE transmits digitized phase and amplitude samples to 89027 over the high speed serial link.

In the receive operation, the information is received by the 89C026XE from the 89027 as two signals which are 90 degrees phase shifted from each other. These analog signals are then digitized by the A/D converter resident on the 89C026XE. By using DSP algorithms, the received signals are processed using adaptive equalization for telephone line delay, amplitude distortion and gain adjustment is executed and the signal demodulated. Following demodulation, the data is unscrambled, and if necessary, returned to asynchronous format.

Figure 5. 89C026XE Block Diagram
## 89C026XE PINOUT

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Function (89026)</th>
<th>Direction</th>
<th>Pin No.</th>
</tr>
</thead>
<tbody>
<tr>
<td>CCLKIN</td>
<td>12.96 MHz master clock from 89027</td>
<td>In</td>
<td>67</td>
</tr>
<tr>
<td>CCLKIN</td>
<td>270 KHz from 89027</td>
<td>In</td>
<td>44</td>
</tr>
<tr>
<td>RST</td>
<td>Chip reset (active low)</td>
<td>In</td>
<td>16</td>
</tr>
<tr>
<td>I</td>
<td>In-phase received signal</td>
<td>In</td>
<td>11</td>
</tr>
<tr>
<td>Q</td>
<td>Quadrature-phase received signal</td>
<td>In</td>
<td>10</td>
</tr>
<tr>
<td>STR</td>
<td>Symbol Timing from 89027</td>
<td>In</td>
<td>24</td>
</tr>
<tr>
<td>ED</td>
<td>Energy Detect input</td>
<td>In</td>
<td>9</td>
</tr>
<tr>
<td>TSYNC</td>
<td>Transmitter sync pulse to 89027</td>
<td>Out</td>
<td>35</td>
</tr>
<tr>
<td>SDATA</td>
<td>Serial Data to 89027</td>
<td>Out</td>
<td>17</td>
</tr>
<tr>
<td>SCLK</td>
<td>Serial Clock to 89027</td>
<td>Out</td>
<td>18</td>
</tr>
<tr>
<td>OH</td>
<td>Off-Hook control to DAA</td>
<td>Out</td>
<td>33</td>
</tr>
<tr>
<td>SH</td>
<td>Switch-Hook from dataphone</td>
<td>In</td>
<td>5</td>
</tr>
<tr>
<td>RI</td>
<td>Ring Indicator from DAA</td>
<td>In</td>
<td>42</td>
</tr>
<tr>
<td>AR</td>
<td>Aux Relay control to DAA</td>
<td>Out</td>
<td>38</td>
</tr>
<tr>
<td>TCL1</td>
<td>NVRAM Data I/O</td>
<td>I/O</td>
<td>20</td>
</tr>
<tr>
<td>TCL0</td>
<td>NVRAM CLK</td>
<td>Out</td>
<td>19</td>
</tr>
<tr>
<td>B/C</td>
<td>103/V.21 default option</td>
<td>In</td>
<td>15</td>
</tr>
<tr>
<td>S/A</td>
<td>NVRAM CE</td>
<td>Out</td>
<td>21</td>
</tr>
<tr>
<td>D/S</td>
<td>Dumb/Smart mode select</td>
<td>In</td>
<td>6</td>
</tr>
<tr>
<td>CONFIG</td>
<td>Custom Firmware Disable</td>
<td>In</td>
<td>8</td>
</tr>
<tr>
<td>TM</td>
<td>Test Mode Indicator</td>
<td>Out</td>
<td>39</td>
</tr>
<tr>
<td>TXD</td>
<td>Transmitted data from DTE</td>
<td>In</td>
<td>27</td>
</tr>
<tr>
<td>RXD</td>
<td>Received data to DTE</td>
<td>Out</td>
<td>29</td>
</tr>
<tr>
<td>RTS</td>
<td>Request to send from DTE</td>
<td>In</td>
<td>22</td>
</tr>
<tr>
<td>CTS</td>
<td>Clear to Send to DTE</td>
<td>Out</td>
<td>23</td>
</tr>
<tr>
<td>DSR</td>
<td>Data Set Ready to DTE</td>
<td>Out</td>
<td>30</td>
</tr>
<tr>
<td>DCD</td>
<td>Data Carrier Detect to DTE</td>
<td>Out</td>
<td>31</td>
</tr>
<tr>
<td>DTR</td>
<td>Data Terminal Ready from DTE</td>
<td>In</td>
<td>25</td>
</tr>
<tr>
<td>RCLK</td>
<td>Received clock to DTE</td>
<td>Out</td>
<td>34</td>
</tr>
<tr>
<td>TCLK</td>
<td>Transmit clock to DTE</td>
<td>Out</td>
<td>28</td>
</tr>
<tr>
<td>XTCLK</td>
<td>External timing clock from DTE</td>
<td>In</td>
<td>26</td>
</tr>
<tr>
<td>SI</td>
<td>Speed Indicator to DTE</td>
<td>Out</td>
<td>32</td>
</tr>
<tr>
<td>REMLB</td>
<td>Remote Loopback Command from DTE</td>
<td>In</td>
<td>7</td>
</tr>
<tr>
<td>LCLLB</td>
<td>Local Loopback Command from DTE</td>
<td>In</td>
<td>4</td>
</tr>
<tr>
<td>VCC</td>
<td>Positive power supply (+5V)</td>
<td>+5V</td>
<td>1</td>
</tr>
<tr>
<td>VPD</td>
<td>Ram back-up power</td>
<td>+5V</td>
<td>14</td>
</tr>
<tr>
<td>VREF</td>
<td>A/D converter reference</td>
<td>+5V</td>
<td>13</td>
</tr>
<tr>
<td>VSS1</td>
<td>Digital ground</td>
<td>GND</td>
<td>36</td>
</tr>
<tr>
<td>VSS2</td>
<td>Digital ground</td>
<td>GND</td>
<td>68</td>
</tr>
<tr>
<td>AGND</td>
<td>Analog ground</td>
<td>AGND</td>
<td>12</td>
</tr>
<tr>
<td>VBBS</td>
<td>Back-bias generator output</td>
<td>Out</td>
<td>37</td>
</tr>
<tr>
<td>EA</td>
<td>External Memory enable</td>
<td>In</td>
<td>2</td>
</tr>
<tr>
<td>A0-AD15</td>
<td>External memory access address/data</td>
<td>I/O</td>
<td>60-45</td>
</tr>
<tr>
<td>AA</td>
<td>Auto Answer</td>
<td>Out</td>
<td>60</td>
</tr>
<tr>
<td>JS</td>
<td>Jack Select</td>
<td>Out</td>
<td>59</td>
</tr>
</tbody>
</table>
### 89C026XE PINOUT (Continued)

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Function (89026)</th>
<th>Direction</th>
<th>Pin No.</th>
</tr>
</thead>
<tbody>
<tr>
<td>NMI</td>
<td>No-maskable Interrupt(VSS)</td>
<td>In</td>
<td>3</td>
</tr>
<tr>
<td>X2</td>
<td>Crystal output(NC)</td>
<td>Out</td>
<td>66</td>
</tr>
<tr>
<td>CLKOUT</td>
<td>Clk output</td>
<td>Out</td>
<td>65</td>
</tr>
<tr>
<td>TEST</td>
<td>Factory test(VCC)</td>
<td>In</td>
<td>64</td>
</tr>
<tr>
<td>INST</td>
<td>External memory instruction fetch</td>
<td>Out</td>
<td>63</td>
</tr>
<tr>
<td>ALE</td>
<td>Address latch enable</td>
<td>Out</td>
<td>62</td>
</tr>
<tr>
<td>RD</td>
<td>External memory read</td>
<td>Out</td>
<td>61</td>
</tr>
<tr>
<td>READY</td>
<td>External memory ready(VCC)</td>
<td>In</td>
<td>43</td>
</tr>
<tr>
<td>BHE</td>
<td>External memory bus high enable</td>
<td>Out</td>
<td>41</td>
</tr>
<tr>
<td>WR</td>
<td>External memory write</td>
<td>Out</td>
<td>40</td>
</tr>
</tbody>
</table>

**NOTES:**
1. Pins marked with (VSS) must be connected to VSS.
2. Pins marked with (NC) are to be left unconnected.
3. Pins marked with (VCC) must be connected to VCC.
4. With internal ROM enabled, AD0-AD1 are used as AA and JS.

### 89C026XE PIN DESCRIPTION

**XTCLK**
Transmitter timing from DTE, when external clock option is selected.

**TXD**
The serial data from DTE to be transmitted on the line. A logic 'high' is mark. In synchronous mode, 89026 samples this data on the rising edges of TCLK.

**TCLK**
Clock output from 89026 as timing source for data exchange from DTE to modem. Serial data is read on the rising edges of the TCLK. This output is High in asynchronous mode.

**RXD**
The serial data to DTE. 'Mark' is a logic High. In synchronous mode, the rising edge of RCLK occurs in the middle of RXD.

**RCLK**
Synchronous clock output. Rising edge of RCLK occurs in the middle of each RXD bit. This pin remains High in asynchronous mode.

**VBBs**
This pin to be connected to AGND through a 0.01 μF capacitor.

**TM**
A Low indicates maintenance condition in the modem.

### DCD
In async operation, DCD remains Low regardless of data carrier (default), or it can be programmed to indicate received carrier signal is within the required timing and amplitude limits. In sync operation Low indicates the received carrier signal is within the required timing and amplitude limits.

### DSR
Low indicates modem is off-hook, and it is in data transmission mode, and the answer tone is being exchanged. CTS Low indicates modem is prepared to accept data.

### RTS
In async mode RTS is ignored. Under command control, in sync mode RTS can be ignored, or the modem can respond with a Low on CTS.

### DTR
&D0 command will cause the modem to ignore DTR. For &D1 the modem assumes the asynchronous command state on a Low to High transition of the DTR circuit. The &D2 command does the same as &D1 except the state of DTR will enable/disable auto answer. A Low to High transition of DTR after the &D3 command will cause the modem to assume the initialization state.

### B/C
Low configures the modem to CCITT V.21. High will configure the modem to Bell 103, when at 300 bps speed. This pin only affects the modem in 300 bps operation.
**TCL1, TCL0**

These pins are used as the serial clock and data for interface to an NVRAM. Refer to Figure 3. TCL0 is used to output a clock and serial data is read in on TCL1.

**AR**

This Auxiliary relay control is for switching a relay for voice or data calls. High is voice, Low is data.

**RI**

A Low signal from DAA indicates line ringing. This input is ignored when the modem is configured for leased line. This signal should follow the ring cadence.

**OH**

Low controls off hook. High indicates on hook. When dialing, this control is used to pulse dial the line.

**SH**

Used as a telephone voice to data switch or vice versa. Any logic level transition will toggle the modem state. This input is ignored, if a software command attempts to switch the modem between voice and data.

**AA**

Used as an indicator for Auto Answer status and Ring indicator. Active low.

**LCLLB**

A Low will set the modem in the local analog loopback test mode. Logic Low levels applied simultaneously to REMLB and LCLLB pins, sets the modem to the local digital loopback.

**REMLB**

A logic Low on this pin initiates a remote loopback condition.

**SI**

Selects one of the two data rates or ranges of rates in the DTE to correspond to the rate in modem. Low selects the higher rate (2400 CCITT/1200 BELL) or range of rates. High selects the Low rate or range of rates.

**D/S**

A Low on this pin will indicate the smart mode which will respond to all commands. A High will ignore all commands.

**VREF**

Voltage reference for the analog to digital converter should be connected to the 89027 AVcc.

**VPD**

The internal RAM power down supply voltage to be connected to 5 Volts during normal operation.

**S/Ä**

The function of this pin is re-defined as external NVRAM CE.

**CONFIG**

Low indicates availability of custom software modules in off-chip memory.

**EA**

When high, memory access from address 2000H to 4000H are directed to on-chip ROM. When low, all Memory access is directed to off-chip memory.

**JS**

Low is used to pulse A and A1 leads to control a 1A2 Key System jack.

---

**89C026XE ABSOLUTE MAXIMUM RATINGS**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Min</th>
<th>Max</th>
</tr>
</thead>
<tbody>
<tr>
<td>Temperature Under Bias</td>
<td>-10</td>
<td>+80°C</td>
</tr>
<tr>
<td>Storage Temperature</td>
<td>-40</td>
<td>+125°C</td>
</tr>
<tr>
<td>Voltage from Any Pin to Vss or AGND</td>
<td>-0.3V</td>
<td>+7.0V</td>
</tr>
<tr>
<td>Average Output Current from Any Pin</td>
<td>10 mA</td>
<td></td>
</tr>
<tr>
<td>Power Dissipation</td>
<td>1.5 Watts</td>
<td></td>
</tr>
</tbody>
</table>

*Notice: Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions above those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

**NOTICE: Specifications contained within the following tables are subject to change.**

---

4-34
OPERATING CONDITIONS

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Parameter</th>
<th>Min</th>
<th>Max</th>
<th>Units</th>
</tr>
</thead>
<tbody>
<tr>
<td>TA</td>
<td>Ambient Temperature Under Bias</td>
<td>0</td>
<td>+70</td>
<td>C</td>
</tr>
<tr>
<td>VCC</td>
<td>Digital Supply Voltage</td>
<td>4.75</td>
<td>5.25</td>
<td>V</td>
</tr>
<tr>
<td>FREQ</td>
<td>CLKIN Frequency 12.96 Mhz</td>
<td>-0.01%</td>
<td>+0.01%</td>
<td></td>
</tr>
<tr>
<td>CLIN2</td>
<td>Frequency 270 KHz</td>
<td>-0.01%</td>
<td>+0.01%</td>
<td></td>
</tr>
<tr>
<td>VPD</td>
<td>Power-Down Supply Voltage</td>
<td>4.75</td>
<td>5.25</td>
<td>V</td>
</tr>
</tbody>
</table>

NOTE:
V_{BBS} should be connected to AGND through a 0.01 μF capacitor. AGND, V_{SS} and the 89027 V_{SS}, AGND must be nominally at the same potential.

DC CHARACTERISTICS

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Parameter</th>
<th>Min</th>
<th>Max</th>
<th>Units</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>VIL</td>
<td>Input Low Voltage</td>
<td>-0.3</td>
<td>+0.8</td>
<td>V</td>
<td>Except RST</td>
</tr>
<tr>
<td>VIL1</td>
<td>Input Low Voltage, RST</td>
<td>-0.3</td>
<td>+0.7</td>
<td>V</td>
<td></td>
</tr>
<tr>
<td>VIH</td>
<td>Input High Voltage</td>
<td>2.0</td>
<td>V_{CC}+0.5</td>
<td>V</td>
<td>Except RST, NMI, CLKIN</td>
</tr>
<tr>
<td>VIH1</td>
<td>Input High Voltage, RST Rising</td>
<td>2.4</td>
<td>V_{CC}+0.5</td>
<td>V</td>
<td></td>
</tr>
<tr>
<td>VIH2</td>
<td>Input High Voltage, RST Falling</td>
<td>2.1</td>
<td>V_{CC}+0.5</td>
<td>V</td>
<td></td>
</tr>
<tr>
<td>VIH3</td>
<td>Input High Voltage, NMI, CLKIN</td>
<td>2.4</td>
<td>V_{CC}+0.5</td>
<td>V</td>
<td></td>
</tr>
<tr>
<td>VOL</td>
<td>Output Low Voltage</td>
<td>0.45</td>
<td>V</td>
<td>See Note 1.</td>
<td></td>
</tr>
<tr>
<td>VOH</td>
<td>Output High Voltage</td>
<td>2.4</td>
<td>V</td>
<td>See Note 2.</td>
<td></td>
</tr>
<tr>
<td>ICC</td>
<td>V_{CC} Supply Current</td>
<td>55</td>
<td>mA</td>
<td>All outputs disconnected</td>
<td></td>
</tr>
<tr>
<td>IPD</td>
<td>V_{PD} Supply Current</td>
<td>1</td>
<td>mA</td>
<td>Normal operation and Power-Down</td>
<td></td>
</tr>
<tr>
<td>IREF</td>
<td>V_{REF} Supply Current</td>
<td>15</td>
<td>mA</td>
<td></td>
<td></td>
</tr>
<tr>
<td>IL</td>
<td>Input Leakage Current</td>
<td>±10</td>
<td>μA</td>
<td>V_{IN}=0 to V_{CC} See Note 3</td>
<td></td>
</tr>
<tr>
<td>IH</td>
<td>Input High Current to EA</td>
<td>100</td>
<td>μA</td>
<td>V_{IH}=2.4V</td>
<td></td>
</tr>
<tr>
<td>IIL</td>
<td>Input Low Current</td>
<td>-100</td>
<td>μA</td>
<td>V_{IL}=0.45V See Note 4</td>
<td></td>
</tr>
<tr>
<td>IIL1</td>
<td>Input Low Current to RST</td>
<td>-2</td>
<td>mA</td>
<td>V_{IL}=0.45V</td>
<td></td>
</tr>
<tr>
<td>IIL2</td>
<td>Input Low Current S/A, SH, RI, READY</td>
<td>-50</td>
<td>μA</td>
<td>V_{IL}=0.45V</td>
<td></td>
</tr>
<tr>
<td>Cs</td>
<td>Pin Capacitance (Any Pin to V_{SS})</td>
<td>10</td>
<td>pF</td>
<td>1 MHz</td>
<td></td>
</tr>
</tbody>
</table>

NOTES:
1. I_{OL} = 0.36 mA for pins TClO, TCl1, B/C, RTS, CTS, DSR, DCD, SI, AR, and OH. Also if AD0 - AD15 are configured as I/O ports.
2. I_{OL} = 2.0 mA for TM, CLKOUT, ALE, BHE, RD, WR, RXD, TCLK, and AD0 - AD15 when used as external memory bus.
3. I_{OH} = 20 μA for pins TClO, TCl1, B/C, RTS, CTS, DSR, DCD, SI, AR, and OH.
4. I_{OH} = 200 μA for TM, CLKOUT, ALE, BHE, RD, WR, RXD, TCLK, and AD0 - AD15 when used as external memory bus.
5. Power must be applied to the device in the following sequence: V_{SS} first, then V_{CC}.
A.C. CHARACTERISTICS \( V_{CC}, V_{PD} = 4.75V \) to 5.25V; \( T_A = 0^\circ C \) to 70°C; CLKIN = 12.96 MHz

Test Conditions: Load capacitance on output pins = 80 pF  
\( T_{OSC} = 1/12.96 \) MHz

The memory system must meet these specifications to work with 89C026XE

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Parameter</th>
<th>Min</th>
<th>Max</th>
<th>Units</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>( T_{AVVV} )</td>
<td>Address Valid to READY Setup</td>
<td>( 2T_{OSC} - 75 ) ns</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( T_{LLYV} )</td>
<td>ALE Low to READY Setup</td>
<td>( T_{OSC} - 72 ) ns</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( T_{LYLY} )</td>
<td>Non READY Time</td>
<td>No upper limit ns</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( T_{CLLYX} )</td>
<td>READY Hold after CLKOUT Low</td>
<td>( T_{OSC} - 72 ) ns</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( T_{CLLYX} )</td>
<td>READY Hold after ALE Low</td>
<td>( T_{OSC} - 15 ) ns</td>
<td>( 2T_{OSC} - 40 ) ns</td>
<td></td>
<td>(Note 1)</td>
</tr>
<tr>
<td>( T_{AVGV} )</td>
<td>Address Valid to Buswidth Setup</td>
<td>( 2T_{OSC} - 70 ) ns</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( T_{LLGV} )</td>
<td>ALE Low to Buswidth Setup</td>
<td>( T_{OSC} - 70 ) ns</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( T_{CLGX} )</td>
<td>Buswidth Hold after CLKOUT Low</td>
<td>0 ns</td>
<td>( 3T_{OSC} - 60 ) ns</td>
<td></td>
<td></td>
</tr>
<tr>
<td>( T_{AVDV} )</td>
<td>Address Valid to Input Data Valid</td>
<td>( 3T_{OSC} - 60 ) ns</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( T_{RLDV} )</td>
<td>RS Active to Input Data Valid</td>
<td>( T_{OSC} - 23 ) ns</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( T_{CLDV} )</td>
<td>CLKOUT Low to Input Data Valid</td>
<td>( T_{OSC} - 50 ) ns</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( T_{RHDZ} )</td>
<td>End of RD to Input Data Float</td>
<td>( T_{OSC} - 20 ) ns</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( T_{RXDX} )</td>
<td>Data Hold after RD Inactive</td>
<td>0 ns</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

NOTES:
1. If max is exceeded, additional wait states will occur.
The 89C026XE will meet these specifications:

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Parameter</th>
<th>Min</th>
<th>Max</th>
<th>Units</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>FCLKIN</td>
<td>CLKIN Frequency</td>
<td>12.95870</td>
<td>12.96129</td>
<td>MHz</td>
<td>12.96 ± 0.01%</td>
</tr>
<tr>
<td>FCLKIN2</td>
<td>CLKIN2 Frequency</td>
<td>269.973</td>
<td>270.027</td>
<td>KHz</td>
<td>270 ± 0.01%</td>
</tr>
<tr>
<td>TXHCH</td>
<td>XTAL1 High to CLKOUT High or Low</td>
<td>40</td>
<td>110</td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>TCHCL</td>
<td>CLKOUT Cycle Time</td>
<td>2TOSC</td>
<td>ns</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TCLCL</td>
<td>CLKOUT High Period</td>
<td>Tosc - 10</td>
<td>Tosc + 10</td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>TCLLH</td>
<td>CLKOUT Falling Edge to ALE Rising</td>
<td>-5</td>
<td>15</td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>TLLCH</td>
<td>ALE Falling Edge to CLKOUT Rising</td>
<td>-15</td>
<td>15</td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>TLLHL</td>
<td>ALE Falling Edge to ALE Falling</td>
<td>Tosc - 10</td>
<td>Tosc + 10</td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>TAVLL</td>
<td>Address Setup to ALE Falling Edge</td>
<td>Tosc - 15</td>
<td>ns</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TLLAX</td>
<td>Address Hold after ALE Falling Edge</td>
<td>Tosc - 35</td>
<td>ns</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TLLRW</td>
<td>ALE Falling Edge to RD Falling Edge</td>
<td>Tosc - 40</td>
<td>ns</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TRLCL</td>
<td>RD Low to CLKOUT Falling Edge</td>
<td>10</td>
<td>30</td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>TRLLH</td>
<td>RD Low to Address Float</td>
<td>Tosc - 5</td>
<td>ns</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TRRHL</td>
<td>RD Rising Edge to ALE Rising Edge</td>
<td>Tosc</td>
<td>Tosc + 25</td>
<td>ns</td>
<td>(Note 2)</td>
</tr>
<tr>
<td>TRLAX</td>
<td>RD Low to Address Float</td>
<td>10</td>
<td>ns</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TLLWL</td>
<td>ALE Falling Edge to WR Falling Edge</td>
<td>Tosc - 10</td>
<td>ns</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TCLWL</td>
<td>CLKOUT Low to WR Rising Edge</td>
<td>0</td>
<td>25</td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>TQVWH</td>
<td>Data Stable to WR Rising Edge</td>
<td>Tosc - 20</td>
<td>ns</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TCHWH</td>
<td>CLKOUT High to WR Rising Edge</td>
<td>-10</td>
<td>10</td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>TWLWH</td>
<td>WR Low Period</td>
<td>Tosc - 30</td>
<td>ns</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TWHQX</td>
<td>Data Hold after WR Rising Edge</td>
<td>Tosc - 10</td>
<td>ns</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TWHLH</td>
<td>WR Rising Edge to ALE Rising Edge</td>
<td>Tosc - 10</td>
<td>Tosc + 15</td>
<td>ns</td>
<td>(Note 2)</td>
</tr>
<tr>
<td>TWHBX</td>
<td>BHE, INST HOLD after WR Rising Edge</td>
<td>Tosc - 10</td>
<td>ns</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

NOTES:
1. This specification is not tested, but is verified by design analysis and/or derived from other tested parameters.
Figure 6. Bus Signal Timings
89027 OVERVIEW

The 89027 is a 28 pin CMOS analog front end device, which performs most of the complex filtering functions required in modem transmitters and receivers. A general block diagram of this chip is provided in Figure 7. Most of the analog signal processing functions in this chip are implemented with CMOS switched capacitor technology. The 89027 functions are controlled by 89C026XE, through a high speed serial data link.

During FSK transmit operation, the 89027 receives digitally synthesized mark and space sinusoid amplitude information from the 89C026XE. The 89027 converts the signal to its analog equivalent, filters it, and transmits it to the telephone line. For QAM transmission, the signal constellation points are transferred to the 89027. This information is modulated into an analog signal, passed through spectral shaping filters, combined with the necessary guard tone, smoothed by a low pass filter, and transmitted to the line. Prior to transmitting either FSK or QAM signals to the telephone line, the 89027 adjusts the signal gain through an on-board programmable gain amplifier.

During the receive operation, the received FSK and QAM signals are passed through anti-alias filters, bandsplit filters, automatic gain control and carrier detect circuits, a Hilbert transform filter, and the output sent to the 89C026XE processor as analog signals.

Other functions provided by the 89027 are: an on-board two wire to four wire circuit with disable capability, an audio monitor output with software configurable gain, and a programmable gain transmit signal.

The 89027 is available in 28 pin plastic DIP and PLCC packages.

Figure 7. 89027 Block Diagram
89027 PINOUT

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Function (89027)</th>
<th>Direction</th>
<th>Pin No.</th>
</tr>
</thead>
<tbody>
<tr>
<td>VCC</td>
<td>Positive Power Supply (Digital)</td>
<td>+5V</td>
<td>28</td>
</tr>
<tr>
<td>VBB</td>
<td>Negative Power Supply</td>
<td>−5V</td>
<td>15</td>
</tr>
<tr>
<td>VSS</td>
<td>Digital Ground</td>
<td>DGND</td>
<td>24</td>
</tr>
<tr>
<td>AGND</td>
<td>Analog Ground</td>
<td>AGND</td>
<td>21</td>
</tr>
<tr>
<td>AVCC</td>
<td>Positive Power Supply (Analog)</td>
<td>+5</td>
<td>7</td>
</tr>
<tr>
<td>X1</td>
<td>Xtal Oscillator</td>
<td>In</td>
<td>23</td>
</tr>
<tr>
<td>X2</td>
<td>Xtal Oscillator</td>
<td>Out</td>
<td>25</td>
</tr>
<tr>
<td>CLKOUT</td>
<td>12.96 MHz Clock Output to 89C026XE</td>
<td>Out</td>
<td>26</td>
</tr>
<tr>
<td>CLKOUT2</td>
<td>270 KHz Clock Output to 89C026XE</td>
<td>Out</td>
<td>19</td>
</tr>
<tr>
<td>RST</td>
<td>Chip reset (active low)</td>
<td>In</td>
<td>20</td>
</tr>
<tr>
<td>HYB</td>
<td>Enable on-chip hybrid (1)</td>
<td>In</td>
<td>10</td>
</tr>
<tr>
<td>AZ1</td>
<td>Auto-zero capacitor</td>
<td>Out</td>
<td>16</td>
</tr>
<tr>
<td>AZ2</td>
<td>Auto-zero capacitor</td>
<td>In</td>
<td>17</td>
</tr>
<tr>
<td>SDATA</td>
<td>Serial data from 89C026XE</td>
<td>In</td>
<td>2</td>
</tr>
<tr>
<td>SCLK</td>
<td>Serial clock from 89C026XE</td>
<td>In</td>
<td>1</td>
</tr>
<tr>
<td>TSYNC</td>
<td>Transmitter sync from 89C026XE</td>
<td>In</td>
<td>3</td>
</tr>
<tr>
<td>STR</td>
<td>Symbol timing to 89C026XE</td>
<td>Out</td>
<td>27</td>
</tr>
<tr>
<td>ED</td>
<td>Receiver energy detect to 89C026XE</td>
<td>Out</td>
<td>18</td>
</tr>
<tr>
<td>I</td>
<td>In phase received signal to 89C026XE</td>
<td>Out</td>
<td>13</td>
</tr>
<tr>
<td>Q</td>
<td>Quadrature-phase received signal to 89C026XE</td>
<td>Out</td>
<td>14</td>
</tr>
<tr>
<td>AO1</td>
<td>Transmitter output</td>
<td>Out</td>
<td>6</td>
</tr>
<tr>
<td>AO2</td>
<td>Receiver input</td>
<td>In</td>
<td>12</td>
</tr>
<tr>
<td>AMP</td>
<td>Output to monitor speaker</td>
<td>Out</td>
<td>11</td>
</tr>
<tr>
<td>TX0</td>
<td>Transmitter level control (LSB) (1)</td>
<td>In</td>
<td>9</td>
</tr>
<tr>
<td>TX1</td>
<td>Transmitter level control (1)</td>
<td>In</td>
<td>8</td>
</tr>
<tr>
<td>TX2</td>
<td>Transmitter level control (1)</td>
<td>In</td>
<td>5</td>
</tr>
<tr>
<td>TX3</td>
<td>Transmitter level control (MSB)(1)</td>
<td>In</td>
<td>4</td>
</tr>
</tbody>
</table>

NOTE:
1. When held high, these pins should be connected through 10K resistors to AVGL.

89027 Pinout Description

TX0-3
These four pins control the transmitted signal level. The output level can be adjusted from −1 dBm to −16 dBm in 1 dB steps.

HYB
This pin enables the on-chip hybrid. A line impedance matching network must be connected between AO1 and AO2 when HYB is enabled. If HYB is disabled and an external 4W/2W hybrid is used, the hybrid receive path must be amplified by 6 dB.
**ABSOLUTE MAXIMUM RATINGS**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Min</th>
<th>Typ</th>
<th>Max</th>
<th>Units</th>
</tr>
</thead>
<tbody>
<tr>
<td>Temperature Under Bias</td>
<td>-10</td>
<td></td>
<td>+80°C</td>
<td></td>
</tr>
<tr>
<td>Storage Temperature</td>
<td>-40</td>
<td></td>
<td>+125°C</td>
<td></td>
</tr>
<tr>
<td>All Input and Output Voltages with Respect to V_\text{BB}</td>
<td>-0.3V</td>
<td></td>
<td>+13.0V</td>
<td></td>
</tr>
<tr>
<td>All Input and Output Voltages with Respect to V_\text{CC} &amp; AV_\text{CC}</td>
<td>-13.0V</td>
<td></td>
<td>0.3V</td>
<td></td>
</tr>
</tbody>
</table>

**NOTE:**
1. Applies to pins SCLK, SDATA, TSYNC, RST, HYB, TX0–TX3 only.

**POWER DISSIPATION**

Ambient Temp = 0° to 70°C, V_\text{CC} = AV_\text{CC} = 5 ± 5%, V_\text{SS} = AGND = 0.

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Parameter</th>
<th>Min</th>
<th>Typ</th>
<th>Max</th>
<th>Units</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alc_1</td>
<td>AV_\text{CC} Operating Current</td>
<td>19</td>
<td>25</td>
<td>mA</td>
<td></td>
</tr>
<tr>
<td>lcc_1</td>
<td>V_\text{CC} Operating Current</td>
<td>7</td>
<td>10</td>
<td>mA</td>
<td></td>
</tr>
<tr>
<td>lbb_1</td>
<td>V_\text{BB} Operating Current</td>
<td>-19</td>
<td>-25</td>
<td>mA</td>
<td></td>
</tr>
<tr>
<td>Alc_\text{s}</td>
<td>AV_\text{CC} Standby Current</td>
<td>0.2</td>
<td>1</td>
<td>mA</td>
<td></td>
</tr>
<tr>
<td>lcc_\text{s}</td>
<td>V_\text{CC} Standby Current</td>
<td>7</td>
<td>10</td>
<td>mA</td>
<td></td>
</tr>
<tr>
<td>lbb_\text{s}</td>
<td>V_\text{BB} Standby Current</td>
<td>-0.6</td>
<td>-2</td>
<td>mA</td>
<td></td>
</tr>
<tr>
<td>Pdo</td>
<td>Operating Power Dissipation</td>
<td>225</td>
<td>300</td>
<td>mW</td>
<td></td>
</tr>
<tr>
<td>Pds</td>
<td>Standby Power Dissipation</td>
<td>40</td>
<td>70</td>
<td>mW</td>
<td></td>
</tr>
</tbody>
</table>

**DC CHARACTERISTICS** (Ta = 0°C to 70°C, AV_\text{CC} = V_\text{CC} = 5V ± 5%, V_\text{BB} = 5V ± 5%, AGND = V_\text{SS} = 0V), supply voltage must be at the same potential as the 89C026XE power supply. Typical Values are for Ta = 25°C and nominal power supply values. Power must be applied in the following sequence: V_\text{SS}, AGND, V_\text{BB}, V_\text{CC}, and AV_\text{CC}. V_\text{CC}, AV_\text{CC} and 89C026XE V_\text{REF} must be nominally at the same potential.

Inputs: TX0, TX1, TX2, TX3, HYB, RST

Outputs: CLKOUT

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Parameter</th>
<th>Min</th>
<th>Max</th>
<th>Units</th>
<th>Test Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>l_\text{II}</td>
<td>Input Leakage Current</td>
<td>-10</td>
<td>+10</td>
<td>\mu A</td>
<td>V_\text{SS} \leq V_\text{in} \leq V_\text{CC}</td>
</tr>
<tr>
<td>l_\text{III}</td>
<td>Input Low Voltage</td>
<td>V_\text{SS}</td>
<td>0.8</td>
<td>V</td>
<td></td>
</tr>
<tr>
<td>V_\text{III}</td>
<td>Input High Voltage</td>
<td>2.0</td>
<td>V_\text{CC}</td>
<td></td>
<td></td>
</tr>
<tr>
<td>l_\text{IV}</td>
<td>Output Low Voltage</td>
<td>0.4</td>
<td>V</td>
<td></td>
<td>lol \geq -1.6 \text{mA}, 1 \text{TTL load}</td>
</tr>
<tr>
<td>V_\text{IV}</td>
<td>Output High Voltage</td>
<td>2.4</td>
<td>V</td>
<td></td>
<td>lch \leq 50 \mu A, 1 \text{TTL load}</td>
</tr>
<tr>
<td>V_\text{CO}</td>
<td>CLKOUT Low Voltage</td>
<td>0.4</td>
<td>V</td>
<td></td>
<td>C_1 = 60 \text{pF}</td>
</tr>
<tr>
<td>V_\text{COH}</td>
<td>CLKOUT High Voltage</td>
<td>2.4</td>
<td>V</td>
<td></td>
<td>C_1 = 60 \text{pF}</td>
</tr>
</tbody>
</table>
AC CHARACTERISTICS (Ta = 25°C, VCC = AVCC = 5 V, VSS = AGND = 0, VBB = -5 V)

ANALOG INPUTS: A02

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Min</th>
<th>Typ</th>
<th>Max</th>
<th>Units</th>
<th>Test Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>AO2 Input Voltage Range</td>
<td></td>
<td></td>
<td>-9</td>
<td>dBm</td>
<td></td>
</tr>
<tr>
<td>AO2 Input Resistance</td>
<td>10</td>
<td></td>
<td></td>
<td>Mohms</td>
<td>-3.5 &lt; Vin &lt; +3.5 V</td>
</tr>
<tr>
<td>AO2 Allowed DC offset</td>
<td>-30</td>
<td>+30</td>
<td></td>
<td>mV</td>
<td>Relative to AGND</td>
</tr>
</tbody>
</table>

AUTO ZERO CAPACITANCE

Capacitance = 0.015 μF
Tolerance = ±10 %
Voltage Rating = 10V
Type = Non-Electrolytic, low leakage.

CRYSTAL REQUIREMENTS

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Min</th>
<th>Typ</th>
<th>Max</th>
<th>Unit</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Frequency Accuracy (0°C−70°C)</td>
<td>-0.0035%</td>
<td>+0.0035%</td>
<td>12.960 Mhz</td>
<td></td>
<td>Refer to Figure 8</td>
</tr>
<tr>
<td>Rx</td>
<td>10</td>
<td>0.024</td>
<td>16</td>
<td>ohms</td>
<td></td>
</tr>
<tr>
<td>Cx</td>
<td>5.1</td>
<td>5.6</td>
<td>33 pF</td>
<td>pF</td>
<td></td>
</tr>
<tr>
<td>Co</td>
<td></td>
<td></td>
<td>+5%</td>
<td>33 pF</td>
<td>2 Load Capacitors</td>
</tr>
<tr>
<td>CL</td>
<td></td>
<td></td>
<td></td>
<td>±5%</td>
<td></td>
</tr>
</tbody>
</table>

Crystal Type: Series Resonant

ANALOG OUTPUTS: A01, AMP

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Min</th>
<th>Typ</th>
<th>Max</th>
<th>Units</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Load Resistance AO1 AMP</td>
<td>600</td>
<td></td>
<td></td>
<td>ohms</td>
<td></td>
</tr>
<tr>
<td>Load Resistance AMP</td>
<td>10</td>
<td></td>
<td></td>
<td>Kohms</td>
<td></td>
</tr>
<tr>
<td>Load Capacitance AMP</td>
<td></td>
<td>100</td>
<td>pF</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Audio Amp Gain AO1 to Amp</td>
<td>-9</td>
<td>-18</td>
<td>-24</td>
<td>dB</td>
<td>Software</td>
</tr>
<tr>
<td></td>
<td>-24</td>
<td>-75</td>
<td></td>
<td>dB</td>
<td>Selectable</td>
</tr>
<tr>
<td>Audio Amp Gain(1) AO2 to Amp</td>
<td>+12</td>
<td>+3</td>
<td>-4</td>
<td>dB</td>
<td>Software</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>-65</td>
<td>dB</td>
<td>Selectable</td>
</tr>
</tbody>
</table>

NOTE:
1. Assumes on-chip hybrid is enabled. When on-chip hybrid is disabled, gain with respect to AO2 is reduced by 6 dB.
NOTE:
1. For FSK, PSK, QAM xmit signal measured at AO1.

REFERENCE MANUAL

Overview
For reference purposes please refer to the 89024 Reference Manual which also contains a full description of the AT commands and S-registers supported by the 89C024XE Modem chip set.

ORDERING INFORMATION
Intel literature number: 296235-001
MNP APPLICATION PACKAGE
MNP CLASS 5 SOFTWARE
FOR THE 89C024XE MODEM CHIP SET

- MNP Class 1–5
- Error-Free Data Transfer
- Up to 200% Improvement in Throughput
- Hardware and Software Flow Control
- Single Processor
- Automatic Speed Matching
- Data Buffering
- DTE Interface Rates to 9600 bps

The MNP Application Package is a software solution which enables the 89C024XE Modem Chip Set to support the popular MNP error correction protocol, including Class 5 data compression. The firmware package includes: the 89C024XE High Level Protocol system source library, the MNP Engine object library, and support documentation. This firmware provides performance identical to the 89C024XE Modem Chip Set, and an extended AT command set which controls the MNP features.

Figure 1. 89C024XE MNP Implementation

*Microcom Network Protocol is a trademark of Microcom Inc.*
Additional Commands Implemented for MNP Feature Control.

An Maximum MNP Block Size
%An Set Auto-Reliable Fallback Character
Bn Transmit Break
 Gn Set Auto-Reliable Buffer
 Gn Set Modern Port Flow Control
Jn Bits per Second Rate Adjust
Kn Set Break Control
Nn Set Operating Mode
O Originent Reliable Link
On Set Serial Port Flow Control
Tn Set Inactivity Timer
Un Accept Reliable Link
Vn Modify Result Code Form
Xn Set XON/XOFF Pass-Through
Yn Switch to Reliable Mode
Zn Switch to Normal Mode

Functional Description

The MNP Application Package provides a cost-effective MNP solution, with minimum additional hardware required. Typically, two EPROMs, an 8 Kbyte static RAM, and a minimum number of latches and memory decoding logic are required. The memory system must be configured to satisfy the 89C024XE timing requirements. Optimum performance is achieved with a 16-bit bus EPROM, 8-bit bus RAM, 0-wait state memory system. A 1-wait state memory system will provide satisfactory Class 4 operation, but may impact throughput during MNP Class 5 operation.

The 89C024XE High Level Protocol system source library consists of the Initialization, Command Decoder, MNP Command Decoder, Call Progress Management, Handshake, Data Mode and Data Pump Modules. These modules can be customized to meet the designer's requirements.

The MNP Engine object library contains the relocatable object modules which execute the MNP class 1-5 protocol.

MNP

Class 1: Half-duplex operation, data throughput efficiency up to 70%.
Class 2: Full-duplex operation, data throughput efficiency up to 84%.
Class 3: Full-duplex operation, data throughput efficiency up to 108%.
Class 4: Full-duplex operation. Adaptive packet sizing optimizes data packet size for line conditions. Data throughput efficiency up to 120%.
Class 5: Full-duplex operation. Adaptive packet sizing and data compression. Data throughput efficiency up to 200%.

Technical Specifications

For technical specifications on CCITT V.22bis, V.22, V.21, Bell 212 and 103, see Intel 89C024XE data sheet.

Data Rate Compatibility: 110, 300 1200, 2400, 4800 and 9600 bps with speed matching.

Remote Modem Interface: MNP Class 1, 2, 3, 4 or 5 Error Correction Protocol.

Flow Control: X-ON/X-OFF; RTS/CTS;

Break Handling: Relayed with Attention Packets; Destructive and Non-Destructive/Expedited and Non-Expedited.

Terminal Data Rate Compatibility: 110, 300, 1200, 2400, 4800, 9600 bps.
The 89024 MEKPC is a 2400 bps, Hayes AT command set compatible PC card modem. It is provided as a
evaluation tool for Intel's 89024 Modem Chip Set and 82050 UART. It is not intended for use on phone lines,
and it is not FCC approved.
FUNCTIONAL DESCRIPTION

The MEKPC is a full functional 2400 bps pc modem card based on the 89024 modem chip set. It can be used with any IBM PC or PC compatible machine. It uses one of two serial communication ports available in a PC, COM1 and COM2. COM port selection is made via a jumper option on the MEKPC. The 82050 UART provides the interface between the 89024 and the PC bus. The 82050 is 100% software compatible with the 16450/8250A UART’s, allowing the modem to work with popular commercial communications software such as Cross Talk, Smartcom, SmarTerm, and PC term. A block diagram of the MEKPC is shown in Figure 1.

89024 MODEM CHIP SET

The 89024 is a highly integrated 2400 bps modem chip set providing a complete modem system in two chips, the 89027 and the 89026. The 89027 is an analog front end which implements the filters for modem receivers and transmitters. The 89026 is an application specific processor which performs modulation/demodulation and user interface functions. It also implements the industry standard “AT” command set.

82050 UART

The 82050 Asynchronous Communications Controller provides 100% software compatibility with the 16450/8250A in a 28 pin package. The smaller package size and its simpler system interface provides substantial board space savings. The 82050 is fabricated using CHMOS III technology for decreased power consumption and increased reliability.

COMMUNICATION SOFTWARE

For asynchronous PC communications, a software package developed by Intel called iTERM is provided. iTerm allows a PC to emulate an ASCII terminal. In addition, user friendly features and menu driven commands for setting terminal parameters and loading the function keys with AT command strings are provided.

DOCUMENTATION

Detailed information on the MEKPC89024, iTERM, 89024, and 82050 is provided. Following is a list of the enclosed documents:
1. MEKPC89024 User’s Guide
2. 89024 Reference Manual
3. 89024 Data Sheet
4. 82050 Data Sheet
5. 89024 Firmware Evaluation Report

89024 Reference Manual

The 89024 Reference Manual details design information for the 89024 Modem Chip Set. It provides descriptions and specifications of the two chips comprising the 89024, the 89026 and the 89027. In addition, it describes the control interface between the two chips.

The reference manual also provides a full description of all the “AT” commands and S-registers supported by the 89024 Modem Chip Set.

Intel literature number: 296235-001

MEKPC89024 ORDERING INFORMATION

Order Number
MEKPC89024, Q_0120
89024 MEK2
89024 ENHANCED MODEM EVALUATION KIT

- Single Board Evaluation Kit for the 89024 2400 bps Modem
- Recommended for Bit Error Rate Testing
- Intended as an 89024 Software Customization and Development Tool
- Comprehensive User's Manual
- PC Communications Package Included
- Includes Power Supply & User Wire-Wrap Area

OVERVIEW

The 89024 MEK2 is a stand alone Sync/Async Intelligent Modem evaluation kit. This MULTI-BUS II form-factor circuit board is a fully assembled and functional modem that can be used for demonstrating the capabilities of the 89024 2400 bps Intelligent modem chip-set. It is also a versatile tool for evaluating the chip-set Bit Error Rate performance, as well as customizing the 89024 software.

The board is equipped with a power supply module, eliminating the need for a lab power supply, as well as a comprehensive user's manual and an Intel PC Communications Software package (ITERM).

NOTE:
This product does not comply with FCC part 68 and part 15 requirements. It is intended for laboratory evaluation only.

Figure 1. 89024 MEK2 Block Diagram
SOFTWARE DEVELOPMENT

The 89024 MEK2 provides EPROM sockets for evaluating userdeveloped software and provides the necessary clocking provisions to interface to an SBE-96 In-Circuit Emulator board.

For evaluation of modem signal quality, a constellation (eye pattern) decoder circuit has been included on-board. All hardware/software configurable features of the 89024 chip-set are strap configurable on this board.

HARDWARE OVERVIEW

The board provides a convenient vehicle for conducting Bit Error Rate tests. It is a good example of simple double-sided PCB layout that meet stringent modem noise requirements. The DAA interface supports Voice/Data Communications, as well as, 1A2 Key System A lead control. A loud-speaker and amplifier provide audible indication of call progress to the user. A series of LED indicators display the status of essential modem functions.

SERIAL INTERFACE

A female DB-25 connector provides an RS-232/V.24 Sync/Async interface to the DTE. For Async PC communications, iTERM Communications Disk may be used to drive the modem. This software package allows a PC to emulate an ASCII terminal. The program has several user-friendly menus which accommodate setting terminal parameters and loading the PC Function Keys with AT command strings.

DOCUMENTATION

Detailed information on the 89024 MEK2, iTERM and 89024 is provided. Following is a list of the enclosed documents.

1. 89024 MEK2 User’s Guide
2. 89024 Reference Manual
3. 89024 Data Sheet
4. 89024 Firmware Evaluation Report

89024 Reference Manual

The 89024 Reference Manual details design information for the 89024 Modem Chip Set. It provides descriptions and specifications of the two chips comprising the 89024, the 89026 and the 89027. In addition, it describes the control interface between the two chips.

The reference manual also provides a full description of all the “AT” commands and S-registers supported by the 89024 Modem Chip Set.

Intel literature number: 296235-001

ORDERING INFORMATION

MEK2,Q_0122
MEK3 MODEM EVALUATION KIT

- Single Board Evaluation Kit for the 89024, 89C024XE and 89C024XE MNP Modems
- Onboard Constellation Decoder
- Software Customization and Development Platform for Intel Modem Chipsets
- Comprehensive User's Manual
- ITERM PC Communication Package
- User Wire-Wrap Area for H/W Customization
- Includes Power Supply

OVERVIEW

The MEK3 is a modem evaluation kit for Intel's modem line of products. These include 89024, 89C024XE and 89C024XE-MNP Modems. This evaluation board comes assembled and tested as a functional modem. It is also designed to provide onboard prototyping area for purposes of customization. The packaged unit comes with the 89C024XE chip-set socketed on the board itself and the 89024 chipset contained in a separate box.

The board is equipped with a power supply module, eliminating the need for a lab power supply. A comprehensive user's manual and a copy of the Intel PC communications Software package ITERM is also included.

Figure 1. MEK3 Block Diagram

NOTE:
This product is not FCC part 68 and part 15 approved. It is intended for laboratory evaluation only.
SOFTWARE DEVELOPMENT

The MEK3 provides EPROM sockets for evaluating user developed software and provides the necessary clocking provisions to interface to an SBE-96 In-Circuit Emulator for 89024 and ICE-196 for 89C024XE chip sets.

For evaluation of modem signal quality, a constellation (eye pattern) decoder circuit has been included on-board. All hardware/software configurable features of the 89024 and 89C024XE chip-sets are strap configurable on this board.

HARDWARE OVERVIEW

The board provides a convenient vehicle for conducting Bit Error Rate tests. It is a good example of simple double-sided PCB layout that meet stringent modem noise requirements. The DAA interface supports Voice/Data Communications, as well as, 1A2 Key System A lead control. A loud-speaker and amplifier provide audible indication of call progress to the user. A series of LED indicators display the status of essential modem functions. The board also provides MNP Class 5 operation in conjunction with 89C024XE modem chip set.

SERIAL INTERFACE

A female DB-25 connector provides an RS-232/V.24 Sync/Async interface to the DTE. For Async PC communications, iTERM Communications Disk may be used to drive the modem. This software package allows a PC to emulate an ASCII terminal.

The program has several user-friendly menus which accommodate setting terminal parameters and loading the PC Function Keys with AT command strings.

DOCUMENTATION

Detailed information on the MEK3, iTERM, the 89C024XE and 89024 kits and 89C024XE MNP is provided. Following is a list of the included documents.

1. MEK3 User’s Guide
2. 89024 Data Sheet
3. 89C024XE Data Sheet
4. 89024 Reference Manual

Reference Manual

The 89024 Reference Manual is available. It describes the 89024 chip-set and provides the specifications for the 89026 and 89027. This manual also describes the control interface between the two chips. It also provides a full description of all the “AT” commands and S-registers supported by the 89024 and 89C024XE Modem chip-sets.

Intel literature number: 296235-001
iATC Advanced Telecommunications Components for Analog and ISDN Applications
The Subscriber Line Datalink is a three-wire interface for synchronous data transfer between master and slave devices. Four full-duplex time-multiplexed 64 Kbps channels are supported on a serial ping-pong link. Each channel transfers a byte of data every 125 μs.

The SLD interface was developed primarily for telecommunications applications, where 8 KHz synchronous data transfers are the norm. It provides a standardized physical interface for the transfer of circuit-switched voice and data, signalling, and control channels to and from the individual subscriber circuits.

**INTERFACE DESCRIPTION**

The three wires of the SLD interface consist of a data clock (SCL), a data direction signal (SDIR), and a ping-pong data lead (SLD). The data clock and direction signals can be common to all slave devices connected to the interface controller. A separate SLD line is connected to each slave device. The slave devices on this interface receive the clock signals from the controller, and only drive the data line when so indicated by the direction signal. The controller generates both the SDIR direction signal and the SCL data clock, which are derived from the system clock.

The SLD line itself supports a 512 Kbps rate, as defined by the SCL clock. The data on this line is formatted as 32 bits of receive (towards slave) data and 32 bits of transmit (from slave) data. This pattern is repeated at an 8 KHz rate, with the direction of transmission being defined by the 8 KHz SDIR signal. When SDIR is high, data is transferred to the slave device, and when SDIR is low, data is transferred back to the master. Hence, the SDIR signal has a duty cycle of approximately 50%. The transmit and receive direction data is further divided into eight bytes, four transferred in each direction. The effective data rate over the SLD interface is 256 Kbps in each direction. Because all SLD lines handled by a controller share the same direction signal, these separate links are all synchronous. The exact use of the data channels on the SLD is determined by the devices connected to it.

**USE OF THE SLD LINK**

Figure 1 shows the SLD interface. The first and fifth bytes (channel A) represent the first voice/data channel, which corresponds to the B1 channel in ISDN applications. The second and sixth bytes (channel B) contain the second voice/data channel, which corresponds to the B2 channel for ISDN. The third byte is used as a control channel to program the features of the SLD slave device, while the seventh byte can be used to read back status. The fourth and eighth bytes are used to transport signalling information, or additional control and status information. The SLD link provides an efficient means of routing all this information between the master and slave devices.
SLD TIMING

The duty cycle of the 512 KHz SCL clock is typically 50%. However, because this signal is usually derived from the system clock, it may not be practical to generate a 50% duty cycle. For example, a 33% duty cycle clock may be generated if the system clock is 1.536 MHz (24 timeslot systems). All slave devices built to interface to the SLD should accept duty cycles of from 30% to 70%.

A special case exists for systems with a 1.544 MHz clock. It is not possible to derive a 512 KHz SCL clock from this system clock. For this case, SCL is allowed to have an instantaneous bit rate of 514.67 KHz, with a stretched clock cycle inserted to achieve a 512 Kbps average over the 125 μs frame. This stretched clock cycle may occur as the last clock of the frame, or as the first clock of the frame (see Figure 2).

Figure 2. SLD Timing for 1.544 MHz
GENERAL SLD TIMING SPECIFICATION

Following is a general SLD timing specification which can be used as a guideline in the design of SLD master or slave devices. It allows for data reception by the master on rising or falling edges of SCL and for data reception by the slave on falling edges, and if adhered to, is compatible to all currently existing SLD standard Intel components.

Figure 3. General SLD Timing
### General SLD Timing

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Parameter</th>
<th>Min</th>
<th>Typ</th>
<th>Max</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>T_{DSM}</td>
<td>Data Setup Time, Master(^{(1)})</td>
<td>150</td>
<td></td>
<td></td>
<td>ns</td>
</tr>
<tr>
<td>T_{DHM}</td>
<td>Data Hold Time, Master(^{(1)})</td>
<td>0</td>
<td></td>
<td></td>
<td>ns</td>
</tr>
<tr>
<td>T_{DSS}</td>
<td>Data Setup Time, Slave</td>
<td>200</td>
<td></td>
<td></td>
<td>ns</td>
</tr>
<tr>
<td>T_{DHS}</td>
<td>Data Hold Time, Slave</td>
<td>150</td>
<td></td>
<td></td>
<td>ns</td>
</tr>
<tr>
<td>T_{DOFF1}</td>
<td>SDIR to Slave Data High Z</td>
<td></td>
<td>50</td>
<td></td>
<td>ns</td>
</tr>
<tr>
<td>T_{DON1}</td>
<td>SDIR to Master Data On</td>
<td>70</td>
<td></td>
<td></td>
<td>ns</td>
</tr>
<tr>
<td>T_{DOFF2}</td>
<td>Master Data High Z to SDIR</td>
<td>20</td>
<td></td>
<td></td>
<td>ns</td>
</tr>
<tr>
<td>T_{DON2}</td>
<td>SDIR to Slave Data On</td>
<td>0</td>
<td></td>
<td></td>
<td>ns</td>
</tr>
<tr>
<td>T_{DIRR}</td>
<td>SCL to SDIR Rising Edge(^{(2)})</td>
<td>-150</td>
<td>100</td>
<td></td>
<td>ns</td>
</tr>
<tr>
<td>T_{DIRFR}</td>
<td>SCL Rising Edge to SDIR Falling Edge(^{(2)})</td>
<td>-150</td>
<td></td>
<td></td>
<td>ns</td>
</tr>
<tr>
<td>T_{DIRFF}</td>
<td>SDIR Falling Edge to SCL Falling Edge(^{(2)})</td>
<td>200</td>
<td></td>
<td></td>
<td>ns</td>
</tr>
<tr>
<td></td>
<td>SCL Duty Cycle(^{(3)})</td>
<td>30</td>
<td>50</td>
<td>70</td>
<td>%</td>
</tr>
<tr>
<td></td>
<td>SCL Frequency(^{(4)})</td>
<td></td>
<td>512</td>
<td>514.7</td>
<td>KHz</td>
</tr>
<tr>
<td></td>
<td>Rise and Fall Times, All Signals</td>
<td></td>
<td></td>
<td>50</td>
<td>ns</td>
</tr>
<tr>
<td></td>
<td>SDIR Period</td>
<td></td>
<td>125</td>
<td></td>
<td>μs</td>
</tr>
</tbody>
</table>

**NOTES:**

1. SLD master can receive on falling or rising edges.
2. It is the responsibility of the master to control SDIR properly to allow reception of data at SLD turn-around points. The TDIR times above do not guarantee data reception on both rising and falling edges.
3. Not all slave devices will accept this duty cycle range. Refer to the timing for the specific slave devices the master will interface with.
4. SCL may be 514.7 KHz (instantaneous) for 1.544 MHz system clocks. However, not all slave devices will accept this. Refer to the timing for the specific slave devices the master will interface with. SCL must have 64 pulses per SDIR cycle in any case.
The Intel iATC 29C48 Feature Control Combo is a low cost, user-programmable, fully integrated PCM Codec with transmit/receive filters fabricated in a CMOS technology. This technology is built on CHMOS and will allow the 29C48 to realize the same excellent transmission performance as in the Intel 2913/2914 combo while achieving the low power consumption typical of CMOS circuits.

The 29C48 supports the analog subscriber with a variety of added per-line features to the normal BORSCHT functions associated with the analog line circuit. Some of these features include secondary analog input channel, programmable transmit and receive gain, custom hybrid balancing network selection, and programmable $\mu$ or A-law conversions. Additionally, the 29C48 can operate on either the A or B channel of the SLD interface, allowing two combos to be connected to one SLD link. In order to facilitate the SLIC interface in this configuration, the 29C48 generates SLIC chip select signals for the proper routing of signaling information.

A unique feature of the 29C48 is programmable tone injection. This feature and its SLD interface makes it particularly easy to use in conjunction with Intel's advanced tranceivers, such as the iATC 29C53AA, in subscriber equipment environments. The 29C53AA handles transfer of voice and feature control information to the 29C48.

---

**Figure 1. Pin Configurations**

![Pin Configurations](image-url)
Figure 2. Block Diagram
Table 1. Pin Names

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>VFX</td>
<td>Analog Input</td>
</tr>
<tr>
<td>VFR</td>
<td>Analog Output</td>
</tr>
<tr>
<td>GNDD</td>
<td>Digital Ground</td>
</tr>
<tr>
<td>GNDA</td>
<td>Analog Ground</td>
</tr>
<tr>
<td>VCC</td>
<td>Power (+5V)</td>
</tr>
<tr>
<td>VBB</td>
<td>Power (-5V)</td>
</tr>
<tr>
<td>B/Ä</td>
<td>Channel Selection</td>
</tr>
<tr>
<td>SCS</td>
<td>SLIC Chip Select</td>
</tr>
</tbody>
</table>

Table 2. Pin Description

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>VCC</td>
<td>Most positive supply; input voltage is +5V ± 5%.</td>
</tr>
<tr>
<td>VBB</td>
<td>Most negative supply; input voltage is -5V ± 5%.</td>
</tr>
<tr>
<td>GNDA</td>
<td>Analog ground return line. Not internally connected to GNDD.</td>
</tr>
<tr>
<td>GNDD</td>
<td>Digital ground return line. Not internally connected to GNDA.</td>
</tr>
<tr>
<td>VFX</td>
<td>Analog voice input to transmit channel. Input impedance is typically larger than 100 KΩ.</td>
</tr>
<tr>
<td>TG1</td>
<td>Inverting input to transmit gain adjusting op-amp. Feedback point for external gain adjusting resistor network or frequency compensation network. Input impedance is typically larger than 10 MΩ.</td>
</tr>
<tr>
<td>TG2</td>
<td>Output of the transmit gain adjusting op-amp. Will drive external gain adjusting resistor network as well as frequency compensation network with an impedance of at least 10 KΩ.</td>
</tr>
<tr>
<td>VFR</td>
<td>Receive voice output. Capable of directly driving transformer hybrids or impedance loads of 600 Ω.</td>
</tr>
<tr>
<td>EBN</td>
<td>Input to the hybrid balancing circuit. Input impedance is typically larger than 10 MΩ.</td>
</tr>
<tr>
<td>EBN1</td>
<td>Input connected to a grounded switch. The switch’s on resistance is not greater than 600 Ω.</td>
</tr>
<tr>
<td>EBN2</td>
<td>Input connected to a grounded switch. The switch’s on resistance is not greater than 600 Ω.</td>
</tr>
<tr>
<td>EBN3/TI</td>
<td>This pin is multiplexed according to the feature control registers. When programmed to be EBN3, it is an input connected to a grounded switch. The switch’s on resistance is not greater than 600 Ω. If this pin is programmed to be TI, an analog signal applied on this pin will be added to the received voice signal before the receive power amplifier.</td>
</tr>
<tr>
<td>SCL</td>
<td>Subscriber clock. This is an input which should be 512 KHz with a duty cycle ranging from 25% to 75%. Input will accept TTL levels.</td>
</tr>
<tr>
<td>SDIR</td>
<td>Subscriber direction signal and frame sync input. When high, SLD becomes an input and data is received by the 29C48. When low, the output buffer on the 29C48 SLD pin is enabled and data is transmitted by the 29C48. Input will accept TTL levels.</td>
</tr>
<tr>
<td>SLD</td>
<td>Subscriber Line Datalink. A 512 Kbps bi-directional serial data port, which is clocked by SCL. SLD becomes a TTL compatible input when SDIR is high and an output capable of driving one TTL load when SDIR is low, during the appropriate SLD fields for the assigned channel.</td>
</tr>
<tr>
<td>B/Ä</td>
<td>Pin strapped to assign the 29C48 to process either A or B channel information from the SLD bus. A low level (GNDD) on this pin selects channel A, a high level (VCC) channel B.</td>
</tr>
<tr>
<td>SCS</td>
<td>This pin is a TTL compatible output capable of driving one TTL load: when low, it informs a SLIC device connected to the same SLD bus as the 29C48 that it can process the receive and transmit signaling data of the present SLD frame.</td>
</tr>
<tr>
<td>SAI</td>
<td>Secondary analog input.</td>
</tr>
</tbody>
</table>
FUNCTIONAL DESCRIPTION

The 29C48 is a combined channel filter and PCM codec for use in ISDN subscriber equipment or analog line interface circuit boards in digital switching systems.

The 29C48 incorporates features which make it particularly suited to subscriber applications. Tone injection allows easy implementation of DTMF feedback and side tone injection, and secondary analog signal input allows remote control and monitoring.

(See Figure 3 for a typical ISDN subscriber equipment application.)

For analog line interface circuit boards this device resides between the circuitry which provides the "BORSHT" functions for a given line, and the shared line board controller. It provides the transmit and receive voice-path filtering and companded analog-to-digital and digital-to-analog conversions necessary to interface a full duplex voice telephone circuit with the PCM highways of a time division multiplexed (TDM) system. (See Figures 4a and 4b for typical line card applications.)

Figure 3. Subscriber Equipment

Figure 4a. Analog Line Card with Discrete or Electronic Parallel Control SLICs
**TRANSMIT AND RECEIVE OPERATION**

**Transmit Filter**

A low pass anti-aliasing section is included on chip. This section typically provides 35 dB attenuation at the sampling frequency. No external components are required to provide the necessary anti-aliasing function for the switched capacitor section of the transmit filter.

The passband section provides flatness and stopband attenuation which fulfills the AT&T D3/D4 specification and the CCITT G.714 recommendation. The 29C48 specifications meet the digital class 5 central office switching systems requirements. The transmit filter transfer characteristics and specifications will be within the limits shown in Figure 12.

A high pass section configuration rejects low frequency noise from 50 and 60 Hz power lines, 17 Hz European electric railroads, ringing frequencies and their harmonics, and other low frequency noise. The transmit filter also provides additional loss at 12 KHz and 16 KHz to support metering pulses.

**Encoding**

The output of the transmit filter or the secondary analog input is internally sampled by the encoder and held on an internal sample and hold capacitor. DC offset is corrected by an on-chip auto zero circuit. The signal is then encoded and presented as PCM data on the SLD lead. (First or second byte of the transmit half-frames depending upon the channel assignment of the device.)

**Decoding**

The PCM word received on the SLD lead (first or second byte of the receive half-frame, depending upon the channel assignment of the device) is sent to the decoder after a serial to parallel conversion. The decoded value is held on an internal sample and hold capacitor.

**Receive Filter**

The receive section of the filter provides a passband flatness and stopband rejection which fulfills the AT&T D3/D4 specification and the CCITT G.714 recommendation. It also provides additional loss at 12 KHz and 16 KHz. The receive filter transfer characteristics and specifications will be within the limits shown in Figure 13.

**GENERAL OPERATION**

**External Gain Setting**

Both transmit and receive gain levels can be modified by external resistors during line card assembly. The value of transmit gain is adjusted by connecting resistors RT1 and RT2 (see Figure 5) at the two external gain setting control pins, TG1 and TG2. These two pins are the input and output of an on-board gain amplifier stage, and the resistors provide the necessary input and feedback for gain control. External gain of up to 20 dB can be set without degrading the performance of the amplifier. The value of external gain is given by:

\[ A = 1 + \frac{RT1}{RT2} \]

For unity gain, pins TG1 and TG2 are tied together.
For the receive section, the external gain can be set by the external resistors, RR1 and RR2. There are two possible ways of implementing the gain control. The first is illustrated in Figure 6a, where the value of the receive gain is given by:

\[ A = \frac{RR2}{RR1 + RR2} \]

The value of RR1 + RR2 should not be less than 600\(\Omega\) to avoid degrading the output power stage’s performance. The second way of implementing the receive gain is shown in Figure 6b, where pin EBN3/T1 is used. The value of the receive gain in this configuration is given by:

\[ A = 1 + \frac{RR1}{RR2} \]

Hybrid Balancing Network

Three external balancing networks can be applied to the 29C48 by the user to accommodate varying subscriber loop characteristics (see Figure 7 for external connections). Feature control allows the grounding of any combination of these networks in order to best suit a particular application. Feature control also allows the user to select a gain of 0.0 or +6.0 dB in the balance signal path to suit the type of SLIC used.

FREQUENCY COMPENSATION

The user may, if desired, compensate for the frequency response characteristics of the SLIC by adjusting the frequency response of the transmission chain. This can be accomplished in the same way as the external gain setting is done in the transmit and receive directions. But, instead of using purely resistive impedances, resistor and capacitor networks have to be used to achieve complex impedances. The two compensation schemes are shown in Figures 8a and 8b. The gains in the transmit and receive directions are respectively:

for Figure 8a \( A = 1 + \frac{ZX1}{ZX2} \)

for Figure 8b \( A = 1 + \frac{ZR1}{ZR2} \)

SECONDARY ANALOG INPUT

Although the main application of the 29C48 will be for voice transmission, it also offers a secondary unfiltered input channel. Narrow band analog signals can be supplied through this channel for remote loop testing and various control uses.

The secondary analog input channel is accessed under software control through the SAI input. When the SAIE bit in the feature control register #3 is set to a

![Figure 5. Transmit Gain Setting](image)

![Figure 6a. Receive Gain Setting](image)

![Figure 6b. Receive Gain Setting](image)
logical one, the 29C48 will encode and transmit the signal present at the SAI input. The 29C48 will switch back to transmission of the voice signal as soon as the SAIE bit is set back to a logical zero.

**TONE INJECTION**

When specified by the feature control memory, an audio frequency signal applied to the EBN3/T1 pin will be added to the receive voice signal at the power amplifier. This feature allows easy implementation of DTMF feedback and side tone injection in ISDN telephone applications, as well as injection of call waiting or metering tones in line card applications. A typical application is shown in Figure 9. Here VFR is the combination of the receive voice signal (V0) and two tones (V1 and V2).

\[ VFR = 2V0 - (V1 + V2)/2 \]

**CHANNEL ASSIGNMENT**

Two 29C48s can be attached to the same SLD line to exchange information with the SLD master during each SLD frame.

The B/\overline{A} pin of the 29C48 is used to assign a voice channel of the SLD frame to the device. When the B/\overline{A} pin is tied low, the 29C48 operates as an A-channel combo, receiving and transmitting voice during the first and fifth bytes of the SLD frame. When this pin is tied high, the 29C48 operates as a B-channel combo, receiving and transmitting voice during the second and sixth bytes of the SLD frame.

---

**Figure 7. Balance Networks**

**Figure 8a. Transmit Frequency Compensation**

**Figure 8b. Receive Frequency Compensation**

**Figure 9. External Tone Injection**
The feature control receive and transmit channels of the SLD frame are shared by the two 29C48s. A 29C48 will accept or return feature control information only if it has been instructed to do so during the first byte of a feature control frame. This is accomplished by setting the logic level of the channel selection bit (LSB) in feature control byte #1 to match the logic level of the B/A pin of the appropriate 29C48. The selected 29C48 will keep exchanging feature control information until a new framing byte makes a new selection. The status of the channel selection bit is sent back during the seventh byte of the SLD frame in which a 00 was received in the F/WE bits of the 3rd byte of the same SLD frame.

The 29C48 does not process data received in the signaling channel. However, it generates chip select signals during the appropriate time slots in order to facilitate the SLIC interface. (See section on SLIC Chip Select.) The 29C48 enters into a high impedance state during the signaling transmit channel, the eighth byte of the SLD frame.

### SLIC Chip Select

In order to facilitate interfacing to an SLD compatible SLIC, especially when two SLICs share the same SLD line, the 29C48 includes a programmable chip select signal.

During the receive cycle of the SLD frame, the SCS pin of the 29C48 whose channel selection pin (B/A) has the same logic state as the channel selection bit (see previous section on Channel Assignment) is pulled low during the receive signaling byte.

During the transmit cycle of the SLD frame, the SCS signal can operate in two modes. In the first mode, called 'byte mode,' the SCS pin of the selected 29C48 is pulled low during the transmit signaling byte, as described above for the receive direction.

A second mode, called the 'half-byte mode,' is provided. In this mode, during the transmit cycle of the SLD frame, the SCS pin of the channel A combo is pulled low during the least significant four bits (last four bits) of the transmit signaling byte. During the same frame, the SCS pin of the channel B combo is pulled low during the most significant four bits (first four bits) of the transmit signaling byte. This allows signaling data from both A and B channel SLD compatible SLICs to be processed by a line card controller during the same frame.

To minimize power consumption, operation of the SCS signal during the receive half of the SLD frame can be disabled through the feature control memory. Operation of this signal in the transmit direction remains unaffected to allow continued monitoring of subscriber status by a line card controller. The SCS signal remains active in the power down mode.

The eight possible sequences for SCS are shown in Figure 10.

### Precision Voltage References

Voltage references are generated on-chip and are trimmed during the manufacturing process. Separate references are supplied for both the transmit and receive sections of the chip, each trimmed independently. These references determine the gain and dynamic range of the device and provide the user a significant margin for error in other board components.

### SLD Interface

The 29C48 is intended for use with the 29C53AA ISDN transceiver or a SLD compatible line card controller. They manage the transfer of all voice and feature control data to and from the Feature Control Combo. The interface between the two consists of just three leads, two of which are clock signals and the third a serial bus for communication.

The subscriber direction (SDIR) lead provides an 8 KHz signal which divides each frame into transmit and receive halves. During the first half when SDIR is high (RCV half-cycle), the 29C48 receives data and in the second (XMIT half-cycle) the 29C48 transmits data. Frame synchronization and all internal timing for the digital circuitry is derived from the rising edge of the SDIR signal.

The subscriber clock (SCL) input generated by the 29C53AA is a fixed 512 KHz clock signal allowing 64 bits (8 bytes) of data to be transferred on the SLD lead during each 125 µs frame. The SCL duty cycle can range from 25% to 75%.

The Subscriber Line Datalink (SLD) is a bi-directional serial bus that transfers four bytes of serial data to and from the 29C48 each frame. During the first half of each frame, RCV channel information is expected by the 29C48 as one byte consisting of voice and one byte of feature control information, while the other two bytes of the RCV half-frame are simply ignored. Similarly, during the second half-frame, one
byte of voice and, if so instructed, one byte of feature control information is sent by the 29C48. The 29C48 places its SLD lead in a high impedance state while the other device connected to the SLD line transmits its own information, and also while the one byte of signaling information is transmitted by an SLD compatible SLIC. The most significant bit (bit 7) of each byte is sent first on the SLD line. The data format of an SLD frame is shown in Figure 11.

Upon power supply application, feature control read or write of the 29C48 is disabled for 9 SLD frames. During this time, the 29C48 resets and enters the power down mode. The SCS output remains high until the 29C48 has been configured as an A or B channel device by the first write to feature control byte #1.

PROGRAMMABLE FEATURES

The 29C48 is configured by a set of five feature control bytes (FCB).

---

**Receive SCS Disabled (See Section on SLIC Chip Select)**

1) **A—Channel Selected (See Section on Channel Assignment)/Byte Mode**

<table>
<thead>
<tr>
<th>RECEIVE HALF FRAME</th>
<th>TRANSMIT HALF-FRAME</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>B</td>
</tr>
<tr>
<td>C</td>
<td>S</td>
</tr>
<tr>
<td>A-CHANNEL SCS</td>
<td>B-CHANNEL SCS</td>
</tr>
</tbody>
</table>

2) **B—Channel Selected/Byte Mode**

<table>
<thead>
<tr>
<th>RECEIVE HALF-FRAME</th>
<th>TRANSMIT HALF-FRAME</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>B</td>
</tr>
<tr>
<td>C</td>
<td>S</td>
</tr>
<tr>
<td>A-CHANNEL SCS</td>
<td>B-CHANNEL SCS</td>
</tr>
</tbody>
</table>

3) **A—Channel Selected/Half-Byte Mode**

<table>
<thead>
<tr>
<th>RECEIVE HALF-FRAME</th>
<th>TRANSMIT HALF-FRAME</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>B</td>
</tr>
<tr>
<td>C</td>
<td>S</td>
</tr>
<tr>
<td>A-CHANNEL SCS</td>
<td>B-CHANNEL SCS</td>
</tr>
</tbody>
</table>

4) **B—Channel Selected/Half-Byte Mode**

<table>
<thead>
<tr>
<th>RECEIVE HALF-FRAME</th>
<th>TRANSMIT HALF-FRAME</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>B</td>
</tr>
<tr>
<td>C</td>
<td>S</td>
</tr>
<tr>
<td>A-CHANNEL SCS</td>
<td>B-CHANNEL SCS</td>
</tr>
</tbody>
</table>

---

**Figure 10. SCS Timing Diagram**

5-13
These bytes of information are stored in internal registers which are serially multiplexed to and from the SLD interface in the third and seventh byte locations. The first two bits of each byte consist of a multiframe synchronization and write enable code. The framing bit (bit 7, MSB) establishes the beginning of a feature control frame when set to a logical zero, and increments the feature control counter when set to one. The second (bit 6) enables the writing to the 29C48 when it is the logical complement of the framing bit. In addition to the two header bits, feature control byte #1 also includes a channel selection bit (bit 0, LSB). This bit is used to designate one of the two 29C48s sharing an SLD link for feature control information exchange. (See previous section on channel Assignment.)

When writing new feature control information to the 29C48, the first byte should contain a framing (F) and write enable (WE) header of 01 (F = 0 and WE = 1), and an appropriate channel selection bit. This designates a new frame of information to trans-

---

**Receive SCS Enabled**

5) A—Channel Selected (See Section on Channel Assignment)/Byte Mode

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>S</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>S</th>
</tr>
</thead>
<tbody>
<tr>
<td>A CHANNEL SCS</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>B CHANNEL SCS</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

270153-14

6) B—Channel Selected/Byte Mode

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>S</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>S</th>
</tr>
</thead>
<tbody>
<tr>
<td>A CHANNEL SCS</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>B CHANNEL SCS</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

270153-30

7) A—Channel Selected/Half-Byte Mode

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>S</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>S</th>
</tr>
</thead>
<tbody>
<tr>
<td>A CHANNEL SCS</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>B CHANNEL SCS</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

270153-31

8) B—Channel Selected/Half-Byte Mode

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>S</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>S</th>
</tr>
</thead>
<tbody>
<tr>
<td>A CHANNEL SCS</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>B CHANNEL SCS</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

270153-32

---

Figure 10. SCS Timing Diagram (Continued)
Voice A, Voice B: A and B channel voice bytes respectively.
Control: Feature control information. This information is exchanged with the 29C48 whose channel selection pin matches the channel selection bit of the latest framing feature control byte.
Signaling: Signaling information which controls the subscriber line. The 29C48 enters into a high impedance state during the transmit signaling time slot, and generates a chip select signal (see section on SLIC chip select).

**Figure 11. 29C48 SLD Interface**

The subsequent bytes should each have \( F = 1 \) to advance the counter, and \( WE = 0 \) to enable the write operation.

The SLD master can also request to verify the feature control register contents by sending a 00 or 11 at the beginning of the byte to be read. To read the first byte, a 00 F/WE code and an appropriate channel selection bit should be sent while each subsequent byte should have a 11 header. An internal six-stage counter is set on the first byte verified then incremented once each 125 \( \mu \)s frame. It is reset only upon detection of a 01 or 00 F/WE. Once the counter is greater than five, neither read nor write modes may be selected by sending the 29C48 a 11 or 10 framing and write enable code. While in this state, the 29C48 will then echo in byte 7 the data it received in byte 3. Another feature control information exchange cycle can only be initiated by establishing a new feature control frame (sending \( F = 0 \)).

**FCB #1—Power Up/Down, Loop Back Mode, \( \mu \)A-/Law, Channel Select Register**

**POWER UP AND DOWN**

The 29C48 can be instructed to go into the power down or standby mode for reduced power consumption. In this mode, all analog inputs and outputs are placed in a high impedance state, inhibiting voice signals. A code of all ones will be output in the voice byte on the SLD. Signaling and feature control information will continue to be processed to allow the 29C48 to be read or reprogrammed.

The state of the feature control combo can be changed from standby to active by the first feature control byte only. All other register contents will be preserved during power down provided the power supplies remain connected.

**LOOP BACK MODE SELECT**

Three modes of remote testing are incorporated in the 29C48 and can be selected by appropriate coding in this register. The loopback features allow a number of tests to be performed to determine line quality and balancing. These include digital loop back, analog loop back, and subscriber loop back.

In the digital loopback mode, the combo retransmits the PCM word it receives in the voice A or B byte of the SLD back to the SLD master in the same frame. This feature allows path verification and testing of the circuit up to the combo.

When the analog loopback mode is selected, the analog output VFR is internally connected to the analog input VFX. This feature allows functional testing of the combo as well as gain adjustment.
In the third test mode, subscriber loopback, the digital output of the A/D converter is internally connected to the input of the D/A converter. The analog signal input to VFX is sent through the transmit filter, encoded, then decoded, filtered and output to VFR. This mode is used primarily for simplifying analog to analog testing from the subscriber side of the line card. Simultaneous selection of more than one loopback mode is prohibited.

CONVERSION LAWS

The 29C48 can be selected for either µ-law or A-law operation. A user can select either conversion law by assigning the corresponding bit. A logical 1 in bit 1 would select µ-law while a logical 0 would select A-law conversions. Both conversions follow CCITT recommendation G.711.

FEATURE CONTROL EXCHANGE CHANNEL SELECT

The LSB of feature control byte #1 is the channel selection bit. It is used to select one of the two 29C48s sharing an SLD line for feature control information exchange. A logical zero will select the channel A combo, and a logical one will select the channel B combo.

FCB #2—Receive Programmable Gain Register

The receive gain levels can be adjusted by applying external resistors as mentioned earlier, or by selective programming of this register. A range from 0 to −15.5 in 0.5 dB increments can be set for the receive channel.

FCB #3—Secondary Analog Channel, Chip Select, and Tone Injection Register

SECONDARY ANALOG INPUT

The 29C48 can be instructed to switch the input of its encoder to the secondary analog input by setting the SAIE bit to a logical one. Transmission of the voice signal will resume as soon as SAIE is set back to a logical zero.

PROGRAMMABLE SLIC CHIP SELECT

Although the 29C48 does not process signaling information, it generates chip select signals in order to help in interfacing to SLD compatible SLICS.

During the transmit half-frame, the chip select works in two possible modes determined by the CSM bit. In the byte mode, the SCS pin of the 29C48 selected by the channel selection bit in feature control byte #1 will be pulled low during the transmit signaling byte. In the half-byte mode, the SCS pin of the A-channel 29C48 will be pulled low during the four least significant bits of the transmit signaling byte, and the SCS pin of the B-channel 29C48 will be pulled low during the four most significant bits of the transmit signaling byte.

Generation of chip select signals during the receive half frame can be disabled by setting the CSD bit to a logical zero.
TONE INJECTION

When the TIE bit is set to a logical one, audio signal applied at the EBN3/T1 pin will be added to the output of the receive programmable gain module. This feature can be used for easy implementation of side tone injection and DTMF feedback, as well as injection at the line card of call waiting tones, ringing or metering pulses.

FCB #4—Transmit Programmable Gain Register

The gain setting of the transmit section of the chip operates in the same manner as the receive gain register. A 12 dB range from 0.0 dB to +12.0 dB in 0.5 dB increments is available.

FCB #5—Balance Network Select and Gain Register

BALANCE NETWORKS

Three external balance networks can be used with the 29C48. Feature control allows the selection of network EBN1, EBN2, and EBN3 individually or in combination in order to best suit a particular application.

EBN3 selection is not effective when TIE is set to a logical one.

GAIN SETTING

An additional 6 dB gain in the balance signal path can be realized by setting the BNG bit to a logical one. A logical zero provides unity gain.
ABSOLUTE MAXIMUM RATINGS

Temperature Under Bias ........... −10°C to +85°C
Storage Temperature ............ −65°C to +150°C
All Input and Output Voltages
with Respect to VBB ............ −0.3V to 13V
All Input and Output Voltages
with Respect to VCC ............ −13V to 0.3V
Power Dissipation ............... 1.35W

Notice: Stresses above those listed under “Absolute Maximum Ratings” may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions above those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

NOTICE: Specifications contained within the following tables are subject to change.

TA = 0°C to +70°C, VCC = +5V ±5%, VBB = −5V ±5%; SCL (50% duty), SDIR, SLD applied GNDD = 0V, GNDA = 0V. Typical values are for TA = 25°C and nominal power supply values

D.C. CHARACTERISTICS

Digital Interface

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Parameter</th>
<th>Min</th>
<th>Typ</th>
<th>Max</th>
<th>Units</th>
<th>Test Conditions</th>
</tr>
</thead>
<tbody>
<tr>
<td>IIL</td>
<td>Input Leakage Current</td>
<td>±10</td>
<td>μA</td>
<td>0</td>
<td>V</td>
<td>0 ≤ Vin ≤ VCC</td>
</tr>
<tr>
<td>VIL</td>
<td>Input Low Voltage</td>
<td>−0.3</td>
<td></td>
<td>0.8</td>
<td>V</td>
<td></td>
</tr>
<tr>
<td>VIH</td>
<td>Input High Voltage</td>
<td>2.0</td>
<td></td>
<td></td>
<td>V</td>
<td></td>
</tr>
<tr>
<td>VOIL</td>
<td>Output Low Voltage</td>
<td>0.4</td>
<td></td>
<td></td>
<td>V</td>
<td>IOH ≥ −1.6 mA, 1 TTL Load</td>
</tr>
<tr>
<td>VOH</td>
<td>Output High Voltage</td>
<td>2.4</td>
<td></td>
<td></td>
<td>V</td>
<td>IOH ≤ 50 μA, 1 TTL Load</td>
</tr>
</tbody>
</table>

Power Dissipation

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Parameter</th>
<th>Min</th>
<th>Typ</th>
<th>Max</th>
<th>Units</th>
<th>Test Conditions</th>
</tr>
</thead>
<tbody>
<tr>
<td>ICC1</td>
<td>VCC Operating Current</td>
<td>7.5</td>
<td>10</td>
<td>mA</td>
<td></td>
<td>Idle Channel</td>
</tr>
<tr>
<td>ICCO</td>
<td>VCC Standby Current</td>
<td>0.6</td>
<td>1.0</td>
<td>mA</td>
<td></td>
<td>Idle Channel</td>
</tr>
<tr>
<td>ISBB</td>
<td>VBB Standby Current</td>
<td>0.3</td>
<td>0.6</td>
<td>mA</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

A.C. CHARACTERISTICS—TRANSMISSION PARAMETERS
(TG1 = TG2, Transmit Programmable Gain = 6 dB. Receive Programmable Gain = 0 dB)

Gain and Dynamic Range

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Parameter</th>
<th>Min</th>
<th>Typ</th>
<th>Max</th>
<th>Units</th>
<th>Test Conditions</th>
</tr>
</thead>
<tbody>
<tr>
<td>EmW</td>
<td>Encoder Milliwatt Response</td>
<td>±0.25</td>
<td>dB</td>
<td></td>
<td></td>
<td>Signal Input of 0 dBm0</td>
</tr>
<tr>
<td>DmW</td>
<td>Digital Milliwatt Response</td>
<td>±0.25</td>
<td>dB</td>
<td></td>
<td></td>
<td>f = 1.02 KHz</td>
</tr>
<tr>
<td>DmWμV</td>
<td>Digital Milliwatt Response</td>
<td>6.11</td>
<td>dBm</td>
<td>Vrms</td>
<td></td>
<td>RL = 600Ω f = 1.02 KHz</td>
</tr>
<tr>
<td>DmWAV</td>
<td>Digital Milliwatt Response</td>
<td>6.17</td>
<td>dBm</td>
<td>Vrms</td>
<td></td>
<td>RL = 600Ω f = 1.02 KHz</td>
</tr>
<tr>
<td>OTLPμX</td>
<td>Zero Transmission Level Point</td>
<td>0.09</td>
<td>dBm</td>
<td>Vrms</td>
<td>μ-law</td>
<td>Referenced to 600Ω</td>
</tr>
<tr>
<td>OTLPAX</td>
<td>Zero Transmission Level Point</td>
<td>0.15</td>
<td>dBm</td>
<td>Vrms</td>
<td>A-law</td>
<td>Referenced to 600Ω</td>
</tr>
<tr>
<td>ΔGp</td>
<td>Programmable Gain Accuracy</td>
<td>±0.20</td>
<td>dB</td>
<td></td>
<td></td>
<td>f = 1.02 KHz for All Steps</td>
</tr>
</tbody>
</table>
GAIN TRACKING

Reference level = 0 dBm0 at 1.02 KHz, TG1 = TG2, Transmit Programmable Gain = 6 dB,
Receive Programmable Gain = 0 dB, AT&T PUB 43801 and CCITT G.714—Method 2

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Parameter</th>
<th>Min</th>
<th>Typ</th>
<th>Max</th>
<th>Units</th>
<th>Test Conditions</th>
</tr>
</thead>
<tbody>
<tr>
<td>G\text{T}</td>
<td>Transmit Gain Tracking Error</td>
<td>±0.25</td>
<td>dB</td>
<td>+3 to -40 dBm0</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Sinusoidal Input; µ or A-law</td>
<td>±0.50</td>
<td>dB</td>
<td>-40 to -50 dBm0</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>±1.2</td>
<td>dB</td>
<td>-50 to -55 dBm0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>G\text{R}</td>
<td>Receive Gain Tracking Error</td>
<td>±0.25</td>
<td>dB</td>
<td>+3 to -40 dBm0</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Sinusoidal Input; µ or A-law</td>
<td>±0.50</td>
<td>dB</td>
<td>-40 to -50 dBm0</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>±1.2</td>
<td>dB</td>
<td>-50 to -55 dBm0</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

ANALOG INTERFACE, RECEIVE CHANNEL

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Parameter</th>
<th>Min</th>
<th>Typ</th>
<th>Max</th>
<th>Units</th>
<th>Test Conditions</th>
</tr>
</thead>
<tbody>
<tr>
<td>R\text{OR}</td>
<td>Output Resistance, VFR</td>
<td>1</td>
<td>Ω</td>
<td></td>
<td></td>
<td>Relative to GNDA</td>
</tr>
<tr>
<td>V\text{OSR1}</td>
<td>Output Offset, VFR</td>
<td>50</td>
<td>mV</td>
<td></td>
<td></td>
<td>Relative to GNDA</td>
</tr>
<tr>
<td>C\text{LR}</td>
<td>Load Capacitance, VFR</td>
<td>100</td>
<td>pF</td>
<td></td>
<td></td>
<td>Relative to GNDA</td>
</tr>
<tr>
<td>V\text{OR1}</td>
<td>Max Output Voltage Swing Across RL, VFR</td>
<td>±3.2</td>
<td>Vp</td>
<td>RL ≥ 600 Ω(1)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

ANALOG INTERFACE, TRANSMIT PRIMARY AND SECONDARY CHANNELS

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Parameter</th>
<th>Min</th>
<th>Typ</th>
<th>Max</th>
<th>Units</th>
<th>Test Conditions</th>
</tr>
</thead>
<tbody>
<tr>
<td>I\text{BX}</td>
<td>Input Leakage Current, EBN, TG1, TI</td>
<td>100</td>
<td>nA</td>
<td></td>
<td></td>
<td>Operating Range(2)</td>
</tr>
<tr>
<td>R\text{IX1}</td>
<td>Input Resistance, VFX</td>
<td>100</td>
<td>KΩ</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>R\text{IX2}</td>
<td>Input Resistance, EBN, TG1, TI</td>
<td>10</td>
<td>MΩ</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TG\text{max}</td>
<td>Max Transmit Gain Adjust</td>
<td>20</td>
<td>dB</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>V\text{OTG}</td>
<td>Max Output Voltage Swing TG2</td>
<td>±1.6</td>
<td>V</td>
<td>RL ≥ 10K Ω(3)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>C\text{LX}</td>
<td>Load Capacitance, TG2</td>
<td>20</td>
<td>pF</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>R\text{LX}</td>
<td>Load Resistance, TG2</td>
<td>10</td>
<td>KΩ</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>R\text{GND}</td>
<td>On Resistance to GNDA, EBN1, EBN2, EBN3</td>
<td>150</td>
<td>600</td>
<td>Ω</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

NOTES:
1. The 29C48 power amplifier is designed to drive signals in excess of the maximum encoding level, which is 3.14 dBm0 for A-Law and 3.17 dBm0 for µ-Law.
2. -3.2V < VFX, EBN, TI < +3.2V; -1.6V < TG1 < +1.6V.
3. Transmit programmable gain must be set to 0 dB to encode this level without clipping in later stages.
### DISTORTION (Primary Channel)

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Parameter</th>
<th>Min</th>
<th>Typ</th>
<th>Max</th>
<th>Units</th>
<th>Test Conditions</th>
</tr>
</thead>
<tbody>
<tr>
<td>SDx, SDp</td>
<td>Signal to Distortion, μ or A-law Sinusoidal Input; CCITT G.714—Method 2 Half Channel</td>
<td>35</td>
<td>29</td>
<td>25</td>
<td>dB</td>
<td>0 to -30 dBm0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>-30 to -40 dBm0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>-40 to -45 dBm0</td>
</tr>
<tr>
<td>DPx, DPp</td>
<td>Single Frequency Distortion Products in Band (2nd or 3rd Harmonic Half Channel)</td>
<td>-60</td>
<td>-47</td>
<td>dBm0</td>
<td></td>
<td>Input = 1.02 KHz 0 dBm0 AT&amp;T Advisory #64 (3.8)</td>
</tr>
<tr>
<td>IMD1</td>
<td>Intermodulation Distortion, End to End Measurement</td>
<td>-35</td>
<td>dB</td>
<td></td>
<td></td>
<td>CCITT G.712(7.1)</td>
</tr>
<tr>
<td>IMD2</td>
<td>Intermodulation Distortion, End to End Measurement</td>
<td>-49</td>
<td>dBm0</td>
<td></td>
<td></td>
<td>CCITT G.712(7.2)</td>
</tr>
<tr>
<td>SOS</td>
<td>Spurious Out of Band Signals, End to End Measurement</td>
<td>-25</td>
<td>dBm0</td>
<td></td>
<td></td>
<td>CCITT G.712(6.1)</td>
</tr>
<tr>
<td>SIS</td>
<td>Spurious in Band Signals, End to End Measurement</td>
<td>-40</td>
<td>dBm0</td>
<td></td>
<td></td>
<td>CCITT G.712(9)</td>
</tr>
<tr>
<td>DAX</td>
<td>Transmit Absolute Group Delay</td>
<td>220</td>
<td>μs</td>
<td></td>
<td></td>
<td>0 dBm0, 1.4 KHz Includes Delay Through A/D</td>
</tr>
<tr>
<td>DDX</td>
<td>Transmit Differential Delay; Relative to DAX</td>
<td>170</td>
<td>μs</td>
<td></td>
<td></td>
<td>f = 500–600 Hz</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>95</td>
<td>μs</td>
<td></td>
<td>f = 600–1000 Hz</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>45</td>
<td>μs</td>
<td></td>
<td>f = 1000–2600 Hz</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>75</td>
<td>μs</td>
<td></td>
<td>f = 2600–2800 Hz</td>
</tr>
<tr>
<td>DAR</td>
<td>Receive Absolute Group Delay</td>
<td>140</td>
<td>μs</td>
<td></td>
<td></td>
<td>0 dBm0, 0.3 KHz Includes Delay Through D/A</td>
</tr>
<tr>
<td>DDR</td>
<td>Receive Differential Envelope Delay; Relative to DAR</td>
<td>35</td>
<td>μs</td>
<td></td>
<td></td>
<td>f = 500–600 Hz</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>35</td>
<td>μs</td>
<td></td>
<td>f = 600–1000 Hz</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>110</td>
<td>μs</td>
<td></td>
<td>f = 1000–2600 Hz</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>135</td>
<td>μs</td>
<td></td>
<td>f = 2600–2800 Hz</td>
</tr>
</tbody>
</table>

### NOISE (Primary Channel)

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Parameter</th>
<th>Min</th>
<th>Typ</th>
<th>Max</th>
<th>Units</th>
<th>Test Conditions</th>
</tr>
</thead>
<tbody>
<tr>
<td>NxCl</td>
<td>Transmit Noise, C-Message Weighted</td>
<td>15</td>
<td>dBrnC0</td>
<td></td>
<td></td>
<td>TG1 = TG2; Transmit Programmable Gain = 6 dB</td>
</tr>
<tr>
<td>Nxp1</td>
<td>Transmit Noise, Psophometrically Weighted</td>
<td>-75</td>
<td>dBm0p</td>
<td></td>
<td></td>
<td>TG1 = TG2; Transmit Programmable Gain = 6 dB</td>
</tr>
<tr>
<td>Nrc1</td>
<td>Receive Noise, C-Message Weighted</td>
<td>11</td>
<td>dBrnC0</td>
<td></td>
<td></td>
<td>Unity Gain; Idle Code; Receive Programmable Gain = 0 dB</td>
</tr>
<tr>
<td>Nrp1</td>
<td>Receive Noise, Psophometrically Weighted</td>
<td>-79</td>
<td>dBm0p</td>
<td></td>
<td></td>
<td>Unity Gain; Idle Code; Receive Programmable Gain = 0 dB</td>
</tr>
<tr>
<td>PSRR1</td>
<td>VCC or VBB Power Supply Rejection Transmit Channel</td>
<td>-30</td>
<td>dB</td>
<td></td>
<td></td>
<td>Idle Channel; 200 mV P-P Signal on Supply, DC to 50 KHz (1)</td>
</tr>
<tr>
<td>PSRR2</td>
<td>VCC or VBB Power Supply Rejection Receive Channel</td>
<td>-30</td>
<td>dB</td>
<td></td>
<td></td>
<td>Idle Channel; 200 mV P-P Signal on Supply, DC to 50 KHz (1)</td>
</tr>
</tbody>
</table>

**NOTE:**
CROSSTALK

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Parameter</th>
<th>Min</th>
<th>Typ</th>
<th>Max</th>
<th>Units</th>
<th>Test Conditions</th>
</tr>
</thead>
<tbody>
<tr>
<td>CTTR</td>
<td>Crosstalk, Transmit Voice to Receive Voice</td>
<td>-90</td>
<td>-75</td>
<td>dB</td>
<td></td>
<td>Input = 0 dBm0, Unity Gain 1.02 kHz; Idle Code on SLD Voice Byte</td>
</tr>
<tr>
<td>CRTT</td>
<td>Crosstalk, Receive Voice to Transmit Voice</td>
<td>-80</td>
<td>-72</td>
<td>dB</td>
<td></td>
<td>0 dBm0, 1.02 KHz Signal at SLD Receive Voice Byte; VFX = GNDA</td>
</tr>
</tbody>
</table>

TRANSMIT VOICE FREQUENCY CHARACTERISTICS

TG1 = TG2, Transmit Programmable Gain = 6 dB

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Parameter</th>
<th>Min</th>
<th>Typ</th>
<th>Max</th>
<th>Units</th>
<th>Test Conditions</th>
</tr>
</thead>
<tbody>
<tr>
<td>G_RX</td>
<td>Gain Relative to Gain at 1.02 KHz</td>
<td></td>
<td></td>
<td></td>
<td>dB</td>
<td>0 dBm0 Signal Input at VFX</td>
</tr>
<tr>
<td>16.67 Hz</td>
<td>-30</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>50 Hz</td>
<td>-25</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>60 Hz</td>
<td>-23</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>200 Hz</td>
<td>-1.8</td>
<td></td>
<td></td>
<td></td>
<td>dB</td>
<td></td>
</tr>
<tr>
<td>300 Hz</td>
<td>-0.125</td>
<td></td>
<td></td>
<td></td>
<td>dB</td>
<td></td>
</tr>
<tr>
<td>3300 Hz</td>
<td>-0.70</td>
<td></td>
<td></td>
<td></td>
<td>dB</td>
<td></td>
</tr>
<tr>
<td>4000 Hz</td>
<td>-14</td>
<td></td>
<td></td>
<td></td>
<td>dB</td>
<td></td>
</tr>
<tr>
<td>4600 Hz and Above</td>
<td>-32</td>
<td></td>
<td></td>
<td></td>
<td>dB</td>
<td></td>
</tr>
</tbody>
</table>

Figure 12. Transmit Voice Frequency Characteristics
RECEIVE VOICE FREQUENCY CHARACTERISTICS
Receive Programmable Gain = 0 dB, Feature Control Bit TIE = 0

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Parameter</th>
<th>Min</th>
<th>Typ</th>
<th>Max</th>
<th>Units</th>
<th>Test Conditions</th>
</tr>
</thead>
<tbody>
<tr>
<td>$G_{RR}$</td>
<td>Gain Relative to Gain at 1.02 KHz</td>
<td></td>
<td>+0.125 dB</td>
<td></td>
<td></td>
<td>0 dBm0 Input on SLD</td>
</tr>
<tr>
<td></td>
<td>Below 200 Hz</td>
<td></td>
<td>+0.125 dB</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>200 Hz</td>
<td>-0.5</td>
<td>+0.125 dB</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>300 to 3000 Hz</td>
<td>-0.125</td>
<td>+0.125 dB</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>3300 Hz</td>
<td>-0.35</td>
<td>+0.03 dB</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>3400 Hz</td>
<td>-0.70</td>
<td>-0.1 dB</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>4000 Hz</td>
<td></td>
<td>-14 dB</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>4600 Hz &amp; Above</td>
<td></td>
<td>-30 dB</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

NOTES:
1. Typical transfer function of the receive filter as a separate component.
2. Typical transfer function of the receive filter driven by the sample and hold output of the Intel 2910A and 2911A codecs. The combined filter/codec response meets the stated specifications.

Figure 13. Receive Voice Frequency Characteristics
### A.C. CHARACTERISTICS—TIMING PARAMETERS

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Parameter</th>
<th>Min</th>
<th>Typ</th>
<th>Max</th>
<th>Units</th>
<th>Test Conditions</th>
</tr>
</thead>
<tbody>
<tr>
<td>T(_{SCL})</td>
<td>SCL Pulse Width</td>
<td>486</td>
<td></td>
<td>1465</td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>T(_{DC})</td>
<td>SCL Duty Cycle</td>
<td>25</td>
<td></td>
<td>75</td>
<td>%</td>
<td></td>
</tr>
<tr>
<td>T(_{RC})</td>
<td>Rise, Fall Times, SCL</td>
<td>50</td>
<td></td>
<td></td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>T(_{RD})</td>
<td>Rise, Fall Times, SLD</td>
<td>50</td>
<td></td>
<td></td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>T(_{RS})</td>
<td>Rise, Fall Times, SCS</td>
<td>50</td>
<td></td>
<td></td>
<td>ns</td>
<td>50 pF Load</td>
</tr>
<tr>
<td>T(_{TFS})</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T(_{TDIRR})</td>
<td>SCL to SDIR Delay</td>
<td>-500</td>
<td></td>
<td>500</td>
<td>ns</td>
<td>Receive Cycle</td>
</tr>
<tr>
<td>T(_{TDIRF})</td>
<td>SCL to SDIR Delay</td>
<td>-500</td>
<td></td>
<td>500</td>
<td>ns</td>
<td>Transmit Cycle</td>
</tr>
<tr>
<td>T(_{DD})</td>
<td>SCL to SLD Delay</td>
<td>0</td>
<td></td>
<td>200</td>
<td>ns</td>
<td>29C48 Transmitting</td>
</tr>
<tr>
<td>T(_{SD})</td>
<td>Set-up Time, SLD to SCL</td>
<td>100</td>
<td></td>
<td></td>
<td>ns</td>
<td>29C48 Receiving</td>
</tr>
<tr>
<td>T(_{HD})</td>
<td>Hold Time, SCL to SLD</td>
<td>100</td>
<td></td>
<td></td>
<td>ns</td>
<td>29C48 Receiving</td>
</tr>
<tr>
<td>T(_{HZ1})</td>
<td>SDIR to SLD Active</td>
<td>0</td>
<td></td>
<td>100</td>
<td>ns</td>
<td>Byte 1, Bit 1 29C48 Transmitting, Channel A</td>
</tr>
<tr>
<td>T(_{HZ2})</td>
<td>SCL to SLD High Impedance</td>
<td>0</td>
<td></td>
<td>100</td>
<td>ns</td>
<td>Channel A, B, or Feature Control as Appropriate (Channel A/B Operation)</td>
</tr>
<tr>
<td>T(_{HZ3})</td>
<td>SCL to SLD Active</td>
<td>0</td>
<td></td>
<td>100</td>
<td>ns</td>
<td>Channel A, B, or Feature Control as Appropriate (Channel A/B Operation)</td>
</tr>
<tr>
<td>T(_{SCSF})</td>
<td>SCL to SCS Low</td>
<td>TFS</td>
<td></td>
<td>250</td>
<td>ns</td>
<td>50 pF Load</td>
</tr>
<tr>
<td>T(_{HZSCSF})</td>
<td>SLD High Impedance to SCS Low</td>
<td>0</td>
<td></td>
<td></td>
<td>ns</td>
<td>Transmit Feature Control</td>
</tr>
<tr>
<td>T(_{SCSR})</td>
<td>SCL to SCS High</td>
<td>TRS</td>
<td></td>
<td>200</td>
<td>ns</td>
<td>50 pF Load</td>
</tr>
</tbody>
</table>

*In cases where T\(_{TDIRF}\) is positive, T\(_{DD}\) is to be measured from the SDIR edge.*

### RECEIVE CYCLE

![Receive Cycle Diagram](image-url)
TRANSMIT CYCLE

A.C. TESTING INPUT, OUTPUT WAVEFORM

A.C. Testing inputs are driven at 2.4 for a logic "1" and 0.45 for a logic "0". Timing measurements are made at 2.0V for a logic "1" and 0.8V for a logic "0".
The Intel Advanced Telecommunication Component (iATC) 29C53AA Digital Loop Controller (DLC) is a 4-wire transceiver/controller that is CCITT I.430 compatible and can function at either loop end. This part has integrated those features which are pertinent to the transceiver function and offers efficient interfacing to other system components such as CODEC/Filter and microcontrollers through the SLD and microprocessor interface ports. It is primarily intended for use in Integrated Services Digital Networks (ISDN) as a basic rate digital data transceiver which transfers data at 144 Kbps as three separate channels—two 64 Kbps digitized-voice/data channels (B channels), and a 16 Kbps signaling/data channel (D channel). The B- and D-channel routing along with D-channel processing (packet framing) is programmable through either the microprocessor or SLD interface ports. The 29C53AA's loop interface uses a 100% pulse-width pseudo-ternary line code which meets CCITT's "S" interface recommendations. It is capable of interfacing with up to eight 29C53AAs in a passive or extended bus configuration as well as point-to-point.
<table>
<thead>
<tr>
<th>Symbol</th>
<th>Pin No.</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>VCC</td>
<td>8</td>
<td>POSITIVE SUPPLY: Input voltage is +5V ±5%</td>
</tr>
<tr>
<td>Vss</td>
<td>22</td>
<td>GROUND: 0V</td>
</tr>
<tr>
<td>CLK</td>
<td>23</td>
<td>MASTER CLOCK: The 3.84 MHz system clock input is the reference for the loop and the SLD interface.</td>
</tr>
<tr>
<td>Res</td>
<td>7</td>
<td>RESET: (Active high input). A high level on this pin initializes control registers and places interface outputs in a high impedance state. Operation begins after the high level is removed.</td>
</tr>
<tr>
<td>LX+, LX−</td>
<td>13, 14</td>
<td>POLARIZED TRANSMIT LOOP INTERFACE PINS: These pins will directly drive the twisted pair line through a pulse transformer. The transmitted line code is 100% pulse width pseudo-ternary.</td>
</tr>
<tr>
<td>LR−, LR+</td>
<td>15, 16</td>
<td>RECEIVE LOOP INTERFACE PINS: The receiver is not sensitive to polarity.</td>
</tr>
<tr>
<td>SLD</td>
<td>2</td>
<td>SUBSCRIBER LINE DATALINK: This pin transfers serial data between the 29C53AA and other SLD based components (e.g., 29C48).</td>
</tr>
<tr>
<td>SCL</td>
<td>4</td>
<td>SUBSCRIBER CLOCK: 512 KHz signal may be either generated or received by the 29C53AA. This signal clocks the data on the SLD pin.</td>
</tr>
<tr>
<td>SDIR</td>
<td>5</td>
<td>SUBSCRIBER DIRECTION: An 8 KHz signal may be either generated or received by the 29C53AA to indicate SLD data direction and framing. A high level indicates master to slave transfer; a low level indicates slave to master transfers.</td>
</tr>
<tr>
<td>CS</td>
<td>3</td>
<td>CHIP SELECT: (Active low input). A low level on this pin enables the 29C53AA bus interface for the next bus cycle. The value is latched by the falling edge of ALE.</td>
</tr>
<tr>
<td>RD</td>
<td>10</td>
<td>READ STROBE: (Active low input). When low, data is transferred from the selected register to the data pins AD (0−7). When no local microprocessor is connected, this pin should be tied to VSS.</td>
</tr>
<tr>
<td>WR</td>
<td>9</td>
<td>WRITE STROBE: (Active low input). When WR changes from low to high, data on pins AD (0−7) is latched into the 29C53AA. When no local microprocessor is connected, this pin should be tied to VSS.</td>
</tr>
<tr>
<td>AD (0−7)</td>
<td>19−21, 24−28</td>
<td>ADDRESS/DATA PINS: This is a standard MCS microprocessor bus used to transfer address and data between the local microprocessor and the internal registers of the 29C53AA. When a local microprocessor is not used, these pins should be tied to VSS.</td>
</tr>
<tr>
<td>ALE</td>
<td>6</td>
<td>ADDRESS LATCH ENABLE: Address is latched from AD(1−5) on falling edge of this signal. State of CS is also latched at this time.</td>
</tr>
<tr>
<td>INT</td>
<td>1</td>
<td>INTERRUPT REQUEST: This is an open drain active low output. (See text for the interrupt conditions.)</td>
</tr>
<tr>
<td>P1, P2</td>
<td>18, 17</td>
<td>PERIPHERAL INTERFACE INPUTS: These are standard CHMOS high impedance inputs that are sampled at a 4 KHz rate (once per “s” frame). The sampled data is stored in the LPS register (bits 5 and 6). If any peripheral input bits have changed value since the previous frame, an interrupt condition is indicated; only present status is available.</td>
</tr>
<tr>
<td>P3</td>
<td>12</td>
<td>PERIPHERAL INTERFACE INPUT/OUTPUT PIN: When configured as an input, this pin has the same characteristics as P1 and P2. The sampled data is stored in the LPS register (bit 7). When programmed as an output, this pin outputs the data stored in the PEC register (bit 1). The pin is configured by bit 2 of the PEC register. An alternate function of this pin and P4 is to indicate the status of the SLD interface. See the section on the SLD interface.</td>
</tr>
<tr>
<td>P4</td>
<td>11</td>
<td>PERIPHERAL INTERFACE OUTPUT PIN: This pin outputs data stored in the PEC register (bit 0) or SLD status.</td>
</tr>
</tbody>
</table>
1.0 INTRODUCTION

The 29C53AA Digital Loop Controller is an advanced, programmable digital transceiver providing the layer one interface at the S or T reference point in Integrated Services Digital Network (ISDN) basic access applications. It provides access to the two B channels and the D channel in accordance with CCITT recommendation I.430, and supports both point-to-point and multipoint topologies. It can be used in linecard (NT) applications, or with the 29C48 programmable CODEC/Filter and appropriate data communications devices in voice/data subscriber (TE) applications.

The 29C53AA may be incorporated at either end of a subscriber loop interface (at the line card or digital telephone/terminal). As shown in Figure 2, the 29C53AA has four separate interfaces: a serial SLD system interface; a parallel peripheral interface; a parallel microprocessor interface and a 4-wire CCITT compatible S-interface (subscriber loop interface).

THE BLOCK DIAGRAM

Figure 2 represents a block diagram of the 29C53AA. Its three major blocks, the line interface unit, the D-channel processor and the SLD interface unit are interconnected by two buses. The parallel bus (PBUS) is used to transfer processed D-channel data and general status and control information, while the serial bus (SBUS) is used to transfer B-channel data and unprocessed D-channel data between the line interface unit and the SLD interface unit.

The SLD interface unit consists of shift registers and serial to parallel converters. Data from both the SBUS and the SLD interface is stored here in appropriate parallel registers before it is loaded into shift registers and passed on. All of the timing circuitry for the SLD interface is located here. This block also contains a command processor which is responsible for executing commands received in the SLD C byte.

The D-channel processor has three major sections. An HDLC section performs some of the basic LAPD protocol functions such as zero insertion or deletion, flag recognition or insertion for frame delineation, abort flag recognition, idle state transmission, and end of packet frame check sequence for both data directions. The FIFO section consists of two 32-byte buffers, one for transmit and one for receive. The control and status section monitors the FIFO data levels and the HDLC section for progress. Interrupts or requests for service may be generated for conditions such as FIFO fullness level, loss of sync, frame check error, overflows and aborts.

The line interface unit contains the line drivers and receivers for the S interface. Connection is made to the transmission lines through a pulse transformer. Formatting, timing and synchronization are also provided here. The receiver includes filters, AGC circuitry, threshold detectors and a loop delay shift register. The loop delay shift register maintains the proper internal frame relationship regardless of loop length (it allows extra propagation delay time for long loops or line repeaters). The received D-channel bits are logically looped back to create the E-channel bits in an NT application through the E-channel circuitry.

The microprocessor interface circuitry allows the 29C53AA to function as a peripheral to a microcontroller or microprocessor. All internal registers are directly accessible.

The spare bits processing block provides access to the FA, N, and A bits. It also provides access to the S and M bits, and supports the S and Q channels.

The peripheral interface circuitry provides an auxiliary port for controlling auxiliary peripherals such as power controllers, etc. It can be programmed to provide SLD status as well.

SLD INTERFACE

The SLD interface provides communication with other devices incorporating SLD interfaces.

As shown in Figure 3, the SLD interface consists of three lines: the SLD bidirectional data line; the 512 KHz SCL clock line; and the 8 KHz SDIR data direction line. SLD data is updated on the rising edge of SCL and is latched on its falling edge. The 125 μs SLD frame period consists of 32 bits transferred in master to slave direction followed by 32 bits in the slave to master direction. The 32 bits compose four 8-bit bytes in the following order: B1 and B2 (voice or data bytes); C (control byte); and (signaling or status byte). Unprocessed D-channel data may be transported over the S-byte in bits 0 and 1, or over the B2 byte.

The 29C53AA can be operated as an SLD master or slave. As an SLD master, it generates the SCL and SDIR signals. When SDIR is high, the SLD pin outputs data. As a slave, it receives SCL and SDIR sig-
nals and SDIR enables the SLD output driver when it is low. The SLD bus is always active; no powered-down or inactive mode is defined.

In a network termination (NT) application (line card), whether a microprocessor is connected to the 29C53AA or not, the SLD control and signaling bytes may be used for 29C53AA configuration and D-channel transfers. The command bytes are interpreted and executed by the 29C53AA’s command processor circuit. The command processor generates internal PBUS cycles to carry out those commands. Internal prioritization resolves PBUS collisions between microprocessor-interface generated and command-processor generated cycles. In case of collisions, the microprocessor interface has higher priority to minimize access time but both cycles will be completed.

**“S” TRANSCEIVER**

The 4-wire “S” transceiver circuit in the 29C53AA conforms to CCITT recommendation 1.430. This transceiver provides the internal drivers for transformer coupling to standard telephone type twisted pair cables.

The “S” transceiver line code is 100% pulse width pseudo-terenary code, with binary ones represented by no line signal, and binary zeros by a positive or negative pulse. Pulses alternate polarity except when a code violation is created for establishing the frame reference timing. The nominal pulse amplitude is 750 mV when a suitable pulse transformer is used.

The 29C53AA transmitter is typically connected to the line through a pulse transformer and series resistance (see Figure 12). The series resistance, when used with protection diodes, provides additional protection against surges. It also increases the output impedance.

The nominal bit rate is 192 Kbps. Figure 4 shows the frame structure. The 250 μs frame transfers two octets of B1, B2 and four bits of D data. The E bits in the master to slave direction echo received D-channel data. The “S” interface slave compares the receive E-channel data to its transmitted D-channel data for D-channel contention as defined in CCITT recommendation 1.430. If these bits do not agree, then the slave will abort its transmission effort. The S, M, FA, A, and N bits are all accessible and programmable. The 29C53AA supports the layer 1 maintenance multiframing for the Q and S channels.

The activation protocol described in 1.430 is supported by the 29C53AA. An inactive receiver can achieve bit synchronization to an incoming signal with approximately 15 mark-mark transitions. Info 2 or 3 frame alignment is not officially recognized until reception of 16 frames, to allow settling of the 29C53AA’s adaptive receive data thresholds. The full activation sequence will complete in approximately 10 ms.

The 29C53AA is not sensitive to the polarity of the wire pair connected to LR + and LR -. Pulses are always interpreted as zeros and framing relies on violations; not on absolute polarity. System configurations may dictate that care be taken in connecting the LX outputs. In a multi-drop bus configuration all TE transmitters must be connected with the same polarity so that positive pulse to negative pulse contention does not take place in the framing and D-channel bits.

![Figure 3. SLD Interface](image-url)
The 29C53AA, functioning as an “S” interface master in a multi-drop application, can interface with up to eight slave systems. In this multiplexing operation a slave initiates a data transfer to the master, by requesting access and transferring the data in accordance with the D-channel line access protocol (L.440). Figure 5 shows typical applications of the 29C53AA.

The frame alignment timing diagram Figure 6(b) shows the relationship of the “S” interface data to the SLD data. Figure 6(a) shows the block diagram used for the timing diagram. The top timing diagram of Figure 6(b) shows the transmitted “S” data stream from the network terminator (master). The dotted lines depict up to 20 μs propagation delay to the “S” receiver at the terminal equipment (slave) end. The terminal equipment’s transmitted “S” interface frame is designed to have a fixed 2-bit frame alignment delay from that of its received frame. The adjustment for loop propagation delay is accounted for in the network terminator’s receive circuitry (loop delay section of block diagram). The loop delay circuitry will compensate for up to 10 bit periods of round trip propagation delay which allows line repeaters to be placed in a loop that is several thousand meters long.

**MICROPROCESSOR INTERFACE**

This interface is designed to operate with standard Intel microprocessors such as the MCS®-48, MCS-51, MCS-85 and 8086 families. All of the 29C53AA’s internal registers are accessible and are available by a single microprocessor cycle access. The 29C53AA latches address information from
AD1–AD5, and does not use AD0 for addressing. This provides compatibility with 16-bit microprocessors.

The maskable interrupt pin is activated by the following interrupt status features: D-channel errors; loss of sync on “S” loop; change in spare bits or peripheral interface data; FIFO data transfer requests.

Alternatively, the 29C53AA can operate in the standalone mode in line card and NT applications. This mode is determined after a reset, provided all the microprocessor interface pins have been tied to VSS, except for the interrupt pin. The 29C53AA is then controlled using the C byte of the SLD interface.

PERIPHERAL INTERFACE

The peripheral interface uses four pins to provide control to, and to accept status from, external devices. Two pins are inputs, one is an output and one is configurable either as an input or an output. The configurable pin defaults to the input mode on power up.

The peripheral interface can also be used to indicate SLD status. Figure 7 shows the timing diagram of P3 and P4. B1, B2 and D-channel data on the SLD pin can be selected or gated by using these signals. As noted on the P3 timing, the D-channel is imbedded in the last two bits (0, 1) of the signaling byte. (This must be programmed in the DPC register).
NOTE: Depicts the beginning of the "S" interface frame (leading edge of F bit) and the beginning of the SLD interface frame.
INTERNAL CONTROL AND STATUS REGISTERS

All of the 29C53AA's internal control and status registers may be accessed through the microprocessor interface or through the SLD interface when in SLD slave mode. When a microprocessor accesses a register, the address and CS inputs are latched on the trailing edge of ALE. The address is latched from pins AD1–AD5 of the microprocessor port.

In an SLD access, the 29C53AA receives a control byte containing an operation code and an argument. The three most significant bits contain the operation code and the remaining five bits contain the argument. The operation code defines eight transfer types.

The 3-bit operation code in the control byte from the SLD master should normally be 111, indicating the idle state. The transferring of data to and from the 29C53AA is accomplished by indicating the type and the number of bytes to transfer in a non-idle control byte. When a polled response is requested, the 29C53AA responds to the poll operation code 000. This can be used for the transfer of one or several bytes of information.

The D-channel block transfers from the 29C53AA to the SLD master preface the data bytes with a byte header specifying the number of following bytes (less than or equal to the maximum specified) and the status of the packet they belong to. All transferred data bytes belong to the same packet; the transfers occur until the selected number of bytes are transferred or an EOP (end of packet) is detected. The EOP may occur even when there are additional bytes in the FIFO. The header byte contains the byte count in the lowest five bits and the packet status in the upper three bits.

Data transfers over the SLD line cannot be made in both directions simultaneously. Multiple commands and data bytes may follow each other directly from the line card controller to the 29C53AA if the previous command has been fully executed.

It is possible to fully configure the 29C53AA over the SLD interface. Provisions are also made to perform this transfer at a 2 byte-per frame rate using both the C and S bytes of the SLD. The first control byte of a configuration transfer to the 29C53AA specifies the type of operation to be performed and the number of data bytes to follow. The system interface command unit loads the internal registers with the information as it is received. When the specified number of data bytes have been transferred, the 29C53AA assumes the next input is a control byte.

The order of the bytes in a configuration or status block transfer is determined by the addresses of the internal registers. A multiple-byte transfer, beginning with register 00H, transfers the data to or from that register and increments the address counter.

The register table below identifies the address of each 29C53AA register. The status registers are read-only registers while all control registers are read/write registers. Because all the register addresses do not fit into the 5-bit address space, a register test mode has been included which permits reading the contents of control registers at addresses which normally are status registers. Where no register is assigned a location in the register test mode, the normal status register located at this address is read.
<table>
<thead>
<tr>
<th>Parallel Port Address</th>
<th>Internal Address(1)</th>
<th>Access</th>
<th>Symbol</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>00000</td>
<td>RD</td>
<td>EXS</td>
<td>Interrupt Status</td>
</tr>
<tr>
<td>00</td>
<td>00000</td>
<td>WR (RT)</td>
<td>EXM</td>
<td>Interrupt Mask</td>
</tr>
<tr>
<td>02</td>
<td>00001</td>
<td>RD</td>
<td>DPS</td>
<td>D-Channel Processor Status</td>
</tr>
<tr>
<td>02</td>
<td>00001</td>
<td>WR (RT)</td>
<td>DPC</td>
<td>D-Channel Processor Control</td>
</tr>
<tr>
<td>04</td>
<td>00010</td>
<td>RD</td>
<td>LPS</td>
<td>Loop and Peripheral Interface Status</td>
</tr>
<tr>
<td>04</td>
<td>00010</td>
<td>WR (RT)</td>
<td>LCR</td>
<td>Loop Interface Control</td>
</tr>
<tr>
<td>06</td>
<td>00011</td>
<td>RDWR</td>
<td>PEC</td>
<td>Peripheral Interface and E-Channel Control</td>
</tr>
<tr>
<td>08</td>
<td>00100</td>
<td>RD</td>
<td>RFN</td>
<td>Receive FIFO Status - # of Bytes Used</td>
</tr>
<tr>
<td>08</td>
<td>00100</td>
<td>WR (RT)</td>
<td>SCR</td>
<td>SLD Interface Control</td>
</tr>
<tr>
<td>0A</td>
<td>00101</td>
<td>RD</td>
<td>XFN</td>
<td>Transmit FIFO Status - # of Free Bytes</td>
</tr>
<tr>
<td>0C</td>
<td>00110</td>
<td>RD</td>
<td>SBR</td>
<td>Spare Bits Receive Status</td>
</tr>
<tr>
<td>0C</td>
<td>00110</td>
<td>WR (RT)</td>
<td>SBX</td>
<td>Spare Bits Transmit</td>
</tr>
<tr>
<td>0E</td>
<td>00111</td>
<td>RDWR</td>
<td>LLB</td>
<td>Loop Interface Loopback Control</td>
</tr>
<tr>
<td>10</td>
<td>01000</td>
<td>RD</td>
<td>RFO</td>
<td>Receive FIFO Output</td>
</tr>
<tr>
<td>12</td>
<td>01001</td>
<td>WR</td>
<td>XFI</td>
<td>Transmit FIFO Input</td>
</tr>
<tr>
<td>14</td>
<td>01010</td>
<td>RDWR</td>
<td>GCR</td>
<td>General Command Register</td>
</tr>
<tr>
<td>16</td>
<td>01011</td>
<td>RDWR</td>
<td>DPR</td>
<td>D-Channel Priority Counter</td>
</tr>
<tr>
<td>18</td>
<td>01100</td>
<td>RD</td>
<td>RFXF</td>
<td>Receive FIFO Interrupt Level</td>
</tr>
<tr>
<td>1A</td>
<td>01101</td>
<td>RDWR</td>
<td>XFXF</td>
<td>Transmit FIFO Interrupt Level</td>
</tr>
<tr>
<td>1C</td>
<td>01110</td>
<td>RD</td>
<td>PLENH</td>
<td>Packet Length High Byte</td>
</tr>
<tr>
<td>1C</td>
<td>01110</td>
<td>WR (RT)</td>
<td>DUTH</td>
<td>D-Channel Byte Counter Underflow and Overflow Threshold</td>
</tr>
<tr>
<td>1E</td>
<td>01111</td>
<td>RD</td>
<td>PLENL</td>
<td>Packet Length Low Byte</td>
</tr>
<tr>
<td>1E</td>
<td>01111</td>
<td>WR (RT)</td>
<td>DOTH</td>
<td>D-Channel Byte Counter Overflow Threshold</td>
</tr>
<tr>
<td>20</td>
<td>10000</td>
<td>RDWR</td>
<td>SBC</td>
<td>Spare Bit Control</td>
</tr>
<tr>
<td>22</td>
<td>10001</td>
<td>RDWR</td>
<td>PSR</td>
<td>Position Selection</td>
</tr>
<tr>
<td>24</td>
<td>10010</td>
<td>RD</td>
<td>RSR</td>
<td>Receive Service Request</td>
</tr>
<tr>
<td>26</td>
<td>10011</td>
<td>RD</td>
<td>XSR</td>
<td>Transmit Service Request</td>
</tr>
<tr>
<td>30</td>
<td>11000</td>
<td>RDWR</td>
<td>B1LS</td>
<td>B1 Data in Loop to SLD Direction</td>
</tr>
<tr>
<td>32</td>
<td>11001</td>
<td>RDWR</td>
<td>B2LS</td>
<td>B2 Data in Loop to SLD Direction</td>
</tr>
<tr>
<td>34</td>
<td>11010</td>
<td>RDWR</td>
<td>CR</td>
<td>Control Byte from SLD</td>
</tr>
<tr>
<td>36</td>
<td>11011</td>
<td>RDWR</td>
<td>SR</td>
<td>Signaling Byte from SLD</td>
</tr>
<tr>
<td>38</td>
<td>11100</td>
<td>RDWR</td>
<td>B1SL</td>
<td>B1 Data in SLD to Loop Direction</td>
</tr>
<tr>
<td>3A</td>
<td>11101</td>
<td>RDWR</td>
<td>B2SL</td>
<td>B2 Data in SLD to Loop Direction</td>
</tr>
<tr>
<td>3C</td>
<td>11110</td>
<td>RDWR</td>
<td>CX</td>
<td>Control Byte to SLD</td>
</tr>
<tr>
<td>3E</td>
<td>11111</td>
<td>RDWR</td>
<td>SX</td>
<td>Signaling Byte to SLD</td>
</tr>
</tbody>
</table>

**NOTE:**

1. Address represents AD1–AD5. AD0 is not used by the 29C53AA for addressing.

5-34
29C53AA Register Definitions

In the register descriptions that follow, the acronym, name, five bit address, and whether the register can be written or read (or read only in RT mode) are provided in the heading. For easy reference, the registers are listed in alphabetical order.

**B1LS B1 Data “S” to SLD Direction 11000 R, W**

<table>
<thead>
<tr>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>B1 data</td>
</tr>
</tbody>
</table>

This register provides access to the B1 channel data flow in the direction from the "S" loop to SLD interface. Data can be read or overwritten by the microprocessor in intercept mode (see SCR register). Also, data can be accessed so the MSB is in bit 7 (default mode) or flipped so that the LSB is in bit 7 by issuing the appropriate GCR command.

**B2LS B2 Data “S” to SLD Direction 11001 R, W**

<table>
<thead>
<tr>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>B2 data</td>
</tr>
</tbody>
</table>

This register provides access to the B2 channel data flow in the direction from the "S" loop to SLD interface. Data can be read or overwritten by the microprocessor in intercept mode (see SCR register). Also, data can be accessed so the MSB is in bit 7 (default mode) or flipped so that the LSB is in bit 7 by issuing the appropriate GCR command.

**B1SL B1 Data SLD to “S” Direction 11100 R, W**

<table>
<thead>
<tr>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>B1 data</td>
</tr>
</tbody>
</table>

This register provides access to the B1 channel data flow in the direction from the SLD interface to the "S" loop. Data can be read or overwritten by the microprocessor in intercept mode (see SCR register). Also, data can be accessed so the MSB is in bit 7 (default mode) or flipped so that the LSB is in bit 7 by issuing the appropriate GCR command.

**B2SL B2 Data SLD to “S” Direction 11101 R, W**

<table>
<thead>
<tr>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>B2 data</td>
</tr>
</tbody>
</table>

This register provides access to the B2 channel data flow in the direction from the SLD interface to "S" loop. Data can be read or overwritten by the microprocessor in intercept mode (see SCR register). Also, data can be accessed so the MSB is in bit 7 (default mode) or flipped so that the LSB is in bit 7 by issuing the appropriate GCR command.

**CR Control Byte Receive 11010 R, W**

<table>
<thead>
<tr>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>data</td>
</tr>
</tbody>
</table>

This address provides access to the receive control byte register pair in the SLD register bank. In SLD master mode, this register contains the received control (C) byte from the SLD link. This is typically control information read back from an SLD slave device such as the 29C48. Register contents are only valid in SSM (SLD master) mode.

**CX Control Byte Transmit 11110 R, W**

<table>
<thead>
<tr>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>data</td>
</tr>
</tbody>
</table>

This address provides access to the transmit control byte register pair in the SLD register bank. In SLD master mode, data placed in this register is transmitted over the SLD link in the C byte. This is typically control information sent to an SLD slave device such as the 29C48. Data is only transmitted in SSM (SLD master) mode.

**DO TH Overflow Threshold (Low Byte) 01111 W (RT)**

<table>
<thead>
<tr>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Overflow Threshold Bits 0–7</td>
</tr>
</tbody>
</table>

This overflow threshold (maximum packet length) may be specified in the range 1–4095. An exception is generated (see DPS register) if the threshold is equaled or exceeded. Setting this register to 00H (default) disables this function. The most significant four bits of the overflow threshold are set in the DUTH register.

**DPC D-Channel Processor Control 00001 W (RT)**

<table>
<thead>
<tr>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>unused</td>
<td>DRT</td>
<td>B2D</td>
<td>D-Ch. Routing</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

DRT D-channel routing through B2 on the SLD if bits 2–0 are 101.

<table>
<thead>
<tr>
<th>BITS</th>
<th>5</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 0</td>
<td>route through B2 bits 0, 1</td>
<td></td>
</tr>
<tr>
<td>0 1</td>
<td>route through B2 bits 2, 3</td>
<td></td>
</tr>
<tr>
<td>1 0</td>
<td>route through B2 bits 4, 5</td>
<td></td>
</tr>
<tr>
<td>1 1</td>
<td>route through B2 bits 6, 7</td>
<td></td>
</tr>
</tbody>
</table>

B2D When the raw D channel is routed to the SLD in the B2 byte (DPC bits 2–0 = 101), this bit sets the value for the unused bits of the transmit direction B2 byte (all zeroes or all ones).
D-channel routing is programmed in bits 2–0 of this register for both NT and TE operation. D-channel processing can be bypassed to provide a clear 16K bits per second channel over the SLD line if desired. Bit 1/3/5/7 of the B2 channel or bit 1 of the S channel will be the first one transferred over the “S” bus in this case.

<table>
<thead>
<tr>
<th>BITS</th>
<th>D-CHANNEL MODE</th>
</tr>
</thead>
<tbody>
<tr>
<td>2 1 0</td>
<td>Processor inactive, disconnect from SBUS</td>
</tr>
<tr>
<td>0 0 0</td>
<td>Loopback test mode (1)</td>
</tr>
<tr>
<td>0 1 0</td>
<td>Reserved</td>
</tr>
<tr>
<td>0 1 1</td>
<td>Processor active on SBUS, normal operation</td>
</tr>
<tr>
<td>1 0 0</td>
<td>Processor inactive, raw D through SLD S byte bits 1, 0</td>
</tr>
<tr>
<td>1 0 1</td>
<td>Processor inactive, raw D through SLD B2 byte as set in DPC 4, 5</td>
</tr>
<tr>
<td>1 1 0</td>
<td>Reserved</td>
</tr>
<tr>
<td>1 1 1</td>
<td>Reserved</td>
</tr>
</tbody>
</table>

**NOTE:**
1. Bit PEC.4 (EMO) must also be set to one to use the D-channel loopback test mode when operating as a loop slave (TE) if the loop interface is not synchronized.

**DPR** D-Channel Priority 01011 R, W

<table>
<thead>
<tr>
<th>BITS</th>
<th>D-CHANNEL MODE</th>
</tr>
</thead>
<tbody>
<tr>
<td>7 6 5 4 3 2 1 0</td>
<td>unused</td>
</tr>
</tbody>
</table>

ZIS When ZIS = 1, the transmitter will not perform zero insertion on the outgoing D-channel frame. The data in the FIFO will be transmitted as is. When ZIS = 0, zero insertion is enabled which is normal operation. The default for this bit is 0.

PRI When PRI = 0, the higher priority class (8) is selected for the D-channel priority logic. When PRI = 1, the lower priority class (10) is selected. This bit defaults to 1, selecting the lower priority class.

**DPS** D-Channel Processor Status 00001 R

<table>
<thead>
<tr>
<th>BITS</th>
<th>D-CHANNEL MODE</th>
</tr>
</thead>
<tbody>
<tr>
<td>7 6 5 4 3 2 1 0</td>
<td>XB</td>
</tr>
</tbody>
</table>

XB Transmitter busy. This bit is set to 1 whenever D-channel processor is transmitting. This bit does not affect the XDP bit in EXS, or cause an interrupt. This bit is zero when the transmitter is inactive or awaiting priority on the D-channel.

Transmit status, encoded as follows:

<table>
<thead>
<tr>
<th>BITS</th>
<th>Transmit Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>2 1 0</td>
<td>Packet underflow (packet too short, threshold set in DUTH)</td>
</tr>
<tr>
<td>0 0 0</td>
<td>Still receiving, no EOP yet</td>
</tr>
<tr>
<td>0 0 1</td>
<td>Good EOP</td>
</tr>
<tr>
<td>0 1 0</td>
<td>FCS error</td>
</tr>
<tr>
<td>0 1 1</td>
<td>FIFO underrun (read when empty)</td>
</tr>
<tr>
<td>1 0 0</td>
<td>FIFO overrun (FIFO full when next byte received)</td>
</tr>
<tr>
<td>1 0 1</td>
<td>Packet overflow (packet too long, threshold set in DOTh)</td>
</tr>
<tr>
<td>1 1 0</td>
<td>Packet underflow (packet too short, threshold set in DUTH)</td>
</tr>
<tr>
<td>1 1 1</td>
<td>Abort or loss of sync</td>
</tr>
</tbody>
</table>

**DUTH** Underflow and Overflow (High Byte) Threshold 01110 W (RT)

<table>
<thead>
<tr>
<th>BITS</th>
<th>D-CHANNEL MODE</th>
</tr>
</thead>
<tbody>
<tr>
<td>7 6 5 4 3 2 1 0</td>
<td>underflow threshold</td>
</tr>
</tbody>
</table>

The underflow threshold (minimum packet length) may be specified in the range 1–15. An exception is generated (see DPS register) if the threshold is not exceeded. Setting this register to 00H (default) disables this function. The upper four bits of the overflow threshold are also contained here.
EXM Exception Mask 00000 W (RT)

<table>
<thead>
<tr>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>XSR</td>
<td>RSR</td>
<td>PC</td>
<td>SX</td>
<td>LS</td>
<td>XDP</td>
<td>RDP</td>
</tr>
</tbody>
</table>

Setting a bit to 1 in this register enables the associated exception to generate an interrupt, and to appear in the S byte of the SLD line. This register is initialized to 00H, all interrupt sources masked.

EXS Exception Status 00000 R

<table>
<thead>
<tr>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>X</td>
<td>XSR</td>
<td>RSR</td>
<td>PC</td>
<td>SX</td>
<td>LS</td>
<td>XDP</td>
<td>RDP</td>
</tr>
</tbody>
</table>

This is the main status register. It should be polled first when investigating the source of an exception. Some status is expanded in additional registers. When unmasked, bits set in EXS cause an interrupt, and affect the SLD signaling byte status.

RSR request for transmit FIFO data transfer. The transmit FIFO is at the programmed level of emptiness (set in XFXF). Or, if XFXF = 0, the closing flag of a packet has been transmitted.

PC change noted in peripheral interface inputs. Read LPS bits 5–7 for status.

SX exception noted in spare bit unit (multiframing or A bit change). Read SBC for more information.

LS loss or gain of synchronization to the “S” loop. Read LPS for loop status.

XDP D-channel processor exception, transmit side. Read DPS bits 4–6 to determine cause.

RDP D-channel processor exception, receive side. Read DPS bits 0–2 to determine cause.

GCR General Command Register 01010 R, W

<table>
<thead>
<tr>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>X</td>
<td>command code</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Writing one of the listed codes to the GCR register causes the specified event to execute once. A new command should not be written to GCR for five cycles of CLK. Where the code contains X's, those bits have no effect. Reading the GCR register reads the previous command.

<table>
<thead>
<tr>
<th>CODE</th>
<th>COMMAND</th>
<th>EVENT</th>
</tr>
</thead>
<tbody>
<tr>
<td>111XXXXX</td>
<td>GRST</td>
<td>Reset all internal units.</td>
</tr>
<tr>
<td>011XXXXX1</td>
<td>GDRAB</td>
<td>Clear receive FIFO to next EOP (ignore data from loop until FLAG if none presently in FIFO).</td>
</tr>
<tr>
<td>011XXXXX0</td>
<td>GDRCL</td>
<td>Clear entire receive FIFO.</td>
</tr>
<tr>
<td>011XXXXX10</td>
<td>GDXAB</td>
<td>Abort and clear the transmit packet currently being constructed by the microprocessor.</td>
</tr>
<tr>
<td>011XXXXX00</td>
<td>GCXCL</td>
<td>Stop D-channel transmission, clear transmit FIFO, and indicate idle on transmitted D-channel.</td>
</tr>
<tr>
<td>010XXXXX0</td>
<td>GDXMK</td>
<td>Mark EOP in transmit FIFO (no effect if FIFO empty).</td>
</tr>
<tr>
<td>010XXXXX1</td>
<td>GLMX4, GLMX2</td>
<td>Transmit INFO 4 (T = 0) or INFO 2 (T = 1) regardless of receive state. GLMX2 must be issued once before GLMX4 will cause INFO 4 to be transmitted. GTD must be issued to after GLMX4 return to normal operation.</td>
</tr>
<tr>
<td>010XXXXX00</td>
<td>GSSY</td>
<td>Synchronize S frame to the next SLD frame (valid only when the loop is inactive).</td>
</tr>
<tr>
<td>001XXXXXNF</td>
<td>GB1F, GB2F</td>
<td>N = select B1 (0) or B2 (1). F = 0, don't flip (bit reverse) B1/B2 between PBUS and SBUS (MSB first). F = 1, flip B1/B2 between PBUS and SBUS (LSB first).</td>
</tr>
<tr>
<td>000XXXX0A</td>
<td>GTST</td>
<td>Set PBUS address A5 to A (A = 1 for register test mode).</td>
</tr>
<tr>
<td>000XXXX1D</td>
<td>GTA, GTD</td>
<td>Set D = 1 for loop activation command, set D = 0 for loop deactivation command.</td>
</tr>
<tr>
<td>otherwise</td>
<td>no effect</td>
<td>no action taken.</td>
</tr>
</tbody>
</table>
LCR Loop Interface Control 00010 W (RT)

<table>
<thead>
<tr>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>unused</td>
<td>B1E</td>
<td>B2E</td>
<td>XMIT</td>
<td>MODE</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

B1E, B2E B-Channel Transmit Enables (no effect in LSM modes)
- 0 disable transmitter for Bn subframes
- 1 enable transmitter for Bn subframes

XMIT Transmitter Enable (no effect in LSM modes)
- 0 power/down/disable transmitter (output is high impedance)
- 1 enable transmitter

MODE Loop Interface Mode

bits
2 1 0

0 0 0 LOFF power down/disable transceiver
0 0 1 LOFF power down/disable transceiver
0 1 0 reserved
0 1 1 reserved

1 0 0 LSSA Adaptive Receive Timing, Slave (TE)
1 0 1 LSMH Hybrid Receive Timing, Master (NT)
Extended passive bus or point-to-point
1 1 0 LSMA Adaptive Receive Timing, Master (NT)
1 1 1 LSMF Fixed Receive Timing, Master (NT)
Short passive bus

LLB Loop Interface Loopback Control 00111 R, W

<table>
<thead>
<tr>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>DL</td>
<td>B2L</td>
<td>B1L</td>
<td>0</td>
<td>DS</td>
<td>B2S</td>
<td>B1S</td>
</tr>
</tbody>
</table>

For data looped back towards the loop, the analog circuitry and data formatting and deformating is included. For data looped back towards the system (SLD) interface, the loopback occurs at the interface to the internal SBUS. None of the data formatting, or analog circuitry is included. The loop interface must be active and synchronized to use these loopback features.

DL 1 = Loopback D toward loop, 0 = disable loopback
B2L 1 = Loopback B2 toward loop, 0 = disable loopback
B1L 1 = Loopback B1 toward loop, 0 = disable loopback
DS 1 = Loopback D toward system, 0 = disable loopback
B2S 1 = Loopback B2 toward system, 0 = disable loopback
B1S 1 = Loopback B1 toward system, 0 = disable loopback

When operating as a loop slave (TE), bit PEC.4 (EM0) must also be set to one to use the D-channel loopback toward the system interface (set by the DS bit) if the loop interface is not synchronized.

LPS Loop, Peripheral Interface Status 00010 R

<table>
<thead>
<tr>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>P3S</td>
<td>P2S</td>
<td>P1S</td>
<td>encoded loop interface status</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

P3S state of peripheral interface pin P3 when configured as an input
P2S state of peripheral interface pin P2
P1S state of peripheral interface pin P1

Loop interface status code:

BITS

4 reserved for future use, should be zero
3 mode: master(1)/slave(0)
2 transmitter enable (both LCR.3 AND LCR.2 are set)

1, 0 11 = active
10 = initialize (init)
01 = initialize
11 = inactive
### 29C53AA LOOP INTERFACE STATES

<table>
<thead>
<tr>
<th>BITS 43210</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1XXXXX</td>
<td>Reserved for future use</td>
</tr>
<tr>
<td>01111</td>
<td>LSM3 active: sending INFO4</td>
</tr>
<tr>
<td>01110</td>
<td>LSM2 remote init: receiving INFO 1</td>
</tr>
<tr>
<td>01101</td>
<td>LSM1 local init: send INFO 2 on local request</td>
</tr>
<tr>
<td>01100</td>
<td>LSM0 inactive, powered up (receiver active)</td>
</tr>
<tr>
<td>01011</td>
<td>— (unused) —</td>
</tr>
<tr>
<td>01010</td>
<td>LMP2 passive resync, &lt; 16 frames correct</td>
</tr>
<tr>
<td>01001</td>
<td>LMP1 passive resync, &lt; 3 pulses per frame</td>
</tr>
<tr>
<td>01000</td>
<td>— (unused) —</td>
</tr>
<tr>
<td>00111</td>
<td>LSS3 active: sending INFO3</td>
</tr>
<tr>
<td>00110</td>
<td>LSS2 remote init: sync to INFO 2/INFO 4</td>
</tr>
<tr>
<td>00101</td>
<td>LSS1 local init: send INFO 1 on local request</td>
</tr>
<tr>
<td>00100</td>
<td>LSS0 inactive, powered up (receiver active)</td>
</tr>
<tr>
<td>00011</td>
<td>LSP3 active, receive only, successful passive resync</td>
</tr>
<tr>
<td>00010</td>
<td>LSP2 passive resync, &lt; 16 frames correct</td>
</tr>
<tr>
<td>00001</td>
<td>LSP1 passive resync, &lt; 3 Pulses per frame</td>
</tr>
<tr>
<td>00000</td>
<td>LOFF inactive, powered down (receiver inactive)</td>
</tr>
</tbody>
</table>

PEC Peripheral Interface, E Channel Control 00011

<table>
<thead>
<tr>
<th>7 6 5 4 3 2 1 0</th>
<th>P30EN P3 output enable (also enabled by SST). 0 = input, 1 = output</th>
</tr>
</thead>
<tbody>
<tr>
<td>0  AA EM1 EM0 SST P30EN P3 P4</td>
<td>P3 P4 output value</td>
</tr>
</tbody>
</table>

**AA** auto-answer mode enabled (1), or disabled (0)

**EM1, EM0**

Master, LSM modes

| 0X | generate E channel from received D |
| 10 | force E channel to logical zero |
| 11 | force E channel to logical one |

Slave, LSS mode

| X0 | normal E-channel function, contention resolution mechanism is enabled |
| 01 | ignore E channel, always transmit D-channel data without waiting for priority. Loss of priority exception is suppressed |
| 11 | Force loss of priority |

**SST** present SLD status on P3 and P4 if 1, or if 0, P3 and P4 are I/O pins controlled by PEC bits 0–2.

P30EN P3 output enable (also enabled by SST). 0 = input, 1 = output

P3 P3 output value

P4 P4 output value

PLENH D-Channel Packet Length High Byte 01110 R

| 7 6 5 4 3 2 1 0 | length |

This register holds the upper byte of the length of the current D-channel packet. The count is updated as the packet is read out of the receive FIFO. Reads of PLENH should be done a minimum of 3 CLK cycles after reading RFO.

PLENL D-Channel Packet Length Low Byte 01111 R

| 7 6 5 4 3 2 1 0 | length |

This register holds the lower byte of the length of the current D-channel packet. The count is updated as the packet is read out of the receive FIFO. Reads of PLENL should be done a minimum of 3 CLK cycles after reading RFO.
PSR Position Selection Register 10001 R, W

<table>
<thead>
<tr>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>EOP</td>
<td>MORE</td>
<td>FULL</td>
<td>number of bytes avail.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

A one in any bit of this register enables that bit onto the microprocessor bus when the XSR and RSR registers are read. Those bits of PSR in the zero state disable the corresponding bit positions of the microprocessor bus of the 29C53AA so that those bits remain in a high impedance state when RSR or XSR are read.

This register allows the processor to poll up to eight 29C53AA transceivers on the same microprocessor bus at one time, with each having its own status bit position in the byte. Using this method the processor can check status on all 29C53AA devices with a single read.

See also RSR (Receive Service Request) and XSR (Transmit Service Request).

This register cannot be accessed via SLD command.

RFN Receive FIFO Number 00100 R

<table>
<thead>
<tr>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>EOP</td>
<td>MORE</td>
<td>FULL</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

This register indicates the number of bytes in the receive FIFO as detailed below.

EOP Minimum of one byte in the receive FIFO is marked as the end of a packet if EOP = 1. If EOP = 0, no byte marked as end of packet.

MORE If MORE = 1, more information is available in the receive FIFO beyond the first packet, delineated by an EOP marker. If MORE = 0, no data lies beyond the first packet.

FULL The entire FIFO is filled if 1. The FIFO is not full if 0.

number If at least one EOP is marked, this value is the number of bytes to the first EOP. If no EOP is marked, this value indicates the number of valid bytes in the receive FIFO.

RFO Receive FIFO Output 01000 R

<table>
<thead>
<tr>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>data</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Read from this register to read from the receive FIFO. Do not write to this location, as one byte will be lost from the packet (read out from the FIFO). Reads from this location should be separated by 4 CLK cycles.

RFXF Receive FIFO Exception Fullness 01100 R, W

<table>
<thead>
<tr>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>unused</td>
<td>RFXF</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

If the number of bytes accumulated in the FIFO is equal to or exceeds this 5-bit number, or if the receive FIFO contains a byte marked EOP, the receive FIFO exception RSR is activated. Setting this register to zero disables this function. However, a packet tagged EOP will still set the RSR exception.

RSR Receive Service Request 10010 R

<table>
<thead>
<tr>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>each bit set to state of RSR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

All bits in this register reflect the same status, the state of the RSR bit in the EXS register. When this register is read, only those bits indicated by a 1 in the corresponding bit positions in the PSR register are enabled onto the microprocessor bus. The remaining bit positions on the microprocessor port pins remain in a high impedance state during a read. This allows up to eight 29C53AA transceivers connected to the same microprocessor bus to be polled for status with one read. This feature is useful, for example, in linecard applications where multiple 29C53AA devices are controlled by one microprocessor.

See also PSR (Position Selection Register) and XSR (Transmit Service Request).

SBC Spare Bit Control 10000 R, W

<table>
<thead>
<tr>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>MFE</td>
<td>MDX</td>
<td>MS</td>
<td>SIE</td>
<td>MIE</td>
<td>MX</td>
<td>AXE</td>
<td>AX</td>
</tr>
</tbody>
</table>

All exceptions are generated at the beginning of the ISDN frame following the event which causes the exception.

MFE Multiframe function enable. When MFE = 1, the spare bit unit is in the multiframe mode. When MFE = 0, the 29C53AA will not perform any of the multiframe procedures.

MDX Multiframe Q/S data change exception indication. When MFE = 1, this bit is set to 1 if a Q/S bit quartet is received that is different from the previous one received, and a spare bit exception (SX in EXS) is generated. This bit is cleared upon reading SBC.

MS Multiframe sync indication. When MFE = 1, this bit indicates the multiframe synchronization status (1 = in sync, 0 = out of sync). When MS changes from 1 to 0, a spare bit exception is generated (SX in EXS). When MFE = 0, this bit is always a one.
SIE  Single frame interrupt enable. Setting this bit to a 1 enables the generation of a spare bit exception (SX in EXS) every ISDN frame. The single frame exception does not have an indication as such, but the indication should be inferred if the SX bit in EXS and the SIE bit are set. If SIE = 0, no single frame exception is generated.

MIE  Multiframe interrupt enable. When MFE = 1, setting MIE to a 1 enables the generation of a spare bit exception (SX in EXS) when the MX bit becomes set. MIE = 0 masks the MX bit from producing a spare bit exception.

MX  Multiframe exception indication. When MFE = 1, this bit becomes set to 1 on the first frame of the 20 frame multiframe, indicating that new Q or S transmit data can be written to SBX, and that new Q or S receive data is available in SBR. MX is cleared upon reading SBC. If MFE = 0, MX is not set. At the TE the multiframe exception indication is not dependent on multiframe sync being established.

AXE  Activation bit change exception enable. Setting AXE to a 1 enables the generation of a spare bit exception (SX in EXS) when AX becomes set. For AXE = 0, the A bit exception is suppressed.

AX  Activation bit exception indication. This bit becomes set to 1 after a change in the received A bit value at the TE, only after the loop interface has acquired synchronization. The state of the A bit can be read in SBR. This bit is cleared upon reading SBC.

SBR  Spare Bit Receive 00110 R

<table>
<thead>
<tr>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
</table>
| QS4/X | QS3/X | QS2/F | QS1/S | S/M | N | FA | A#

(MFE = 1/MFE = 0)

QSI- Q or S bit quartet received. These bits are only valid when MFE = 1. QS1 is received first.

S  S bit received. Valid only in LSS (TE) mode. When MFE = 1 the S bit is located at SBR.3, and when MFE = 0, at SBR.4.

M  M bit received. Valid only in LSS mode.

N  N bit received. Valid only in LSS mode.

FA  FA bit received.

A  A bit received. Valid only in LSS mode.

NOTE:
The Q and S bit quartets are updated every 20 frames, while the remaining bits are updated every frame.

SBX  Spare Bit Transmit 00110 W (RT)

<table>
<thead>
<tr>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
</table>
| QS4/X | QS3/X | QS2/F | QS1/S | S/M | N=FA | FA | A#

(MFE = 1/MFE = 0)

QSI- Q or S bit quartet transmission. These bits are only valid when MFE = 1. QS1 is transmitted first.

FAE#  FA echo enable. Valid only when MFE = 0 and in LSS (TE) mode. If FAE# is 0, the 29C53AA automatically echoes the received FA bit from the NT in the FA bit position of its transmit frame. If FAE# is 1, the transmitted FA bit follows the state of bit SBR.1. The default is FAE# = 0, and the FA bit is echoed.

S  S bit transmitted. Valid only in LSS (NT) modes. When MFE = 1, the S bit is located at SBR.3, and when MFE=0, at SBR.4. When MFE = 1, the S bit is transmitted in the frames of the multiframe when FA is not equal to 1.

M  M bit transmitted. Valid only in NT mode and MFE = 0.

N = FA Control for the N bit. Valid only in the NT mode. If N = FA is set to 1 the N bit transmitted will equal the FA bit value. The N = FA bit should be set to 0 for normal operation so that the N bit will be the complement of the FA bit. The default is N = FA set to 0.

FA  FA bit transmitted. When MFE = 0 (and FAE# = 1 for LSS mode) this bit controls the value of the transmitted FA bit (normally 0). If MFE = 1, this bit controls the value of the transmitted FA bit during frames not involved in the multiframe procedure.

A#  Control for the A bit. Valid only in the NT mode. The complement of this bit is sent in the A bit position of the transmit frame. For the default value of zero the A bit is transmitted as a one. During activation, the A bit is set to zero during INFO 2 regardless of the state of A#.

SCR  System Interface Control 00100 W (RT)

<table>
<thead>
<tr>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>unused</td>
<td>B2M</td>
<td>B1M</td>
<td>SIM</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

This register configures the system, or SLD interface. It defaults to OOH upon power up, which is the
SLD slave mode. In this mode, the 29C53AA expects to receive its timing reference from the SLD interface. Therefore, in a terminal or other application where the SLD interface will be a signal source, the SLD mode master mode should be programmed immediately after reset.

Bits 5, 4 B2 Mode
- 0 0 enable normal SLD transfers (default mode).
- 1 0 microprocessor intercept of B2
- 0 1 loop back B2 toward system interface
- 1 1 loop back B2 toward loop interface

Bits 3, 2 B1 Mode
- 0 0 enable normal SLD transfers (default mode).
- 1 0 microprocessor intercept of B1
- 0 1 loop back B1 toward system interface
- 1 1 loop back B1 toward loop interface

Bits 1, 0 System Interface Mode
- 0 0 SSS SLD slave (default mode). The 29C53AA SIDR and SCL pins are inputs, and the 29C53AA transmits data on SLD when SIDR is low
- 1 0 SSN SLD interface for intelligent NT2. The 29C53AA generates SCL and SIDR, and transmits data on SLD when SIDR is low
- 0 1 SSM SLD master. The 29C53AA generates SCL and SIDR, and transmits data on SLD when SIDR is high.
- 1 1 reserved

SDC SLD Data Transfer Configuration 00101 W (RT)

<table>
<thead>
<tr>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>unused</td>
<td>CE</td>
<td>MP</td>
<td>MCS</td>
<td>MF</td>
<td>SS</td>
<td>SC</td>
</tr>
</tbody>
</table>

This register has no effect in SLD master mode (SSM).

CE Command Enable
- 0 Ignore commands received on the SLD line (except when RD and WR are both low during reset)
- 1 Enable execution of SLD commands

MP Multiple Transfer to SLD Master
- 0 respond to poll command received in control byte
- 1 respond immediately beginning in the next half of the SLD frame

MCS Multiple Byte Configuration and Status Transfer
0 use control byte only (one byte per SLD frame)
1 use both control and signaling bytes (two bytes per SLD frame)

MF Multiple Byte FIFO transfers (both directions)
0 use control byte only (one byte per SLD frame)
1 use both control and signaling bytes (two bytes per SLD frame)

SS Single Status Byte Transfer (29C53AA to SLD master)
0 respond to poll command received in SLD control byte
1 respond immediately beginning in the next half of the SLD frame

SC Single Command/Configuration Byte Transfer (SLD master to 29C53AA)
0 data follows in control byte of next frame
1 data follows in the same frame’s signaling byte

SR Signaling Byte Receive 11011 R, W

<table>
<thead>
<tr>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>data</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

This address provides access to the receive signaling byte register pair in the SLD register bank. In SLD master mode, this register contains the received signaling (S) byte from the SLD link. This is typically signaling or status information read back from an SLD slave device. Register contents are only valid in SSM (SLD Master) mode.

SX Signaling Byte Transmit 11111 R, W

<table>
<thead>
<tr>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>data</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

This address provides access to the transmit signaling byte register pair in the SLD register bank. In SLD master mode, data placed in this register is transmitted over the SLD link in the S byte. This is typically signaling or status information sent to an SLD slave device. Register contents are only transmitted in SSM (SLD Master) mode.

XFI Transmit FIFO Input 01001 W

<table>
<thead>
<tr>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>data</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

This is the input address for the transmit FIFO. This register should not be read, as it will cause the previous contents of XFI to be entered into the FIFO, and the FIFO count (XFN) to be incremented. Writes to this register should be separated by 3 CLK cycles.
This register indicates the number of empty bytes in the transmit FIFO. The count requires 5 CLK cycles to update after a write to XFI.

If the number of untransmitted bytes in the transmit FIFO is equal to this 5-bit number, the transmit service request (XSR) bit is set in EXS.
ABSOLUTE MAXIMUM RATINGS

Temperature Under Bias ............. \(-10^\circ C \text{ to } +80^\circ C\)
Storage Temperature ................. \(-65^\circ C \text{ to } +150^\circ C\)
Voltage on any Pin ........ \(V_{SS} -0.5V \text{ to } V_{CC} + 0.5V\)
Maximum Voltage on \(V_{CC}\)
with Respect to \(V_{SS}\) ............... \(+7V\)
Total Power Dissipation ............. \(500\text{ mW}\)

*Notice: Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions above those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

NOTICE: Specifications contained within the following tables are subject to change.

D.C. CHARACTERISTICS \(V_{CC} = +5V \pm 5\%; V_{SS} = 0V; T_A = 0^\circ C \text{ to } 70^\circ C;\)
Typical Values are at \(T_A = 25^\circ C\) and Nominal Power Supply Values

DIGITAL INTERFACES

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Parameter</th>
<th>Min</th>
<th>Typ</th>
<th>Max</th>
<th>Units</th>
<th>Test Conditions</th>
</tr>
</thead>
<tbody>
<tr>
<td>WLK</td>
<td>Input/Output Leakage Current (Excluding LR(+), LR(-), LX(+), LX(-))</td>
<td>±10</td>
<td>μA</td>
<td></td>
<td></td>
<td>(V_{SS} \leq V_{IN} \leq V_{CC})</td>
</tr>
<tr>
<td>VIL</td>
<td>Input Low Voltage</td>
<td>0.8</td>
<td>V</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>VIH</td>
<td>Input High Voltage</td>
<td>2.0</td>
<td>V</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>VOL</td>
<td>Output Low Voltage</td>
<td>0.45</td>
<td>V</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>VOH1</td>
<td>Output High Voltage</td>
<td>2.4</td>
<td>V</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>VOH2</td>
<td>Output High Voltage</td>
<td>0.9 (V_{CC})</td>
<td>V</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>Pin Cap. (ex. LR(\pm), LX(\pm))</td>
<td>10</td>
<td>pF</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

POWER SUPPLY CURRENT (Averaged over 1 ms)

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Parameter</th>
<th>Min</th>
<th>Typ</th>
<th>Max</th>
<th>Units</th>
<th>Test Conditions</th>
</tr>
</thead>
<tbody>
<tr>
<td>ICC (P)</td>
<td>Power Down (Standby)</td>
<td>4</td>
<td>8</td>
<td>mA</td>
<td>SLD and CLK Active</td>
<td></td>
</tr>
<tr>
<td>ICC (I)</td>
<td>Idle Operating Current</td>
<td>8</td>
<td>12</td>
<td>mA</td>
<td>Receiver, SLD, CLK Active</td>
<td></td>
</tr>
<tr>
<td>VCC (N)</td>
<td>Normal Operating Current</td>
<td>20</td>
<td>mA</td>
<td></td>
<td>Everything is Active (Excluding Current for Output Loads)</td>
<td></td>
</tr>
</tbody>
</table>

A.C. Characteristics \(V_{CC} = 5V \pm 5\%; V_{SS} = 0V; T_A = 0^\circ C - 70^\circ C; CLK = 3.84 MHz\)

RECEIVER

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Parameter</th>
<th>Min</th>
<th>Typ</th>
<th>Max</th>
<th>Units</th>
<th>Test Conditions</th>
</tr>
</thead>
<tbody>
<tr>
<td>VRD</td>
<td>Minimum Received Differential Pulse Voltage</td>
<td>300</td>
<td>400</td>
<td>mV</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ZIR</td>
<td>LR(+), LR(-) Input Impedance</td>
<td>30</td>
<td>60</td>
<td>KΩ</td>
<td>Each Pin</td>
<td></td>
</tr>
<tr>
<td>CIR</td>
<td>LR(+), LR(-) Input Capacitance</td>
<td>10</td>
<td>20</td>
<td>pF</td>
<td>Each Pin</td>
<td></td>
</tr>
</tbody>
</table>
## TRANSMITTER

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Parameter</th>
<th>Min</th>
<th>Typ</th>
<th>Max</th>
<th>Units</th>
<th>Test Conditions</th>
</tr>
</thead>
<tbody>
<tr>
<td>VXD</td>
<td>Transmit Differential Pulse Voltage</td>
<td>1900</td>
<td>2000</td>
<td>2100</td>
<td>mV</td>
<td>$R_L = 2500$ (Note 1)</td>
</tr>
<tr>
<td>ZOX1</td>
<td>LX+, LX- Output Impedance</td>
<td>30</td>
<td>60</td>
<td></td>
<td>KΩ</td>
<td>Each Pin, Transmitting Binary One</td>
</tr>
<tr>
<td>ZOX2</td>
<td>LX+, LX- Output Impedance</td>
<td>2</td>
<td>5</td>
<td></td>
<td>Ω</td>
<td>Transmitting Binary Zero, Each Pin</td>
</tr>
<tr>
<td>COX</td>
<td>Output Capacitance</td>
<td>30</td>
<td>40</td>
<td></td>
<td>pF</td>
<td>Each Pin</td>
</tr>
<tr>
<td>CL</td>
<td>Capacitive Load between LX+, LX-</td>
<td></td>
<td></td>
<td>1500</td>
<td>pF</td>
<td>Parallel with 300Ω Directly across LX+ and LX- (Note 2)</td>
</tr>
<tr>
<td>tMR</td>
<td>Transmit Pulse Rise Time</td>
<td></td>
<td></td>
<td>400</td>
<td>ns</td>
<td>Test Load</td>
</tr>
<tr>
<td>IXL</td>
<td>Source, Sink Current Limit</td>
<td>10</td>
<td></td>
<td>14.4</td>
<td>mA</td>
<td>$R_L = 75$</td>
</tr>
<tr>
<td>PHU</td>
<td>Pulse Height Unbalance</td>
<td>0.5</td>
<td>3</td>
<td></td>
<td>%</td>
<td>Test Load</td>
</tr>
<tr>
<td>tpW</td>
<td>Pulse Width</td>
<td>5.16</td>
<td>5.21</td>
<td>5.26</td>
<td>μs</td>
<td>$CLK = 3.84$ MHz ± 100 ppm (Note 3)</td>
</tr>
</tbody>
</table>

## TIMING

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Parameter</th>
<th>Min</th>
<th>Typ</th>
<th>Max</th>
<th>Units</th>
<th>Test Conditions</th>
</tr>
</thead>
<tbody>
<tr>
<td>J</td>
<td>Timing Extraction Jitter (&quot;S&quot; Slave Mode)</td>
<td>0</td>
<td></td>
<td>±5</td>
<td>%</td>
<td>$CLK = 3.84$ MHz ± 100 ppm</td>
</tr>
<tr>
<td>PD</td>
<td>Total Phase Deviation LX with Respect to LR</td>
<td>−7</td>
<td></td>
<td>+15</td>
<td>%</td>
<td>$CLK = 3.84$ MHz ± 100 ppm</td>
</tr>
</tbody>
</table>

### NOTES:

1. This is essentially the open circuit voltage.
2. This is a stability test. Overshoot less than 25%, damping time less than 1.5 μs.
3. Free running, measured between zero crossing of adjacent pulses. During DPLL adjustment in LSS (TE) mode, the framing pulse may be wider or narrower by one cycle of CLK (260 ns).

![Transmitter Test Load](image)

![Differential Output](image)
**Figure 8. SLD Interface Timing (29C53AA As Master)**

**SLD INTERFACE TIMING (29C53AA as Master)**

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Parameter</th>
<th>Min</th>
<th>Max</th>
<th>Units</th>
</tr>
</thead>
<tbody>
<tr>
<td>TKC</td>
<td>CLK to SCL Delay</td>
<td>150</td>
<td></td>
<td>ns</td>
</tr>
<tr>
<td>TKS</td>
<td>CLK to SDIR Delay</td>
<td>150</td>
<td></td>
<td>ns</td>
</tr>
<tr>
<td>TKP</td>
<td>CLK to P3/P4 Delay(4)</td>
<td>150</td>
<td></td>
<td>ns</td>
</tr>
<tr>
<td>TKDE</td>
<td>CLK to SLD Driver Enabled</td>
<td>0</td>
<td></td>
<td>ns</td>
</tr>
<tr>
<td>TKDV</td>
<td>CLK to SLD Data Valid</td>
<td></td>
<td>150</td>
<td>ns</td>
</tr>
<tr>
<td>TKDH</td>
<td>SLD Data Hold After Clock Edge</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TKDD</td>
<td>CLK to SLD Float</td>
<td></td>
<td>150</td>
<td>ns</td>
</tr>
<tr>
<td>TDVS</td>
<td>SLD Data Input Setup Time to SCL</td>
<td>50</td>
<td></td>
<td>ns</td>
</tr>
<tr>
<td>TSDV</td>
<td>SLD Data Hold Time After SCL</td>
<td>80</td>
<td></td>
<td>ns</td>
</tr>
<tr>
<td>TCLK</td>
<td>CLK Period(5)</td>
<td>230</td>
<td>1000</td>
<td>ns</td>
</tr>
<tr>
<td>TCLKL</td>
<td>CLK Low Time</td>
<td>115</td>
<td>550</td>
<td>ns</td>
</tr>
<tr>
<td>TCLKH</td>
<td>CLK High Time</td>
<td>115</td>
<td>550</td>
<td>ns</td>
</tr>
</tbody>
</table>

**NOTES:**

1. 29C53AA samples SLD input data on SCL falling edge.
2. 29C53AA changes SLD output data on SCL rising edge.
3. 29C53AA SLD out is enabled 1½ CLK cycles after the SDIR rising edge and disabled ½ CLK cycle before the SDIR falling edge (as master only).
4. P3/P4, when programmed to output SLD status, changes state while SCL is low, approximately one CLK period ahead of SCL rising edge.
5. Range over which the 29C53AA will function. The frequency of CLK must be 3.84 MHz ± 100 ppm to meet layer 1 recommendations for free running bit rate.
## SLD INTERFACE TIMING (29C53AA as Slave)

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Parameter</th>
<th>Min</th>
<th>Max</th>
<th>Units</th>
</tr>
</thead>
<tbody>
<tr>
<td>TDIRR</td>
<td>SCL to SDIR Rising Edge</td>
<td>-150</td>
<td>150</td>
<td>ns</td>
</tr>
<tr>
<td>TDIRRFR</td>
<td>SCL to SDIR Falling Edge</td>
<td>-150</td>
<td></td>
<td>ns</td>
</tr>
<tr>
<td>TDIRFF</td>
<td>SDIR to SCL Falling Edge</td>
<td>150</td>
<td></td>
<td>ns</td>
</tr>
<tr>
<td>TCHDF</td>
<td>SCL High to Data Out Float</td>
<td></td>
<td>50</td>
<td>ns</td>
</tr>
<tr>
<td>TSHDF</td>
<td>SDIR High to Data Out Float</td>
<td></td>
<td>50</td>
<td>ns</td>
</tr>
<tr>
<td>TKDH</td>
<td>Output Data Hold After SCL Edge</td>
<td>0</td>
<td></td>
<td>ns</td>
</tr>
<tr>
<td>TKDV</td>
<td>Output Data Valid After SCL Edge</td>
<td></td>
<td>100</td>
<td>ns</td>
</tr>
<tr>
<td>TDVS</td>
<td>SLD Input Data Setup Time</td>
<td>50</td>
<td></td>
<td>ns</td>
</tr>
<tr>
<td>TSDV</td>
<td>SLD Input Data Hold Time</td>
<td>80</td>
<td></td>
<td>ns</td>
</tr>
<tr>
<td>TCHDE</td>
<td>Enable SLD Output After SCL</td>
<td>0</td>
<td></td>
<td>ns</td>
</tr>
<tr>
<td>TSLDE</td>
<td>Enable SLD Output After SDIR</td>
<td>0</td>
<td></td>
<td>ns</td>
</tr>
<tr>
<td>TSSH</td>
<td>SLD Status Hold After SCL&lt;sup&gt;(3)&lt;/sup&gt;</td>
<td>80</td>
<td>TCLK + TCLKL + 100</td>
<td>ns</td>
</tr>
</tbody>
</table>

### NOTES:
1. 29C53AA samples SLD input data on SCL falling edge.
2. 29C53AA changes SLD output data on SCL rising edge.
3. P3/P4, when programmed to output SLD status, are generated on the first CLK rising edge after SCL high to low transition is detected. SCL is sampled on CLK falling edge.
Figure 10. Microprocessor Bus Timing

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Parameter Description</th>
<th>Min</th>
<th>Max</th>
<th>Units</th>
</tr>
</thead>
<tbody>
<tr>
<td>TAL</td>
<td>Address Setup Before ALE Trailing Edge</td>
<td>10</td>
<td></td>
<td>ns</td>
</tr>
<tr>
<td>TLA</td>
<td>Address Hold After ALE Trailing Edge</td>
<td>20</td>
<td></td>
<td>ns</td>
</tr>
<tr>
<td>TWW</td>
<td>Write Control Signal Width</td>
<td>100</td>
<td></td>
<td>ns</td>
</tr>
<tr>
<td>TDW</td>
<td>Data Setup Before WR Trailing Edge</td>
<td>40</td>
<td></td>
<td>ns</td>
</tr>
<tr>
<td>TWD</td>
<td>Data Hold After WR Trailing Edge</td>
<td>15</td>
<td></td>
<td>ns</td>
</tr>
<tr>
<td>TAA</td>
<td>ALE Pulse Width</td>
<td>30</td>
<td></td>
<td>ns</td>
</tr>
<tr>
<td>TWI</td>
<td>Active CS Cycle Disallowed After WR(1)</td>
<td>$2.5 \times TCLK + 30$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TRR</td>
<td>Read Control Signal Width</td>
<td>100</td>
<td></td>
<td>ns</td>
</tr>
<tr>
<td>TRD</td>
<td>Access Time from RD Leading Edge</td>
<td>100</td>
<td></td>
<td>ns</td>
</tr>
<tr>
<td>TAD</td>
<td>Access Time from ALE Trailing Edge</td>
<td>250</td>
<td></td>
<td>ns</td>
</tr>
<tr>
<td>TDF</td>
<td>Float Delay After RD Trailing Edge</td>
<td>40</td>
<td></td>
<td>ns</td>
</tr>
<tr>
<td>TCI</td>
<td>Active CS Cycle Time for FIFO Access, RFO XFI for other registers</td>
<td>$4 \times TCLK$</td>
<td>$3 \times TCLK$</td>
<td>ns</td>
</tr>
<tr>
<td>TAC</td>
<td>ALE to Control Pulse</td>
<td>10</td>
<td></td>
<td>ns</td>
</tr>
</tbody>
</table>

**NOTE:**
1. Allow 3 extra clock cycles for GCR commands to execute.

---

Figure 11. A.C. Testing Input, Output Waveform
Figure 12. 29C53AA Application Diagram

NOTES:
1. The SLD port will be connected to an SLD master in NT applications, or to SLD slave devices such as the 29C48 programmable CODEC/Filter or appropriate data communication devices in TE applications.
2. Series resistance is used to increase the output impedance during transmission of a binary zero to greater than 20Ω on the loop side. It also serves as protection against surges when used in combination with external protection diodes.
3. A lower turns ratio (e.g., 1:1.8) may be used at the cost of a lower ratio of received signal to locally generated noise.
4. Appropriate protection circuitry can be added depending upon the application.
5. Pullup selected for approximately 1 mA \( I_{OL} \).
The 89151 T-Link Communications Controller (TCC) is a highly integrated communications controller which provides a complete implementation of the T-Link rate adaption protocol. The T-Link Communications Controller is used to rate adapt synchronous and asynchronous terminals to the public switched digital network, or an ISDN, providing transparent, digital end to end communications. The 89151 TCC includes a terminal interface port supporting synchronous or asynchronous terminals, a synchronous network interface port, and a general purpose parallel microprocessor port. The 89151 TCC can operate as a peripheral to a wide variety of microprocessors.
The T-Link Rate Adaption Protocol

T-Link is a full duplex byte oriented rate adaptation protocol designed to transfer either asynchronous or asynchronous data over a switched digital circuit at data rates from 300 bit/s to 64 Kbit/s. T-Link can operate over a 64 Kbit/s clear channel, or over a 64 Kbit/s restricted channel with a 1's density requirement. T-link can also be used on a 64 Kbit/s channel with capacity restricted to 56 Kbit/s due to the use of inband signaling or a 1's density requirement of today's T1 networks. T-link provides:

End to End Synchronization
Support of terminals with synchronous data rates from 1200 bit/s to 64 Kbit/s
Support of terminals with asynchronous data rates from 300 bit/s to 19.2 Kbit/s
Exchange of EIA or CCITT DTE/DCE lead status
User data (synchronous or asynchronous) transfer
Error correction for user data rates of 9600 bit/s or less

Networks providing circuit switched 64 Kbit/s data transmission are ideal for providing high speed, wide area data transfer. T-link provides a rate adaption protocol which can be used by a Terminal Adaption Device (TAD) to connect present DTEs to such networks. The T-link protocol can be used on existing networks as well as on the ISDN (Figure 2).

T-Link Communications Controller Overview

The 89151 T-Link Communications Controller provides a T-Link building block for implementing a Terminal Adaption Device that may be used for an ISDN TA or other TAD application. The complete rate adaption protocol is built into the 89151. The 89151 includes three interfaces which facilitate its use in DCE applications supporting the T-Link rate adaption protocol. There are the serial terminal interface, the serial network interface and the parallel microprocessor interface. Refer to Figure 1.

The terminal interface supports synchronous and asynchronous serial terminals. As T-Link carries the terminal status across the connection, the standard EIA handshake leads are provided. Clocking can be sourced by the attached device or by the 89151.

The network interface is designed to pass serial data between the 89151 and the network side of the terminal adaptor. It can operate in one of two modes. In one mode, the network interface consists of a serial input, serial output, a 2.56 MHz bit clock, and an 8 KHz synchronization clock. This mode is called IDL. In the second mode, it supports the SLD interface.

The microprocessor interface is used to pass commands and status between a local microprocessor and the 89151. Options for T-Link control can be selected. This interface includes an eight bit parallel data bus, read (RDB) and write (WRB) signals, a ready (RDY) signal and address lines (A0, A1). This port supports a wide variety of microprocessors and microcontrollers.

Figure 2. Rate Adaption for the Switched Digital Network
Applications

Figure 3 illustrates a typical TAD implemented using the 89151 T-Link communications Controller. The serial terminal interface can support a variety of protocols, such as RS232C/V.24, V.35, RS449, etc. The human interface provides keypad, switches and display for example. The network interface block provides the physical connection to the network and is dependent on the type of network involved. The microprocessor is in charge of responding to user inputs, setting up the connection and controlling T-link options. The entire T-Link rate adaptation protocol is performed by the 89151, providing a transparent data link once the connection is setup.

A specific example of an ISDN terminal adaptor is shown in Figure 4. Here the 89151 is used along with the 29C53 Digital Loop Controller, which provides access to the "S" reference point of the ISDN basic rate interface. The 89151 T-Link Communications Controller connects directly to the SLD interface of the 29C53, and is used to rate adapt a non-ISDN terminal to one of the 64 Kbit/s basic rate B channels. The TCC can operate over either the B1 or B2 channel, and so it can share the SLD interface with another SLD slave device, such as the 29C48 programmable CODEC/Filter. Channel data can be complemented by a TCC package pin. The layer 2 and 3 ISDN signaling functions are performed by the 80188.
ISP188
ISDN SOFTWARE PACKAGE FOR THE 80188

- Complies with CCITT Recommendations for Layers 1, 2 and 3 of the ISDN User Network Interface
- Device Drivers for Intel's IATC29C48/C50A, IATC29C53, and 82530
- Intel IAPX 80188 Based
- Software License and Source Code Included
- Written in Microsoft "C" Language
- IBM PC Development Environment
- PC Plug-In Development Boards Available
- Debug Monitor/Display Supported
- Comprehensive Support Services Available from DGM&S, Inc.*
- Reference Sold by Intel

The ISDN Software Package for the 80188 (ISP188) is specifically designed for ISDN terminal applications using Intel's Advanced Telecommunication Components (iATC). The software supports the iATC 29C53 Digital Loop Controller, the iATC 29C48/29C50A Feature Control CODEC/filter combos (for voice conversion), and the 82530, Serial Communications Controller.

ISP188 is based on the 8086 architecture and can be used with the 80188/186, 80286 and 80386 microprocessors. The software source modules are written in "C" language using the Microsoft compiler. The modules are well defined to permit integration with customer supplied software.

ISP188 supports the recommendations set forth by CCITT for the datalink and network layers (I.440, I.441, and I.450, I.451) of the OSI Reference Model. The iATC 29C53 supports the physical layer (I.430). Combined, the 29C53 and ISP188 implement the standards now in place for the "S" reference point (layers 1, 2, and 3).

ISP188 package includes a software license for incorporation into OEM products and a copy of the source code is provided on a 5¼ inch floppy disk.

A PC co-processor board is available that contains the Intel components and other necessary hardware to use ISP188 to establish a voice call and simultaneous circuit switched data or PC to PC file transfer through ISDN switched access. In addition, demonstration routines are included to show examples of the same capabilities in a back to back PC environment (local, no switch involved). Additionally, a debug port is provided on the board and supported by the software so that diagnostic messages can be enabled and sent to a printer or terminal over an RS 232-C connection.

The PC card and software combined can be used for ISDN hardware or software development or be included in an OEM product. Typical applications for ISP188 include digital telephones, feature telephones, integrated voice/data terminals (IVDT), and terminal adapters.

*Dale, Gesek, McWilliams & Sheridan, Inc.

Figure 1. OSI Reference Model
FUNCTIONAL DESCRIPTION

ISP188 is designed to establish, maintain, and tear-down voice and circuit switched data links on the basic rate 2B+D “S” bus interface. The software package provides the “out of band” “D” channel signalling software for call control. This includes layer 2 LAPD, and layer 3 for basic voice services and circuit switch data calls on the B channels.

The software was designed and tested with a PC plug-in card which functions as a communication coprocessor in an IBM PC, XT, or AT host environment. The co-processor board contains the Intel ISDN components for which the software was designed. All of the software modules run on the board, while some support software runs under MS-DOS. A Shared Memory Interface (SM1 and SM2) which is contained on the co-processor is used for communication between the host environment and the board. Figure 2 outlines the software modules, where they run and devices for which drivers are supplied.

ISP188 is configured to be compatible with the AT&T #5ESS Central Office switch and is compatible with the 5E4.1 Generic Basic Rate Interface specification. A PC/XT/AT host configured with the coprocessor board and loaded with ISP188 may be connected to an AT&T #5ESS Basic Rate Interface.

Layer 1, ISDN Hardware Device Drivers

The device drivers allow the software to interface with the hardware components. The following hardware device drivers are provided:

- D-channel and link control (29C53)
- B-channel control (82530, 29C48/C50A)
- B-channel data transfer (82530)
- Voice analog control (29C48/C50), DTMF generation, alerting tones
- PC bus interface via 8K x 8 FIFO (SM1 & SM2)

The D-channel and link control driver interfaces the software to the 29C53 Digital Loop Controller. It supports the activation, deactivation, error detection and D-channel data transfer functions on the “S” bus. The device driver supports the following primitives as its interface to layer 2 of the ISDN D-channel:

- PH-DATA (Transmit and receive packets from layer 2)
- PH-ACTIVATE (Activate or sense activation of the S bus link)
- PH-DEACTIVATE (Deactivate or sense the deactivation of the S bus link)
- PH-MPH_ERROR (Detect and report link level error conditions to the management entity)

The B-channel control driver is used to control access to the B-channels. Access is through the following procedures:

- Enable/Disable voice to B1 or B2 (29C48/C50A)
- Enable/Disable 82530 channel A to B1 or B2
- Enable/Disable 8250 channel B to B1 or B2
- Enable/Disable 82530 channel A to 29C48/C50A—this permits tone generation to speaker or ear piece
Figure 2. ISP188 Block Diagram
Additionally, the B-channel control driver is used to force B-channel data to mark one or zeros when no device is attached. It also is used to select hardware rate adaption or data inversion of the B-channels.

The B-channel Data Transfer driver supports the transmission of LAPB/LAPD HDLC packets on either B-channel via the 82530 using DMA data transfer at 64 Kb/s or interrupt driven at 16 Kb/s. The interfaces are similar to the D-channel and link control device drivers and use the same routines.

The Voice Analog Control driver is used to control the voice analog section of the co-processor board. It is used to initialize the 29C48/C50A, set volume levels, generate and send tones to speaker and ear piece, and detect status of the ON/OFF hook inputs.

**ISDN Layer 2 (LAPD) Module**

The layer 2 of the ISDN D-channel signaling implementation is compatible with CCITT Recommendation I.441. The design supports multiple interfaces to the data link layers of the D-channel and B-channels. Each interface maintains its own state tables, data areas and input/output physical interface.

The layer 2 software supports Terminal Endpoint Identifier (TEI) assignment, single and multi-frame operation, and modulo 8 and 128 operation.

Additionally, layer 2 supports an interface to packet layer X.25 data on the B-channel using the LAPD procedures.

The interface to layer 1 supports the CCITT recommended primitives.

The interface to layer 3 supports the CCITT recommendation with the following primitives implemented:

- DL-UNIT DATA REQUEST/INDICATION—unacknowledged information transfer
- DL-DATA REQUEST/INDICATION—acknowledged information transfer
- DL-ESTABLISH REQUEST/INDICATION—establish single or multi-frame operation
- DL-RELEASE REQUEST/INDICATION—terminate single or multi-frame operation
- MDL-ASSIGN REQUEST/INDICATION—obtain TEI assignment
- MDL-REMOVE REQUEST—remove previously assigned TEI

**ISDN Layer 3 D-Channel Signalling Module**

The layer 3 implementation conforms to the CCITT Recommendation I.451. The implementation also supports two call references, one per B-channel. This implementation supports the following network functions:

- Initiate a call on either B-channel (voice or data)
- Answer a call on either B-channel
- Terminate a call on either B-channel (caller or called party hang up)

Each B-channel will operate independently of the other. That is, one B-channel can be terminated while the other B-channel initiates or continues a call.

The layer 3 to layer 2 interface conforms to the CCITT recommended primitives and layer 3 conforms to the call state/event table definitions of I.451.

The implementation provides for the easy expansion of layer 3 to support additional network facilities such as call waiting, call transfer, etc.

**Connection Manager Module**

The Connection Manager Module interprets call request information and manages the connection and disconnection of the voice and data between the PC source and the B-channels. The B-channels can be connected for voice or HDLC framed data phase packets. This module is defined to support the connection of several data sources that could exist in future development.

**Management Module**

The Management Module provides several functions for the ISDN software system. Its primary function is to detect, indicate and respond to error conditions detected in the ISDN system software and hardware. A secondary function performed by the management module is the initialization of the LSI hardware chips in the ISDN system.

**HDLC Frame Interface Module**

This module interfaces through the PC bus via the shared memory interface (SM1 and SM2) to a PC driver interface that supports the transfer of data in variable length buffers up to 512 bytes. This data is then sent to the HDLC driver where it is framed with...
flags and CRC and sent/received on either B channel using DMA or interrupt driven data transfer mode. The interface can also specify rate adaption to B channel intermediate rates of 56, 48, 32, 16 or 8 Kb/s. The HDLC data stream may also be inverted.

Packet Data Interface Module

This module interfaces through the PC bus via SM1 and SM2 to a PC driver that supports the transfer of X.25 data phase packets. Layer 3 data phase X.25 packets are handled from the PC and sent as LAPB or LAPD packets at 64 Kb/s. Other transmission bit rates may also be specified.

The Kernel

A machine readable version of a real-time multi-processing executive is contained in the software package. The kernel runs on the coprocessor and supports service calls for the following:

- Dynamic Memory Management
- Timer Support
- Semaphores
- Inter-Process Communication
- Queue Control
- Interrupt to Semaphore Mapping

The software is built into a collection of "processes" with the kernel using the "system builder" software which is provided.

Complete documentation is available on the kernel and is listed below.

DEVELOPMENT ENVIRONMENT

All software is developed in "C" language, with the exception of the low-level hardware device drivers and interrupt handlers. These are implemented in assembly language using the Microsoft assembler. The Microsoft "C" compiler version 4.0 or later is used.

The software is supported by the environment provided by the real-time multi-tasking kernel.

The software package was designed and tested using the PC coprocessor board in an IBM PC (XT, AT). The board can be used for development of custom upper layer software or in production to upgrade PCs to ISDN workstations (additional software required for specific applications).

The PC co-processor development board contains the following major functional elements:

- iAPX 188 microprocessor
- ISDN "S" bus interface with full support for D and B-channels (Intel's 29C53 Digital Loop Controller)
- Voice interface for hand or headset (includes Intel's 29C48 or 29C50A Feature Control Combo)
- Interface to the PC bus (includes Intel's 82530 Serial Communications Controller and 8256 UART)
- 512K to 1 megabyte of RAM (includes Intel's 8208 DRAM Controller)

The 82530 is used as the B-channel controller. Since the 80188 provides only two DMA channels, full duplex DMA transmission (transmit and receive) can be supported for data transfer via the 82530 on only one B-channel. Interrupt driven data transfer on the second B-channel is possible when voice has not been selected. Independent rate adaption is supported on each of the B-channels when they are attached to the 82530. The following rate adaption schemes are supported by the hardware:

- 64 Kb/s (clear channel no rate adaption)
- 56 Kb/s (V.110 single stage)
- 48 Kb/s (V.110 single stage)
- 32 Kb/s (V.110 second stage)
- 16 Kb/s (V.110 second stage)
- 8 Kb/s (V.110 second stage)

HDLC framing and flag stuffing (X.31-DMI mode 2)

The 82530 SCC supports the following signaling protocols on the B-channels:

- Async
- IBM BSC
- SNA SDLC
- HDLC (LAPD/LAPB)

Complete documentation is available on the coprocessor plug-in card and is listed below.
DEMONSTRATION SOFTWARE

Three demo packages are included in the software package. The first is for voice calls run on the PC using MS-DOS and the PC keypad for dialing. The package will demonstrate establishing a call, answering a call, and call termination. The second package demonstrates sending and receiving a PC file using HDLC data frame transport. The third demo permits keyboard data on each PC to be transmitted on the B-channel at 64 Kb/s to the other PC's screen. Voice and data call demos can be overlapped.

ORDERING INFORMATION

ISP188 is reference sold by Intel. Software licensing and sales are conducted directly with DGM&S, Inc. and can be contacted at the address below. Complete development and support packages are available.

Marketing Department
DGM&S, Inc., Communication Technologies
1025 Briggs Road
Mt. Laurel, New Jersey 08054
(609) 866-1212

COMPATIBILITY

ISP188 is being tested for compatibility in several switch environments as well as ISDN field trails. For the latest information on compatibility testing, please contact DGM&S, Inc.

ADDITIONAL LITERATURE

Related literature is available from DGM&S, Inc. on the following subjects:
- IBM PC ISDN "S" Co-processor Adapter (GEN-071-2)
- DGM&S Kernel including The Process Builder (GEN-048)
- DGM&S Shared Memory Interface—SM1 (GEN-047)
- PC Application Platform (GEN-093)
- ISDN ISP188 Hardware—Software Description (GEN-092-2)
PC53 ISDN BOARD

- IBM, PC, XT, AT Compatible Intelligent Communications Card**
- 2B + D ISDN Basic Rate Interface Supported
- Complemented with ISP188 ISDN Software Package
- On Board 80188 for ISDN Protocol Processing
- 29C53 Digital Loop Controller Provides ISDN Physical Layer Access
- 82530 for B-Channel Protocol
- Voice/Handset Interface Provided via 29C48
- Auxiliary Serial Port for Debug or Data
- Switch Hook Detect
- 512 Kbyte RAM Expandable to 1 Mbyte on Board
- Supports Data Transfers over Any or All Three Channels (Two B-Channels, One D-Channel)

The PC53 ISDN Board is an intelligent communications card for connection to four-wire ISDN networks conforming to the CCITT I series recommendations. The board is a full size card and can be used in the IBM PC/XT/AT** or compatible computer system. The PC53 uses Intel's VLSI technology to provide a self-contained co-processor for ISDN communications applications in a PC environment. The board includes an 80188 CPU, 512 Kbyte dynamic RAM (expandable to 1 MB), B-channel protocol controller, ISDN "S" interface, and telephone handset interface, and can be used as a communications adapter card, or for ISDN software development. The on-board 80188 processor, with appropriate software, can perform the call processing and signaling required for the ISDN network, off loading this task from the host CPU. When installed with the ISP188 software package, the system is compatible with the AT&T #5ESS* ISDN switch containing the 5E4.1 Generic Program.

Figure 1. PC53 Board

**IBM, PC, XT and AT are registered trademarks of International Business Machines.

*#5ESS is a trademark of AT&T.
FUNCTIONAL DESCRIPTION

The PC53 ISDN co-processor implements the hardware functions required to support the CCITT I-series "S" interface. The board is designed to run as a slave processor board in the PC host system (Figure 1). The co-processor design relieves the host processor of much of the communication functions. The board uses no system memory address space, but rather one block of eight I/O ports with switch selectable base address.

As an intelligent integrated voice and data adapter, the PC53 permits basic access to ISDN facilities over the 4-wire "S" bus interface, providing access to the two full duplex 64 Kb/s B-channels (B1 and B2) and to the 16 Kb/s D-channel for signaling or packet data. This is referred to as the 2B+D interface.

Physical access to the two B-channels and the D-channel is provided with the 29C53 Digital Loop Controller. This device also provides hardware support for the D-channel signaling protocol.

The PC53 is implemented using the 80188 microprocessor, providing a high integration, high performance design. Since the 80188 is software compatible with the PC host processor, application software can be developed on the host computer. Hardware and software support is included for downloading code from the host system to the onboard RAM of the PC53 for execution.

The PC53 can be expanded to one megabyte of RAM for use as program and data memory.
(512 Kbytes provided). In addition, 8 Kbytes of dual-port RAM is used for host to PC53 communications. The PC53 contains no ROM; all software is downloaded from the main processor. The PC processor downloads software and messages to the PC53 via the dual-port RAM and controls the PC53 via a set of control lines through an I/O port.

The design permits the PC53 to establish, maintain and clear voice and/or data calls via D-channel signaling. All signaling is supported on the PC53 with the 80188 microprocessor and the D-channel processor in the 29C53 ISDN chip.

The PC53 includes an interface to a handset or headset speaker/microphone and provides a 29C48 codec/filter for the conversion between analog voice and the digital voice (Figure 2). The digitized voice may be routed over either B-channel.

Switch hook detection capabilities are provided by using a four-pin connector on the outer edge of the board. This can be used to connect to a handset cradle for detection of on/off hook condition.

The PC53 ISDN board also supports data transfer on either or both B-channels. A multiprotocol communication adapter, the 82530, is used to route data to or from either B-channel. This device supports both byte and bit synchronous protocols at 64 Kb/s for the B-channels. The PC53 provides full duplex DMA support for one of the B-channels when used in SDLC or HDLC mode, providing a 64 Kb/s transfer rate. The other channel is interrupt driven, with throughput depending upon other tasks being handled by the 80188.

The PC53 also includes an auxiliary serial communication port which can be used as a protocol or software debug monitor port, or as an auxiliary data channel. In the latter case, the PC53 could be used to interface to a communication port on the PC or a terminal, and bit rate adapt the asynchronous data to either of the 64 Kb/s B-channels, or packetize the data and route it to the 16 Kb/s D-channel.

Functionally, the PC53 ISDN board is divided into the co-processor subsystem, the PC interface, dual-port subsystem, ISDN interface, and an analog voice interface (Figure 2). The remaining sections of this document describe the subsystems in more detail.

**CO-PROCESSOR SUBSYSTEM**

The ISDN co-processor subsystem is driven by the 80188 microprocessor which runs at an 8.192 MHz clock rate. The processor has many programmable internal functional blocks with operation dependent on the configuration software. These include:

- 3 programmable timers
- Programmable interrupt controller
- 13 programmable chip select lines
- 2 channel DMA controller

The hardware design takes advantage of these blocks to provide a high integration/high performance communications card. The 80188 is run without wait states for most operations except simultaneous dual port accesses and accesses to the 82530 Serial Communications Controller. Thus, most operations not involving the dual port RAM or 82530 will execute in 4 clock cycles, or 488 ns.

**Programmable Chip Select Lines**

The programmable chip select lines reduce the amount of external decode logic required. The actual address that the line responds to is a function of the software. The programmable chip select lines are connected as follows:

- **UCS:** Selects dual port RAM/disables dynamic RAM
- **MCS0–3:** Not used
- **LCS:** Not used
- **PCS0:** Selects 29C53 ISDN transceiver
- **PCS1:** B1 channel control
- **PCS2:** Selects 8256 UART
- **PCS3:** B2 channel control
- **PCS4:** Selects 82530 SCC
- **PCS5:** Latched A1
- **PCS6:** Latched A2
Interrupts

The 80188 programmable interrupt controller is used to handle the system interrupts. The actual interrupt vectors and priorities are determined by the software. The interrupt pins are wired as follows:

<table>
<thead>
<tr>
<th>Pin</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NMI</td>
<td>PC processor or watchdog timer</td>
</tr>
<tr>
<td>INT0</td>
<td>29C53</td>
</tr>
<tr>
<td>INT1</td>
<td>82530</td>
</tr>
<tr>
<td>INT2</td>
<td>PC processor</td>
</tr>
<tr>
<td>INT3</td>
<td>8256</td>
</tr>
</tbody>
</table>

Timers

The 80188 has three internal programmable timers. TMRO can be programmed as a real time clock for the co-processor. TMR2 is then used as a prescaler. TMR1 has been intended for use as a watchdog timer. The TMR1OUT pin is ORed with the non-maskable interrupt signal from the PC (set by I/O port) and fed back to the NMI pin of the 80188. Thus the timer will produce an NMI to the co-processor upon time-out. Software can determine the source of the NMI by reading the timer status.

Dynamic RAM

Up to one megabyte of dynamic RAM may be installed on the board. The RAM is divided into four banks of 256 Kbytes each. RAM with an access time of 150 ns or less is used to avoid wait states. This RAM is used both for program memory and for data buffers.

All access to the dynamic RAM is controlled by the Intel 82C08 dynamic RAM controller chip. The chip handles all RAM refresh and also decodes memory accesses to the RAM.

While one megabyte of dynamic RAM may be installed on the board, the top 8 Kbytes of the address space is used by the dual port RAM. The RAM controller is disabled by the programmable chip select line UCS when the dual port RAM is accessed. The RAM controller is also disabled during an I/O operation.

MUART (Multi-Function Universal Asynchronous Receiver-Transmitter)

The Intel 8256 MUART is a multi-function chip containing:

- Programmable serial asynchronous communications interface
- On-board baud rate generator
- Five 8-bit programmable timer/counters; four can be cascaded to form two 16-bit timer/counters
- Two programmable parallel I/O ports
- Eight level interrupt controller
- Programmable system clock

The MUART appears to the co-processor as 16 read/write registers selected by programmable chip select line PCS2.

The MUART parallel ports are used to control and monitor various system functions. These include:

- Interrupt requests to/from PC host
- Voice channel volume control
- DTMF tone routing
- Microphone muting

The MUART is also used to provide an asynchronous serial port. This port is intended to support software debug and software or data link diagnostics. It may also be used as an interface to an asynchronous terminal for rate adaption to one of the ISDN channels. Data rates from 50 to 19.2 kb/s can be supported.

PC HOST INTERFACE

The PC53 interfaces with the host processor through the 62-pin card edge-connector. The PC53 board appears to the main processor as a bank of 8 read/write I/O locations with switch selectable base address. Thus, the PC53 does not use any of the available PC memory address space.

The 8 I/O ports allow the host processor to control and monitor various PC53 board functions, and to pass data to and from the board. The following functions are supported:
a. PC53 board control

The main processor can directly control the co-processor by writing to the co-processor control latch. This is a 6-bit port which directly drives several of the co-processor control lines. The latch is cleared by the PC bus reset line. The co-processor reset bit in the slave control latch is active low so that the co-processor is held in the reset condition until released by the host processor. The slave control latch includes the following functions:

- Reset PC53 board
- Send NMI to 80188
- Send INT2 request to 80188
- Enable interrupts from 80188 to PC bus
- Enable FIFO address register 1 increment
- Enable FIFO address register 2 increment
- Sense IRQ latch status (interrupt request to PC)

b. Dual-port FIFO control

The PC processor may transfer data to or from the dual port RAM (FIFO) using standard I/O instructions.

- Load FIFO address register 1 or 2
- Clear FIFO address register 1 or 2
- Read or write to high byte of FIFO (no change of FIFO addresses)
- Read or write FIFO via address register 1 or 2 and optionally increment address register

DUAL-PORT SUBSYSTEM

To the ISDN 80188 processor, the dual-port RAM appears as an 8 Kbyte block of RAM addressed at the top 8K of memory. When the dual-port RAM is selected, the dynamic RAM controller is disabled.

To the PC processor, the dual-port RAM appears as two I/O locations configured as push/pop queues. The actual address accessed is controlled by two 12-bit queue pointers, which can be set by the processor as described in the PC Host Interface section.

Either processor may access the dual-port RAM with no wait states if there is no contention. The dual-port logic resolves contention by giving access to one processor and holding the other processor by asserting a not ready signal.

The dual-port RAM is used to pass data to and from the PC53 for transmission/reception of data, to pass commands, and to download code to the PC53 board during software installation.

ISDN INTERFACE SUBSYSTEM

The ISDN interface consists of the Intel 29C53 “S” interface transceiver, the Intel 82530 Serial Communications Controller, and the 29C48 codec/filter.

Physical Layer Interface

The 29C53 Digital Loop Controller provides the layer one connection to the “S” interface. It provides access to the two 64 Kb/s B-channels in either serial or parallel fashion. Clear access (no protocol processing) is provided for the B-channels. The 29C53 also provides access to the 16 Kb/s D-channel, and provides layer 2 packet framing functions in hardware, as well as data buffering using FIFOs.

Line transformers and protection circuitry are included to complete the line interface. Termination resistors can be installed via jumpers if not included in the site wiring.

B-Channel Processing

The 82530 Serial Communications Controller is a dual port multiprotocol controller which is used to provide B-channel protocol support. The 82530 is interfaced to the 29C53 serial port (SLD) and can access either or both B-channels under processor control.

Hardware support also exists to bit rate adapt 8, 16, 32, 48, and 56 Kb/s data to one of the 64 Kb/s B-channels.

Voice Conversion

The 29C48 codec/filter is also connected to the SLD serial port of the 29C53, and allows one of the B-channels to be used for voice transmission. Either B-channel can be used.
**AUDIO SUBSYSTEM**

The audio subsystem provides for a handset or headset interface for voice capability. The audio components include the 29C48 codec/filter, a tone generator, digitally controlled potentiometers, and an analog switch.

The 29C48 receives and transmits digitized audio via the serial SLD interface to the 29C53. The transmit and receive gains can be controlled via the internal gain control registers of the 29C48, or by setting the digitally controlled potentiometers. The potentiometers contain non-volatile memory, so they retain their settings during power down conditions.

The tone generator is used to generate DTMF signals and to provide signaling tones to the auxiliary audio output (speaker). The DTMF tones may be routed to the handset earpiece, to the “S” interface, or to the speaker. The tone generator allows a variety of alerting, or ringing tones to be generated for signaling an incoming call to the user.

The analog switch provides for microphone and earpiece mute functions, and for DTMF tone routing.

**AVAILABLE SOFTWARE**

The PC53 ISDN board is complemented by the ISP188 ISDN software package. ISP188 is sold separately by DGM&S and includes a source copy and license. With a licensing agreement in place, no royalties are required for use of ISP188 binary code in OEM products.

ISP188 is a software package specifically designed by Intel and DGM&S for use with Intel's ISDN products. It contains modules that map to the data link layer (layer 2) and the network layer (layer 3) of the OSI model and is compatible with CCITT's recommendations I.440/441, and I.450/451 for the “S” reference point.
The 29C53 contained on the PC53 board is compatible with the physical layer (I.430 or layer 1). Combined, the 29C53 and ISP188 implement the standards now in place for the "S" reference point (layers 1, 2, and 3). Layers 4 through 7 are not specified for ISDN and will allow many users to utilize the PC53 and ISP188 to meet the ISDN standards but still differentiate products by application and custom features (Figure 4).

ISP188 contains device drivers at the physical layer for the 29C53, 29C48 and 82530. Modules at layer 2 support LAPD procedures and will allow multiple interfaces to the data link layers of the D-channel and B-channels. Furthermore, at layer 3, ISP188 supports two call references, one per B-channel. This implementation supports initiating a call on either B-channel with voice or data, answering a call on either B-channel and terminating a call on either B-channel by either party (caller or called party hang up). Each B-channel operates independently.

Complete documentation on the ISP188 is contained in a separate data sheet.

---

**Figure 4. OSI Reference Model**

- **PC53 Hardware Supported**
- **ISP188 Software Supported**
SPECIFICATIONS

HOST REQUIREMENTS
IBM PC, PC/XT, PC/AT, or compatibles

PROCESSOR
Intel 80188 operating at 8.192 MHz

S BUS CONTROLLER
Intel 29C53

CODEC/FILTER
Intel 29C48

B CHANNEL PROCESSOR
Intel 82530

TONE GENERATOR
PCD 3312

MEMORY
RAM Supplied: 512 Kbytes
Total Capacity: 1 Mbytes

INTERFACES
Accessed through end plate:
TC1 — Handset Interface
  4 way modular jack connector
  Carbon type microphone interface
TC2 — "S/T" Interface
  144 Kb/s: two 64 Kb/s B-channels, one 16 Kb/s D-channel
  CCITT 1.430 compatible
  8 way modular jack connector

J7 — Switch Hook Interface
  4 way right angle header connector

Access internal to PC:
P1 — 62-pin PC bus edge connector
J2 — Auxiliary Serial Port
  50 Kb/s to 19.2 Kb/s asynchronous
  RS232 TXD, RXD, CTS and GND signals from Intel 8256
  Additional signals:
    Two programmable inputs, one programmable output
    +5V, +12V, -12V Power Connections
    10 way header connector
J1 — SLD bus port
  SLD signals: SLD (Data), SCL (Clock), SDIR (direction)
  Other signals: 29C53 P3 and P4, 3.84 MHz clock, VCC, GND
  8 way header connector

Physical Characteristics
Width: 13.3 inches (33.78 cm)
Height: 4.2 inches (10.67 cm)

Environmental Characteristics
Temperature: 0°C to 55°C
            (32°F to 131°F)
Humidity: 5% to 90% Operating
          5% to 95% Non-Operating

Power Requirements
+5V  ± 5%  1.5A typ., 2.2A max.
+12V ± 10% 30 mA typ., 60 mA max.
-12V ± 10% 30 mA typ., 60 mA max.
IDK29C53
ISDN DEVELOPMENT KIT FOR 29C53

- Includes the Following:
  - Two PC53 ISDN Boards
  - Executable Copy of ISP188 Software Package
  - Executable Copy of NT Simulator Based on ISP188
  - Demonstration Program
  - Two Telephone Handsets and Cradles
  - All Necessary Cables

- Compatible for Use with AT&T's #5ESS* ISDN Switch
- Complete PC Based Support Package for Hardware and Software
- Used for Evaluation, Development and Prototyping of ISDN Terminal Applications
- For Use with the IBM PC, XT, AT or Compatibles**

The ISDN Development Kit for the 29C53 (IDK29C53) contains the necessary hardware and software to develop and demonstrate ISDN terminal applications. The user must provide the development environment for the kit which requires two PC, XT or AT compatible systems. One system is used as the terminal endpoint (TE) where specific applications will be developed and the second system is used to simulate the network terminator (NT) function and will respond appropriately to the TE's request for service. The minimum requirements for a development system are a hard disk, floppy disk drive, and 512 Kbytes of RAM. The minimum requirements for an NT system are a floppy disk drive, and 512 Kbytes of RAM.

IDK29C53 is intended to assist users of Intel ISDN products to investigate and develop applications in a productive and time efficient manner. The IDK29C53 supports ISDN terminal applications based on the PC53 ISDN Board, or those based on the 80188 microprocessor, 29C53 ISDN transceiver, the 29C48 codec/filter, and the 82530 serial communications controller. Additionally, those applications based on the ISP188 ISDN software package are supported.

The PC53 ISDN Board and source version of ISP188 are available for purchase and incorporation into OEM products that are developed. Typical applications for ISP188 and/or the PC53 ISDN Board include digital telephones, integrated voice/data terminals (IVDT), terminal adapters and ISDN test systems.

**IBM, PC, XT and AT are registered trademarks of International Business Machines
* #5ESS abd AT&T are trademarks of American Telephone & Telegraph
**Functional Description**

The following components are included in the IDK29C53 (Figure 1):

- 2 PC53 ISDN boards
- 1 ISP188—TE executable binary system
- 1 ISP188—NT executable binary system simulator
- 2 Carbon microphone telephone handsets & cords
- 2 Handset cradles and cords
- 1 Eight wire back to back development cable
- 2 Eight wire ISDN “S” interface cable
- 1 Serial communication cable for debug monitor
- 1 Documentation package

**PC53 ISDN Board (Co-Processor)**

The PC53 is a full length intelligent communications card used for connection to four-wire ISDN networks conforming to the CCITT I series recommendations for the ‘S’ interface.

The board is designed to run as a slave processor board in the PC host system. The co-processor design relieves the host processor of many of the communication functions. The board uses no system memory address space, but rather one block of eight I/O ports with switch selectable base address.

The PC53 ISDN co-processor is implemented using the 80188 microprocessor, providing a high integration, high performance design. Since the 80188 is software compatible with the PC host processor, application software can be developed on the host computer. Hardware support is included for downloading code from the host system to the on-board RAM of the co-processor for execution.

Physical access to the two B channels and the D channel is provided with the 29C53 Digital Loop Controller (see Figure 2). Voice conversion from analog to digital and vice versa is provided by the 29C48 Feature Control Combo (codec/filter).

---

**Figure 2. PC53 Block Diagram**

5-68
The interfaces to the board include the 62 pin card edge-connector for connection to the host system bus. Communication with the host system is via an 8 Kbyte dual port RAM (see Figure 3). There is also an eight pin ISDN "S" interface, a four pin telephone handset interface, a four pin interface for switch hook detection, and a ten pin auxiliary serial connection used to connect a terminal during the development stage.

Complete documentation on the PC53 is contained in a separate data sheet and user's manual.
SOFTWARE

An executable copy of the ISP188, ISDN software package for the 80188, is provided in the kit.

The following "C" functions are provided by the executable code.

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Connect Functions</td>
<td></td>
</tr>
<tr>
<td>isn-dial:</td>
<td>make a voice call</td>
</tr>
<tr>
<td>isn-connect:</td>
<td>establish a data connection</td>
</tr>
<tr>
<td>isn-connect-accept:</td>
<td>accept incoming data connection</td>
</tr>
<tr>
<td>isn-listen:</td>
<td>pre-accept incoming data connection</td>
</tr>
<tr>
<td>isn-voice-accept:</td>
<td>accept incoming voice connection</td>
</tr>
<tr>
<td>Disconnect Functions</td>
<td></td>
</tr>
<tr>
<td>isn-dial-disconnect:</td>
<td>disconnect voice call</td>
</tr>
<tr>
<td>isn-disconnect:</td>
<td>disconnect data connection</td>
</tr>
</tbody>
</table>

ISP188 is a software package specifically designed by Intel and DGM&S for use with Intel's ISDN components and boards. It contains modules that map to the data link layer (layer 2) and the network layer (layer 3) of the OSI model and is compatible with CCITT's recommendations I.440/441, and I.450/451 for the "S" reference point.

The 29C53 contained on the PC53 board is compatible with the physical layer (I.430 or layer 1). Combined, the 29C53 and ISP188 implement the standards now in place for the "S" reference point (layers 1, 2, and 3). Layers 4 through 7 are not specified for ISDN and will allow many users to utilize the PC53 and ISP188 to meet the ISDN standards but still differentiate products by application and custom features (see Figure 4).

ISP188 contains device drivers at the physical layer for the 29C53, 29C48 and 82530. Modules at layer 2 support LAPD procedures and will allow multiple interfaces to the data link layer of the D-channel and the B-channels. Furthermore, at layer 3, ISP188 supports two call references, one per B-channel. This implementation supports initiating a call on either B-channel with voice or data, answering a call on either B-channel and terminating a call on either B-channel by either party (caller or called party hang up). Each B-channel operates independently.

Data Transmission Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>isn-transmit:</td>
<td>transmit data</td>
</tr>
</tbody>
</table>

Data Reception Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>isn-receive-wait:</td>
<td>wait for received data</td>
</tr>
<tr>
<td>isn-receive-immediate:</td>
<td>returns received data if available</td>
</tr>
</tbody>
</table>

Rejection Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>isn-connect-reject:</td>
<td>reject an incoming connection</td>
</tr>
</tbody>
</table>

Audio Control Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>isn-audio-vol:</td>
<td>adjust audio volume</td>
</tr>
<tr>
<td>isn-mute-mic:</td>
<td>mute microphone</td>
</tr>
<tr>
<td>isn-mute-ear:</td>
<td>mute earpiece</td>
</tr>
</tbody>
</table>

Status Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>isn-status:</td>
<td>returns layer 3 state variable</td>
</tr>
<tr>
<td>isn-report-to-pc:</td>
<td>enable/disable error reporting</td>
</tr>
<tr>
<td>isn-init:</td>
<td>initialize isdn subsystem</td>
</tr>
</tbody>
</table>

Additionally, a version of ISP188 is used to simulate switch responses to service requests from the TE. The NT simulator is not used to develop applications for the NT side of the "S" interface. Rather its function is to support TE application development by providing an NT simulator against which to run TE applications. This version of ISP188 is only available in executable form and is not intended for use in OEM products.

ISP188 is fully supported by DGM&S of Mt. Laurel, New Jersey. A source copy of ISP188 that includes an OEM license is available as a separate package from DGM&S. Complete documentation on ISP188 is contained in a separate data sheet from Intel. Additional literature is available from DGM&S and their address is listed below.

TELEPHONE HANDSETS

The telephone handsets are carbon microphone type handsets with a modular plug connection. One handset is connected by a four pin modular connector to each PC53 board.
HANDSET CRADLES

The handset cradles are used to hold the handset and signal the software the switch hook status; on-hook or off-hook. When the handset is lifted from the cradle, the off-hook condition is detected and a call sequence mode is entered so that a number can be dialed from the keyboard to establish a voice call. Where there is an incoming call, the off-hook condition will stop call alerting (ringing) and allow the call request to be connected.

The cradle is connected to the PC53 board by a four pin connector and cable which is supplied.

DEMONSTRATION CABLE

The demonstration cable is used to connect the two PC53s together back to back. This allows local development and demonstration of communication between the two boards located in separate PC systems. Two of the four pairs of wires in the cable are reversed on one end so that the transmit pair from one PC53 board becomes the receive for the other board. The cable is connected to the boards by a standard eight pin modular plug using the "S" interface connection on the board.

ISDN "S" INTERFACE CABLE

The "S" interface cable is used to connect the PC53 board to the ISDN network at the wall outlet. The interface cable is four wire pairs and uses a standard eight pin modular plug.

SERIAL COMMUNICATIONS CABLE

The serial communications cable is used during the development and prototype stage to connect an external terminal to the PC53 board. This will allow diagnostic information to be displayed (or printed) while debugging new applications.

The connection on the board is a ten pin header on the top side of the board. The serial communication
cable has a ten pin header connector on one end and a DB-25 RS232 async connector on the other. Only four pins are current used.

SOFTWARE LICENSE

The binary (executable) version does not carry any license for incorporation of any part of the supplied code into OEM products. The binary (executable) versions of the software are supplied for use on the PC53 boards supplied in the kit.

Resale of any part of the ISP88 software requires a license from DGM&S. With a licensing agreement in place no royalties are required for use of ISP188 binary code in OEM products.

SUPPORT

The IDK29C53 is comprised of two major components: the PC53 ISDN boards which are sold and supported by Intel, and the executable copies of ISP188 which are sold and supported by DGM&S. The two components are combined in a kit for the convenience of the user in purchasing. Your Intel representative may be contacted for information regarding support for either the software or hardware portions of the kit. Software support has been established by DGM&S in North America, Japan and Europe.

COMPATIBILITY

The PC53 ISDN board and the ISP188 software package have jointly been tested by AT&T and found to be fully compatible to AT&T's #5ESS* 5E4.1 generic program. Further compatibility testing for other major central office switches and revised versions of the #5ESS are planned. For the latest compatibility information please contact DGM&S or your Intel representative.

DOCUMENTATION

The following documentation is included with the IDK29C53:

IDK29C53 data sheet
PC53 data sheet
PC53 users manual
ISP188 data sheet
ISP188 binary users manual

ORDERING INFORMATION

<table>
<thead>
<tr>
<th>Part Number</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IDK29C53</td>
<td>29C53 ISDN Development Kit including manual</td>
</tr>
</tbody>
</table>

RELATED PRODUCTS

<table>
<thead>
<tr>
<th>Part Number</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PC53</td>
<td>PC53 ISDN Board</td>
</tr>
<tr>
<td>TEK29C53</td>
<td>29C53 Terminal Evaluation Kit including manual</td>
</tr>
<tr>
<td>LEK29C53</td>
<td>29C53 Linecard Evaluation Kit including manual</td>
</tr>
</tbody>
</table>

Software Products Available from DGM&S

<table>
<thead>
<tr>
<th>Part Number</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ISDNB-TE</td>
<td>ISP188 User Binary (Terminal Side)</td>
</tr>
<tr>
<td>ISDNB-NT</td>
<td>ISP188 User Binary (Network Side)</td>
</tr>
<tr>
<td>ISDNS-PC</td>
<td>ISP188 Source with OEM license</td>
</tr>
</tbody>
</table>

SOFTWARE SUPPORT CONTACT

DGM&S, Inc., Communication Technologies
1025 Briggs Road
Mt. Laurel, New Jersey 08054
(609) 866-1212 Ext. 188
LEK29C53
29C53 LINE CARD EVALUATION KIT

- Single Board Evaluation Kit Supporting Four 29C53 Transceivers
- On-Board Monitor for Access of 29C53 from a Terminal
- Large User Breadboard Area
- Comprehensive User's Manual
- Line Interface Provided Including Pulse Transformer

The LEK29C53 comes fully assembled, and contains the components necessary for the evaluation of the 29C53 in linecard applications. The kit allows evaluation of point-to-point as well as multipoint operation of the 29C53 when used with the 29C53 Terminal Evaluation Kit (TEK29C53). A monitor program, included in a preprogrammed ROM, allows the user to access the registers of the 29C53 transceivers, as well as those of the 2952 linecard controller, using simple English commands. Access to the monitor is via an RS232C compatible link, allowing any start-stop terminal or computer system with such a physical interface to be used with the LEK29C53. The LEK29C53 includes sockets for four 29C53 transceivers. Two are supplied with the kit. This highly flexible kit has been designed to provide a rapid introduction to the 29C53, thereby leading to shorter development time and increased productivity.

Figure 1. 29C53 Linecard Evaluation Kit
FUNCTIONAL DESCRIPTION

Overview
The LEK29C53 is a single board evaluation kit which contains a linecard controller, 8031 microcontroller, ROM including monitor, user RAM, clock generation circuitry, and sockets for four 29C53 transceivers (two devices supplied). In addition, a large breadboard area is provided for implementation of the loop interface, and any other custom circuits. The LEK29C53 can be used with the TEK29C53 for full evaluation of the 29C53 in point-to-point and multipoint applications. The LEK29C53 can also be operated independently, with two of the four transceiver sockets configurable as loop slaves, allowing evaluation of S interface performance with just one board. A block diagram of the LEK29C53 is given in Figure 1.

Software
A system monitor is provided in the preprogrammed EPROM. Simple commands allow access to the registers of the 29C53. The monitor also provides for the down loading of 8031 HEX code to the onboard RAM, which can be operated as program memory for testing user code. A sample program is included in EPROM and can be run by command from the terminal interface.

User Interface
The serial port on the 8031 microcontroller is used to establish a serial communication link to a terminal for interaction with the onboard monitor program. This interface can be modified by changing jumpers to support standard DTE or DCE configurations.

Documentation
The LEK29C53 User's Manual provides a hardware description, description of monitor commands, circuit diagram, monitor listing and sample program listing. A 29C53 Reference Manual is also included with the kit.

SPECIFICATIONS

Control Processor:
Intel 8031 microcontroller.
12 MHz clock rate.

Memory
ROM - Socket accepts JEDEC pinouts to 32K X 8.
RAM - Socket will accept 8K X 8 static RAM (provided), or JEDEC pinouts to 32K X 8 for ROM.

Clock Generation:
Selectable 2.048 MHz or 4.096 MHz system clock.
Provision for external system clock.
3.84 MHz clock for 29C53.

Terminal Interface:
25 pin D type connector.
300 to 4800 Baud.

Digital Loop Controller (29C53):
Sockets for four Intel iATC 29C53 transceivers.
All four configured as loop masters, or two as loop masters, two as loop slaves.

Physical and Electrical Characteristics:
Width: 12 in.(30 cm)
Height: 1 5/8 in.(4 cm)
Depth: 7 in.(18 cm)
Power requirements: 5V ± 5% @ 2.5 A
± 12V generated onboard by a DC to DC converter
Operating temperature: 10 to 40° C

ORDERING INFORMATION
29C53 Linecard Evaluation Kit, including manual — LEK29C53

Related Products:
29C53 Terminal Evaluation Kit, including manual — TEK29C53
TEK29C53
29C53 TERMINAL EVALUATION KIT

- Single Board Supports Evaluation of 29C53 Transceiver in Terminal and Terminal Adapter Applications
- Onboard Monitor for Access of 29C53 from a Terminal
- Line Interface Provided Including Pulse Transformer

The TEK29C53 comes fully assembled, and contains the components necessary for the evaluation of the 29C53 in terminal and terminal adapter applications. The kit allows evaluation of point-to-point as well as multipoint operation of the 29C53 when used with the 29C53 Linecard Evaluation Kit (LEK29C53). Alternatively, the TEK29C53 can be configured to communicate directly with other TEK29C53 kits. A programmable combo is supplied with the kit for evaluation of applications supporting voice. A monitor program, included in a preprogrammed EPROM, allows the user to access the registers of the 29C53 transceiver using simple English commands. Access to the monitor is via an RS232C compatible link, allowing any start-stop terminal or computer system with such a physical interface to be used with the TEK29C53. This highly flexible ISDN terminal evaluation kit has been designed to provide a rapid introduction to the 29C53, thereby leading to shorter development time and increased productivity.

* Sockets are available for both the 29C48 and the 29C50A

Figure 1. 29C53 Terminal Evaluation Kit

270236–1

September 1988
Order Number: 270236-001

5-75
FUNCTIONAL DESCRIPTION

Overview

The TEK29C53 is a single board evaluation kit which contains the 29C53 transceiver, 8031 microcontroller, EPROM including monitor, user RAM, clock generation circuitry, and 82530 serial communications controller. In addition, a large breadboard area is provided for implementation of the loop interface, phone interface, or any other customer circuits. A programmable SLD combo, interfaced through the SLD port of the 29C53, provides access to one of the B channels of the S interface for applications supporting voice. The 82530 communications controller provides access to the second B channel, also via the SLD port of the 29C53, for packet data transfers. The second channel of the 82530 can be used as an interface to synchronous or asynchronous terminals. For full evaluation of the 29C53 in point-to-point and multipoint applications, the TEK29C53 Linecard Evaluation Kit can be used with the TEK29C53, or the TEK29C53 can itself be configured as a loop master for communication with one or more other TEK29C53 boards. A block diagram of the TEK29C53 is given in Figure 1.

Software

A system monitor is provided in the preprogrammed EPROM. Simple commands allow access to the registers of the 29C53. The 82530 can also be programmed using the monitor commands. Additionally, the monitor program provides for the downloading of 8031 HEX code to the onboard RAM, which can be operated as program memory for testing user code. A sample program is included in EPROM and can be run by command from the terminal interface.

User Interface

The serial port on the 8031 microcontroller is used to establish a serial communication link to a terminal for interaction with the onboard monitor program. This interface can be modified by changing jumpers to support standard DTE or DCE configurations.

Documentation

The TEK29C53 User's Manual provides a hardware description, description of monitor commands, circuit diagram, monitor listing and sample program listing. A 29C53 Reference Manual is also included with the kit.

SPECIFICATIONS

Control Processor:
Intel 8031 microcontroller. 12 MHz clock rate.

Memory:
ROM - Socket accepts JEDEC pinouts to 32K X 8. RAM - Socket will accept 8K X 8 static RAM (provided), or JEDEC pinouts to 32K X 8 for ROM.

Clock Generation:
3.84 MHz clock for 29C53.

Terminal Interface for Monitor:
25 pin D type connector. 300 to 4800 Baud.

Serial Data Port:
25 pin D type connector, user configurable.

Digital Loop Controller:
29C53, programmed operation in master or slave mode.

Serial Communications Controller:
Intel 82530 supporting two serial channels, asynchronous or synchronous.
Feature Control Combo

Intel 29C48 with additional socket available for the Intel 29C50A.

Physical and Electrical Characteristics:

Width: 12 in.(30 cm)
Height: 2 in.(5 cm)
Depth: 7 in.(18 cm)

Power requirements: 5V ± 5% @ 2.5 A
- 5V, ±12V generated on-board by a DC to DC converter

Operating temperature: 10 to 40° C

ORDERING INFORMATION

29C53 Terminal Evaluation Kit, including manual — TEK29C53

Related Products:

29C53 Linecard Evaluation Kit, including manual — LEK29C53
29C53 Transceiver Line Interfacing

JAGTINDER S. BOLARIA
TELECOM PRODUCT MARKETING

Order Number: 270209-003
INTRODUCTION

Presently, the majority of the transmission from the telephone to the Central Switching system is analog. For this purpose the circuitry interfacing to the twisted pair line is optimized to operate between 300 and 3400 Hz. The essential line interface functions consist of isolation, over voltage protection, signaling, power feeding and a ringing signal insertion. With the advent of ISDN (Integrated Services Digital Network) these functions have to be reassessed.

ISDN is implemented with digital transmission from the subscriber to the switch, which in turn offers the user various data services in addition to the voice service. CCITT has various recommendations for the implementation of the ISDN network. Of these, I.430 details the basic rate access i.e. the physical communications between a terminal and the first level of switching. For I.430, Intel offers a transceiver which is capable of operating at either end of the loop, namely the 29C53.

The 29C53 is a four wire (two for transmit and two for receive) transceiver operating over the "S" loop. The data transmitted by the 29C53 at the switch and the terminal is at a rate of 192 kb/s; the effective data throughput is 144 kb/s. This data consists of two bearer channels of 64 kb/s each (B1 + B2) and a 16 kb/s D channel. The 29C53, additionally, incorporates some protocol processing for the D channel. This transceiver has four interfaces, namely the microprocessor port, a general purpose I/O port, the SLD port and the "S" loop interface. It is the loop interface requirements that are addressed by this application note.

This note will analyze the line interface requirement at both the line card and the terminal, and will offer general implementations. These implementations will address power feeding, the protection circuitry, the line transformers and power extraction. Throughout this brief, the approach has been to present various alternate concepts which may assist the designer in addressing a specific application.

LINE INTERFACE

Both at the line card and the terminal, there is a need to provide isolation for the circuitry from the line itself. As well as isolation, it is also necessary to protect the equipment from any overvoltage conditions on the line. Additionally the system may be designed to provide phantom power feeding i.e. the switching system delivers power to the terminal over the "S" loop. Unlike its analog counterpart the digital line card does not need to send a ringing signal owing to the fact that all signaling is accommodated via the D channel.

POWER FEEDING

Figure 1 shows the CCITT recommended technique of phantom power feeding as described in section 9 of I.430. The current splits evenly between the two secondary windings. This in turn produces equal and opposite fluxes in the transformer, that cancel each other out, thus preventing the core from saturating. The equality of the fluxes in the secondary will depend on the longitudinal balance of the transformer and the transmission line.

The scheme shown on Figure 1 may be wasteful of power when feeding short lines. One way around this would be to have a constant current feed, which will make the power consumption independent of the length of line. Figure 2 shows such an implementation.
One way of reducing the power dissipation over the loop is to provide a variable voltage source, instead of the traditional fixed voltage. This can be accomplished by using a DC to DC converter, or a switching regulator. The feedback circuit of the switching regulator can be used to ensure that the regulator provides just enough voltage to maintain a pre-defined feed current down any length of line. The DC to DC converter can have a built in threshold detector, which would be used to release the line in case excessive currents are being drawn.

In the event of mains power loss, it is often required to maintain a minimal voice service powered off the line. Figure 3 shows the block diagram of a digital telephone, illustrating the necessary components required to maintain a voice service.

![Figure 3. Digital Telephone](29C48 29C53 80CS1)

The 80C51 is a low power microcontroller while the 29C48 is an SLD compatible combo (codec and filter). The gains through the 29C48 can be set externally or programmed by the microcontroller via the SLD interface. The 29C48 is designed to allow insertion of sidetone and DTMF (dual tone multi-frequency); both these features are presently used to provide feedback to the user.

**PROTECTION**

Next, let us examine the question of protection. A telecommunication system comprises subscribers linked together through the cable plant and a switching network. The cable plant consists of multiple pairs of transmission lines, either suspended on poles, or buried in the earth. In either case, transient energy can be coupled from lightning (or other electromagnetic events) and conducted to the switch or the terminal. The other major source of transient energy is the commercial AC power system, where high currents that accompany faults can induce overvoltage in the lines, or the power lines can fall and make contact with the telephone lines. The latter is sometimes referred to as a mains or power cross.

It is generally agreed, as shown in Figure 4, that two or more levels of protection are required. The primary protector is usually placed on the line at a distance greater than 25m from the line card. The impedance of the line will ensure that the primary protector will operate first and the secondary protector will not be exposed to the full surge. If the primary protector is to be placed closer to the secondary, then a small resistor can be inserted in series with the line between the primary and the secondary protector (1). A 5Ω 3W resistor or a positive temperature coefficient resistor may be used. During a surge, the voltage drop across the resistor will increase allowing the voltage across the primary protector to build up thus driving it to conduction.

The primary protection can be a gas discharge tube, such as the General Instrument three terminal PMT3-(310). These devices consist of spaced metallic gaps enclosed in a combination of gases at low pressure. In the event of a surge, the gap breaks down, diverting the transient and thus rerouting the energy. These devices can be operated a number of times and present a capacitance of less than 5 pF. Since the templates in Figures 10 and 11 of I.430 specify a low output capacitance for the terminal and the network terminator, the low output capacitance feature of the gas discharge tube makes it ideal for ISDN i.e. it will have a minimal effect on the line drivers.

The secondary protection can be provided by Schottky diodes chosen for the low voltage drop and capacitance across them. The diodes are placed between the power supplies and the loop interface pins on the 29C53, thus forming a diode bridge across the line. This will ensure that the voltage on these pins does not exceed the power supplies by more than approximately 300 mV, thus fulfilling the specification that the voltage on any pin...
Protection may not exceed the power supply by more than 500 mV. The 5V and ground connections to the diodes should be as close as possible to the 29C53 power supply pins, which in turn should be decoupled by a 0.1 μF capacitor. The capacitor serves a secondary function of bypassing surge currents. The particular diodes chosen are dependent on the expected surge current, however, BAT85 from Philips used in this application can withstand 200 mA forward current while presenting a maximum of 10 pF capacitance across it. The maximum current through the diodes can be limited by placing a resistor in series with the diodes and the transformer. The value of this resistance can be extracted from the transformer design discussion. To further limit the current to the 29C53, the series resistance can be split, with part of it on the 29C53 side of the diodes, and part of it on the transformer side of the diodes. For the receive direction it is possible to replace the diode bridge by placing a resistance in series with the 29C53 receive pins. This series resistance will limit the surge current that the 29C53 is exposed to. The value of this resistance is limited by the input impedance presented by the 29C53 and the loss that can be tolerated in the received signal. The receive differential input impedance of the 29C53 is 100 KΩ, hence a 10 KΩ resistor in each arm will reduce the received signal by 17%.

In case of a mains cross, the loop can be made to self recover by using thermal devices such as the positive temperature coefficient thermister (PTC). Keystone Carbon Company has a range of PTCs specific to telephone line applications that they refer to as resettable fuses. Economic considerations may make this unjustifiable in which case a fusible resistor or link may be used.

Further protection may be deemed necessary, in which case two varistors can be placed across the line close to the transformer. The varistor has a volt-current relationship similar to a diode i.e. after a specified voltage across the varistor is reached, the current through it will rise dramatically; thus clamping the voltage to the specified level. A typical varistor that may be used as a back-up protection is the GE V220MA4B. This device typically presents a 21 pF capacitance.

The ideas discussed thus far are encompassed in Figure 5 for a minimal component count protection scheme.
LINE TRANSFORMER

A transformer is used at both the terminal and the line card to provide isolation from the line. A well balanced I.430 transformer resolves the issue of DC currents since they induce self-cancelling fluxes. Generally speaking, a pulse transformer with minimum leakage inductance and self capacitance is required. The impedance templates in I.430 specify the minimum value of the inductance required at the line side. This value can be calculated to be 20 mH. A further requirement is to minimize the winding resistance, so that a minimal voltage is dropped across it. A 2.5:1 ratio transformer can be used with the 29C53 to produce the proper pulse amplitude. The transformer design discussed below can be used with the 29C53 at either the line card or the terminal. Alternatively it can be used for example purposes to aid designs.

The RM series of ferrite cores are chosen to facilitate easy winding and PCB mounting, additionally the RM series is available internationally from various vendors—Ferroxcube in the U.S. and Mullard in Europe, to name two. The RM6 core was selected to be the smallest size that accommodates wiring which does not exceed the maximum allowable DC resistance. The core material has to have a high enough permeability to allow the 20 mH inductance with a minimum number of turns hence, the Ferroxcube core material 3E2A was selected. This material has a very high inductance factor, \( A_L \). This is given by the manufacturer as the inductance (in mH) per 1000 turns.

For the core RM6PL00-3E2A

\[ A_L = 6710 \pm 25\% \]

Therefore minimum

\[ A_L = 5032 \approx 5000 \]

The number of turns, \( N_s \), required for 20 mH is given by:

\[ N_s = 10^3 \sqrt{L/A_L} \quad L \text{ - required inductance in mH} \]

\[ N_s = 70 \text{ turns} \quad \text{- assume 25 mH is required} \]

The 29C53 side winding will require 2.5 times this number of turns.

\[ N_p = 175 \text{ turns} \]

The transformer is now ready to be wound, the 32 gauge wire will just fill the RM6PCB1 bobbin. The bobbin is started by bifilar winding the 175 turns. Bifilar winding is accomplished by taking two separate pieces of wire and winding them simultaneously. The finish of one winding is then soldered to the start of the other and often, as is the case in this implementation, the point of connection of the two wires (center tap) is brought out to a pin of the transformer. The remaining ends (start and finish) now comprise the winding. The transformer is now followed by 1\( \frac{3}{4} \) layers of insulating tape. The insulating tape used was the Permacel P-256 which forms a dielectric capable of withstanding 5 KV, this serves to protect the line card and the subscribers.
from lightning induced surges. The 70 turns are then bifilar wound; this results in a well balanced transformer. The start of one winding should be connected to the finish of the other and brought out to a pin, thus creating a center tap on the line winding. The transformer is then finished with 1½ layer of insulating tape. The transformer thus designed gave satisfactory results in the lab and is characterized by the following:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Secondary inductance</td>
<td>$L_s = 26 \text{ mH}$</td>
</tr>
<tr>
<td>Secondary leakage inductance</td>
<td>$L_s = 20 \mu\text{H}$</td>
</tr>
<tr>
<td>Secondary winding resistance</td>
<td>$R_s = 1.5\Omega$</td>
</tr>
<tr>
<td>Primary winding resistance</td>
<td>$R_p = 2.7\Omega$</td>
</tr>
</tbody>
</table>

The capacitance between the two bifilar windings was measured to be 100 pF and this may be too high for certain applications. For this case the bifilar winding can be replaced by the cross winding technique shown in Figure 6a. The two windings are now wound in opposite directions, one wire is on top on the top side while the other is on top on the bottom side. This technique reduced the above mentioned capacitance to less than 50 pF.

The 29C53 has been designed to drive voltages as specified in the I.430, since the transformer presents a series resistance, some of this voltage will be dropped across it. For the transformer designed above, the overall series resistance is $(2.7 + 1.5 \times 6.25) = 12\Omega$ which will result in a 3.8% error over the allowed peak transmit signal in I.430. This is acceptable as I.430 allows a 10% error for the peak voltage. If series resistors are required to protect the Schottky diodes, their value may be calculated by having the maximum allowed peak voltage error. Note that equal value resistors should be placed on both arms of the line. If larger values of protection resistors are required, the above procedure may be repeated with a larger core. This will allow the same inductance to be achieved with a fewer turns and the larger core will make it possible to use a thicker wire. Both of these factors will contribute to reduce the winding resistance, hence a larger value diode protection resistor may be used. Alternatively, the transformer turns ratio can be decreased so that the output voltage is increased and hence more of it can be dropped across the series resistance. This in turn means that the value of this protection resistor can be increased. However, note that the 29C53 is only capable of driving loads greater than 200Ω. If a turns ratio of 1.8:1 is used then the overall series resistance can be 64Ω. This also increases the output impedance to 20Ω while transmitting a pulse. As discussed earlier this resistor can be larger on the 29C53 receive pins.

Some establishments may require further line isolation from the transformer in which case a Faraday shield can be placed in between the primary and the secondary windings. The Faraday shield can be made by wrapping 1½ layers of a copper tape (such as the permacel P-389) between the two windings. The copper tape should be insulated from the windings and should be brought out to the local ground. As well as isolating, the Faraday shield also serves to reduce the interwinding capacitance.

The transformer designed was connected up as shown in Figure 6b to measure its longitudinal balance.

![Figure 6a. Crosswinding](image)

**Figure 6a. Crosswinding**

The center tap on the primary (29C53 side) is coupled to ground via a 10 nF capacitor. In this manner longitudinal signals on the primary are bypassed to ground. Measurements produced greater than 70 dB of longitudinal signal rejection.
When designing the System board, special care should be paid to the layout. The transformer and the 29C53 should both be placed on a ground plane. The connecting tracks from the 29C53 to the transformer should be as short as possible. The two devices should be placed close to the edge where the transmission lines interface, while the high frequency logic should be placed on the opposite edge. The analog ground wiring should follow a star configuration and should have a separate isolated lead originating from the system ground where it enters the board.

Though the analysis of pulse transformers is beyond the scope of this brief (2), one should be aware of the pertinent parameters affecting the good reproduction of the pulse. The pulse transformer is generally analyzed by different equivalent circuits, depicting the varying phases of the pulse.

Figure 7 shows these circuits. The pulse shape is then optimized by considering the transient response of the equivalent circuits.

The pulse response of the transformer is characterized by a finite rise time, a decaying top period and finite fall time as depicted in Figure 7d. The fastest rise time that can be obtained without overshoot is for the critically damped case and is given by:

\[ t_r = 3.35 \sqrt{\alpha L_C} \quad \text{where} \quad \alpha = R_L / (R_g + R_L) \]

For the top period, there will be some decay leading to a fractional droop, this is given by:

\[ D = \tau \frac{L_P}{R} \quad \text{where} \quad \tau = \text{pulse width} \]

\[ R = R_L \text{ and } R_g \text{ in parallel} \]

The fall period is characterized by the second order circuit of Figure 7c; the primary concern here to prevent severe undershoot or backswing when the 29C53 transmitter is in the high impedance mode. This can best be achieved by having an overdamped system, which is the case when:

\[ L_P > 4CRL_L^2 \]

Commercially available pulse transformers exist which are compatible with the 29C53. Some examples are given in Table 1. Most manufacturers will modify their design to meet the requirements of a particular application.

![Figure 7. (a) Equivalent Circuits for Rise Period (b) Top and Decay Period (c) Fall Period (d) The Pulse Response](image-url)
### TABLE 1. Manufacturers of Pulse Transformers

<table>
<thead>
<tr>
<th>Manufacturer</th>
<th>Location</th>
<th>Winding Ratio</th>
<th>Part No.</th>
</tr>
</thead>
<tbody>
<tr>
<td>AIE Magnetics</td>
<td>St. Petersburg, FL</td>
<td>1.8:1</td>
<td>325-0228</td>
</tr>
<tr>
<td></td>
<td>(813) 347-2181</td>
<td>2.5:1</td>
<td>325-0172</td>
</tr>
<tr>
<td>Schott Corporation</td>
<td>Nashville, TN</td>
<td>1.8:1</td>
<td>11207</td>
</tr>
<tr>
<td></td>
<td>(615) 889-8800</td>
<td>2.5:1</td>
<td>11124</td>
</tr>
<tr>
<td>CTM Magnetics</td>
<td>Tempe, AZ</td>
<td>1.8:1</td>
<td>22087</td>
</tr>
<tr>
<td></td>
<td>(602) 967-9447</td>
<td>2.5:1</td>
<td>25585</td>
</tr>
<tr>
<td>Pulse Engineering</td>
<td>San Diego, CA</td>
<td>1.8:1</td>
<td>64994</td>
</tr>
<tr>
<td></td>
<td>(619) 268-2400</td>
<td>2.5:1</td>
<td>64996</td>
</tr>
</tbody>
</table>

### POWER EXTRACTION

The same transformer can be used at both the line card and the terminal, and the same protection scheme can be used at both ends of the loop. The need now arises to provide power to the terminal. There are a number of ways of providing power to the terminal, for instance a secondary cell can be used as battery back-up in conjunction with a main supply. There is also some scope for trickle charging secondary cells from the line or from a small solar cell array, but the drawback with secondary cells tends to be their short life span. This disadvantage can be offset by using special purpose primary cells as a back-up supply, these do not need any charging circuitry and can be expected to have life expectancy twice that of the secondary cells. Finally, the power can be fed from the switch, in which case a regulator is required at the terminal to extract the power off the line. Figure 8 illustrates this approach.

A DC to DC converter is required to convert the line voltage to 5V for the local circuitry. In order to obtain the lowest losses in the conversion process, it is necessary to use a high efficiency regulator, specifically, a switched mode regulator. Basically, there are three types of switched mode power supplies, the forward, the push pull and the flyback converter (3). This section is devoted to the flyback implementation of a DC to DC converter. The flyback is the most suitable converter for this application, as it provides the highest achievable efficiency and the simplest drive circuitry. Figure 9 shows a block diagram of a flyback converter.

---

**Figure 8. Power Extraction**

**Figure 9. Flyback Converter**
Figure 10. DC to DC Converter
In the flyback inductor, energy is inductively stored during the switch on period, and then passed to the load during the switch off, or the flyback period. During the switch on period, the output diode does not conduct so that the energy in the choke (although appearing as a transformer, this element will be referred to as the choke in accordance with its function) builds up with rising current. While the switch is off the choke voltage reverses in polarity causing the output diode to conduct whereupon the inductive energy is discharged into the output capacitor to form a DC voltage. Regulation is achieved by modulating the oscillator duty cycle, which effectively varies the switch on/off periods. In Figure 9 the diode bridge ensures the correct polarity for the converter while the opto-isolator completes the input to output isolation.

Figure 10 shows a discrete circuit implementation of a DC to DC converter. This circuit was designed to regulate a 5V output for 20-60V input voltage. This implementation provides a maximum power of at least 450 mW. The DC to DC converter consists of an oscillator, a pulse width modulator incorporating an error amplifier and isolating stage, the start up circuitry and the flyback converter. When T5 is on, the choke stores energy and reverse biases diodes D8, D9 and D10. While T5 is off, the choke voltage is negative, hence diodes D8, 9 and 10 are all forward biased and thus build a DC voltage on their respective capacitors. Note that due to the reverse winding technique, the voltage in the output windings are opposite in polarity to the switch winding. The 5V output is regulated by comparing it to a reference voltage, the error in the comparison is then used to modify the transistor T5 on time in such a way so as to keep the 5V output constant.

The diode bridge D1-D4 ensures a certain polarity of the DC voltage for the converter, this is necessary in case the network uses polarity reversal for signaling. The decoupling capacitor C1 serves a secondary function of bypassing any induced surge current. One half of the Schmitt NAND gate CD4093 is used to form a 25 KHz oscillator.

At the output, the opto-isolator in conjunction with the regulating diode TL431 is used to generate an error current. The current through the regulating diode is proportional to the voltage difference between the output and the reference. This device is available from Texas Instruments and Motorola amongst others. Figure 11 illustrates its function.

For the regulator diode, the output voltage is given by:

\[ V_{out} = (1 + \frac{R10}{R9}) \cdot V_{ref} \]

where \( V_{ref} \) is typically 2.5V.

If \( R10 = R9 \)

then \( V_{out} = 5V \)

The current through the regulating diode will increase or decrease with a respective change in the output voltage. This change in current is coupled to the output of the oscillator through the opto-isolator. The opto-isolator used is a Hewlett Packard 6N139, which has Darlington transistor stage providing high current gain that results in a lower power dissipation in the opto-isolator. The current through the isolator differentiates the output of the oscillator through capacitor C3. This differentiated signal is then squared off to define the switching transistor T5 on period. T5 is an IRFD110 MOSFET and is available from International Rectifier. The isolator current and hence the output voltage control the amount of differentiation or the transistor T5 on period as illustrated in Figure 12. Thus regulation is achieved, as the on period is reduced with increasing output voltage and vice versa.
The two transistors T2 and T3 provide a low source impedance driving stage for the switching transistor. The fast current sinking and sourcing will ensure fast switching of transistor T5.

The input capacitance of the MOSFET IRFD110 is a maximum of 200 pF. Without the buffer stage the MOSFET will stay in the linear region longer before saturating, thus resulting in a slower switching speed. The slow switching in turn will result in a lower overall efficiency for the converter.

The resistor R6 and transistor T4 provide current overload protection. Transistor T4 will conduct when the voltage across R6 exceed 0.6V or conversely, the current through it is greater than 150 mA. With T4 conducting, the drive to the MOSFET is nulled by the associated NAND gate.

The transformer choke is a three winding transformer consisting of the switching winding, the output winding, which is split for the +5V and -5V and the self bias winding. The transformer is designed for complete energy transfer under no load conditions and incomplete energy transfer under full load conditions. Figure 13 shows the wave forms of the two modes.

At full load, the incomplete energy transfer mode exhibits a lower peak switching transistor current, while the complete mode at lower power assures a smaller core. The inductance required to achieve this is 6.5 mH for the switch winding. The core used was an RM6CA400-3B7. The number of turns required to achieve this inductance is 130 and for a 20-60V line voltage, 50 turns are required for a +5V output, hence use 50 turns for the -5V too. The self bias winding uses 70 turns. The transformer was wound with 130 turns of 34AWG, followed by 50 bifilar turns of 32AWG and finished off with 70 turns of 32AWG. The dot scheme in Figure 10 should be adhered to. The bobbin is then immersed in varnish such as the Dolph's BC356 to dispel any moisture and to provide a protective coating. Alternatively, a commercially available DC to DC converter transformer such as the 326-0533 can be purchased from AIE Magnetics.

At start up, the converter is powered by the linear regulator D5, R1 and T1, which sets the power supply at 5.3V. After start up the self bias winding forces the voltage on C4 to be between 7 and 15 volts, which will back bias diode D6, thus turning off the linear regulator. Under this condition the power supply provides a self bias voltage to keep it running, while little power is

![Figure 13. (a) Current Voltage Waveforms for Complete Energy Transfer (b) Waveforms for Incomplete Energy Transfer](image-url)
dissipated in the start up regulator. Transistor T1 is selected so that the base-collector can sustain the high voltage stress when it is off. The \(-5\) V supply will only be regulated if the load on that winding is the same as that on the \(+5\) V winding. If this is not possible, it may be necessary to use a linear post regulator to obtain a regulated \(-5\) V supply.

Figure 14 shows the volt-current oscillograms for a 30V line voltage and 400 mW output power. This shows the flyback converter working in the incomplete energy transfer mode. The results obtained in the lab gave an overall efficiency of better than 67% and a power supply ripple of less than 25 mV. The no load power consumption was less than 50 mW. Regulation of the output voltage was better than 150 mV.

The design was wire wrapped to illustrate the concept of power extraction and can of course, be optimized for better performance. Special care should be paid to the layout; Figure 15 shows good layout principles. Use star ground connections to avoid current loops in the ground.

All lead lengths going to the switching MOSFET should be minimized and in particular the gate lead. The resistor in series with the MOSFET should be placed as close to the gate lead as possible. These precautions will avoid undesired oscillations in the MOSFET. The output stage uses Schottky diode and low ESR capacitors to reduce power dissipation. In the event of any undesired EMI radiation the transformer can be placed in an electromagnetic container and the converter can be enclosed in a copper container.
POWER FAILURE CONSIDERATIONS

Without power the line interface pins of the 29C53 appear as diode drops across the line. This means that the transmitter of the Network Terminator and the powered on terminals in a multidrop configuration will be terminated by a diode instead of the usual 50Ω. In the event of a failure, it therefore becomes necessary to isolate the offending terminal from the line. This can be done by providing a switch in the transmit path that is normally closed and opens when no power is applied.

When there is power, the two MOSFETS will be on and appear as a small on resistance, which has to be included in the line transformer design analysis. When there is no power, the MOSFETS appear as back to back diodes, thus stopping any AC flow. The VN0300 MOSFETS manufactured by Siliconix may be used, when on they present a 1.2Ω resistance each. Note that in order to ensure that the MOSFETS conduct it is necessary to have a 10V supply in the system. If this is not possible the MOSFETS can be replaced by a Reed relay which presents a lower on resistance and capacitance but has the disadvantage of consuming more power. A low power relay could not be located hence a vendor was requested to customize one. Figure 17 shows the isolation technique using the Wabash 1992-2-1 25 mW relay which will operate at 3.8V and release at 0.5V.

In the receive path, it is only necessary to increase the impedance seen by the line. One way to implement this principle is to use a MOSFET bilateral switch in the transmit path and to place series resistors in the receive path, such that the impedance seen by the line is greater than 2500Ω. Figure 16 illustrates this approach. A noteworthy point is that the series resistors in the receive path not only provide terminal isolation in case of failure but also protect the terminal from current surges.

Figure 16. Isolation of Equipment in Case of Power Failure

Figure 17. Power Failure Isolation
CONCLUSION

Specific implementations have been provided for the general aspects of line interfacing at both the line card and the terminal end. These solutions can be taken as they are and placed in the particular application or used to aid a system design.

The fixed voltage or constant current feed are both simpler and more economical to realize in discrete form; however the constant current variable voltage scheme may be more suitable in an integrated form. The power converter discussed was based on a low cost simple implementation and it is certainly possible to optimize it to obtain conversion efficiencies in the 75% range. As an alternative to discrete implementations, a low power switch mode power supply is commercially available from Fairchild and Motorola, to name two.

The protection circuits and the transformer, however, can only be provided in discrete form at the present time. The concepts presented in the protection section emphasized low capacitance and maximum protection. The section took an overkill approach and as such a subset of the discussed ideas should suffice most applications. The transformer designed pointed out the relevant parameters to consider and can be used as it is or modified to the particular application. Of course the ISDN transformer is also commercially available.

REFERENCES

1. Protecting against surge voltages in short and long branch circuits. By Shanawaz M. Khan, Communications Systems Equipment Design, December 1984
3. Design of solid state power supplies. By Eugene R. Hnatek, Von Nostrand Reinhold Company
ISDN Applications with 29C53 and 80188

HERBERT WEBER
TELECOM OPERATIONS

Order Number: 270247-003
TERMINAL ADAPTOR (TA)

A terminal adaptor, or "TA," is the link between existing non-ISDN equipment like terminals, facsimile, printers and the ISDN network. The function of this application is to effectively replace equipment such as a modem. Usually provided as a separate box, it processes RS232 or X.21 data and places it on the 4 wire 'S' loop. No change at the terminal is required to make it ISDN compatible.

The design is based on a 29C53 transceiver for the ISDN connection and an 80188 microprocessor in combination with an 82530 communications controller for the data connection. Benefits of the application are:

- Data rates up to 19.2 Kb/s using an RS232 interface or up to 48 Kb/s using an X.21 interface.
- Compact design and low cost.
- Virtually error free transmission.

Link Setup

The user sets up a call in the same manner as a Hayes* modem user does, i.e. a command is transferred to the adaptor via the RS232 interface. The command takes the form of an ASCII string in which the first 2 characters are "AT".

The 80188 accepts the command and begins the call setup procedure by communicating the call's destination to the NT (or CO). This is achieved by passing call setup messages to a link level protocol, which is passed to the NT over the physical level (S bus). The partitioning of the tasks is as follows:

82530

Full duplex, dual channel serial communications controller capable of working in asynchronous, bit or byte synchronous modes. The 82530 receives commands from the terminal's RS232 or X.21 interface and passes them on to the 80188.

80188

After having received the dialing information from the keyboard, the 80188 sets up the call via the 29C53 D-channel by sending the appropriate CCITT message up the link.

- Call setup message generation (CCITT I.451).

*Hayes is a registered trademark of Hayes Microcomputer Products, Inc.

Upper portion of link access procedure (CCITT I.440) handling:
- Multiple logic channels
- Sequence control
- Error correction (retransmission)
- Flow control

EPLD

- Interface conversion, serial to/from SLD
- B-channel assignment

29C53

- Physical level interface (CCITT I.430)
- Lower portion of the link access procedure:
  - Zero insertion/deletion
  - CRC generation and checking
  - Flag appending and detection
- D-channel message buffering

The 80188 passes the information for the D-channel messages via the parallel bus into the FIFO's of the 29C53.

The NT grants a B-channel (if available) to the TA and the channel is now ready for data transfer.

Data Transfer

An indication is given to the user's terminal via the RS232 or X.21 port that communication may commence. Any subsequent data, from the terminal, is treated as follows:

Data from the terminal passes via the 82530 to RAM via one of the 80188 DMA channels.

The 80188 fetches the data from RAM, depacketizes and packetizes it before sending it back to the 82530 where a protective HDLC protocol is added.

From the 82530 the data reaches the EPLD to be inserted into the B1 or B2 channel on the SLD bus. The 29C53 sends it out over the "S" interface.
Figure 1. Terminal Adaptor
ISDN PHONE WITH BUILT IN TERMINAL ADAPTOR (TA)

Figure 2 shows the concept of an ISDN phone with hookup to standard sync/async terminals. No change at the terminal is required to make it ISDN compatible.

The design is based on a 29C53 transceiver for the ISDN connection, a 29C48 combo for the voice connection and an 80188 microprocessor in combination with an 82530 communications controller for the data connection. Benefits of the application are:

- Data rates up to 19.2 kb/s using an RS232 interface or up to 48 kb/s using an X.21 interface.
- Compact design and low cost.
- Virtually error free transmission.

**Link Setup**

Applies both for speech and data links. The 80188 accepts the command and begins the call setup procedure by communicating the call’s destination to the NT (or CO). This is achieved by passing call setup messages to a link level protocol, which is passed to the NT over the physical level (S-bus). The partitioning of the tasks is as follows:

**8279**

The 8279 keyboard and display controller scans the telephone number pad and supports a small telephone display. Calls are initiated either through the terminal keyboard using an extended Hayes Smart Modem command set or via the telephone number pad.

**82530**

Full duplex, dual channel serial communications controller capable of working in asynchronous, bit or byte synchronous modes. The 82530 receives commands from the terminal's RS232 or X.21 interface and passes them on to the 80188.

**80188**

After having received the dialing information from either keyboard, the 80188 sets up the call via the 29C53 D-channel by sending the appropriate message up the link.

- Call setup message generation (CCITT I.451)
- Upper portion of link access procedure (CCITT I.440) handling:
  - Multiple logic channels
  - Sequence control
  - Error correction (retransmission)
  - Flow control

**EPLD**

- Interface conversion, serial to/from SLD
- B-channel assignment

**29C53**

- Physical level interface (CCITT I.430)
- Lower portion of the link access procedure:
  - Zero insertion/deletion
  - CRC generation and checking
  - Flag appending and detection
- D-channel message buffering

The 80188 passes the information for the D-channel messages via the parallel bus into the FIFO’s of the 29C53.

The NT grants a B-channel (if available) to the TA and the channel is now ready for data transfer.

**Information Transfer**

**VOICE**

The voice transfer is supported by the 29C48 which transmits the voice on either the B1 or B2 channel (controlled by the EPLD) into the 29C53 and onward to the S-bus.

**DATA**

An indication is given to the user’s terminal via the RS232 or X.21 port that communication may commence. Any subsequent data, from the terminal, is treated as follows:

Data from the terminal passes via the 82530 to RAM via one of the 80188 DMA channels.

The 80188 fetches the data from RAM, depacketizes and packetizes it before sending it back to the 82530 where a protective HDLC protocol is added.

From the 82530, the data reaches the EPLD to be inserted into the B1 or B2 channel on the SLD bus. The 29C53 sends it out over the “S” interface.
Figure 2. ISDN Phone With Built In Terminal Adaptor
PERSONAL COMPUTER INTERFACE

Like the terminal adaptor, the ISDN PC adaptor provides a link to the ISDN network. The ISDN Co-Processor shown in Figure 3 implements the hardware functions required to support the CCITT I-series "S" interface.

The ISDN Co-Processor is using the 80188 microprocessor in combination with an 82530 serial communications controller for data processing, dual port RAM as interface and buffer to the host bus, the 29C48 Codec/filter for voice support and the 29C53 transceiver for the ISDN connection.

The ISP188 ISDN Software Package is optimized for this hardware configuration.

Co-Processor

The PC adaptor is an intelligent communications subsystem designed to function as a slave processor board in the PC. This relieves the host processor of much of the communications function.

82530

Full duplex, dual channel serial communications controller. One of the two channels is attached to either of the B channels and operates at 64 kb/s. The second channel is available to external datacom equipment via an optional serial port, or for connection to the second B channel.

29C48

Voice conversion and interface to the four wire handset is performed by this software programmable integrated Codec/filter combo. Designed for ISDN terminal applications it offers programmable gain in transmit and receive direction for user loudness control and adaptation to local network requirements as well as sidetone insertion and tone injection for locally produced feedback signals.

The 29C48 can access either B1 or B2 channel by setting the B Sel pin accordingly.

29C53

"S" bus transceiver and D channel controller in a single chip. The 29C53 provides the physical level interface to the "S" bus in accordance to CCITT I.430 and the lower portion of the link access protocol. Activation, deactivation, zero insertion/deletion, CRC generation and checking and flag appending and detection are performed by the 29C53, the higher level portions of LABD are executed on the 80188 and passed on to the 29C53 via the parallel bus into the FIFO's.

B channel access is via the SLD serial port. Voice signals are directly passed on to the 29C48. Data is extracted and injected by the EPLD (Erasable, Programmable Logic Device) which performs the B channel assignment and the interface conversion to the 82530.

Figure 3. Personal Computer Interface
The addition of ISDN line cards to a PABX provides the user with access to the ISDN network. While the analog line card provides access for standard telephone as well as for modems, ISDN terminal adaptors, terminals and phones are connected to the ISDN line card via a 4 wire "S" loop. The described application provides all functions to separate voice and switched data (B-channels) from signaling and packetized data (D-channel).

The 29C53 and 80188 together handle the processing of D-channel protocols and messages as follows:

1. The 29C53 executes all bit level HDLC processing, puts the "raw" messages into its FIFO and raises the interrupt signal.

2. A special status register in the 29C53s allows the 80188, through a single status read operation, to determine which of the 29C53s is requiring interrupt servicing, i.e. has D-channel messages(s) in its FIFO.

3. The 80188 accesses the FIFO concerned and the data is transferred to RAM.

4. The 80188 determines whether the data is for signaling (S-packet) or is a message to be sent out over the packet (P-packet) switched network.

   Signaling information can be processed locally or sent via the linecard controller.

   If the data is of "P" type, meant for the packet switched network, it is DMA'd into the 82530 serial communications controller which performs the necessary HDLC transmission, again without any CPU involvement.

5. The 80188 software is responsible for sending out acknowledgements for received messages from the 29C53's D-channel and can thus support large window sizes.

B-channel information is directly passed from the "S" loop over the 29C53 and line card controller to the switch backplane.

For transmission in the opposite direction, the procedure is equivalent to the one described above.
### OTHER AVAILABLE TELECOM LITERATURE

<table>
<thead>
<tr>
<th>Title</th>
<th>Order Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>29C53 Reference Manual</td>
<td>270151-001</td>
</tr>
<tr>
<td>29C53 Terminal Evaluation Kit (TEK) Manual</td>
<td></td>
</tr>
<tr>
<td>ISP188 ISDN Software Package for the 80188</td>
<td>290149-001</td>
</tr>
<tr>
<td>IDK29C53 ISDN Development Kit for 29C53</td>
<td></td>
</tr>
<tr>
<td>PC53 ISDN Board</td>
<td></td>
</tr>
</tbody>
</table>
PCM Codec/Filter and Combo
The Intel 2910A is a fully integrated PCM (Pulse Code Modulation) Codec (Coder-Decoder), fabricated with N-channel silicon gate technology. The high density of integration allows the sample and hold circuits, the digital-to-analog converter, the comparator and the successive approximation register to be integrated on the same chip, along with the logic necessary to interface a full duplex PCM link and provide in-band signaling.

The primary applications are in telephone systems:

- Transmission — T1 Carrier
- Switching — Digital PBX's and Central Office Switching Systems
- Concentration — Subscriber Carrier/Concentrators

The wide dynamic range of the 2910A (78dB) and the minimal conversion time (80μsec minimum) make it an ideal product for other applications, like:

- Date Acquisition
- Telemetry
- Secure Communications Systems
- Signal Processing Systems

Figure 1. Block Diagram

Figure 2. Pin Configuration

Figure 3. Pin Names

November 1986
Order Number: 006785-002
## Pin Description

<table>
<thead>
<tr>
<th>Pin No.</th>
<th>Symbol</th>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>CAP1x</td>
<td>Hold</td>
<td>Connections for the transmit holding capacitor. Refer to Applications section.</td>
</tr>
<tr>
<td>2</td>
<td>CAP2x</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>VFx</td>
<td>Input</td>
<td>Analog input to be encoded into a PCM word. The signal on this lead is sampled at the same rate as the transmit frame synchronization pulse FSx, and the sample value is held in the external capacitor connected to the CAP1x and CAP2x leads until the encoding process is completed.</td>
</tr>
<tr>
<td>4</td>
<td>AUTO</td>
<td>Output</td>
<td>Most significant bit of the encoded PCM word (+5V for negative, -5V for positive inputs). Refer to the Codec Applications section.</td>
</tr>
<tr>
<td>5</td>
<td>GRDA</td>
<td>Ground</td>
<td>Analog return common to the transmit and receive analog circuits. Not connected to GRDD internally.</td>
</tr>
<tr>
<td>6</td>
<td>SIGR</td>
<td>Output</td>
<td>Signaling output. SIGR is updated with the 8th bit of the receive PCM word on signaling frames, and is latched between two signaling frames. TTL interface.</td>
</tr>
<tr>
<td>7</td>
<td>VDD</td>
<td>Power</td>
<td>+12V±5%; referenced to GRDA.</td>
</tr>
<tr>
<td>8</td>
<td>D_R</td>
<td>Input</td>
<td>Receive PCM highway (serial bus) interface. The Codec serially receives a PCM word (8 bits) through this lead at the proper time defined by FSR, CLKR, DCr, and CLKe.</td>
</tr>
<tr>
<td>9</td>
<td>PDN</td>
<td>Output</td>
<td>Active high when Codec is in the power down state. Open drain output.</td>
</tr>
<tr>
<td>10</td>
<td>VF_R</td>
<td>Output</td>
<td>Analog output. The voltage present on VF_R is the decoded value of the PCM word received on lead D_R. This value is held constant between two conversions.</td>
</tr>
<tr>
<td>11</td>
<td>NC</td>
<td>No Connects</td>
<td>Recommended practice is to strap these NC’s to GRDA.</td>
</tr>
<tr>
<td>12</td>
<td>NC</td>
<td>Connects</td>
<td>Ground return common to the logic power supply, VCC.</td>
</tr>
<tr>
<td>13</td>
<td>GRDD</td>
<td>Ground</td>
<td>Output of the transmit side onto the send PCM highway (serial bus). The 8-bit PCM word is serially sent out on this pin at the proper time defined by FSX, CLKX, DCx, and CLKe. TTL three-state output.</td>
</tr>
<tr>
<td>14</td>
<td>DX</td>
<td>Output</td>
<td>Normally high, this signal goes low while the Codec is transmitting an 8-bit PCM word on the DX lead. (Timeslot information used for diagnostic purposes and also to gate the data on the DX lead.) Open drain output.</td>
</tr>
<tr>
<td>15</td>
<td>TSX</td>
<td>Output</td>
<td>Master receive clock defining the bit rate on the receive PCM highway. Typically 1.544 Mbps for a T1 carrier system. Maximum rate 2.1 Mbps. 50% duty cycle. TTL interface.</td>
</tr>
<tr>
<td>16</td>
<td>VCC</td>
<td>Power</td>
<td>+5V±5%, referenced to GRDD.</td>
</tr>
<tr>
<td>17</td>
<td>CLKR</td>
<td>Input</td>
<td>Frame synchronization pulse for the receive PCM highway. Resets the on-chip timeslot counter for the receive side. Maximum repetition rate 12 KHz. Also used to differentiate between non-signaling frames and signaling frames on the receive side. TTL interface.</td>
</tr>
</tbody>
</table>
Pin Description (Continued)

<table>
<thead>
<tr>
<th>Pin No.</th>
<th>Symbol</th>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>19</td>
<td>CLKX</td>
<td>Input</td>
<td>Master transmit clock defining the bit rate on the transmit PCM highway. Typically 1.544 Mbps for a T1 carrier system. Maximum rate 2.1 Mbps. 50% duty cycle. TTL interface.</td>
</tr>
<tr>
<td>20</td>
<td>FSX</td>
<td>Input</td>
<td>Frame synchronization pulse for the transmit PCM highway. Resets the on-chip timeslot counter for the transmit side. Maximum repetition rate 12 KHz. Also used to differentiate between non-signaling frames and signaling frames on the transmit side. TTL interface.</td>
</tr>
<tr>
<td>21</td>
<td>SIGX</td>
<td>Input</td>
<td>Signaling input. This digital input is transmitted as the 8th bit of the PCM word on the DX lead, on signaling frames. TTL interface.</td>
</tr>
<tr>
<td>22</td>
<td>VBB</td>
<td>Power</td>
<td>-5V ± 5%, referenced to GRDA.</td>
</tr>
<tr>
<td>23</td>
<td>DC</td>
<td>Input</td>
<td>Data input to program the Codec for the chosen mode of operation. Becomes an active low chip select when CLKC is tied to VCC. TTL interface.</td>
</tr>
<tr>
<td>24</td>
<td>CLKC</td>
<td>Input</td>
<td>Clock input to clock in the data on the DC lead when the timeslot assignment feature is used; tied to VCC to disable this feature. TTL interface.</td>
</tr>
</tbody>
</table>

FUNCTIONAL DESCRIPTION

The 2910A PCM Codec provides the analog-to-digital and the digital-to-analog conversions necessary to interface a full duplex (4 wires) voice telephone circuit with the PCM highways of a time division multiplexed (TDM) system.

In a typical telephone system the Codec is used between the PCM highways and the channel filters. The Codec provides two major functions:

- Encoding and decoding of analog signals (voice and call progress tones)
- Encoding and decoding of the signaling and supervision information

On a non-signaling frame, the Codec encodes the incoming analog signal at the frame rate (FSX) into an 8-bit PCM word which is sent out on the DX lead at the proper time. Similarly, the Codec fetches an 8-bit PCM word from the receive highway (DR lead) and decodes an analog value which will remain constant on lead VF until the next receive frame. Transmit and receive frames are independent. They can be asynchronous (transmission) or synchronous (switching) with each other.

For channel associated signaling, the Codec transmit side will encode the incoming analog signal as previously described and substitute the signal present on lead SIGX for the least significant bit of the encoded PCM word. Similarly, on a receive signaling frame, the Codec will decode the 7 most significant bits according to the CCITT G733 recommendation and will output the least significant bit value on the SIGR lead until the next signaling frame. Signaling frames on the send and receive sides are independent of each other, and are selected by a double-width frame sync pulse on the appropriate channel.

![Figure 4. Typical Line Termination](image-url)
The 2910A Codec is intended to be used on line and trunk terminations. The call progress tones (dial tone, busy tone, ring-back tone, re-order tone), and the prerecorded announcements, can be sent through the voice-path; digital signaling (off hook and disconnect supervision, rotary dial pulses, ring control) is sent through the signaling path.

Circuitry is provided within the Codec to internally define the transmit and receive timeslots. In small systems this may eliminate the need for any external timeslot exchange; in large systems it provides one level of concentration. This feature can be bypassed and discrete timeslots sent to each Codec within a system.

In the power-down mode, most functions of the Codec are directly disabled to reduce power dissipation to a minimum.

**CODEC OPERATION**

**Codec Control**

The operation of the 2910A is defined by serially loading an 8-bit word through the Dc lead (data) and the CLKc lead (clock). The loading is asynchronous with the other operations of the Codec, and takes place whenever transitions occur on the CLKc lead. The Dc input is loaded in during the trailing edge of the CLKc input.

![Codec Control Diagram](image)

The control word contains two fields:

Bit 1 and Bit 2 define whether the subsequent 6 bits apply to both the transmit and receive side (00), the transmit side only (01), the receive side only (10), or whether the Codec should go into the standby, power-down mode (11). In the last case (11), the following 6 bits are irrelevant.

The last 6 bits of the control word define the timeslot assignment, from 000000 (timeslot 1) to 111111 (timeslot 64). Bit 3 is the most significant bit and bit 8 the least significant bit and last into the Codec.

<table>
<thead>
<tr>
<th>Bit 1</th>
<th>Bit 2</th>
<th>Mode</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>X &amp; R</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>X</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>R</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>Standby</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit 3 4 5 6 7 8</th>
<th>Timeslot</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 0 0 0 0 0 0</td>
<td>1</td>
</tr>
<tr>
<td>0 0 0 0 0 0 1</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>1 1 1 1 1 1 1</td>
<td>64</td>
</tr>
</tbody>
</table>

The Codec will retain the control word (or words) until a new word is loaded in or until power is lost. This feature permits dynamic allocation of timeslots for switching applications.

**Microcomputer Control Mode**

In the microcomputer mode, each Codec performs its own timeslot computation independently for the transmit and receive channels by counting clock pulses (CLKx and CLKr). All Codecs tied to the same data bus receive identical framing pulses (FSx and FSr). The framing pulses reset the on-chip timeslot counters every frame; hence the timeslot counters of all devices are synchronized. Each Codec is programmed via CLKc and Dc for the desired transmit and receive timeslots according to the description in the Codec Control Section. All Codecs tied to the same DR bus will, in general, have different receive timeslots, although that is not a device requirement. There may be separate busses for transmit and receive or all Codecs may transmit and receive over the same bus, in which case the transmit and receive channels must be synchronous (CLKx = CLKr). There are no other restrictions on timeslot assignments; a device may have the same transmit and receive timeslot even if a single bus is used.

There are several requirements for using the CLKc-Dc interface in the microcomputer mode.

1) A complete timeslot assignment, consisting of eight negative transitions of CLKc, must be made in less than one frame period. The assignment
can overlap a framing pulse so long as all 8 control bits are clocked in within a total span of 125 \( \mu \text{s} \) (for an 8 KHz frame rate). \( \text{CLK}_C \) must be left at a TTL low level when not assigning a timeslot.

2) A dead period of two frames must always be observed between successive timeslot assignments. The two frame delay is measured from the rising edge of the first \( \text{CLK}_C \) transition of the previous timeslot assigned.

3) When the device is in the power-down state (Standby), the following three-step sequence must be followed to power-up the codec to avoid contention on the transmit PCM highway.
   a) Assign a dummy transmit timeslot. The dummy should be at least two timeslots greater than the maximum valid system timeslot (usually 24 or 32). For example, in a 24 timeslot system, the dummy could be any timeslot between 26 and 64. This will power-up the transmit side, but prevent any spurious \( D_X \) or \( T_S_X \) outputs.
   b) Two frames later, assign the desired transmit timeslot.
   c) Two frames later assign the desired receive timeslot.

4) Initialization sequence: The device contains an on-chip power-on clear function which guarantees that with proper sequencing of the supplies (\( V_{CC} \) or \( V_{DD} \) on last), the device will initialize with no timeslot assigned to either the transmit or receive channel. After a supply failure or whenever the supplies are applied, it is recommended that either power down assignment be made first, or the first timeslot assignment be a transmit timeslot or a transmit/receive timeslot. The consequence of making a receive timeslot assignment first, after supply application, is that the transmit channel will assume timeslot 1, potentially producing bus contention.

5) Transmit only/receive only operation is permitted provided that a power down assignment is made first. Otherwise, special circuits which use only one channel should be physically disconnected from the unused bus; this allows a timeslot to be made to an unused channel without consequence.

6) Both frame synchronizing pulses (\( F_S_X \), \( F_S_R \)) must be active at all times after power on clear (after power supplies are turned on). This requirement must be met during powerdown and receive only or transmit only operation, as well as during normal transmit and receive operation.

Example of Microcomputer Control Mode:

The two words 01000001 and 10000010 have been loaded into the Codec. The transmit side is now programmed for timeslot 2 and the receive side for timeslot 3. The Codec will output a PCM word on the transmit PCM highway (bus) during the timeslot 2 of the transmit frame, and will fetch a PCM word from the receive PCM highway during timeslot 3.

Figure 5. Microcomputer Mode Programming Example
In this example the Codec interface to the PCM highway then functions as shown above. (FS_X and FS_R may be asynchronous.)

**Direct Control Mode**

The direct mode of operation will be selected when the CLK_C pin is strapped to the +5 volt supply (V_CC). In this mode, the DC pin is an active low chip select. In other words, when DC is low, the device transmits and receives in the timeslots which follow the appropriate framing pulses. With DC high the device is in the power down state. Even though CLK_C characteristics are simpler for the 2910A it will operate properly when plugged into a 2910 board.

Deactivation of a channel by removal of the appropriate framing pulse (FS_X or FS_R) is not permitted. Specifically, framing pulses must be applied for a minimum of two frames after a change in state of DC in order for the DC change to be internally sensed. In particular, when entering standby in the direct mode, framing pulses must be applied as usual for two frames after DC is brought high.

The Codec will enter the direct mode within three frame times (375 μs) as measured from the time the device power supplies settle to within the specified limits. This assumes that CLK_C is tied to V_CC and that all clocks are available at the time the supplies have settled.

**General Control Requirements**

All bit and frame clocks should be applied whenever the device is active. In particular, an unused channel cannot be deactivated by removal of its associated frame or bit clock while the other channel of the same device remains active.

A single channel cannot be deactivated except by physical disconnection of the data lead (D_X or D_R) from the system data bus. A device (both transmit and receive channels) may be deactivated in either control mode by powering down the device. Both channels are always powered down together.

**Encoding**

The VF signal to be encoded is input on the VF_X lead. An internal switch samples the signal and the hold function is performed by the external capacitor connected to the CAP1_X and CAP2_X leads. The sampling and conversion is synchronized with the

---

**Figure 6. Microcomputer Mode PCM Highway Example**

**Figure 7. Transmit Encoding**
transmit timeslot. The PCM word is then output on the DX lead at the proper timeslot occurrence of the following frame. The A/D converter saturates at approximately ±2.2 volts RMS (±3.1 volts peak).

Decoding
The PCM word is fetched by the DR lead from the PCM highway at the proper timeslot occurrence. The decoded value is held on an internal sample and hold capacitor. The buffered non-return to zero output signal on the VFR lead has a dynamic range of approximately ±2.2 volts RMS (±3.1 volts peak).

Signaling
The duration of the FSX and FSR pulses defines whether a frame is an information frame or a signaling frame:

- A frame synchronization pulse which is a full clock period in duration (CLKX period for FSX, CLKR period for FSR) designates a non-signaling frame.
- A frame synchronization pulse which is two full clock periods in duration (two CLKX periods for FSX, two CLKR periods for FSR) designates a signaling frame.

On the encoding side, when the FSX pulse is widened, the 8th bit of the PCM word will be replaced by the value on the SIGX input at the time when the 8th bit is output on the DX lead.

On the decoding side, when the FSR pulse is widened, the 8th bit of the PCM word is detected and transmitted on the SIGR lead. That output is latched until the next receiving signaling frame.

The remaining 7 bits are decoded according to the value given in the CCITT G733 recommendation. The SIGR lead is reset to a TTL low level whenever the Codec is in the power-down state.

---

Figure 8. Transmit 8th Bit Signaling

Figure 9. Receive 8th Bit Signaling
T1 Framing

The Codec will accept the standard D3/D4 framing format of 193 clock pulses per frame (equivalent to CLKx, CLKr of 1.544 Mb/s). However, the 193rd bit may be blanked (equivalent to CLKx, CLKr of 1.536 Mb/s) if desired.

Standby Mode—Power Down

To minimize power consumption and heat dissipation a standby mode is provided where all Codec functions are disabled except for DC and CLKD leads. These allow the Codec to be reactivated. In the microcomputer mode the Codec is placed into standby by loading a control word (DC) with a “1” in bits 1 and 2 locations. In the direct mode when DC is brought high, the all “1’s” control word is internally transferred to the control register, invoking the standby condition.

While in the standby mode, the DX output is actively held in a high impedance state to guarantee that the PCM bus will not be driven. The SIGR output is held low to provide a known condition and remains this way upon activation until it is changed by signaling.

The power consumption in the standby mode is typically 33 mW.

Power-On Clear

Whether the device is used in the direct or microcomputer mode, an internal reset (power-on clear) is generated, forcing the device into the power down state, when power is supplied by any of the following methods. (1) Device power supplies are turned on in a system power-up situation where either VCC or VDD is applied last. (2) A large supply transient causes either of the two positive supplies to drop to less than approximately 2 volts. (3) A board containing Codecs is plugged into a “hot” system where VCC or VDD is the last contact made. It may be necessary to trim back the edge connector pins or fingers on VCC or VDD relative to the other supply to guarantee that the power-on clear will operate properly when a board is plugged into a “hot” system. Furthermore, the Codec will inhibit activity on TSx and DX during the application of power supplies.

The device is also tolerant of transients in the negative supply (VBG) so long as VBG remains more negative than −3.5 volts. VBG transients which exceed this level should be detected and followed by a system reinitialization.

Precision Voltage Reference for the D/A Converter

The voltage reference is generated on the chip and is calibrated during the manufacturing process. The technique uses the difference in sub-surface charge density between two suitably implanted MOS devices to derive a temperature stable and bias stable reference voltage.

A gain setting op amp, programmed during manufacturing, “trims” the reference voltage source to the final precision voltage reference value provided to the D/A converter. The precision voltage reference determines the initial gain and dynamic range characteristics described in the A.C. Transmission Specification section.

µ-Law Conversion

µ-law represents a particular implementation of a piece-wise linear approximation to a logarithmic compression curve which is:

\[ F(x) = \text{Sgn}(x) \frac{\ln(1 + \mu |x|)}{\ln(1 + \mu)} \quad 0 \leq |x| \leq 1 \]

where \( x \) = input signal

\[ \text{Sgn}(x) = \text{sign of input signal} \]

\[ \mu = 255 \text{ (defined by AT & T)} \]

The 2910A \( \mu = 255 \) law Codec uses a 15 segment approximation to the logarithmic law. Each segment consists of 16 steps. In adjacent segments the step sizes are in a ratio of two to one. Within each segment the step size is constant except for the first step of the first segment of the encoder, as indicated in the attached table. The output levels are midway between the corresponding decision levels. The output levels \( y_n \) are related to the input levels \( x_n \) by the expression:

\[ y_n = \frac{x_n + x_{n+1}}{2} \quad \text{for } 1 \leq n \leq 127 \]

\[ y_0 = x_0 = 0 \text{ for } n = 0 \]

These relationships are implicit in the following table.
### Theoretical \( \mu \)-Law—Positive Input Values (for Negative Input Values, Invert Bit 1)

<table>
<thead>
<tr>
<th>Segment Number</th>
<th>No. of Steps x Step Size</th>
<th>Value at Segment End Points</th>
<th>Decision Value Number ( n )</th>
<th>Decision Value ( x_n(1) )</th>
<th>PCM Word(^{(2)} )</th>
<th>Normalized Value at Decoder Output ( y_n(4) )</th>
<th>Decoder Output Value Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td>16 x 256</td>
<td>8159(^{(5)} ) (128)</td>
<td>(8159)</td>
<td>1 0 0 0 0 0 0 0</td>
<td>8031</td>
<td>127</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>16 x 128</td>
<td>4063</td>
<td>112</td>
<td>4063</td>
<td>1 0 0 0 1 1 1 1</td>
<td>4191</td>
<td>112</td>
</tr>
<tr>
<td>6</td>
<td>16 x 64</td>
<td>2015</td>
<td>96</td>
<td>2015</td>
<td>1 0 0 1 1 1 1 1</td>
<td>2079</td>
<td>96</td>
</tr>
<tr>
<td>5</td>
<td>16 x 32</td>
<td>991</td>
<td>80</td>
<td>991</td>
<td>1 0 1 0 1 1 1 1</td>
<td>1023</td>
<td>80</td>
</tr>
<tr>
<td>4</td>
<td>16 x 16</td>
<td>479</td>
<td>64</td>
<td>479</td>
<td>1 0 1 1 1 1 1 1</td>
<td>495</td>
<td>64</td>
</tr>
<tr>
<td>3</td>
<td>16 x 8</td>
<td>223</td>
<td>48</td>
<td>223</td>
<td>1 1 0 0 1 1 1 1</td>
<td>231</td>
<td>48</td>
</tr>
<tr>
<td>2</td>
<td>16 x 4</td>
<td>95</td>
<td>32</td>
<td>95</td>
<td>1 1 0 1 1 1 1 1</td>
<td>99</td>
<td>32</td>
</tr>
<tr>
<td>1</td>
<td>15 x 2</td>
<td>31</td>
<td>16</td>
<td>31</td>
<td>1 1 1 0 1 1 1 1</td>
<td>33</td>
<td>16</td>
</tr>
<tr>
<td>1</td>
<td>1 x 1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1 1 1 1 1 1 1 0</td>
<td>2</td>
<td>1</td>
</tr>
</tbody>
</table>

**NOTES:**

1. 8159 normalized value units correspond to the value of the on-chip voltage reference.
2. The PCM word corresponding to positive input values between two successive decision values numbered \( n \) and \( n + 1 \) (see column 4) is \((255 - n)\) expressed as a binary number.
3. The PCM word on the highways is the same as the one shown in column 6.
4. The voltage output on the VFR lead is equal to the normalized value given in the table, augmented by an offset. The offset value is approximately 15 mV.
5. \( x_{128} \) is a virtual decision value.
During signaling frames, a 7-bit transfer characteristic is implemented in the decoder. This characteristic is derived from the decoder values in the attached table by assuming a value of "1" for the LSB (8th bit) and shifting the decoder transfer characteristics one half-step away from the origin. For example, the maximum decoder output level for signaling frames has normalized value 7903, whereas it has value 8031 in normal (non-signaling) frames.

APPLICATIONS

![Figure 10. Codec Transfer Characteristic](image1)

![Figure 11. Circuit Interface—without External Auto Zero](image2)
Holding Capacitor

For an 8 KHz sampling system the transmit holding capacitor $\text{CAP}_x$ should be 2000 pF ± 20%.

Auto Zero

The 2910A contains a transparent on-chip auto zero plus a device pin for implementing a sign-bit driven external auto zero feedback loop. The on-chip auto zero reduces the input offset voltage of the encoder ($VF_x$) to less than 3 mV. For most telephony applications, this input offset is perfectly acceptable, since it insures the encoder is biased in the lower 25% of the first segment.

Where lower input offset is required the external auto zero loop may be used to bias the encoder exactly at the zero crossing point. The consequence of the external auto zero loop, aside from extra components, is the addition of the dithering auto-zero signal to the input signal, resulting in slightly higher idle channel noise (approximately 2dB) than when the external loop is not used. Consequently, where the application permits, it is recommended that the external auto zero loop not be used. When not used, the AUTO pin should float.

The circuit interface with auto zero drawing shows a possible connection between the $VF_x$ and AUTO leads with the recommended values of $C_1 = 0.3 \mu\text{F}$, $R_1 = 150 \text{ K}\Omega$, $R_2 = 330 \text{ K}\Omega$, and $R_3 = 470 \text{ K}\Omega$.

Filters Interface

The filters may be interfaced as shown in the circuit interface diagrams. Note that the output pulse stream is of the non-return-to-zero type and hence requires the $(\sin x)/x$ correction provided by the 2912A filter.

Dx Buffering

For higher drive capability or increased system reliability it may be desirable that the $D_x$ output of a group of Codecs be buffered from the system PCM bus with an external three-state or open collector buffers. A buffer can be enabled with the appropriate Codec generated $TS_x$ signal signals. $TS_x$ signal may also be used to activate external zero code suppression logic, since the occurrence of an active state of any $TS_x$ implies the existence of PCM voice bits (as opposed to transparent data bits) on the bus.

![Circuit Interface—With External Auto Zero](image-url)
ABSOLUTE MAXIMUM RATINGS*

Temperature Under Bias ........... -10°C to +80°C
Storage Temperature ............ -65°C to +150°C
All Input or Output Voltages with
  Respect to VBB .................. -0.3V to +20V
VCC, VDD, GRDD, and GRDA with
  Respect to VBB .................. -0.3V to +20V
Power Dissipation ................. 1.35W

*Notice: Stresses above those listed under “Absolute Maximum Ratings” may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions above those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

D.C. CHARACTERISTICS

TA=0°C to +70°C, VDD=+12V ±5%, VCC=+5V ±5%, VBB=−5V ±5%, GRDA=0V, GRDD=0V, unless otherwise specified

DIGITAL INTERFACE

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Parameter</th>
<th>Limits</th>
<th>Units</th>
<th>Test Conditions</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Min</td>
<td>Typ(1)</td>
<td>Max</td>
</tr>
<tr>
<td>IL</td>
<td>Low Level Input Current</td>
<td>10</td>
<td>µA</td>
<td>V&lt;sub&gt;IN&lt;/sub&gt; &lt; V&lt;sub&gt;IL&lt;/sub&gt;</td>
</tr>
<tr>
<td>IH</td>
<td>High Level Input Current</td>
<td>10</td>
<td>µA</td>
<td>V&lt;sub&gt;IN&lt;/sub&gt; &gt; V&lt;sub&gt;IH&lt;/sub&gt;</td>
</tr>
<tr>
<td>VIL</td>
<td>Input Low Voltage</td>
<td>0.6</td>
<td>V</td>
<td></td>
</tr>
<tr>
<td>VIH</td>
<td>Input High Voltage</td>
<td>2.0</td>
<td>V</td>
<td></td>
</tr>
<tr>
<td>VOL</td>
<td>Output Low Voltage</td>
<td>0.4</td>
<td>V</td>
<td>D&lt;sub&gt;x&lt;/sub&gt;, I&lt;sub&gt;OL&lt;/sub&gt; = 4.0 mA</td>
</tr>
<tr>
<td>VOH</td>
<td>Output High Voltage</td>
<td>2.4</td>
<td>V</td>
<td>SIGR&lt;sub&gt;r&lt;/sub&gt;, I&lt;sub&gt;OH&lt;/sub&gt; = 0.08 mA</td>
</tr>
</tbody>
</table>

ANALOG INTERFACE

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Parameter</th>
<th>Limits</th>
<th>Units</th>
<th>Test Conditions</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Min</td>
<td>Typ(1)</td>
<td>Max</td>
</tr>
<tr>
<td>ZAI</td>
<td>Input Impedance when Sampling, VFX</td>
<td>125</td>
<td>300</td>
<td>500</td>
</tr>
<tr>
<td>ZAO</td>
<td>Small Signal Output Impedance, VF&lt;sub&gt;R&lt;/sub&gt;</td>
<td>100</td>
<td>180</td>
<td>300</td>
</tr>
<tr>
<td>VOR</td>
<td>Output Offset Voltage at VF&lt;sub&gt;R&lt;/sub&gt;</td>
<td>±50</td>
<td>mV</td>
<td>all “1s” code sent to D&lt;sub&gt;R&lt;/sub&gt;</td>
</tr>
<tr>
<td>VIX</td>
<td>Input Offset Voltage at VFX</td>
<td>±5</td>
<td>mV</td>
<td>VF&lt;sub&gt;X&lt;/sub&gt; Voltage Required to Produce all “1s” Code at D&lt;sub&gt;X&lt;/sub&gt;</td>
</tr>
<tr>
<td>VOL</td>
<td>Output Low Voltage at AUTO</td>
<td>V&lt;sub&gt;BB&lt;/sub&gt; (V&lt;sub&gt;BB&lt;/sub&gt; + 2)</td>
<td>V</td>
<td>400 KΩ to GRDA</td>
</tr>
<tr>
<td>VOH</td>
<td>Output High Voltage at AUTO</td>
<td>(V&lt;sub&gt;CC&lt;/sub&gt;−2)</td>
<td>V&lt;sub&gt;CC&lt;/sub&gt;</td>
<td>400 KΩ to GRDA</td>
</tr>
</tbody>
</table>

POWER DISSIPATION

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Parameter</th>
<th>Limits</th>
<th>Units</th>
<th>Test Conditions</th>
</tr>
</thead>
<tbody>
<tr>
<td>I&lt;sub&gt;DDO&lt;/sub&gt;</td>
<td>Standby Current</td>
<td>0.7</td>
<td>1.1</td>
<td>mA</td>
</tr>
<tr>
<td>I&lt;sub&gt;CCO&lt;/sub&gt;</td>
<td>Standby Current</td>
<td>4</td>
<td>7.0</td>
<td>mA</td>
</tr>
<tr>
<td>I&lt;sub&gt;BBO&lt;/sub&gt;</td>
<td>Standby Current</td>
<td>1</td>
<td>2.5</td>
<td>mA</td>
</tr>
<tr>
<td>I&lt;sub&gt;DDI&lt;/sub&gt;</td>
<td>Operating Current</td>
<td>11</td>
<td>16</td>
<td>mA</td>
</tr>
<tr>
<td>I&lt;sub&gt;CCI&lt;/sub&gt;</td>
<td>Operating Current</td>
<td>13</td>
<td>21</td>
<td>mA</td>
</tr>
<tr>
<td>I&lt;sub&gt;BBI&lt;/sub&gt;</td>
<td>Operating Current</td>
<td>4</td>
<td>6.0</td>
<td>mA</td>
</tr>
</tbody>
</table>

NOTE:
1. Typical values are for T<sub>A</sub> = 25°C and nominal power supply values.
### A.C. CHARACTERISTICS

$T_A = 0^\circ \text{C}$ to $+70^\circ \text{C}$, $V_{DPD} = +12\text{V} \pm 5\%$, $V_{CC} = +5\text{V} \pm 5\%$, $V_{BB} = -5\text{V} \pm 5\%$, $GRDA = 0\text{V}$, $GRDD = 0\text{V}$, unless otherwise specified.

#### TRANSMISSION

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Parameter Description</th>
<th>Limits</th>
<th>Unit</th>
<th>Test Conditions</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Min</td>
<td>Typ(1)</td>
<td>Max</td>
</tr>
<tr>
<td>S/D</td>
<td>Signal/Tone Distortion Ratio, C-Message Weighted, Half Channel (See Figure 1)</td>
<td>36</td>
<td>dB</td>
<td>VF$_X$ = 1.02 kHz, Sinusoid</td>
</tr>
<tr>
<td></td>
<td></td>
<td>30</td>
<td>dB</td>
<td>$-30\text{ dBm0} \leq VF_X \leq 0 \text{ dBm0}$</td>
</tr>
<tr>
<td></td>
<td></td>
<td>27</td>
<td>dB</td>
<td>$-40 \text{ dBm0} \leq VF_X &lt; -30 \text{ dBm0}$</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>$-45 \text{ dBm0} \leq VF_X &lt; -40 \text{ dBm0}$</td>
</tr>
<tr>
<td>$\Delta G$</td>
<td>Gain Tracking Deviation, Half Channel</td>
<td>$\pm 0.25$</td>
<td>$\pm 0.30$</td>
<td>dB</td>
</tr>
<tr>
<td>$\Delta G_V$</td>
<td>$\Delta G$ Variation with Supplies, Half Channel</td>
<td>$\pm 0.0002$</td>
<td>$\pm 0.0004$</td>
<td>dB/mV</td>
</tr>
<tr>
<td>$\Delta G_T$</td>
<td>$\Delta G$ Variation with Temperature, Half Channel</td>
<td>$\pm 0.001$</td>
<td>$\pm 0.002$</td>
<td>dB/$^\circ\text{C}$</td>
</tr>
<tr>
<td>$N_{IC1}$</td>
<td>Idle Channel Noise, C-Message Weighted</td>
<td>2</td>
<td>7</td>
<td>dBmO</td>
</tr>
<tr>
<td>$N_{IC2}$</td>
<td>Idle Channel Noise, C-Message Weighted</td>
<td>10</td>
<td>13</td>
<td>dBmO</td>
</tr>
<tr>
<td>$N_{IC3}$</td>
<td>Idle Channel Noise, C-Message Weighted</td>
<td>14</td>
<td>18</td>
<td>dBmO</td>
</tr>
<tr>
<td>HD</td>
<td>Harmonic Distortion (2nd or 3rd)</td>
<td>$-48$</td>
<td>$-44$</td>
<td>dB</td>
</tr>
<tr>
<td>IMD</td>
<td>Intermodulation Distortion 2nd Order, 3rd Order</td>
<td>$-45$</td>
<td>$-55$</td>
<td>dB</td>
</tr>
</tbody>
</table>

**NOTES:**

1. Typical values are for $T_A = 25^\circ \text{C}$ and nominal supply values.
2. Measured in one direction, either decoder or encoder and an ideal device, at 23$^\circ \text{C}$, nominal supplies.
3. If the external auto-zero is used $N_{IC1}$ has a typical value of 8 dBm0 and $N_{IC2}$ has a typical value of 13 dBm0.
5. With the D.C. method the positive and negative clipping levels are measured and $A_{IR}$ is calculated. With the A.C. method a sinusoidal input signal to VF$_X$ is used where $A_{IR}$ is measured directly.

![Figure 13. Signal/Total Distortion Ratio (Half-Channel)](image1)

![Figure 14. Gain Tracking Deviation ($\Delta G$) (Half-Channel)](image2)
A.C. CHARACTERISTICS

\( T_A = 0^\circ \text{C} \) to \( +70^\circ \text{C} \), \( V_{DD} = +12 \text{V} \pm 5\% \), \( V_{CC} = +5 \text{V} \pm 5\% \), \( V_{BB} = -5 \text{V} \pm 5\% \), \( GRDA = 0 \text{V} \), \( GRDD = 0 \text{V} \), unless otherwise specified (Continued)

**GAIN AND DYNAMIC RANGE**

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Parameter</th>
<th>Limits</th>
<th>Unit</th>
<th>Test Conditions</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Min</td>
<td>Typ(1)</td>
<td>Max</td>
</tr>
<tr>
<td>DmW</td>
<td>Digital MilliWatt Response</td>
<td>5.53</td>
<td>5.63</td>
<td>5.73</td>
</tr>
<tr>
<td>DmWT</td>
<td>DmWO Variation with Temperature</td>
<td>(-0.001)</td>
<td>(-0.002)</td>
<td></td>
</tr>
<tr>
<td>DmWS</td>
<td>DmWO Variation with Supplies</td>
<td>(\pm 0.07)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>AIR</td>
<td>Input Dynamic Range</td>
<td>2.17</td>
<td>2.20</td>
<td>2.23</td>
</tr>
<tr>
<td>AIRT</td>
<td>Input Dynamic Range with Temperature</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>AIRS</td>
<td>Input Dynamic Range with Supplies</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>AOR</td>
<td>Output Dynamic Range, ( V_{FR} )</td>
<td>2.13</td>
<td>2.16</td>
<td>2.19</td>
</tr>
<tr>
<td>AORT</td>
<td>AOR Variation with Temperature</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>AORS</td>
<td>AOR Variation with Supplies</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**SUPPLY REJECTION AND CROSSTALK**

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Parameter</th>
<th>Limits</th>
<th>Unit</th>
<th>Test Conditions</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Min</td>
<td>Typ(1)</td>
<td>Max</td>
</tr>
<tr>
<td>PSRR₁</td>
<td>( V_{DD} ) Power Supply Rejection Ratio</td>
<td>45</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PSRR₂</td>
<td>( V_{BB} ) Power Supply Rejection Ratio</td>
<td>35</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PSRR₃</td>
<td>( V_{CC} ) Power Supply Rejection Ratio</td>
<td>50</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PSRR₄</td>
<td>( V_{DD} ) Power Supply Rejection Ratio</td>
<td>50</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PSRR₅</td>
<td>( V_{BB} ) Power Supply Rejection Ratio</td>
<td>45</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PSRR₆</td>
<td>( V_{CC} ) Power Supply Rejection Ratio</td>
<td>50</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CTᵣ</td>
<td>Crosstalk Isolation, Receive Side</td>
<td>75</td>
<td>80</td>
<td></td>
</tr>
<tr>
<td>CTₜ</td>
<td>Crosstalk Isolation, Transmit Side</td>
<td>75</td>
<td>80</td>
<td></td>
</tr>
<tr>
<td>CAPₓ</td>
<td>Input Sample and Hold Capacitor</td>
<td>1600</td>
<td>200</td>
<td>2400</td>
</tr>
</tbody>
</table>

**NOTES:**

1. Typical values are for \( T_A = 25^\circ \text{C} \) and nominal power supply values.
2. D.U.T. decoder; impose 200 mVpp, 1.02 KHz on appropriate supply; measurement made at decoder output; decoder in idle channel conditions.
3. D.U.T. encoder; impose 200 mVpp, 1.02 KHz on appropriate supply; measurement made at encoder output; encoder in idle channel conditions.
4. \( VF_x \) of D.U.T. encoder = 1.02 KHz, 0 dBm0. Decoder under quiet channel conditions; measurement made at decoder output.
5. \( VF_x \) = 0 Vrms. Decoder = 1.02 KHz, 0 dBm0. Encoder under quiet channel conditions; measurement made at encoder output.
A.C. CHARACTERISTIC—TIMING SPECIFICATION(1)

$T_A = 0^\circ C$ to $+70^\circ C$, $V_{DD} = +12V \pm 5\%$, $V_{CC} = +5V \pm 5\%$, $V_{BB} = -5V \pm 5\%$, $GRDA = 0V$, $GRDD = 0V$, unless otherwise specified.

### CLOCK SECTION

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Parameter</th>
<th>Limits</th>
<th>Units</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>$t_{CY}$</td>
<td>Clock Period</td>
<td>485</td>
<td>ns</td>
<td>$CLK_{X}$, $CLK_{R}$ (2.048 MHz Systems), $CLK_{C}$</td>
</tr>
<tr>
<td>$t_{r}, t_{f}$</td>
<td>Clock Rise and Fall Time</td>
<td>0-30</td>
<td>ns</td>
<td>$CLK_{X}$, $CLK_{R}$, $CLK_{C}$</td>
</tr>
<tr>
<td>$t_{CLK}$</td>
<td>Clock Pulse Width</td>
<td>215</td>
<td>ns</td>
<td>$CLK_{X}$, $CLK_{R}$, $CLK_{C}$</td>
</tr>
<tr>
<td>$t_{DCD}$</td>
<td>Clock Duty Cycle ($t_{CLK} + t_{CY}$)</td>
<td>45-55</td>
<td>%</td>
<td>$CLK_{X}$, $CLK_{R}$</td>
</tr>
</tbody>
</table>

### TRANSMIT SECTION

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Parameter</th>
<th>Limits</th>
<th>Units</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>$t_{VFX}$</td>
<td>Analog Input Conversion</td>
<td>20</td>
<td>Timeslot</td>
<td>from Leading Edge of Transmit Timeslot (2)</td>
</tr>
<tr>
<td>$t_{DZX}$</td>
<td>Data Enabled on TS Entry</td>
<td>50-180</td>
<td>ns</td>
<td>$0 &lt; C_{LOAD} &lt; 100 \text{ pF}$</td>
</tr>
<tr>
<td>$t_{DHX}$</td>
<td>Data Hold Time</td>
<td>80-230</td>
<td>ns</td>
<td>$0 &lt; C_{LOAD} &lt; 100 \text{ pF}$</td>
</tr>
<tr>
<td>$t_{HZX}$</td>
<td>Data Float on TS Exit</td>
<td>75-245</td>
<td>ns</td>
<td>$C_{LOAD} = 0$</td>
</tr>
<tr>
<td>$t_{SON}$</td>
<td>Timeslot X to Enable</td>
<td>30-220</td>
<td>ns</td>
<td>$0 &lt; C_{LOAD} &lt; 100 \text{ pF}$</td>
</tr>
<tr>
<td>$t_{SOFF}$</td>
<td>Timeslot X to Disable</td>
<td>70-225</td>
<td>ns</td>
<td>$C_{LOAD} = 0$</td>
</tr>
<tr>
<td>$t_{SS}$</td>
<td>Signal Setup Time</td>
<td>0</td>
<td>ns</td>
<td>Relative to Bit 7 Falling Edge</td>
</tr>
<tr>
<td>$t_{SH}$</td>
<td>Signal Hold Time</td>
<td>100</td>
<td>ns</td>
<td>Relative to Bit 8 Falling Edge</td>
</tr>
<tr>
<td>$t_{FSD}$</td>
<td>Frame Sync Delay</td>
<td>15-150</td>
<td>ns</td>
<td></td>
</tr>
</tbody>
</table>

### RECEIVE AND CONTROL SECTIONS

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Parameter</th>
<th>Limits</th>
<th>Units</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>$t_{VFR}$</td>
<td>Analog Output Update</td>
<td>9 $1/16$-9 $1/16$</td>
<td>Timeslot</td>
<td>from Leading Edge of the Channel Timeslot</td>
</tr>
<tr>
<td>$t_{DSR}$</td>
<td>Receive Data Setup</td>
<td>20</td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>$t_{DHR}$</td>
<td>Receive Data Hold</td>
<td>60</td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>$t_{SIGR}$</td>
<td>SIGR Update</td>
<td>1</td>
<td>$\mu$s</td>
<td>from Trailing Edge of the Channel Timeslot</td>
</tr>
<tr>
<td>$t_{FSD}$</td>
<td>Frame Sync Delay</td>
<td>15-150</td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>$t_{DSC}$</td>
<td>Control Data Setup</td>
<td>115</td>
<td>ns</td>
<td>Microcomputer Mode Only</td>
</tr>
<tr>
<td>$t_{DCH}$</td>
<td>Control Data Hold</td>
<td>115</td>
<td>ns</td>
<td>Microcomputer Mode Only</td>
</tr>
</tbody>
</table>

**NOTES:**

1. All timing parameters referenced to 1.5V, except $t_{DZX}$ and $t_{SOFF}$ which reference to high impedance state.
2. The 20 timeslot minimum insures that the complete A/D conversion will take place under any combination of receive interrupt or asynchronous operation of the Codec. If the transmit channel only is operated, the A/D conversion can be completed in a minimum of 11 timeslots. Refer to the Codec Control General Requirement section for instructions on setting a channel in an idle condition.
TIMING WAVEFORMS (Continued)

CONTROL TIMING
**2911A-1**

**PCM CODEC—A LAW**

**8-BIT COMPANDED A/D AND D/A CONVERTER**

- Per Channel, Single Chip Codec
- CCITT G711 and G732 Compatible, Even Order Bits Inversion Included
- Microcomputer Interface with On-Chip Time-Slot Computation
- Simple Direct Mode Interface When Fixed Timeslots Are Used
- ±5% Power Supplies: +12V, +5V, −5V
- 66 dB Dynamic Range, with Resolution Equivalent to 11-Bit Linear Conversion Around Zero
- Precision On-Chip Voltage Reference
- Low Power Consumption 230 mW Typ. Standby Power 33 mW Typ.
- Fabricated with Reliable N-Channel MOS Process

The Intel 2911A is a fully integrated PCM (Pulse Code Modulation) Codec (Coder-Decoder), fabricated with N-channel silicon gate technology. The high density of integration allows the sample and hold circuits, the digital-to-analog converter, the comparator and the successive approximation register to be integrated on the same chip, along with the logic necessary to interface a full duplex PCM link.

The primary applications are in telephone systems:

- Transmission — 30/32 Channel Systems at 2.048 Mbps
- Switching — Digital PBX's and Central Office Switching Systems
- Concentration — Subscriber Carrier/Concentrators

The wide dynamic range of the 2911A (66 dB) and the minimal conversion time (80 μs minimum) make it an ideal product for other applications, like:

- Data Acquisition
- Secure Communications Systems
- Telemetry
- Signal Processing Systems

---

**Figure 1. Pin Configuration**

<table>
<thead>
<tr>
<th>Pin</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>CAP1x, CAP2x</td>
<td>Holding Capacitor</td>
</tr>
<tr>
<td>V福</td>
<td>Analog Input</td>
</tr>
<tr>
<td>V福</td>
<td>Analog Output</td>
</tr>
<tr>
<td>D福, DDC</td>
<td>Digital Input</td>
</tr>
<tr>
<td>D福, TSK</td>
<td>Digital Output</td>
</tr>
<tr>
<td>CLKc, CLKb, CLKR</td>
<td>Clock Input</td>
</tr>
<tr>
<td>FSX, FSR</td>
<td>Frame Sync Input</td>
</tr>
<tr>
<td>AUTO</td>
<td>Auto Zero Output</td>
</tr>
<tr>
<td>V福8</td>
<td>Power (−5V)</td>
</tr>
<tr>
<td>V福c</td>
<td>Power (+5V)</td>
</tr>
<tr>
<td>V福D</td>
<td>Power (+12V)</td>
</tr>
<tr>
<td>PDN</td>
<td>Power Down</td>
</tr>
<tr>
<td>GRDA</td>
<td>Analog Ground</td>
</tr>
<tr>
<td>GRDD</td>
<td>Digital Ground</td>
</tr>
<tr>
<td>NC</td>
<td>No Connect</td>
</tr>
</tbody>
</table>

*Figure 2. Pin Names*
<table>
<thead>
<tr>
<th>Pin No</th>
<th>Symbol</th>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>CAP1X</td>
<td>Hold</td>
<td>Connections for the transmit holding capacitor. Refer to Applications section.</td>
</tr>
<tr>
<td>2</td>
<td>CAP2X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>VFx</td>
<td>Input</td>
<td>Analog input to be encoded into a PCM word. The signal on this lead is sampled at the same rate as the transmit frame synchronization pulse FSx, and the sample value is held in the external capacitor connected to the CAP1X and CAP2X leads until the encoding process is completed.</td>
</tr>
<tr>
<td>4</td>
<td>AUTO</td>
<td>Output</td>
<td>Most significant bit of the encoded PCM word (+5V for negative, −5V for positive values). Refer to the Codec Applications section.</td>
</tr>
<tr>
<td>5</td>
<td>GRDA</td>
<td>Ground</td>
<td>Analog return common to the transmit and receive analog circuits. Not connected to GRDD internally.</td>
</tr>
<tr>
<td>6</td>
<td>VDD</td>
<td>Power</td>
<td>+12V ±5%; referenced to GRDA.</td>
</tr>
<tr>
<td>7</td>
<td>DR</td>
<td>Input</td>
<td>Receive PCM highway (serial bus) interface. The Codec serially receives a PCM word (8 bits) through this lead at the proper time defined by FSR, CLKR, DC, and CLKe.</td>
</tr>
<tr>
<td>8</td>
<td>PDN</td>
<td>Output</td>
<td>Active high when the Codec is in the power down state. Open drain output.</td>
</tr>
<tr>
<td>9</td>
<td>VFR</td>
<td>Output</td>
<td>Analog Output. The voltage present on VFR is the decoded value of the PCM word received on lead DR. This value is held constant between two conversions.</td>
</tr>
<tr>
<td>10</td>
<td>NC</td>
<td>No Connects</td>
<td>Recommended practice is to strap these NC's to GRDA.</td>
</tr>
<tr>
<td>11</td>
<td>NC</td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>GRDD</td>
<td>Ground</td>
<td>Ground return common to the logic power supply; VCC.</td>
</tr>
<tr>
<td>13</td>
<td>DX</td>
<td>Output</td>
<td>Output of the transmit side onto the send PCM highway (serial bus). The 8-bit PCM word is serially sent out on this pin at the proper time defined by FSX, CLKX, DC, and CLKe. TTL three-state output.</td>
</tr>
<tr>
<td>14</td>
<td>TSX</td>
<td>Output</td>
<td>Normally high, this signal goes low while the Codec is transmitting an 8-bit PCM word on the DX lead. (Timeslot information used for diagnostic purposes and also to gate the data on the DX lead.) Open drain output.</td>
</tr>
<tr>
<td>15</td>
<td>VCC</td>
<td>Power</td>
<td>+5V ±5%, referenced to GRDD.</td>
</tr>
<tr>
<td>16</td>
<td>CLKR</td>
<td>Input</td>
<td>Master receive clock defining the bit rate on the receive PCM highway. Typically 2.048 Mbps for a carrier system. Maximum rate 2.1 Mbps. 50% duty cycle. TTL compatible.</td>
</tr>
<tr>
<td>17</td>
<td>FSR</td>
<td>Input</td>
<td>Frame synchronization pulse for the receive PCM highway. Resets the on-chip timeslot counter for the receive side. Maximum repetition rate 12 KHz. TTL interface.</td>
</tr>
<tr>
<td>18</td>
<td>CLKX</td>
<td>Input</td>
<td>Master transmit clock defining the bit rate on the transmit PCM highway. Typically 2.048 Mbps for a carrier system. Maximum rate 2.1 Mbps. 50% duty cycle. TTL interface.</td>
</tr>
<tr>
<td>19</td>
<td>FSX</td>
<td>Input</td>
<td>Frame synchronization pulse for the transmit PCM highway. Resets the on-chip timeslot counter for the transmit side. Maximum repetition rate 12 KHz. TTL interface.</td>
</tr>
<tr>
<td>20</td>
<td>VBB</td>
<td>Power</td>
<td>−5V ±5%, referenced to GRDA.</td>
</tr>
<tr>
<td>21</td>
<td>DC</td>
<td>Input</td>
<td>Data input to program the Codec for the chosen mode of operation. Becomes an active low chip select when CLKC is tied to VCC. TTL interface.</td>
</tr>
<tr>
<td>22</td>
<td>CLKC</td>
<td>Input</td>
<td>Clock input to clock in the data on the DC lead when the timeslot assignment feature is used; tied to VCC to disable this feature. TTL interface.</td>
</tr>
</tbody>
</table>
FUNCTIONAL DESCRIPTION

The 2911A PCM Codec provides the analog-to-digital and the digital-to-analog conversions necessary to interface a full duplex (4 wires) voice telephone circuit with the PCM highways of a time division multiplexed (TDM) system. The Codec is intended to be used on line and trunk terminations.

In a typical telephone system the Codec is located between the PCM highways and the channel filters.

The Codec encodes the incoming analog signal at the frame rate ($FS_X$) into an 8-bit PCM word which is sent out on the $D_X$ lead at the proper time. Similarly, on the receive link, the Codec fetches an 8-bit PCM word from the receive highway ($D_R$ lead) and decodes an analog value which will remain constant on lead $VFR$ until the next receive frame. Transmit and receive frames are independent. They can be asynchronous (transmission) or synchronous (switching) with each other.

Circuitry is provided within the Codec to internally define the transmit and receive timeslots. In small systems this may eliminate the need for any external timeslot exchange; in large systems it provides one level of concentration. This feature can be bypassed and discrete timeslots sent to each Codec within a system.

In the power-down mode, most functions of the Codec are directly disabled to reduce power dissipation to a minimum.

CODEC OPERATION

Codec Control

The operation of the 2911A is defined by serially loading an 8-bit word through the $D_C$ lead (data) and the $CLK_C$ lead (clock). The loading is synchronous with the other operations of the Codec, and takes place whenever transitions occur on the $CLK_C$ lead. The $D_C$ input is loaded in during the trailing edge of the $CLK_C$ input.

The control word contains two fields:

Bit 1 and Bit 2 define whether the subsequent 6 bits apply to both the transmit and receive side (00), the transmit side only (01), the receive side only (10), or whether the Codec should go into the standby, power-down mode (11). In the last case (11), the following 6 bits are irrelevant.

Figure 4. Typical Line Termination

6-21
The last 6 bits of the control word define the timeslot assignment, from 000000 (timeslot 1) to 111111 (timeslot 64). Bit 3 is the most significant bit and bit 8 the least significant bit and last into the Codec.

<table>
<thead>
<tr>
<th>Bit 1</th>
<th>Bit 2</th>
<th>Mode</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>X &amp; R</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>X</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>R</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>Standby</td>
</tr>
</tbody>
</table>

The Codec will retain the control word (or words) until a new word is loaded in or until power is lost. This feature permits dynamic allocation of timeslots for switching applications.

**Microcomputer Control Mode**

In the microcomputer mode, each Codec performs its own timeslot computation independently for the transmit and receive channels by counting clock pulses (CLK<sub>X</sub> and CLK<sub>R</sub>). All Codecs tied to the same data bus receive identical framing pulses (FS<sub>X</sub> and FS<sub>R</sub>). The framing pulses reset the on-chip timeslot counters every frame; hence the timeslot counters of all devices are synchronized. Each Codec is programmed via CLK<sub>C</sub> and DC for the desired transmit and receive timeslots according to the description in the Codec Control Section. All Codecs tied to the same D<sub>R</sub> bus will, in general, have different receive timeslots, although that is not a device requirement. There may be separate busses for transmit and receive or all Codecs may transmit and receive over the same bus, in which case the transmit and receive channels must be synchronous (CLK<sub>X</sub> = CLK<sub>R</sub>). There are no other restrictions on timeslot assignments; a device may have the same transmit and receive timeslot even if a single bus is used.

There are several requirements for using the CLK<sup>C</sup> - DC interface in the microcomputer mode:

1. A complete timeslot assignment, consisting of eight negative transitions of CLK<sup>C</sup>, must be made in less than one frame period. The assignment can overlap a framing pulse so long as all 8 control bits are clocked in within a total span of 125 µs (for an 8 KHz frame rate). CLK<sup>C</sup> must be left at a TTL low level when not assigning a timeslot.

2. A dead period of two frames must always be observed between successive timeslot assignments. The two frame delay is measured from the rising edge of the first CLK<sup>C</sup> transition of the previous timeslot assigned.

3. When the device is in the power-down state (Standby), the following three-step sequence must be followed to power-up the codec to avoid contention on the transmit PCM highway.

   a. Assign a dummy transmit timeslot. The dummy should be at least two timeslots greater than the maximum valid system (usually 24 or 32). For example, in a 24 timeslot system, the dummy could be any timeslot between 26 and 64. This will power-up the transmit side, but prevent any spurious Dx or TSx outputs.

   b. Two frames later, assign the desired transmit timeslot.

   c. Two frames later assign the desired receive timeslot.

4. Initialization sequence: The device contains an on-chip power-on clear function which guarantees that with proper sequencing of the supplies (V<sub>CC</sub> or V<sub>DD</sub> on last), the device will initialize with no timeslot assigned to either the transmit or receive channel. After a supply failure or whenever the supplies are applied, it is recommended that either power down assignment be made first, or the first timeslot assignment be a transmit timeslot or a transmit/receive timeslot. The consequence of making a receive timeslot assignment first, after supply application, is that the transmit channel will assume timeslot 1, potentially producing bus contention.

5. Transmit only/receive only operation is permitted provided that a power down assignment is made first. Otherwise, special circuits which use only one channel should be physically disconnected from the unused bus; this allows a timeslot to be made to an unused channel without consequence.

6. Both frame synchronizing pulses (FS<sub>X</sub>, FS<sub>R</sub>) must be active at all times after power on clear (after power supplies are turned on). This requirement must be met during powerdown and receive only or transmit only operation, as well as during normal transmit and receive operation.

Example of Microcomputer Control Mode:

The two words 01000001 and 10000010 have been loaded into the Codec. The transmit side is now programmed for timeslot 2 and the receive side for
timeslot 3. The Codec will output a PCM word on the transmit PCM highway (bus) during the timeslot 2 of the transmit frame, and will fetch a PCM word from the receive PCM highway during timeslot 3.

![Figure 5. Microcomputer Mode Programming Examples](image)

In this example the Codec interface to the PCM highway then functions as shown below. (FSx and FSR may be asynchronous.)

![Figure 6. Microcomputer Mode PCM Highway Example](image)

**Direct Control Mode**

The direct mode of operation will be selected when the CLKC pin is strapped to the +5V supply (VCC). In this mode, the DC pin is an active low chip select. In other words, when DC is low, the device transmits and receives in the timeslots which follow the appropriate framing pulses. With DC high the device is in the power down state. Even though CLKC characteristics are simpler for the 2911A it will operate properly when plugged into a 2911 board.

Deactivation of a channel by removal of the appropriate framing pulse (FSx or FSR) is not permitted.

Specifically, framing pulses must be applied for a minimum of two frames after a change in state of DC in order for the DC change to be internally sensed. In particular, when entering standby in the direct mode, framing pulses must be applied as usual for two frames after DC is brought high.

The Codec will enter direct mode within three frame times (375 μs) as measured from the time the device power supplies settle to within the specified limits. This assumes that CLKC is tied to VCC and that all clocks are available at the time the supplies have settled.

**General Control Requirements**

All bit and frame clocks should be applied whenever the device is active. In particular, an unused channel cannot be deactivated by removal of its associated frame or bit clock while the other channel of the same device remains active.

A single channel cannot be deactivated except by physical disconnection of the data lead (DX or DR) from the system data bus. A device (both transmit and receive channels) may be deactivated in either control mode by powering down the device. Both channels are always powered down together.
Encoding

The VF signal to be encoded is input on the VFx lead. An internal switch samples the signal and the hold function is performed by the external capacitor connected to the CAP1x and CAP2x leads. The sampling and conversion is synchronized with the transmit timeslot. The PCM word is then output on the Dx lead at the proper timeslot occurrence of the following frame. The A/D converter saturates at approximately ±2.2V RMS (±3.1 volts peak).

![Figure 7. Transmit Encoding](image)

Decoding

The PCM word is fetched by the DR lead from the PCM highway at the proper timeslot occurrence. The decoded value is held on an internal sample and hold capacitor. The buffered non-return to zero output signal on the VFR lead has a dynamic range of ±2.2V RMS (±3.1 volts peak).

Standby Mode—Power Down

To minimize power consumption and heat dissipation a standby mode is provided where all Codec functions are disabled except for DC and CLKC leads. These allow the Codec to be reactivated. In the microcomputer mode the Codec is placed into standby by loading a control word (DC) with a “1” in bits 1 and 2 locations. In the direct mode when DC is brought high, the all “1’s” control word is internally transferred to the control register, invoking the standby condition.

While in the standby mode, the DX output is actively held in a high impedance state to guarantee that the PCM bus will not be driven.

The power consumption in the standby mode is typically 33 mW.

Power-On Clear

Whether the device is used in the direct or microcomputer mode, an internal reset (power-on clear) is generated, forcing the device into the power down state, when power is supplied by any of the following methods. (1) Device power supplies are turned on in a system power-up situation where either VCC or VDD is applied last. (2) A large supply transient causes either of the two positive supplies to drop to approximately 2V. (3) A board containing Codecs is plugged into a “hot” system where VCC or VDD is the last contact made. It may be necessary to trim back the edge connector pins or fingers on VCC or VDD relative to the other supply to guarantee that the power-on clear will operate properly when a board is plugged into a “hot” system. Furthermore, the Codec will inhibit activity on TSX and DX during the application of power supplies.

The device is also tolerant of transients in the negative supply (VBB) so long as VBB remains more negative than −3.5V. VBB transients which exceed this level should be detected and followed by a system reinitialization.

Precision Voltage Reference for the D/A Converter

The voltage reference is generated on the chip and is calibrated during the manufacturing process. The technique uses the difference in sub-surface charge density between two suitably implanted MOS devices to derive a temperature stable and bias stable reference voltage.
A gain setting op amp, programmed during manufac-
turing, "trims" the reference voltage source to the
final precision voltage reference value provided to
the D/A converter. The precision voltage reference
determines the initial gain and dynamic range char-
acteristics described in the A.C. Transmission Speci-
fication section.

CONVERSION LAW

The conversion law is commonly referred to as the A
Law.

\[
F(x) = \begin{cases} 
Sgn(x) \left[ \frac{1 + \log_{10}(A|x|)}{1 + \log_{10} A} \right], & 0 \leq |x| \leq 1/A \\
Sgn(x) \left[ \frac{A|x|}{1 + \log_{10} A} \right], & 1/A \leq |x| \leq 1
\end{cases}
\]

where: \( x \) = the input signal

\( Sgn(x) = \) sign of the input signal

\( A = 87.6 \) (defined by CCITT)

The Codec provides a piecewise linear approxima-
tion of the logarithmic law through 13 segments. Each segment is made of 16 steps with the excep-
tion of the first segment, which has 32 steps. In adja-
cent segments the step sizes are in a ratio of two to
one. Within each segment, the step size is constant.

The output levels are midway between the corre-
sponding decision levels. The output levels \( y_n \) are
related to the input levels \( x_n \) by the expression:

\[
y_n = \frac{x_{n-1} + x_n}{2}, \quad 0 < n \leq 128
\]
## Theoretical A-Law—Positive input Values (for Negative input Values, Invert Bit 1)

<table>
<thead>
<tr>
<th>Segment Number</th>
<th>No. of Steps x Step Size</th>
<th>Value at Segment End Points</th>
<th>Decision Value Number n</th>
<th>Decision Value $x_n(1)$</th>
<th>PCM Word(4)</th>
<th>Normalized Value at Decoder Output $y_n(5)$</th>
<th>Decoder Output Value Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>16 x 128</td>
<td>4096(3)</td>
<td>127</td>
<td>3968</td>
<td>1 1 1 1 1 1 1 1</td>
<td>4032</td>
<td>128</td>
</tr>
<tr>
<td>6</td>
<td>16 x 64</td>
<td>2048</td>
<td>112</td>
<td>2048</td>
<td>1 1 1 1 0 0 0 0</td>
<td>2112</td>
<td>113</td>
</tr>
<tr>
<td>5</td>
<td>16 x 32</td>
<td>1024</td>
<td>96</td>
<td>1024</td>
<td>1 1 1 0 0 0 0 0</td>
<td>1056</td>
<td>97</td>
</tr>
<tr>
<td>4</td>
<td>16 x 16</td>
<td>512</td>
<td>80</td>
<td>512</td>
<td>1 1 0 1 0 0 0 0</td>
<td>528</td>
<td>81</td>
</tr>
<tr>
<td>3</td>
<td>16 x 8</td>
<td>256</td>
<td>64</td>
<td>256</td>
<td>1 1 0 0 0 0 0 0</td>
<td>264</td>
<td>65</td>
</tr>
<tr>
<td>2</td>
<td>16 x 4</td>
<td>128</td>
<td>48</td>
<td>128</td>
<td>1 0 1 1 0 0 0 0</td>
<td>132</td>
<td>49</td>
</tr>
<tr>
<td>1</td>
<td>32 x 2</td>
<td>64</td>
<td>32</td>
<td>64</td>
<td>1 0 1 0 0 0 0 0</td>
<td>66</td>
<td>33</td>
</tr>
</tbody>
</table>

### NOTES:

1. 4096 normalized value units correspond to the value of the on-chip voltage reference.
2. The PCM word corresponding to positive input values between two successive decision values numbered $n$ and $n + 1$ (see column 4) is $(128 + n)$ expressed as a binary number.
3. $x_{128}$ is a virtual decision value.
4. The PCM word on the highways is the same as the one shown in column 6, with the even order bits inverted. The 2911A provides for the inversion of the even order bits on both the send and receive sections.
5. The voltage output on the VFR lead is equal to the normalized value given in the table, augmented by an offset. The offset value is approximately 15 mV.
APPLICATIONS

Holding Capacitor

For an 8 KHz sampling system the transmit holding capacitor \( \text{CAP}_x \) should be 2000 pF \( \pm \) 20%.

![Figure 11. Circuit Interface—Without External Auto Zero](image)

Filters Interface

The filters may be interfaced as shown in the circuit interface diagrams. Note that the output pulse stream is of the non-return-to-zero type and hence requires the \((\sin x)/x\) correction provided by the 2912A filter.

DX Buffering

For higher drive capability or increased system reliability it may be desirable that the \( \text{DS}_x \) output of a group of Codecs be buffered from the system PCM bus with an external three-state or open collector buffers. A buffer can be enabled with the appropriate Codec generated \( \text{TS}_x \) signal or signals. \( \text{TS}_x \) signal may also be used to activate external zero code suppression logic, since the occurrence of an active state of any \( \text{TS}_x \) implies the existence of PCM voice bits (as opposed to transparent data bits) on the bus.

![Figure 12. Circuit Interface—With External Auto Zero](image)

Auto Zero

The 2911A contains a transparent on-chip auto zero plus a device pin for implementing a sign-bit driven external auto zero feedback loop. The on-chip auto zero reduces the input offset voltage of the encoder \( \text{VF}_x \) to less than 3 mV. For most telephony applications, this input offset is perfectly acceptable, since it insures the encoder is biased in the lower 25% of the first segment.

Where lower input offset is required the external auto zero loop may be used to bias the encoder exactly at the zero crossing point. The consequence of the external auto zero loop, aside from extra components, is the addition of the dithering auto-zero signal to the input signal, resulting in slightly higher idle channel noise (approximately 2 dB) than when the external loop is not used. Consequently, where the application permits, it is recommended that the external auto zero loop not be used. When not used, the AUTO pin should float.

The circuit interface with external auto zero drawing shows a possible connection between \( \text{VF}_x \) and AUTO leads with the recommended values of \( C_1 = 0.3 \ \mu \text{F} \), \( R_1 = 150 \ \text{K}\Omega \), \( R_2 = 330 \Omega \), and \( R_3 = 470 \ \text{K}\Omega \).
**ABSOLUTE MAXIMUM RATINGS**

- Temperature Under Bias: −10°C to +80°C
- Storage Temperature: −65°C to +150°C
- All Input or Output Voltages with Respect to $V_{BB} = -0.3V$ to +20V
- $V_{CC}, V_{DD}, GRDA, and GRDA$ with Respect to $V_{BB} = -0.3V$ to +20V
- Power Dissipation: 1.35W

*Notice: Stresses above those listed under “Absolute Maximum Ratings” may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions above those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.*

**D.C. CHARACTERISTICS**

- $T_A = 0°C$ to +70°C, $V_{DD} = +12V$ ±5%, $V_{CC} = +5V$ ±5%, $V_{BB} = -5V$ ±5%, GRDA = 0V, GRDD = 0V, unless otherwise specified.

### DIGITAL INTERFACE

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Parameter</th>
<th>Limits</th>
<th>Unit</th>
<th>Test Conditions</th>
</tr>
</thead>
<tbody>
<tr>
<td>$I_{IL}$</td>
<td>Low Level Input Current</td>
<td>10 µA</td>
<td>$V_{IN} &lt; V_{IL}$</td>
<td></td>
</tr>
<tr>
<td>$I_{IH}$</td>
<td>High Level Input Current</td>
<td>10 µA</td>
<td>$V_{IN} &gt; V_{IH}$</td>
<td></td>
</tr>
<tr>
<td>$V_{IL}$</td>
<td>Input Low Voltage</td>
<td>0.6 V</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$V_{IH}$</td>
<td>Input High Voltage</td>
<td>2.2 V</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$V_{OL}$</td>
<td>Output Low Voltage</td>
<td>0.4 V</td>
<td>$D_X, I_{OL} = 4.0$ mA</td>
<td></td>
</tr>
<tr>
<td>$V_{OH}$</td>
<td>Output High Voltage</td>
<td>2.4 V</td>
<td>$D_X, I_{OH} = 15$ mA</td>
<td></td>
</tr>
</tbody>
</table>

### ANALOG INTERFACE

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Parameter</th>
<th>Limits</th>
<th>Unit</th>
<th>Test Conditions</th>
</tr>
</thead>
<tbody>
<tr>
<td>$Z_{AI}$</td>
<td>Input Impedance when Sampling, $V_{FX}$</td>
<td>125 300 500 Ω</td>
<td>In series with $CAP_X$ to GRDA, $-3.1V &lt; V_{IN} &lt; 3.1V$</td>
<td></td>
</tr>
<tr>
<td>$Z_{AO}$</td>
<td>Small Signal Output Impedance, $V_{FR}$</td>
<td>100 180 300 Ω</td>
<td>$-3.1V &lt; V_{OUT} &lt; 3.1V$</td>
<td></td>
</tr>
<tr>
<td>$V_{OR}$</td>
<td>Output Offset Voltage at $V_{FR}$</td>
<td>$-50$</td>
<td>50 mV</td>
<td>Minimum code to $D_R$</td>
</tr>
<tr>
<td>$V_{IX}$</td>
<td>Input Offset Voltage at $V_{FX}$</td>
<td>$-5$</td>
<td>5 mV</td>
<td>Minimum positive code produced at $D_X$</td>
</tr>
<tr>
<td>$V_{OL}$</td>
<td>Output Low Voltage at AUTO</td>
<td>$V_{BB}$</td>
<td>400 KΩ to GRDA</td>
<td></td>
</tr>
<tr>
<td>$V_{OH}$</td>
<td>Output High Voltage at AUTO</td>
<td>$(V_{CC} - 2)$</td>
<td>400 KΩ to GRDA</td>
<td></td>
</tr>
</tbody>
</table>
D.C. CHARACTERISTICS

\( T_A = 0^\circ \text{C} \) to \( +70^\circ \text{C} \), \( V_{DD} = +12V \pm 5\% \), \( V_{CC} = +5V \pm 5\% \), \( V_{BB} = -5V \pm 5\% \), \( GRDA = 0V \), \( GRDD = 0V \), unless otherwise specified. (Continued)

POWER DISSIPATION

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Parameter</th>
<th>Limits</th>
<th>Unit</th>
<th>Test Conditions</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Min</td>
<td>Typ(1)</td>
<td>Max</td>
</tr>
<tr>
<td>( I_{DD0} )</td>
<td>Standby Current</td>
<td>0.7</td>
<td>1.1</td>
<td>mA</td>
</tr>
<tr>
<td>( I_{CCO} )</td>
<td>Standby Current</td>
<td>4.0</td>
<td>7.0</td>
<td>mA</td>
</tr>
<tr>
<td>( I_{BB0} )</td>
<td>Standby Current</td>
<td>1.0</td>
<td>2.5</td>
<td>mA</td>
</tr>
<tr>
<td>( I_{DD1} )</td>
<td>Operating Current</td>
<td>11</td>
<td>16</td>
<td>mA</td>
</tr>
<tr>
<td>( I_{CC1} )</td>
<td>Operating Current</td>
<td>13</td>
<td>21</td>
<td>mA</td>
</tr>
<tr>
<td>( I_{BB1} )</td>
<td>Operating Current</td>
<td>4.0</td>
<td>6.0</td>
<td>mA</td>
</tr>
</tbody>
</table>

NOTE:
1. Typical values are for \( T_A = 25^\circ \text{C} \) and nominal power supply values.

A.C. CHARACTERISTICS

\( T_A = 0^\circ \text{C} \) to \( +70^\circ \text{C} \), \( V_{DD} = +12V \pm 5\% \), \( V_{CC} = +5V \pm 5\% \), \( V_{BB} = -5V \pm 5\% \), \( GRDA = 0V \), \( GRDD = 0V \), unless otherwise specified.

TRANSMISSION

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Parameter</th>
<th>Limits</th>
<th>Unit</th>
<th>Test Conditions</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Min</td>
<td>Typ(1)</td>
<td>Max</td>
</tr>
<tr>
<td>S/D</td>
<td>Signal to Total Distortion Ratio, ( \text{CCITT G.712 Method 2} ) (Half Channel)</td>
<td>37</td>
<td></td>
<td>dB</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Signal level 0 dBm0 to -30 dBm0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>31</td>
<td></td>
<td>dB</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Signal level to -40 dBm0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>26</td>
<td></td>
<td>dB</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Signal level to -45 dBm0</td>
</tr>
<tr>
<td>( \Delta G )</td>
<td>Gain Tracking Deviation Half Channel</td>
<td>( \pm 0.25 )</td>
<td>( \pm 0.30 )</td>
<td>dB</td>
</tr>
<tr>
<td></td>
<td>Reference Level -10 dBm0</td>
<td>( \pm 0.60 )</td>
<td>( \pm 0.70 )</td>
<td>dB</td>
</tr>
<tr>
<td></td>
<td></td>
<td>( \pm 1.5 )</td>
<td>( \pm 1.8 )</td>
<td>dB</td>
</tr>
<tr>
<td>( \Delta G_{V} )</td>
<td>( \Delta G ) Variation with Supplies Half Channel</td>
<td>( \pm 0.0002 )</td>
<td>( \pm 0.0004 )</td>
<td>dB/mV</td>
</tr>
<tr>
<td></td>
<td></td>
<td>( \pm 0.0004 )</td>
<td>( \pm 0.0008 )</td>
<td>dB/mV</td>
</tr>
<tr>
<td>( \Delta G_{T} )</td>
<td>( \Delta G ) Variation with Temperature Half Channel</td>
<td>( \pm 0.001 )</td>
<td>( \pm 0.002 )</td>
<td>dB/\text{C}</td>
</tr>
<tr>
<td></td>
<td></td>
<td>( \pm 0.002 )</td>
<td>( \pm 0.005 )</td>
<td>dB/\text{C}</td>
</tr>
<tr>
<td>( N_{IC} )</td>
<td>Idle Channel Noise</td>
<td>-85</td>
<td>-78</td>
<td>dBm0p</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Quiet Code. (Note 2)</td>
</tr>
<tr>
<td>( HD )</td>
<td>Harmonic Distortion (2nd or 3rd)</td>
<td>-48</td>
<td>-44</td>
<td>dB</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>( VF_X = 1.02 \text{ KHz, sinusoid} )</td>
</tr>
<tr>
<td>( IMD_{1} )</td>
<td>Intermodulation Distortion G.712(7.1)</td>
<td>-45</td>
<td></td>
<td>dB</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>CCITT G.712</td>
</tr>
<tr>
<td></td>
<td>( IMD_{2} )</td>
<td>G.712(7.2)</td>
<td>-50</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Two Tone Method</td>
</tr>
</tbody>
</table>
A.C. CHARACTERISTICS

$T_A = 0^\circ C$ to $+70^\circ C$, $V_{DD} = +12V \pm 5\%$, $V_{CC} = +5V \pm 5\%$, $V_{BB} = -5V \pm 5\%$, $GRDA = 0V$, $GRDD = 0V$, unless otherwise specified. (Continued)

GAIN AND DYNAMIC RANGE

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Parameter</th>
<th>Limits</th>
<th>Unit</th>
<th>Test Conditions</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Min</td>
<td>Typ(1)</td>
<td>Max</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$D_mW$</td>
<td>Digital Milliwatt Response</td>
<td>5.58</td>
<td>5.66</td>
<td>5.78</td>
</tr>
<tr>
<td>$D_{mW_T}$</td>
<td>$D_mW_0$ Variation with Temperature</td>
<td>$-0.001$</td>
<td>$-0.002$</td>
<td></td>
</tr>
<tr>
<td>$D_{mW_S}$</td>
<td>$D_mW_0$ Variation with Supplies</td>
<td>±0.07</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$A_{IR}$</td>
<td>Input Dynamic Range</td>
<td>2.183</td>
<td>2.213</td>
<td>2.243</td>
</tr>
<tr>
<td>$A_{IRT}$</td>
<td>Input Dynamic Range vs Temperature</td>
<td>−0.5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$A_{IRS}$</td>
<td>Input Dynamic Range vs Supplies</td>
<td>±18</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$A_{OR}$</td>
<td>Output Dynamic Range, $V_{FR}$</td>
<td>2.14</td>
<td>2.17</td>
<td>2.20</td>
</tr>
<tr>
<td>$A_{ORT}$</td>
<td>$A_{OR}$ Variation with Temperature</td>
<td>$-0.5$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$A_{ORS}$</td>
<td>$A_{OR}$ Variation with Supplies</td>
<td>±18</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

SUPPLY REJECTION AND CROSSTALK

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Parameter</th>
<th>Limits</th>
<th>Unit</th>
<th>Test Conditions</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Min</td>
<td>Typ(1)</td>
<td>Max</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$PSRR_1$</td>
<td>$V_{DD}$ Power Supply Rejection Ratio</td>
<td>45</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$PSRR_2$</td>
<td>$V_{BB}$ Power Supply Rejection Ratio</td>
<td>35</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$PSRR_3$</td>
<td>$V_{CC}$ Power Supply Rejection Ratio</td>
<td>50</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$PSRR_4$</td>
<td>$V_{DD}$ Power Supply Rejection Ratio</td>
<td>50</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$PSRR_5$</td>
<td>$V_{BB}$ Power Supply Rejection Ratio</td>
<td>45</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$PSRR_6$</td>
<td>$V_{CC}$ Power Supply Rejection Ratio</td>
<td>50</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$CT_R$</td>
<td>Crosstalk Isolation, Receive Side</td>
<td>75</td>
<td>80</td>
<td></td>
</tr>
<tr>
<td>$CT_T$</td>
<td>Crosstalk Isolation, Transmit Side</td>
<td>75</td>
<td>80</td>
<td></td>
</tr>
<tr>
<td>$CAPX$</td>
<td>Input Sample and Hold Capacitor</td>
<td>1600</td>
<td>2000</td>
<td>2400</td>
</tr>
</tbody>
</table>

NOTES:
1. Typical values are for $T_A = 25^\circ C$ and nominal power supply values.
2. If the external auto zero is used N$_{IC}$ has a typical value of $-76$ dBm0.
3. Tested and guaranteed at $23^\circ C$, nominal supplies.
5. With the D.C. method the positive and negative clipping levels are measured and $A_{IR}$ is calculated. With the A.C. method a sinusoidal input signal to $V_{F_X}$ is used where $A_{IR}$ is measured directly.
6. D.U.T. decoder, impose 200 mVpp, 1.02 KHz on appropriate supply; measurement made at decoder output; decoder in idle channel conditions.
7. D.U.T. encoder, impose 200 mVpp, 1.02 KHz on appropriate supply; measurement made at encoder output; encoder in idle channel conditions.
8. $V_{F_X}$ of D.U.T encoder = 1.02 KHz, 0 dBm0. Decoder under quiet channel conditions; measurements made at decoder output.
9. $V_{F_X} = 0$ Vrms. Decoder = 1.02 KHz, 0 dBm0. Encoder under quiet channel conditions; measurement made at encoder output.
Figure 13. Tracking Deviation ($\Delta G$) (Half Channel)

Figure 14. Signal to Total Distortion Ratio (Half Channel)
A.C. CHARACTERISTICS—TIMING SPECIFICATION\(^{(1)}\)

\( T_A = 0^\circ C \text{ to } +70^\circ C, V_{DD} = +12\text{V }\pm 5\%, V_{CC} = +5\text{V }\pm 5\%, V_{BB} = -5\text{V }\pm 5\%, \) GRDA = 0V, GRDD = 0V, unless otherwise specified.

**CLOCK SECTION**

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Parameter</th>
<th>Limits</th>
<th>Unit</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Min</td>
<td>Max</td>
<td>ns</td>
</tr>
<tr>
<td>( t_{CY} )</td>
<td>Clock Period</td>
<td>485</td>
<td></td>
<td></td>
</tr>
<tr>
<td>( t_r, t_f )</td>
<td>Clock Rise and Fall Time</td>
<td>0</td>
<td>30</td>
<td></td>
</tr>
<tr>
<td>( t_{CLK} )</td>
<td>Clock Pulse Width</td>
<td>215</td>
<td></td>
<td></td>
</tr>
<tr>
<td>( t_{CDC} )</td>
<td>Clock Duty Cycle ((t_{CLK} + t_{CY}))</td>
<td>45</td>
<td>55</td>
<td></td>
</tr>
</tbody>
</table>

**TRANSMIT SECTION**

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Parameter</th>
<th>Limits</th>
<th>Unit</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Min</td>
<td>Max</td>
<td>Timeslot</td>
</tr>
<tr>
<td>( t_{VFX} )</td>
<td>Analog Input Conversion</td>
<td>20</td>
<td></td>
<td></td>
</tr>
<tr>
<td>( t_{DZx} )</td>
<td>Data Enabled on TS Entry</td>
<td>50</td>
<td>180</td>
<td></td>
</tr>
<tr>
<td>( t_{DHX} )</td>
<td>Data Hold Time</td>
<td>80</td>
<td>230</td>
<td></td>
</tr>
<tr>
<td>( t_{HZX} )</td>
<td>Data Float on TS Exit</td>
<td>75</td>
<td>245</td>
<td></td>
</tr>
<tr>
<td>( t_{SON} )</td>
<td>Timeslot X to Enable</td>
<td>30</td>
<td>185</td>
<td></td>
</tr>
<tr>
<td>( t_{SOF} )</td>
<td>Timeslot X to Disable</td>
<td>70</td>
<td>225</td>
<td></td>
</tr>
<tr>
<td>( t_{FSD} )</td>
<td>Frame Sync Delay</td>
<td>15</td>
<td>150</td>
<td></td>
</tr>
</tbody>
</table>

**RECEIVE AND CONTROL SECTIONS**

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Parameter</th>
<th>Limits</th>
<th>Unit</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Min</td>
<td>Max</td>
<td>Timeslot</td>
</tr>
<tr>
<td>( t_{VFR} )</td>
<td>Analog Output Update</td>
<td>9( \frac{1}{16} )</td>
<td>9( \frac{1}{16} )</td>
<td></td>
</tr>
<tr>
<td>( t_{DSR} )</td>
<td>Receive Data Setup</td>
<td>20</td>
<td></td>
<td></td>
</tr>
<tr>
<td>( t_{DHR} )</td>
<td>Receive Data Hold</td>
<td>60</td>
<td></td>
<td></td>
</tr>
<tr>
<td>( t_{FSD} )</td>
<td>Frame Sync Delay</td>
<td>15</td>
<td>150</td>
<td></td>
</tr>
<tr>
<td>( t_{DSC} )</td>
<td>Control Data Setup</td>
<td>115</td>
<td></td>
<td></td>
</tr>
<tr>
<td>( t_{DHC} )</td>
<td>Control Data Hold</td>
<td>115</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**NOTES:**

1. All timing parameters referenced to 1.5V, except \( t_{DZx} \) and \( t_{SOF} \), which reference a high impedance state.
2. The 20 timeslot minimum insures that the complete A/D conversion will take place under any combination of receive interrupt or asynchronous operation of the Codec. Consult an Intel applications specialist or Intel Corporation for applications information which would allow operation with less than 20 timeslots.
**TIMING WAVEFORMS**

**TRANSMIT TIMING**

![Transmit Timing Diagram]

**RECEIVE TIMING**

![Receive Timing Diagram]

**CONTROL TIMING**

![Control Timing Diagram]

**NOTE:**
1. All timing parameters referenced to 1.5V, except $t_{HZX}$ and $t_{SOFF}$ which reference a high impedance state.
The Intel 2912A 2nd generation PCM line filter is a fully integrated monolithic device containing the two filters of a PCM line or trunk termination. It has improved key parameters of power consumption, idle channel noise, and power supply rejection. A single part exceeds both AT&T\textsuperscript{*} D3/D4 and CCITT transmission specs, exceeds digital Class 5 central office switching system stringent specifications, and is fully compatible with the 2912. The primary application for the 2912A is in telephone systems for transmission, switching, or remote concentration.

An advanced version of the switched capacitor technique used for the 2912 is used to implement the transmit and receive passband filter sections of the 2912A. The device is fabricated using Intel’s reliable two layer polysilicon gate NMOS technology. (See Intel Reliability Report RR-24 on the 2910A, 2911 A, and 2912.) The combination of advances in the switched capacitor techniques first used on the 2912 and the NMOS technology results in a monolithic 2912A filter which is packaged in a standard 16-pin DIP.
### Table 1. Pin Description

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Pin No.</th>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>VF(_x)(+)</td>
<td>1</td>
<td>Input</td>
<td>Analog input of the transmit filter. The VF(_x)(+) signal comes from the 2 to 4 wire hybrid in the case of a 2 wire line and goes through the frequency rejection and the antialiasing filters before being sent to the Codec for encoding.</td>
</tr>
<tr>
<td>VF(_x)(\sim)</td>
<td>2</td>
<td>Input</td>
<td>Inverting input of the gain adjustment operational amplifier on the transmit filter.</td>
</tr>
<tr>
<td>GS(_x)</td>
<td>3</td>
<td>Output</td>
<td>Output of the gain adjustment operational amplifier on the transmit filter. Used for gain setting of the transmit filter.</td>
</tr>
<tr>
<td>VF(_R)(O)</td>
<td>4</td>
<td>Output</td>
<td>Analog output of the receive filter. This output provides a direct interface to electronic hybrids. For a transformer hybrid application, VF(_R)(O) is tied to PWRI and a dual balanced output is provided on pins PWRO(+) and PWRO(\sim).</td>
</tr>
<tr>
<td>PWRI</td>
<td>5</td>
<td>Input</td>
<td>Input to the power driver amplifiers on the receive side for interface to transformer hybrids. High impedance input. When tied to V(_BB), the power amplifiers are powered down.</td>
</tr>
<tr>
<td>PWRO(+)</td>
<td>6</td>
<td>Output</td>
<td>Non-inverting side of the power amplifiers. Power driver output capable of directly driving transformer hybrids.</td>
</tr>
<tr>
<td>PWRO(\sim)</td>
<td>7</td>
<td>Output</td>
<td>Inverting side of the power amplifiers. Power driver output capable of directly driving transformer hybrids.</td>
</tr>
<tr>
<td>V(_BB)</td>
<td>8</td>
<td>Power</td>
<td>−5V ±5% referenced to GRDA</td>
</tr>
<tr>
<td>V(_CC)</td>
<td>9</td>
<td>Power</td>
<td>+5V ±5% referenced to GRDA</td>
</tr>
<tr>
<td>VF(_R)(I)</td>
<td>10</td>
<td>Input</td>
<td>Analog input of the receive filter, interface to the Codec analog output for PCM applications. The receive filter provides the −(\sin x) correction needed for sample and hold type Codec outputs to give unity gain. The input voltage range is directly compatible with the Intel 2910A and 2911A Codecs.</td>
</tr>
<tr>
<td>GRDD</td>
<td>11</td>
<td>Ground</td>
<td>Digital ground return for internal clock generator.</td>
</tr>
<tr>
<td>CLK(^{(1)})</td>
<td>12</td>
<td>Input</td>
<td>Clock input. Three clock frequencies can be used: 1.536 MHz, 1.544 MHz or 2.048 MHz; pin 14, CLK0, has to be strapped accordingly. High impedance input, TTL voltage levels.</td>
</tr>
<tr>
<td>PDN</td>
<td>13</td>
<td>Input</td>
<td>Control input for the stand-by power down mode. An internal pull up to +5V is provided for interface to the Intel 2910A and 2911A PDN outputs. TTL voltage levels.</td>
</tr>
<tr>
<td>CLK0(^{(1)})</td>
<td>14</td>
<td>Input</td>
<td>Clock (pin 12, CLK) frequency selection. If tied to V(_BB), CLK should be 1.536 MHz. If tied to Ground, CLK should be 1.544 MHz. If tied to V(_CC), CLK should be 2.048 MHz.</td>
</tr>
<tr>
<td>GRDA</td>
<td>15</td>
<td>Ground</td>
<td>Analog return common to the transmit and receive analog circuits. Not connected to GRDD internally.</td>
</tr>
<tr>
<td>VF(_x)(O)</td>
<td>16</td>
<td>Output</td>
<td>Analog output of the transmit filter. The output voltage range is directly compatible with the Intel 2910A and 2911A Codecs.</td>
</tr>
</tbody>
</table>

**NOTE:**
1. The three clock frequencies are directly compatible with the Intel 2910A and 2911A Codecs. The following table should be observed in selecting the clock frequency.

<table>
<thead>
<tr>
<th>Codec Clock</th>
<th>Clock Bits/Frame</th>
<th>CLK, Pin 12</th>
<th>CLK0, Pin 14</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.536 MHz</td>
<td>192</td>
<td>1.536 MHz</td>
<td>V(_BB) (−5V)</td>
</tr>
<tr>
<td>1.544 MHz</td>
<td>193</td>
<td>1.544 MHz</td>
<td>GRDD</td>
</tr>
<tr>
<td>2.048 MHz</td>
<td>256</td>
<td>2.048 MHz</td>
<td>V(_CC) (+5V)</td>
</tr>
</tbody>
</table>
FUNCTIONAL DESCRIPTION

The 2912A provides the transmit and receive filters found on the analog termination of a PCM line or trunk. The transmit filter performs the anti-aliasing function needed for an 8 KHz sampling system, and the 50/60 Hz rejection. The receive filter has a low pass transfer characteristic and also provides the Sinx/x correction necessary to interface the Intel 2910A (µ Law) and 2911A (A Law) Codecs which have a non-return-to-zero output of the digital to analog conversion. Gain adjustment is provided in the receive and transmit directions.

A stand-by, power down mode is included in the 2912A and can be directly controlled by the 2910A/2911A Codecs.

The 2912A can interface directly with a transformer hybrid (2 to 4 wire conversion) or with electronic hybrids; in the latter case the power dissipation is reduced by powering down the output amplifier provided on the 2912A.

FILTER OPERATION

Transmit Filter Input Stage

The input stage provides gain adjustment in the pass-band. The input operational amplifier has a common mode range of ±2.2 volts, a DC offset of less than 25 mV, a voltage gain greater than 3000 and a unity gain bandwidth of 1 MHz. It can be connected to provide a gain of 20 dB without degrading the noise performance of the filter. The load impedance connected to the amplifier output (GSx) must be greater than 10K Ω in parallel with 25 pF. The input signal on lead VFx1+ can be either AC or DC coupled. The input Op Amp can also be used in the inverting mode or differential amplifier mode. The remaining portion of the transmit filter provides a gain of +3 dB in the pass band.
Receive Filter Output

The VFR\textsubscript{O} lead is capable of driving high impedance electronic hybrids. The gain of the receive section from VFR\textsubscript{I} to VFR\textsubscript{O} is:

\[
\frac{\sin\left(\frac{\pi f}{8000}\right)}{\pi f}
\]

which when multiplied by the output response of the Intel 2910A and 2911A Codecs results in a 0 dB gain in the pass band. The filter gain can be adjusted downward by a resistor voltage divider connected as shown in Figure 5. The total resistive load R\textsubscript{LR} on VFR\textsubscript{O} should not be less than 10K Ω.

![Figure 5. Receive Filter Output Gain Adjustment](image)

Receive Filter Output Driver Amplifier Stage

A balanced power amplifier is provided in order to drive low-impedance loads in a bridged configuration. The receive filter output VFR\textsubscript{O} is connected through gain setting resistors R\textsubscript{1} and R\textsubscript{2} to the amplifier input PWRI. The input voltage range on PWRI is ±3.2 volts and the gain is 6 dB for a bridged output.

With a 600Ω load connected between PWRO\textsuperscript{+} and PWRO\textsuperscript{−}, the maximum voltage swing across the load is ±5.0 volts. The series combination of R\textsubscript{3} and the hybrid transformer must present a minimum A.C. load resistance of 600Ω to the amplifier in the bridged configuration. A typical connection of the output driver amplifiers is shown in Figure 6. These amplifiers can also be used with loads connected to ground.

When the power amplifier is not needed it should be deactivated to save power. This is accomplished by tying the PWRI pin to V\textsubscript{BB} before the device is powered up.

Power Down Mode

Pin 13, PDN, provides the power down control. When the signal on this lead is brought high, the 2912A goes into a standby, power down mode. Power dissipation is reduced to 0.5 mW. In the stand-by mode, all outputs go into a high impedance state. This feature allows multiple 2912As to drive the same analog bus on a time-shared basis.

When power is restored, the settling time of the 2912A is typically 15 ms.

The PDN interface is directly compatible with the Intel 2910A and 2911A PDN outputs. Only one command from the common control is then necessary to power down both the Codec and the Filters of the line or trunk interface.

![Figure 6. Typical Connection of Output Driver Amplifier](image)
ABSOLUTE MAXIMUM RATINGS*

Temperature Under Bias \(\ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldotsmiddleware error}
**D.C. CHARACTERISTICS**  
$T_A = 0^\circ C$ to $+70^\circ C$; $V_{CC} = 5V \pm 5\%$; $V_{BB} = -5V \pm 5\%$; $GRDA = 0V$; $GRDD = 0V$; unless otherwise specified (Continued)

**ANALOG INTERFACE, TRANSMIT FILTER INPUT STAGE**

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Parameter</th>
<th>Min</th>
<th>Typ(1)</th>
<th>Max</th>
<th>Unit</th>
<th>Test Conditions</th>
</tr>
</thead>
<tbody>
<tr>
<td>$I_{BIX}$</td>
<td>Input Leakage Current, $VF_{X1+}$, $VF_{X1-}$</td>
<td>$100$</td>
<td>nA</td>
<td>$-2.2V &lt; VIN &lt; 2.2V$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$R_{IXI}$</td>
<td>Input Resistance, $VF_{X1+}$, $VF_{X1-}$</td>
<td>$10$</td>
<td>MΩ</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$V_{OSXI}$</td>
<td>Input Offset Voltage, $VF_{X1+}$, $VF_{X1-}$</td>
<td>$25$</td>
<td>mV</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CMRR</td>
<td>Common Mode Rejection, $VF_{X1+}$, $VF_{X1-}$</td>
<td>$60$</td>
<td>$75$</td>
<td>dB</td>
<td>$-2.2V &lt; VIN &lt; 2.2V$, $0$ dBm0 $\equiv 1.1$ VRMS, Input at $VF_{X1-}$</td>
<td></td>
</tr>
<tr>
<td>$A_{VOL}$</td>
<td>DC Open Loop Voltage Gain, $GS_X$</td>
<td>$3000$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$f_C$</td>
<td>Open Loop Unity Gain Bandwidth, $GS_X$</td>
<td>$1$</td>
<td>MHz</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$V_{OXI}$</td>
<td>Output Voltage Swing, $GS_X$</td>
<td>$\pm 2.5$</td>
<td>V</td>
<td>$R_L &gt; 10$ KΩ</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$C_{LXI}$</td>
<td>Load Capacitance, $GS_X$</td>
<td>$25$</td>
<td>pF</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$R_{LXI}$</td>
<td>Minimum Load Resistance, $GS_X$</td>
<td>$10$</td>
<td>KΩ</td>
<td>Minimum $R_L$</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**ANALOG INTERFACE, TRANSMIT FILTER**  
(See Figure 9)

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Parameter</th>
<th>Min</th>
<th>Typ(1)</th>
<th>Max</th>
<th>Unit</th>
<th>Test Conditions</th>
</tr>
</thead>
<tbody>
<tr>
<td>$R_{OX}$</td>
<td>Output Resistance, $VF_{XO}$</td>
<td>$20$</td>
<td>$35$</td>
<td>Ω</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$V_{OSX}$</td>
<td>Output DC Offset, $VF_{XO}$</td>
<td>$100$</td>
<td>mV</td>
<td>$VF_{X1+}$ Connected to GRDA, Input Op Amp at Unity Gain</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PSRR$_1$</td>
<td>Power Supply Rejection of $V_{CC}$ at 1 KHz, $VF_{XO}$</td>
<td>$30$</td>
<td>$40$</td>
<td>dB</td>
<td>Note 2</td>
<td></td>
</tr>
<tr>
<td>PSRR$_2$</td>
<td>Power Supply Rejection of $V_{BB}$ at 1 KHz, $VF_{XO}$</td>
<td>$25$</td>
<td>$30$</td>
<td>dB</td>
<td>Note 2</td>
<td></td>
</tr>
<tr>
<td>$C_{LX}$</td>
<td>Load Capacitance, $VF_{XO}$</td>
<td>$25$</td>
<td>pF</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$R_{LX}$</td>
<td>Minimum Load Resistance, $VF_{XO}$</td>
<td>$2.7$</td>
<td>KΩ</td>
<td>Minimum $R_L$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$V_{OX1}$</td>
<td>Output Voltage Swing, 1 KHz, $VF_{XO}$</td>
<td>$\pm 3.2$</td>
<td>V</td>
<td>$R_L &gt; 10$ KΩ or with 2910A or 2911A</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$V_{OX2}$</td>
<td>Output Voltage Swing, 1 KHz, $VF_{XO}$</td>
<td>$\pm 2.5$</td>
<td>V</td>
<td>$R_L &gt; 2.7$ KΩ</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
D.C. CHARACTERISTICS $T_A = 0^\circ \text{C} \text{ to } +70^\circ \text{C}; V_{CC} = 5\text{V } \pm 5\%; V_{BB} = -5\text{V } \pm 5\%; GRDA = 0\text{V}; GRDD = 0\text{V};$ unless otherwise specified (Continued)

ANALOG INTERFACE, RECEIVE FILTER (See Figure 10)

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Parameter</th>
<th>Min</th>
<th>Typ(1)</th>
<th>Max</th>
<th>Unit</th>
<th>Test Conditions</th>
</tr>
</thead>
<tbody>
<tr>
<td>$I_{BR}$</td>
<td>Input Leakage Current, $V_{FR1}$</td>
<td>3</td>
<td>$\mu$A</td>
<td>$-3.2\text{V} &lt; V_{IN} &lt; 3.2\text{V}$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$R_{IR}$</td>
<td>Input Resistance, $V_{FR1}$</td>
<td>1</td>
<td>$\Omega$</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$R_{OR}$</td>
<td>Output Resistance, $V_{FR0}$</td>
<td>100</td>
<td>$\Omega$</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$V_{OSR}$</td>
<td>Output DC Offset $V_{FR0}$</td>
<td>100</td>
<td>mV</td>
<td>$V_{FR1}$ Connected to GRDA</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$PSRR_3$</td>
<td>Power Supply Rejection of $V_{CC}$ at 1 KHz, $V_{FR0}$</td>
<td>30</td>
<td>45</td>
<td>dB</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$PSRR_4$</td>
<td>Power Supply Rejection of $V_{BB}$ at 1 KHz, $V_{FR0}$</td>
<td>30</td>
<td>35</td>
<td>dB</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$C_{LR}$</td>
<td>Load Capacitance, $V_{FR0}$</td>
<td>25</td>
<td>pF</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$R_{LR}$</td>
<td>Minimum Load Resistance, $V_{FR0}$</td>
<td>10</td>
<td>$\Omega$</td>
<td>Minimum $R_L$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$V_{OR}$</td>
<td>Output Voltage Swing, $V_{FR0}$</td>
<td>$\pm 3.2$</td>
<td>V</td>
<td>$R_L = 10\text{ K}\Omega$</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

ANALOG INTERFACE, RECEIVE FILTER DRIVER AMPLIFIER STAGE

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Parameter</th>
<th>Min</th>
<th>Typ(1)</th>
<th>Max</th>
<th>Unit</th>
<th>Test Conditions</th>
</tr>
</thead>
<tbody>
<tr>
<td>$I_{BRA}$</td>
<td>Input Leakage Current, $PWRI$</td>
<td>3</td>
<td>$\mu$A</td>
<td>$-3.2\text{V} &lt; V_{IN} &lt; 3.2\text{V}$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$R_{IRA}$</td>
<td>Input Resistance, $PWRI$</td>
<td>10</td>
<td>$\Omega$</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$R_{ORA}$</td>
<td>Output Resistance, $PWRO+, PWRO-$</td>
<td>1</td>
<td>$\Omega$</td>
<td>$</td>
<td>I_{OUT}</td>
<td>&lt; 10\text{ mA}$</td>
</tr>
<tr>
<td>$V_{OSRA}$</td>
<td>Output DC Offset, $PWRO+, PWRO-$</td>
<td>50</td>
<td>mV</td>
<td>$PWRI$ Connected to GRDA</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$C_{LRA}$</td>
<td>Load Capacitance, $PWRO+, PWRO-$</td>
<td>100</td>
<td>pF</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$V_{ORA1}$</td>
<td>Output Voltage Swing Across $R_L$, $PWRO+, PWRO-$ - Single Ended Connection</td>
<td>$\pm3.2$</td>
<td>V</td>
<td>$R_L = 10\text{ K}\Omega$</td>
<td>$R_L$ Connected to GRDA</td>
<td>$f_o \geq 200$ Hz</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$\pm2.9$</td>
<td>V</td>
<td>$R_L = 600\text{\Omega}$</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>$\pm2.5$</td>
<td>V</td>
<td>$R_L = 300\text{\Omega}$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$V_{ORA2}$</td>
<td>Differential Output Voltage Swing, $PWRO+, PWRO-$ - Balanced Output Connection</td>
<td>$\pm6.4$</td>
<td>V</td>
<td>$R_L = 20\text{ K}\Omega$</td>
<td>$R_L$ Connected between $PWRO+$ and $PWRO-$</td>
<td>$f_o \geq 200$ Hz</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$\pm5.8$</td>
<td>V</td>
<td>$R_L = 1200\text{\Omega}$</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>$\pm5.0$</td>
<td>V</td>
<td>$R_L = 600\text{\Omega}$</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

NOTE:
1. Typical values are for $T_A = 25^\circ \text{C}$ and nominal power supply values.
A.C. CHARACTERISTICS  \( T_A = 0^\circ C \text{ to } +70^\circ C; V_{CC} = 5V \pm 5\%; V_{BB} = -5V \pm 5\%; \text{GRDA} = 0V; \)  
GRDD = 0V; unless otherwise specified

Clock Input Frequency:  
\[ \text{CLK} = 1.536 \text{ MHz } \pm 0.1\%; \text{CLK0} = V_{IL0} \text{ (Tied to } V_{BB}) \]  
\[ \text{CLK} = 2.048 \text{ MHz } \pm 0.1\%; \text{CLK0} = V_{IH0} \text{ (Tied to } V_{CC}) \]  
\[ \text{CLK} = 1.544 \text{ MHz } \pm 0.1\%; \text{CLK0} = V_{II0} \text{ (Tied to } \text{GRDD}) \]

**TRANSMIT FILTER TRANSFER CHARACTERISTICS**  
(See Transmit Filter Transfer Characteristics, Figure 7)

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Parameter</th>
<th>Min</th>
<th>Typ(1)</th>
<th>Max</th>
<th>Unit</th>
<th>Test Conditions</th>
</tr>
</thead>
<tbody>
<tr>
<td>( G_{RX} )</td>
<td>Gain Relative to Gain at 1 KHz</td>
<td>16.67 Hz</td>
<td>-56</td>
<td>-50</td>
<td>dB</td>
<td>0 dBm0 Input Signal</td>
</tr>
<tr>
<td></td>
<td>50 Hz</td>
<td></td>
<td></td>
<td>-25</td>
<td>dB</td>
<td>Gain Setting Op Amp</td>
</tr>
<tr>
<td></td>
<td>60 Hz</td>
<td></td>
<td></td>
<td>-23</td>
<td>dB</td>
<td>Unity Gain</td>
</tr>
<tr>
<td></td>
<td>200 Hz</td>
<td></td>
<td>-1.8</td>
<td>-0.125</td>
<td>dB</td>
<td>0 dBm0 Signal ( = 1.1 ) ( V_{RMS} )</td>
</tr>
<tr>
<td></td>
<td>300 Hz to 3000 Hz</td>
<td></td>
<td>-0.125</td>
<td>0.125</td>
<td>dB</td>
<td>Input at ( VF_Xl^- )</td>
</tr>
<tr>
<td></td>
<td>3300 Hz</td>
<td></td>
<td>-0.35</td>
<td>0.03</td>
<td>dB</td>
<td>0 dBm0 Signal ( = 1.6 ) ( V_{RMS} )</td>
</tr>
<tr>
<td></td>
<td>3400 Hz</td>
<td></td>
<td>-0.7</td>
<td>-0.1</td>
<td>dB</td>
<td>Output at ( VF_XO )</td>
</tr>
<tr>
<td></td>
<td>4000 Hz</td>
<td></td>
<td></td>
<td>-14</td>
<td>dB</td>
<td></td>
</tr>
<tr>
<td></td>
<td>4600 Hz and Above</td>
<td></td>
<td></td>
<td>-32</td>
<td>dB</td>
<td></td>
</tr>
<tr>
<td>( G_{AX} )</td>
<td>Absolute Passband Gain at 1 KHz, ( VF_XO )</td>
<td></td>
<td>2.9</td>
<td>3.0</td>
<td>3.1</td>
<td>dB</td>
</tr>
<tr>
<td>( G_{AXT} )</td>
<td>Gain Variation with Temperature at 1 KHz</td>
<td></td>
<td></td>
<td>0.0002</td>
<td>0.002</td>
<td>dB/( ^\circ )C</td>
</tr>
<tr>
<td>( G_{AXS} )</td>
<td>Gain Variation with Supplies at 1 KHz</td>
<td></td>
<td></td>
<td>0.01</td>
<td>0.07</td>
<td>dB/V</td>
</tr>
<tr>
<td>( C_{RT} )</td>
<td>Cross Talk, Receive to Transmit, Measured at ( VF_XO )</td>
<td></td>
<td>20 log ( VF_XO / VF_RO )</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( N_{CX1} )</td>
<td>Total C Message Noise at Output, ( VF_XO )</td>
<td></td>
<td>6</td>
<td>11</td>
<td>dBm0 (Note 2)</td>
<td>Gain Setting Op Amp at Unity Gain</td>
</tr>
<tr>
<td>( N_{CX2} )</td>
<td>Total C Message Noise at Output, ( VF_XO )</td>
<td></td>
<td>9</td>
<td>13</td>
<td>dBm0 (Note 2)</td>
<td>Gain Setting Op Amp at 20 dB Gain</td>
</tr>
<tr>
<td>( D_{DX} )</td>
<td>Differential Envelope Delay, ( VF_XO ) 1 KHz to 2.6 KHz</td>
<td></td>
<td></td>
<td>60</td>
<td>( \mu )s</td>
<td></td>
</tr>
<tr>
<td>( D_{AX} )</td>
<td>Absolute Delay at 1 KHz, ( VF_XO )</td>
<td></td>
<td></td>
<td>110</td>
<td>( \mu )s</td>
<td></td>
</tr>
<tr>
<td>( D_{P_{X1}} )</td>
<td>Single Frequency Distortion Products</td>
<td></td>
<td></td>
<td>-48</td>
<td>dB</td>
<td>0 dBm0 Input Signal at 1 KHz</td>
</tr>
<tr>
<td>( D_{P_{X2}} )</td>
<td>Single Frequency Distortion Products at Maximum Signal Level of +3 dBm0 at ( VF_XO )</td>
<td></td>
<td></td>
<td>-45</td>
<td>dB</td>
<td>0.16 ( V_{RMS} ) 1 KHz Input Signal at ( VF_Xl^- ) + Gain Setting Op Amp at 20 dB Gain, The + 3 dBm0 Signal at ( VF_XO ) is 2.26 ( V_{RMS} )</td>
</tr>
</tbody>
</table>
A.C. CHARACTERISTICS  $T_A = 0^\circ C$ to $+70^\circ C$; $V_{CC} = 5V \pm 5\%$; $V_{BB} = -5V \pm 5\%$; $GRDA = 0V$; $GRDD = 0V$; unless otherwise specified (Continued)

Clock Input Frequency: $CLK = 1.536 \text{ MHz} \pm 0.1\%$; $CLK_0 = V_{IL0}$ (Tied to $V_{BB}$)
$CLK = 1.544 \text{ MHz} \pm 0.1\%$; $CLK_0 = V_{IL0}$ (Tied to $GRDD$)
$CLK = 2.048 \text{ MHz} \pm 0.1\%$; $CLK_0 = V_{IH0}$ (Tied to $V_{CC}$)

RECEIVE FILTER TRANSFER CHARACTERISTICS (See Receive Filter Transfer Characteristics, Figure 8)

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Parameter Description</th>
<th>Min</th>
<th>Typ(1)</th>
<th>Max</th>
<th>Unit</th>
<th>Test Conditions</th>
</tr>
</thead>
<tbody>
<tr>
<td>$G_{RR}$</td>
<td>Gain Relative to Gain at 1 KHz with $\sin x / x$ Correction of 2910A or 2911A</td>
<td>0.125 dB</td>
<td>0 dBm0 Input Signal</td>
<td>$V_{FRO}$ or $V_{FRI}$ Connected to $20 \log (V_{FRO}/V_{FRI})$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Below 200 Hz</td>
<td>0.125 dB</td>
<td>0 dBm0 Signal $\equiv 1.6 V_{RMS} \times \frac{\pi f}{8000}$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>200 Hz</td>
<td>$-0.5$</td>
<td>$0.125$ dB</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>300 Hz to 3000 Hz</td>
<td>$-0.125$</td>
<td>$0.125$ dB</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3300 Hz</td>
<td>$-0.35$</td>
<td>$0.03$ dB</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3400 Hz</td>
<td>$-0.7$</td>
<td>$-0.1$ dB</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4000 Hz</td>
<td>$-14$</td>
<td>dB</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4600 Hz and Above</td>
<td>$-30$</td>
<td>dB</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$G_{AR}$</td>
<td>Absolute Passband Gain at 1 KHz, $V_{FRO}$</td>
<td>$-0.1$ dB</td>
<td>$0 + 0.1$ dB</td>
<td>$R_L = \infty$ (3, 4)</td>
<td>0 dBm0 Signal Level</td>
<td></td>
</tr>
<tr>
<td>$G_{ART}$</td>
<td>Gain Variation with Temperature at 1 KHz</td>
<td>0.0002 dB/°C</td>
<td>0.002 dB/°C</td>
<td>0 dBm0 Signal Level</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$G_{ARS}$</td>
<td>Gain Variation with Supplies at 1 KHz</td>
<td>0.01 dB/V</td>
<td>0.07 dB/V</td>
<td>0 dBm0 Signal Level, Supplies ±5%</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$C_{TTR}$</td>
<td>Cross Talk, Transmit to Receive, Measured at $V_{FRO}$; $20 \log (V_{FRO}/V_{FRI})$</td>
<td>$-70$ dB</td>
<td>$-60$ dB</td>
<td>$V_{FRI} = 1.1 V_{RMS}$, 1 KHz Output, $V_{FRI}$ Connected to $GRDA$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$N_{CR}$</td>
<td>Total C Message Noise at Output, $V_{FRO}$</td>
<td>2 dBm0</td>
<td>6 dBm0 (Note 2)</td>
<td>$V_{FRO}$ Output or $PWRO+$ and $PWRO-$ Connected with Unity Gain</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$D_{DR}$</td>
<td>Differential Envelope Delay, $V_{FRO}$, 1 KHz to 2.6 KHz</td>
<td>100 µs</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$D_{AR}$</td>
<td>Absolute Delay at 1 KHz, $V_{FRO}$</td>
<td>110 µs</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$D_{PR1}$</td>
<td>Single Frequency Distortion Products</td>
<td>$-48$ dB</td>
<td>0 dBm0 Input Signal at 1 KHz</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$D_{PR2}$</td>
<td>Single Frequency Distortion Products at Maximum Signal Level of $+3$ dBm0 at $V_{FRO}$</td>
<td>$-45$ dB</td>
<td>$+3$ dBm0 Signal Level of $2.26 V_{RMS}$, 1 KHz Input at $V_{FRO}$</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

NOTES:
1. Typical Values are for $T_A = 25^\circ C$ and nominal power supply values.
2. A noise measurement of 12 dBm0 into a 600Ω load at the 2912A device is equivalent to 6 dBm0.
3. For gain under load refer to output resistance specs and perform gain calculation.
4. Output is non-inverting.
Figure 7. Transmit Filter
Figure 8. Receive Filter

NOTES:
1. Typical Transfer Function of the Receive Filter as a Separate Component.
POWER SUPPLY REJECTION TYPICAL VALUES OVER 3 RANGES

All VFxO with VFxI Connected to GRDA; Input Op Amp at Unity Gain

Figure 9. Transmit Filter

All VFxO with VFxI Connected to GRDA

Figure 10. Receive Filter
2913 AND 2914
COMBINED SINGLE-CHIP PCM CODEC AND FILTER

- 2913 Synchronous Clocks Only, 300 Mil Package
- 2914 Asynchronous Clocks, 8th Bit Signaling, Loop Back Test Capability
- AT&T D3/D4 and CCITT Compatible for Synchronous Operation
- Pin Selectable μ-Law or A-Law Operation
- Two Timing Modes:  
  - Fixed Data Rate Mode  
    1.536, 1.544, or 2.048 MHz  
  - Variable Data Rate Mode  
    64 KHz 2.048 MHz
- Exceptional Analog Performance
- 28-Pin Plastic Leaded Chip Carrier (PLCC) for Higher Integration
- Low Power HMOS-E Technology:  
  - 5 mW Typical Power Down  
  - 140 mW Typical Operating
- Fully Differential Architecture Enhances Noise Immunity
- On-Chip Auto Zero, Sample and Hold, and Precision Voltage References
- Direct Interface with Transformer or Electronic Hybrids

The Intel 2913 and 2914 are fully integrated PCM codecs with transmit/receive filters fabricated in a highly reliable and proven N-channel HMOS silicon gate technology (HMOS-E). These devices provide the functions that were formerly provided by two complex chips (2910A or 2911A and 2912A). Besides the higher level of integration, the performance of the 2913 and 2914 is superior to that of the separate devices.

The primary applications for the 2913 and 2914 are in telephone systems:
- Switching—Digital PBX’s and Central Office Switching Systems
- Transmission—D3/D4 Type Channel Banks and Subscriber Carrier Systems
- Subscriber Instruments—Digital Handsets and Office Workstations

The wide dynamic range of the 2913 and 2914 (78 dB) and the minimal conversion time make them ideal products for other applications such as:
- Voice Store and Forward
- Secure Communications Systems
- Digital Echo Cancellers
- Satellite Earth Stations

Figure 1. Pin Configurations
### Table 1. Pin Names

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>VBB</td>
<td>Power (-5V)</td>
<td>GSx</td>
<td>Transmit Gain Control</td>
</tr>
<tr>
<td>PWRO+</td>
<td>Power Amplifier Outputs</td>
<td>VFxI-, VFxI+</td>
<td>Analog Inputs</td>
</tr>
<tr>
<td>GS R</td>
<td>Receive Gain Control</td>
<td>GRDA</td>
<td>Analog Ground</td>
</tr>
<tr>
<td>PDN</td>
<td>Power Down Select</td>
<td>NC</td>
<td>No Connect</td>
</tr>
<tr>
<td>CLKSEL</td>
<td>Master Clock Frequency Select</td>
<td>SIGx</td>
<td>Transmit Signaling Input</td>
</tr>
<tr>
<td>LOOP</td>
<td>Analog Loop Back</td>
<td>ASEL</td>
<td>μ- or A-Law Select</td>
</tr>
<tr>
<td>SIG R</td>
<td>Receive Signaling Output</td>
<td>TSx</td>
<td>Timeslot Strobe/Buffer Enable</td>
</tr>
<tr>
<td>DCLK R</td>
<td>Receive Variable Data Clock</td>
<td>DCLKx</td>
<td>Transmit Variable Data Clock</td>
</tr>
<tr>
<td>D R</td>
<td>Receive PCM Input</td>
<td>Dx</td>
<td>Transmit PCM Output</td>
</tr>
<tr>
<td>FS R</td>
<td>Receive Frame Synchronization Clock</td>
<td>FSX</td>
<td>Transmit Frame Synchronization Clock</td>
</tr>
<tr>
<td>GRDD</td>
<td>Digital Ground</td>
<td>CLKx</td>
<td>Transmit Master Clock</td>
</tr>
<tr>
<td>VCC</td>
<td>Power (+5V)</td>
<td>CLKR</td>
<td>Receive Master Clock (2914 Only, Internally Connected to CLKx on 2913)</td>
</tr>
</tbody>
</table>

### Table 2. Pin Description

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>VBB</td>
<td>Most negative supply; input voltage is −5V ± 5%.</td>
</tr>
<tr>
<td>PWRO+</td>
<td>Non-inverting output of power amplifier. Can drive transformer hybrids or high impedance loads directly in either a differential or single ended configuration.</td>
</tr>
<tr>
<td>PWRO−</td>
<td>Inverting output of power amplifier. Functionally identical and complementary to PWRO+.</td>
</tr>
<tr>
<td>GS R</td>
<td>Input to the gain setting network on the output power amplifier. Transmission level can be adjusted over a 12 dB range depending on the voltage at GS R.</td>
</tr>
<tr>
<td>PDN</td>
<td>Power down select. When PDN is TTL high, the device is active. When low, the device is powered down.</td>
</tr>
<tr>
<td>CLKSEL</td>
<td>Input which must be pinstrapped to reflect the master clock frequency at CLKx, CLK R. CLK R = VBB .......................... 2.048 MHz CLK R = GRDD .......................... 1.544 MHz CLK R = VCC .......................... 1.536 MHz</td>
</tr>
<tr>
<td>LOOP</td>
<td>Analog loopback. When this pin is TTL high, the analog output (PWRO +) is internally connected to the analog input (VFxI+), GS R is internally connected to PWRO −, and VFxI − is internally connected to GSx. A 0 dBm0 digital signal input at D R is returned as a +3 dBm0 digital signal output at D x.</td>
</tr>
<tr>
<td>SIG R</td>
<td>Signaling bit output, receive channel. In fixed data rate mode, SIG R outputs the logical state of the eighth bit of the PCM word in the most recent signaling frame.</td>
</tr>
<tr>
<td>DCLK R</td>
<td>Selects the fixed or variable data rate mode. When DCLK R is connected to VBB, the fixed data rate mode is selected. When DCLK R is not connected to VBB, the device operates in the variable data rate mode. In this mode DCLK R becomes the receive data clock which operates at TTL levels from 64 Kb to 2.048 Mb data rates.</td>
</tr>
<tr>
<td>Symbol</td>
<td>Function</td>
</tr>
<tr>
<td>---------</td>
<td>-------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>DR</td>
<td>Receive PCM input. PCM data is clocked in on this lead on eight consecutive negative transitions of the receive data clock; CLK_R in the fixed data rate mode and DCLK_R in variable data rate mode.</td>
</tr>
<tr>
<td>FS_R</td>
<td>8 KHz frame synchronization clock input/timeslot enable; receive channel. A multi-function input which in fixed data rate mode distinguishes between signaling and non-signaling frames by means of a double or single wide pulse respectively. In variable data rate mode this signal must remain high for the entire length of the timeslot. The receive channel enters the standby state whenever FS_R is TTL low for 300 milliseconds.</td>
</tr>
<tr>
<td>GRDD</td>
<td>Digital ground for all internal logic circuits. Not internally tied to GRDA.</td>
</tr>
<tr>
<td>CLK_R</td>
<td>Receive master and data clock for the fixed data rate mode; receive master clock only in variable data rate mode.</td>
</tr>
<tr>
<td>CLK_X</td>
<td>Transmit master and data clock for the fixed data rate mode; transmit master clock only in variable data rate mode.</td>
</tr>
<tr>
<td>FS_X</td>
<td>8 KHz frame synchronization clock input/timeslot enable, transmit channel. Operates independently but in an analogous manner to FS_R. The transmit channel enters the standby state whenever FS_X is TTL low for 300 milliseconds.</td>
</tr>
<tr>
<td>DX</td>
<td>Transmit PCM output. PCM data is clocked out on this lead on eight consecutive positive transitions of the transmit data clock: CLK_X in fixed data rate mode and DCLK_X in variable data rate mode.</td>
</tr>
<tr>
<td>TS_X/DCLK_X</td>
<td>Transmit channel timeslot strobe (output) or data clock (input) for the transmit channel. In fixed data rate mode, this pin is an open drain output designed to be used as an enable signal for a three-state buffer. In variable data rate mode, this pin becomes the transmit data clock which operates at TTL levels from 64 Kbps to 2.048 Mbps data rates.</td>
</tr>
<tr>
<td>SIG_X/ASEL</td>
<td>A dual purpose pin. When connected to V_BB, A-law operation is selected. When it is not connected to V_BB this pin is a TTL level input for signaling operation. This input is transmitted as the eighth bit of the PCM word during signaling frames on the DX lead. If not used as an input pin, ASEL should be strapped to either V_CC or GRDD.</td>
</tr>
<tr>
<td>NC</td>
<td>No connect.</td>
</tr>
<tr>
<td>GRDA</td>
<td>Analog ground return for all internal voice circuits. Not internally connected to GRDD.</td>
</tr>
<tr>
<td>VF_X+</td>
<td>Non-inverting analog input to uncommitted transmit operational amplifier.</td>
</tr>
<tr>
<td>VF_X-</td>
<td>Inverting analog input to uncommitted transmit operational amplifier.</td>
</tr>
<tr>
<td>GS_X</td>
<td>Output terminal of transmit channel input op amp. Internally, this is the voice signal input to the transmit filter.</td>
</tr>
<tr>
<td>V_CC</td>
<td>Most positive supply; input voltage is +5V ± 5%.</td>
</tr>
</tbody>
</table>
FUNCTIONAL DESCRIPTION

The 2913 and 2914 provide the analog-to-digital and the digital-to-analog conversions and the transmit and receive filtering necessary to interface a full duplex (4 wires) voice telephone circuit with the PCM highways of a time division multiplexed (TDM) system. They are intended to be used at the analog termination of a PCM line or trunk.

SWITCHING

The following major functions are provided:
- Bandpass filtering of the analog signals prior to encoding and after decoding
- Encoding and decoding of voice and call progress information
- Encoding and decoding of the signaling and supervision information

CHANNEL BANKS

A TYPICAL 4-WIRE CHANNEL UNIT WITH SIGNALING USING BORROWED 8TH BIT

Figure 3. Typical Line Terminations
GENERAL OPERATION

System Reliability Features

The combochip can be powered up by pulsing FSX and/or FSR while a TTL high voltage is applied to PDN, provided that all clocks and supplies are connected. The 2913 and 2914 have internal resets on power up (or when VBB or VCC are re-applied) in order to ensure validity of the digital outputs and thereby maintain integrity of the PCM highway.

On the transmit channel, digital outputs DX and TSX are held in a high impedance state for approximately four frames (500 µs) after power up or application of VBB or VCC. After this delay, DX, TSX, and signaling will be functional and will occur in the proper timeslot. The analog circuits on the transmit side require approximately 60 milliseconds to reach their equilibrium value due to the autozero circuit settling time. Thus, valid digital information, such as for on/off hook detection, is available almost immediately, while analog information is available after some delay.

On the receive channel, the digital output SIGR is also held low for a maximum of four frames after power up or application of VBB or VCC. SIGR will remain low thereafter until it is updated by a signaling frame.

To further enhance system reliability, TSX and DX will be placed in a high impedance state approximately 30 µs after an interruption of CLKX. Similarly, SIGR will be held low approximately 30 µs after an interruption of CLKR. These interruptions could possibly occur with some kind of fault condition.

Power Down and Standby Modes

To minimize power consumption, two power down modes are provided in which most 2913/2914 functions are disabled. Only the power down, clock, and frame sync buffers, which are required to power up the device, are enabled in these modes. As shown in Table 3, the digital outputs on the appropriate channels are placed in a high impedance state until the device returns to the active mode.

The Power Down mode utilizes an external control signal to the PDN pin. In this mode, power consumption is reduced to the value shown in Table 3. The device is active when the signal is high and inactive when it is low. In the absence of any signal, the PDN pin floats to TTL high allowing the device to remain active continuously.

The Standby mode leaves the user an option of powering either channel down separately or powering the entire device down by selectively removing FSX and/or FSR. With both channels in the standby state, power consumption is reduced to the value shown in Table 3. If transmit only operation is desired, FSX should be applied to the device while FSR is held low. Similarly, if receive only operation is desired, FSR should be applied while FSX is held low.

Fixed Data Rate Mode

Fixed data rate timing, which is 2910A and 2911A compatible, is selected by connecting DCLKR to VBB. It employs master clocks CLKX and CLKR, frame synchronization clocks FSX and FSR, and output TSX.

<table>
<thead>
<tr>
<th>Device Status</th>
<th>Power-Down Method</th>
<th>Typical Power Consumption</th>
<th>Digital Output Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>Power Down Mode</td>
<td>PDN = TTL Low</td>
<td>5 mW</td>
<td>TSX and DX are placed in a high impedance state and SIGR is placed in a TTL low state within 10 µs.</td>
</tr>
<tr>
<td>Standby Mode</td>
<td>FSX and FSR are TTL Low</td>
<td>12 mW</td>
<td>TSX and DX are placed in a high impedance state and SIGR is placed in a TTL low state 300 milliseconds after FSX and FSR are removed.</td>
</tr>
<tr>
<td>Only Transmit Is on Standby</td>
<td>FSX is TTL Low</td>
<td>70 mW</td>
<td>TSX and DX are placed in a high impedance state within 300 milliseconds.</td>
</tr>
<tr>
<td>Only Receive Is on Standby</td>
<td>FSR is TTL Low</td>
<td>110 mW</td>
<td>SIGR is placed in a TTL low state within 300 milliseconds.</td>
</tr>
</tbody>
</table>

Table 3. Power-Down Methods
CLK<sub>x</sub> and CLK<sub>r</sub> serve both as master clocks to operate the codec and filter sections and bit clocks to clock the data in and out from the PCM highway. FS<sub>x</sub> and FS<sub>r</sub> are 8 KHz inputs which set the sampling frequency and distinguish between signaling and non-signaling frames by their pulse width. A frame synchronization pulse which is one master clock wide designates a non-signaling frame, while a double wide sync pulse enables the signaling function. TS<sub>x</sub> is a timeslot strobe/buffer enable output which gates the PCM word onto the PCM highway when an external buffer is used to drive the line.

Data is transmitted on the highway at DX on the first eight positive transitions of CLK<sub>x</sub> following the rising edge of FS<sub>x</sub>. Similarly, on the receive side, data is received on the first eight falling edges of CLK<sub>r</sub>. The frequency of CLK<sub>x</sub> and CLK<sub>r</sub> is selected by the CLKSEL pin to be either 1.536, 1.544, or 2.048 MHz. No other frequency of operation is allowed in the fixed data rate mode.

Variable Data Rate Mode

Variable data rate timing is selected by connecting DCLKR to the bit clock for the receive PCM highway rather than to V<sub>BB</sub>. It employs master clocks CLK<sub>x</sub> and CLK<sub>r</sub>, bit clocks DCLK<sub>r</sub> and DCLK<sub>x</sub>, and frame synchronization clocks FS<sub>r</sub> and FS<sub>x</sub>.

Variable data rate timing allows for a flexible data frequency. It provides the ability to vary the frequency of the bit clocks, which can be asynchronous in the case of the 2914 or synchronous in the case of the 2913, from 64 KHz to 2.048 MHz. Master clock's inputs are still restricted to 1.536, 1.544, or 2.048 MHz.

In this mode, DCLK<sub>r</sub> and DCLK<sub>x</sub> become the data clocks for the receive and transmit PCM highways. While FS<sub>x</sub> is high, PCM data from DX is transmitted onto the highway on the next eight consecutive positive transitions of DCLK<sub>x</sub>. Similarly, while FS<sub>r</sub> is high, each PCM bit from the highway is received by DR on the next eight consecutive negative transitions of DCLK<sub>r</sub>.

On the transmit side, the PCM word will be repeated in all remaining timeslots in the 125 μs frame as long as DCLK<sub>x</sub> is pulsed and FS<sub>x</sub> is held high. This feature allows the PCM word to be transmitted to the PCM highway more than once per frame, if desired, and is only available in the variable data rate mode. Conversely, signaling is only allowed in the fixed data rate mode since the variable mode provides no means with which to specify a signaling frame.

Signaling

Signaling can only be performed with the 24-pin device in the fixed data rate timing mode (DCLKR = V<sub>BB</sub>). Signaling frames on the transmit and receive sides are independent of one another and are selected by a double-width frame sync pulse on the appropriate channel. During a transmit signaling frame, the codec will encode the incoming analog signal and substitute the signal present on SIG<sub>x</sub> for the least significant bit of the encoded PCM word. Similarly, in a receive signaling frame, the codec will decode the seven most significant bits according to CCITT recommendation G.733 and output the logical state of the LSB on the SIG<sub>r</sub> lead until it is updated in the next signaling frame. Timing relationships for signaling operation are shown in Figure 4.

---

**Figure 4.** Signaling Timing (Used Only with Fixed Data Rate Mode)
Asynchronous Operation

The 2914 can be operated with asynchronous clocks in either the fixed or variable data rate modes. In order to avoid crosstalk problems associated with special interrupt circuitry, the design of the Intel 2913/2914 combochip includes separate digital-to-analog converters and voltage references on the transmit and receive sides to allow independent operation of the two channels.

In either timing mode, the master clock, data clock, and timeslot strobe must be synchronized at the beginning of each frame. CLKx and DCLKx are synchronized once per frame but may be of different frequencies. The receive channel operates in a similar manner and is completely independent of the transmit timing (refer to Variable Data Rate Timing Diagrams). This approach requires the provision of two separate master clocks, even in variable data rate mode, but avoids the use of a synchronizer which can cause intermittent data conversion errors.

Analog Loopback

A distinctive feature of the 2914 is its analog loopback capability. This feature allows the user to send a control signal which internally connects the analog input and output ports. As shown in Figure 5, when LOOP is TTL high the analog output (PWRO+) is internally connected to the analog input (VFxl+), GSx is internally connected to PWRO-, and VFxl− is internally connected to GSx.

With this feature, the user can test the line circuit remotely by comparing the digital codes sent into the receive channel (DR) with those generated on the transmit channel (DX). Due to the difference in transmission levels between the transmit and receive sides, a 0 dBm0 code sent into DR will emerge from DX as a +3 dBm0 code, an implicit gain of 3 dB. Thus, the maximum signal input level which can be tested using analog loopback is 0 dBm0.

Precision Voltage References

No external components are required with the combochip to provide the voltage reference function. Voltage references are generated on-chip and are calibrated during the manufacturing process. The technique uses a difference in sub-surface charge density between two suitably implanted MOS devices to derive a temperature and bias stable reference voltage. These references determine the gain and dynamic range characteristics of the device.

Separate references are supplied to the transmit and receive sections and each is trimmed independently during the manufacturing process. The reference value is then further trimmed in the gain setting op-amps to a final precision value. With this method the combochip can achieve the extremely accurate Digital Milliwatt Responses specified in the transmission parameters, providing the user a significant margin for error in other board components.

Conversion Laws

The 2913 and 2914 are designed to operate in both μ-law and A-law systems. The user can select either conversion law according to the voltage present on the SIGx/ASEL pin. In each case the coder and decoder process a compressed 8-bit PCM word following CCITT recommendation G.711 for μ-law and A-law conversion. If A-law operation is desired, SIGx should be tied to Vss. Thus, signaling is not allowed during A-law operation. If μ = 255-law operation is selected, then SIGx is a TTL level input which modifies the LSB of the PCM output in signaling frames.

Figure 5. Simplified Block Diagram of 2914 Combochip in the Analog Loopback Configuration
TRANSMIT OPERATION

Transmit Filter

The input section provides gain adjustment in the passband by means of an on-chip operational amplifier. This operational amplifier has a common mode range of ±2.17 volts, a DC offset of 25 mV, an open loop voltage gain of 5000, and a unity gain bandwidth of typically 1 MHz. Gain of up to 20 dB can be set without degrading the performance of the filter. The load impedance to ground (GRDA) at the amplifier output (GSx) must be greater than 10 kΩ in parallel with less than 50 pF. The input signal on lead VFxl+ can be either AC or DC coupled. The input op amp can also be used in the inverting mode or differential amplifier mode (see Figure 6).

A low pass anti-aliasing section is included on-chip. This section typically provides 35 dB attenuation at the sampling frequency. No external components are required to provide the necessary anti-aliasing function for the switched capacitor section of the transmit filter.

The passband section provides flatness and stopband attenuation which fulfills the AT&T 03/04 channel bank transmission specification and CCITT recommendation G.714. The 2913 and 2914 specifications meet or exceed digital class 5 central office switching systems requirements. The transmit filter transfer characteristics and specifications will be within the limits shown in Figure 8.

A high pass section configuration was chosen to reject low frequency noise from 50 Hz and 60 Hz power lines, 17 Hz European electric railroads, ringing frequencies and their harmonics, and other low frequency noise. Even though there is high rejection at these frequencies, the sharpness of the band edge gives low attenuation at 200 Hz. This feature allows the use of low-cost transformer hybrids without external components.

Encoding

The encoder internally samples the output of the transmit filter and holds each sample on an internal sample and hold capacitor. The encoder then performs an analog to digital conversion on a switched capacitor array. Digital data representing the sample is transmitted on the first eight data clock bits of the next frame.

An on-chip autozero circuit corrects for DC-offset on the input signal to the encoder. This autozero circuit uses the sign bit averaging technique; the sign bit from the encoder output is long term averaged and subtracted from the input to the encoder. In this way, all DC offset is removed from the encoder input waveform.

RECEIVE OPERATION

Decoding

The PCM word at the Dr lead is serially fetched on the first eight data clock bits of the frame. A D/A conversion is performed on the digital word and the corresponding analog sample is held on an internal sample and hold capacitor. This sample is then transferred to the receive filter.

Receive Filter

The receive filter provides passband flatness and stopband rejection which fulfills both the AT&T D3/D4 specification and CCITT recommendation G.714. The filter contains the required compensation for the \(\frac{\sin x}{x}\) response of such decoders. The receive filter characteristics and specifications are shown in Figure 9.

Receive Output Power Amplifiers

A balanced output amplifier is provided in order to allow maximum flexibility in output configuration. Either of the two outputs can be used single ended (referenced to GRDA) to drive single ended loads. Alternatively, the differential output will drive a bridged load directly. The output stage is capable of driving loads as low as 300 ohms single ended or 600 ohms differentially.
### Table 4. Zero Transmission Level Points

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Parameter</th>
<th>Value</th>
<th>Units</th>
<th>Test Conditions</th>
</tr>
</thead>
<tbody>
<tr>
<td>0TLP1X</td>
<td>Zero Transmission Level Point</td>
<td>+2.76</td>
<td>dBm</td>
<td>Referenced to 600Ω</td>
</tr>
<tr>
<td></td>
<td>Transmit Channel (0 dBm0) µ-Law</td>
<td>1.00</td>
<td>dBm</td>
<td>Referenced to 900Ω</td>
</tr>
<tr>
<td>0TLP2X</td>
<td>Zero Transmission Level Point</td>
<td>+2.79</td>
<td>dBm</td>
<td>Referenced to 600Ω</td>
</tr>
<tr>
<td></td>
<td>Transmit Channel (0 dBm0) A-Law</td>
<td>1.03</td>
<td>dBm</td>
<td>Referenced to 900Ω</td>
</tr>
<tr>
<td>0TLP1R</td>
<td>Zero Transmission Level Point</td>
<td>+5.76</td>
<td>dBm</td>
<td>Referenced to 600Ω</td>
</tr>
<tr>
<td></td>
<td>Receive Channel (0 dBm0) µ-Law</td>
<td>4.00</td>
<td>dBm</td>
<td>Referenced to 900Ω</td>
</tr>
<tr>
<td>0TLP2R</td>
<td>Zero Transmission Level Point</td>
<td>+5.79</td>
<td>dBm</td>
<td>Referenced to 600Ω</td>
</tr>
<tr>
<td></td>
<td>Receive Channel (0 dBm0) A-Law</td>
<td>4.03</td>
<td>dBm</td>
<td>Referenced to 900Ω</td>
</tr>
</tbody>
</table>

The receive channel transmission level may be adjusted between specified limits by manipulation of the GSR input. GSR is internally connected to an analog gain setting network. When GSR is strapped to PWRO−, the receive level is maximized; when it is tied to PWRO+, the level is minimized. The output transmission level interpolates between 0 dB and −12 dB as GSR is interpolated (with a potentiometer) between PWRO+ and PWRO−. The use of the output gain set is illustrated in Figure 7.

Transmission levels are specified relative to the receive channel output under digital milliwatt conditions, that is, when the digital input at DA is the eight-code sequence specified in CCITT recommendation G.711.

### OUTPUT GAIN SET: DESIGN CONSIDERATIONS

(Refer to Figure 7)

PWRO+ and PWRO− are low impedance complementary outputs. The voltages at the nodes are:

\[ V_0^+ \text{ at PWRO}^+ \]
\[ V_0^- \text{ at PWRO}^- \]
\[ V_0 = (V_0^+) - (V_0^-) \text{(total differential response)} \]
\[ R_1 \text{ and } R_2 \text{ are a gain setting resistor network with the center tap connected to the GSR input.} \]

A value greater than 10K ohms for \( R_1 + R_2 \) and less than 100K ohms for \( R_1 \) in parallel with \( R_2 \) is recommended because:

(a) The parallel combination of \( R_1 + R_2 \) and \( R_L \) sets the total loading.

(b) The total capacitance at the GSR input and the parallel combination of \( R_1 \) and \( R_2 \) define a time constant which has to be minimized to avoid inaccuracies.

\[ A = \frac{1 + (R_1/R_2)}{4 + (R_1/R_2)} \]

For design purposes, a useful form is \( R_1/R_2 \) as a function of \( A \).

\[ R_1/R_2 = \frac{4A - 1}{1 - A} \]

(Allowable values for \( A \) are those which make \( R_1/R_2 \) positive.)

Examples are:

If \( A = 1 \) (maximum output), then

\[ R_1/R_2 = \infty \text{ or } V(GSR) = V_0^-; \]

i.e., GSR is tied to PWRO−

If \( A = \frac{1}{2} \), then

\[ R_1/R_2 = 2 \]

If \( A = \frac{1}{4} \), (minimum output) then

\[ R_1/R_2 = 0 \text{ or } V(GSR) = V_0^+; \]

i.e., GSR is tied to PWRO+.
**ABSOLUTE MAXIMUM RATINGS**

Temperature under Bias .................. $-10^\circ C$ to $+80^\circ C$

Storage Temperature .................. $-65^\circ C$ to $+150^\circ C$

$V_{CC}$ and GRDD
with Respect to $V_{BB}$ .................. $-0.3V$ to $+15V$

All Input and Output Voltages
with Respect to $V_{BB}$ .................. $-0.3V$ to $+15V$

Power Dissipation .................. $1.35W$

*Notice: Stresses above those listed under “Absolute Maximum Ratings” may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions above those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.*

**D.C. CHARACTERISTICS**

$T_A = 0^\circ C$ to $70^\circ C$, $V_{CC} = +5V \pm 5\%$, $V_{BB} = -5V \pm 5\%$, GRDA = 0V, GRDD = 0V, unless otherwise specified.

Typical values are for $T_A = 25^\circ C$ and nominal power supply values.

**DIGITAL INTERFACE**

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Parameter</th>
<th>Min</th>
<th>Typ</th>
<th>Max</th>
<th>Unit</th>
<th>Test Conditions</th>
</tr>
</thead>
<tbody>
<tr>
<td>$I_{IL}$</td>
<td>Low Level Input Current</td>
<td>$10 \mu A$</td>
<td>$GRDD \leq V_{IN} \leq V_{IL}(1)$</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$I_{IH}$</td>
<td>High Level Input Current</td>
<td>$10 \mu A$</td>
<td>$V_{IH} \leq V_{IN} \leq V_{CC}$</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$V_{IL}$</td>
<td>Input Low Voltage, except CLKSEL</td>
<td>0.8</td>
<td>$V$</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$V_{IH}$</td>
<td>Input High Voltage, except CLKSEL</td>
<td>2.0</td>
<td>$V$</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$V_{OL}$</td>
<td>Output Low Voltage</td>
<td>0.4</td>
<td>$V$</td>
<td>$I_{OL} = 3.2mA$ at $D_X, T^S_X$ and $SIG_R$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$V_{OH}$</td>
<td>Output High Voltage</td>
<td>2.4</td>
<td>$V$</td>
<td>$I_{OH} = 9.6mA$ at $D_X, I_{OH} = 1.2mA$ at $SIG_R$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$V_{ILO}$</td>
<td>Input Low Voltage, CLKSEL(^{(2)})</td>
<td>$V_{BB}$</td>
<td>$V_{BB}$</td>
<td>$V$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$V_{IIO}$</td>
<td>Input Intermediate Voltage, CLKSEL</td>
<td>$GRDD - 0.5$</td>
<td>0.5</td>
<td>$V$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$V_{IHO}$</td>
<td>Input High Voltage, CLKSEL</td>
<td>$V_{CC} - 0.5$</td>
<td>$V_{CC}$</td>
<td>$V$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$C_{OX}$</td>
<td>Digital Output Capacitance(^{(3)})</td>
<td>5</td>
<td>$pF$</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$C_{IN}$</td>
<td>Digital Input Capacitance</td>
<td>5</td>
<td>10</td>
<td>$pF$</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
D.C. CHARACTERISTICS

$T_A = 0\degree C$ to $70\degree C$, $V_{CC} = +5V \pm 5\%$, $V_{BB} = -5V \pm 5\%$, GRDA = 0V, GRDD = 0V, unless otherwise specified.

Typical values are for $T_A = 25\degree C$ and nominal power supply values (Continued)

POWER DISSIPATION

All measurements made at $f_{DCLK} = 2.048$ MHz, outputs unloaded.

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Parameter</th>
<th>Min</th>
<th>Typ</th>
<th>Max</th>
<th>Unit</th>
<th>Test Conditions</th>
</tr>
</thead>
<tbody>
<tr>
<td>$I_{CC1}$</td>
<td>$V_{CC}$ Operating Current</td>
<td>14</td>
<td>19</td>
<td>mA</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$I_{BB1}$</td>
<td>$V_{BB}$ Operating Current</td>
<td>-18</td>
<td>-24</td>
<td>mA</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
| $I_{CC0}$ | $V_{CC}$ Power Down Current | 0.5 | 1.0 | mA | | PDN $\leq V_{IL}$; after 10 $\mu s$
| $I_{BB0}$ | $V_{BB}$ Power Down Current | -0.5 | -1.0 | mA | | PDN $\leq V_{IL}$; after 10 $\mu s$
| $I_{CCS}$ | $V_{CC}$ Standby Current | 1.2 | 2.4 | mA | | FSX, FSR $\leq V_{IL}$; after 300 ms
| $I_{BBS}$ | $V_{BB}$ Standby Current | -1.2 | -2.4 | mA | | FSX, FSR $\leq V_{IL}$; after 300 ms
| $P_{D1}$ | Operating Power Dissipation | 140 | 200 | mW | | |
| $P_{D0}$ | Power Down Dissipation | 5 | 10 | mW | PDN $\leq V_{IL}$; after 10 $\mu s$
| $P_{ST}$ | Standby Power Dissipation | 12 | 25 | mW | | FSX, FSR $\leq V_{IL}$

NOTES:

1. $V_{IN}$ is the voltage on any digital pin.
2. SIGX and DCLKR are TTL level inputs between GRDD and $V_{CC}$; they are also pin straps for mode selection when tied to VSS. Under these conditions $V_{ILC}$ is the input low voltage requirement.
3. Timing parameters are guaranteed based on a 100 pF load capacitance. Up to eight digital outputs may be connected to a common PCM highway without buffering, assuming a board capacitance of 60 pF.
4. With nominal power supply values.
5. $V_{CC}$ applied last or simultaneously with $V_{BB}$.

ANALOG INTERFACE, TRANSMIT CHANNEL INPUT STAGE

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Parameter</th>
<th>Min</th>
<th>Typ</th>
<th>Max</th>
<th>Unit</th>
<th>Test Conditions</th>
</tr>
</thead>
</table>
| $I_{BXI}$ | Input Leakage Current, $V_{F/X1+}$, $V_{F/X1-}$ | 100 | nA | | $-2.17V \leq V_{IN} \leq 2.17V$
| $R_{IXI}$ | Input Resistance, $V_{F/X1+}$, $V_{F/X1-}$ | 10 | M$\Omega$ | | |
| $V_{OSX1}$ | Input Offset Voltage, $V_{F/X1+}$, $V_{F/X1-}$ | 25 | mV | | |
| CMRR | Common Mode Rejection, $V_{F/X1+}$, $V_{F/X1-}$ | 55 | dB | $-2.17V \leq V_{IN} \leq 2.17V$
| $A_{VOL}$ | DC Open Loop Voltage Gain, $G_{S_X}$ | 5000 | | | |
| $f_{C}$ | Open Loop Unity Gain Bandwidth, $G_{S_X}$ | 1 | MHz | | |
| $C_{LXI}$ | Load Capacitance, $G_{S_X}$ | 50 | pF | | |
| $R_{LXI}$ | Minimum Load Resistance, $G_{S_X}$ | 10 | K$\Omega$ | | |

ANALOG INTERFACE, RECEIVE CHANNEL DRIVER AMPLIFIER STAGE

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Parameter</th>
<th>Min</th>
<th>Typ</th>
<th>Max</th>
<th>Unit</th>
<th>Test Conditions</th>
</tr>
</thead>
<tbody>
<tr>
<td>$R_{ORA}$</td>
<td>Output Resistance, $PWR0+$, $PWR0-$</td>
<td>1</td>
<td>$\Omega$</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$V_{OSRA}$</td>
<td>Single-Ended Output DC Offset, $PWR0+$, $PWR0-$</td>
<td>75</td>
<td>$\pm 150$ mV</td>
<td>Relative to GRDA</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$C_{LRA}$</td>
<td>Load Capacitance, $PWR0+$, $PWR0-$</td>
<td>100</td>
<td>pF</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
A.C. CHARACTERISTICS—TRANSMISSION PARAMETERS

Unless otherwise noted, the analog input is a 0 dBm0, 1020 Hz sine wave. Input amplifier is set for unity gain, noninverting. The digital input is a PCM bit stream generated by passing a 0 dBm0, 1020 Hz sine wave through an ideal encoder. Receive output is measured single ended, maximum gain configuration. All output levels are (sin x)/x corrected. Specifications are for synchronous operation. Typical values are for TA = 25°C and nominal power supply values. (TA = 0°C to +70°C; VCC = +5V ±5%; VBB = -5V ±5%; GRDA = 0V; GRDD = 0V; unless otherwise specified).

GAIN AND DYNAMIC RANGE

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Parameter</th>
<th>Min</th>
<th>Typ</th>
<th>Max</th>
<th>Unit</th>
<th>Test Conditions</th>
</tr>
</thead>
<tbody>
<tr>
<td>EmW</td>
<td>Encoder Milliwatt Response Tolerance</td>
<td>-0.18</td>
<td>±0.04</td>
<td>+0.18</td>
<td>dBm0</td>
<td>Signal input of 1.064 Vrms µ-law</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Signal input of 1.068 Vrms A-law</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>TA = 25°C; VBB = -5V; VCC = +5V</td>
</tr>
<tr>
<td>EmWTS</td>
<td>EmW Variation with Temperature and Supplies</td>
<td>-0.07</td>
<td>±0.02</td>
<td>+0.07</td>
<td>dB</td>
<td>±5% supplies, 0 to 70°C Relative to nominal conditions</td>
</tr>
<tr>
<td>DmW</td>
<td>Digital Milliwatt Response Tolerance</td>
<td>-0.18</td>
<td>±0.04</td>
<td>+0.18</td>
<td>dBm0</td>
<td>Measure relative to OTLP, Signal input per CCITT Recommendation G.711.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Output signal of 1000 Hz; RL = ∞; TA = 25°C;</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>VBB = -5V; VCC = +5V.</td>
</tr>
<tr>
<td>DmWTS</td>
<td>DmW Variation with Temperature and Supplies</td>
<td>-0.07</td>
<td>+0.02</td>
<td>+0.07</td>
<td>dB</td>
<td>±5% supplies, 0 to 70°C</td>
</tr>
</tbody>
</table>

NOTES:
1. 0 dBm0 is defined as the zero reference point of the channel under test (0TLP). This corresponds to an analog signal input of 1.064 Vrms or an output of 1.503 Vrms for µ-law. See Table 4.
2. Unity gain input amplifier: GSX is connected to VFX-; Signal input VFX+; Maximum gain output amplifier; GSR is connected to PWRO-, output to PWRO+.

GAIN TRACKING Reference Level = -10 dBm0

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Parameter</th>
<th>2913-1,2914-1</th>
<th>2913,2914</th>
<th>Unit</th>
<th>Test Conditions</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>2913,2914</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GT1X</td>
<td>Transmit Gain Tracking Error</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Sinusoidal Input; µ-Law</td>
<td>±0.2</td>
<td>±0.3</td>
<td>±0.65</td>
<td>dB</td>
</tr>
<tr>
<td></td>
<td></td>
<td>±0.25</td>
<td>±0.5</td>
<td>±1.2</td>
<td>dB</td>
</tr>
<tr>
<td></td>
<td></td>
<td>±0.25</td>
<td></td>
<td></td>
<td>dB</td>
</tr>
<tr>
<td>GT2X</td>
<td>Transmit Gain Tracking Error</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Sinusoidal Input; A-Law</td>
<td>±0.2</td>
<td>±0.3</td>
<td>±0.65</td>
<td>dB</td>
</tr>
<tr>
<td></td>
<td></td>
<td>±0.25</td>
<td>±0.5</td>
<td>±1.2</td>
<td>dB</td>
</tr>
<tr>
<td></td>
<td></td>
<td>±0.25</td>
<td></td>
<td></td>
<td>dB</td>
</tr>
<tr>
<td>GT1R</td>
<td>Receive Gain Tracking Error</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Sinusoidal Input; µ-Law</td>
<td>±0.2</td>
<td>±0.3</td>
<td>±0.65</td>
<td>dB</td>
</tr>
<tr>
<td></td>
<td></td>
<td>±0.25</td>
<td>±0.5</td>
<td>±1.2</td>
<td>dB</td>
</tr>
<tr>
<td></td>
<td></td>
<td>±0.25</td>
<td></td>
<td></td>
<td>dB</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Measured at PWRO+, RL = 300Ω</td>
</tr>
<tr>
<td>GT2R</td>
<td>Receive Gain Tracking Error</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Sinusoidal Input; A-Law</td>
<td>±0.2</td>
<td>±0.3</td>
<td>±0.65</td>
<td>dB</td>
</tr>
<tr>
<td></td>
<td></td>
<td>±0.25</td>
<td>±0.5</td>
<td>±1.2</td>
<td>dB</td>
</tr>
<tr>
<td></td>
<td></td>
<td>±0.25</td>
<td></td>
<td></td>
<td>dB</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Measured at PWRO+, RL = 300Ω</td>
</tr>
</tbody>
</table>
## A.C. CHARACTERISTICS—TRANSMISSION PARAMETERS (Continued)

### NOISE

All receive channel measurements are single ended.

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Parameter</th>
<th>2913-1, 2914-1</th>
<th>2913, 2914</th>
<th>Unit</th>
<th>Test Conditions</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Min</td>
<td>Typ</td>
<td>Max</td>
<td>Min</td>
</tr>
<tr>
<td>(N_{XC1})</td>
<td>Transmit Noise, C-Message Weighted</td>
<td>13</td>
<td>15</td>
<td>dBm0</td>
<td>VF(X)(+) = GRDA,VF(X)(-) = GS(X)</td>
</tr>
<tr>
<td>(N_{XC2})</td>
<td>Transmit Noise, C-Message Weighted with Eighth Bit Signaling</td>
<td>16</td>
<td>18</td>
<td>dBm0</td>
<td>VF(X)(+) = GRDA,VF(X)(-) = GS(X); 6th Frame Signaling</td>
</tr>
<tr>
<td>(N_{XP})</td>
<td>Transmit Noise, Psychometrically Weighted</td>
<td>-77</td>
<td>-75</td>
<td>dBm0</td>
<td>VF(X)(+) = GRDA,VF(X)(-) = GS(X)</td>
</tr>
<tr>
<td>(N_{RC1})</td>
<td>Receive Noise, C-Message Weighted: Quiet Code</td>
<td>8</td>
<td>11</td>
<td>dBm0</td>
<td>(D_R = 11111111)</td>
</tr>
<tr>
<td>(N_{RC2})</td>
<td>Receive Noise, C-Message Weighted: Sign Bit Toggle</td>
<td>9</td>
<td>12</td>
<td>dBm0</td>
<td>Input to (D_R) is zero code with sign bit toggle at 1 KHz rate</td>
</tr>
<tr>
<td>(N_{RP})</td>
<td>Receive Noise, Psychometrically Weighted</td>
<td>-82</td>
<td>-79</td>
<td>dBm0</td>
<td>(D_R = ) lowest positive decode level</td>
</tr>
<tr>
<td>(N_{SF})</td>
<td>Single Frequency Noise End to End Measurement</td>
<td>-50</td>
<td>-50</td>
<td>dBm0</td>
<td>CCITT G.712.4.2, measure at PWRO+</td>
</tr>
<tr>
<td>(PSRR_1)</td>
<td>(V_{CC}) Power Supply Rejection, Transmit Channel</td>
<td>-30</td>
<td>-30</td>
<td>dB</td>
<td>Idle channel; 200 mV P-P signal on supply; 0 to 50 KHz, measure at (D_X)</td>
</tr>
<tr>
<td>(PSRR_2)</td>
<td>(V_{BB}) Power Supply Rejection, Transmit Channel</td>
<td>-30</td>
<td>-30</td>
<td>dB</td>
<td>Idle channel; 200 mV P-P signal on supply; 0 to 50 KHz, measure at (D_X)</td>
</tr>
<tr>
<td>(PSRR_3)</td>
<td>(V_{CC}) Power Supply Rejection, Receive Channel</td>
<td>-25</td>
<td>-25</td>
<td>dB</td>
<td>Idle channel; 200 mV P-P signal on supply; measure narrow band at PWRO+ , 0 to 50 KHz</td>
</tr>
<tr>
<td>(PSRR_4)</td>
<td>(V_{BB}) Power Supply Rejection, Receive Channel</td>
<td>-25</td>
<td>-25</td>
<td>dB</td>
<td>Idle channel; 200 mV P-P signal on supply; measure narrow band at PWRO+ , 0 to 50 KHz</td>
</tr>
<tr>
<td>(CT_{TR})</td>
<td>Crosstalk, Transmit to Receive</td>
<td>-80</td>
<td>-71</td>
<td>dB</td>
<td>VF(X)(+) = 0 dBm0, 1.02 KHz, (D_R = ) lowest positive decode level, measure at PWRO+</td>
</tr>
<tr>
<td>(CT_{RT})</td>
<td>Crosstalk, Receive to Transmit</td>
<td>-80</td>
<td>-71</td>
<td>dB</td>
<td>(D_R = 0 \text{ dBm0, 1.02 KHz}) VF(X)(+) = GRDA, measure at (D_X)</td>
</tr>
</tbody>
</table>
# A.C. CHARACTERISTICS—TRANSMISSION PARAMETERS (Continued)

## DISTORTION

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Parameter</th>
<th>Min</th>
<th>Typ</th>
<th>Max</th>
<th>Unit</th>
<th>Test Conditions</th>
</tr>
</thead>
<tbody>
<tr>
<td>SD1&lt;sub&gt;X&lt;/sub&gt;</td>
<td>Transmit Signal to Distortion, μ-Law Sinusoidal Input; CCITT G.714-Method 2</td>
<td>36</td>
<td>dB</td>
<td>0 to −30 dBm0</td>
<td>30</td>
<td>dB</td>
</tr>
<tr>
<td>SD2&lt;sub&gt;X&lt;/sub&gt;</td>
<td>Transmit Signal to Distortion, A-Law Sinusoidal Input; CCITT G.714-Method 2</td>
<td>36</td>
<td>dB</td>
<td>0 to −30 dBm0</td>
<td>30</td>
<td>dB</td>
</tr>
<tr>
<td>SD1&lt;sub&gt;R&lt;/sub&gt;</td>
<td>Receive Signal to Distortion, μ-Law Sinusoidal Input; CCITT G.714-Method 2</td>
<td>36</td>
<td>dB</td>
<td>0 to −30 dBm0</td>
<td>30</td>
<td>dB</td>
</tr>
<tr>
<td>SD2&lt;sub&gt;R&lt;/sub&gt;</td>
<td>Receive Signal to Distortion, A-Law Sinusoidal Input; CCITT G.714-Method 2</td>
<td>36</td>
<td>dB</td>
<td>0 to −30 dBm0</td>
<td>30</td>
<td>dB</td>
</tr>
<tr>
<td>DP&lt;sub&gt;X&lt;/sub&gt;</td>
<td>Transmit Single Frequency Distortion Products</td>
<td>−46</td>
<td>dBm0</td>
<td>AT&amp;T Advisory #64 (3.8)</td>
<td>0 dBm0 Input Signal</td>
<td></td>
</tr>
<tr>
<td>DP&lt;sub&gt;R&lt;/sub&gt;</td>
<td>Receive Single Frequency Distortion Products</td>
<td>−46</td>
<td>dBm0</td>
<td>AT&amp;T Advisory #64 (3.8)</td>
<td>0 dBm0 Input Signal</td>
<td></td>
</tr>
<tr>
<td>IMD&lt;sub&gt;1&lt;/sub&gt;</td>
<td>Intermodulation Distortion, End to End Measurement</td>
<td>−35</td>
<td>dB</td>
<td>CCITT G.712 (7.1)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>IMD&lt;sub&gt;2&lt;/sub&gt;</td>
<td>Intermodulation Distortion, End to End Measurement</td>
<td>−49</td>
<td>dBm0</td>
<td>CCITT G.712 (7.2)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SOS</td>
<td>Spurious Out of Band Signals, End to End Measurement</td>
<td>−25</td>
<td>dBm0</td>
<td>CCITT G.712 (6.1)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SIS</td>
<td>Spurious in Band Signals, End to End Measurement</td>
<td>−40</td>
<td>dBm0</td>
<td>CCITT G.712 (9)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>D&lt;sub&gt;AX&lt;/sub&gt;</td>
<td>Transmit Absolute Delay</td>
<td>245</td>
<td>μs</td>
<td>Fixed Data Rate. CLK&lt;sub&gt;X&lt;/sub&gt; = 2.048 MHz</td>
<td>0 dBm0, 1.02 KHz</td>
<td>signal at VF&lt;sub&gt;X&lt;/sub&gt;L+</td>
</tr>
<tr>
<td>D&lt;sub&gt;DX&lt;/sub&gt;</td>
<td>Transmit Differential Envelope Delay Relative to D&lt;sub&gt;AX&lt;/sub&gt;</td>
<td>170</td>
<td>μs</td>
<td>f = 500 − 600 Hz</td>
<td>95</td>
<td>μs</td>
</tr>
<tr>
<td>D&lt;sub&gt;AR&lt;/sub&gt;</td>
<td>Receive Absolute Delay</td>
<td>190</td>
<td>μs</td>
<td>Fixed Data Rate, CLK&lt;sub&gt;R&lt;/sub&gt; = 2.048 MHz; Digital input is DMW codes. Measure at PWRO+</td>
<td></td>
<td></td>
</tr>
<tr>
<td>D&lt;sub&gt;DR&lt;/sub&gt;</td>
<td>Receive Differential Envelope Delay Relative to D&lt;sub&gt;AR&lt;/sub&gt;</td>
<td>45</td>
<td>μs</td>
<td>f = 500 − 600 Hz</td>
<td>35</td>
<td>μs</td>
</tr>
</tbody>
</table>
### A.C. CHARACTERISTICS—TRANSMISSION PARAMETERS (Continued)

#### TRANSMIT CHANNEL TRANSFER CHARACTERISTICS

Input amplifier is set for unity gain; noninverting; maximum gain output.

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Parameter</th>
<th>Min</th>
<th>Typ</th>
<th>Max</th>
<th>Unit</th>
<th>Test Conditions</th>
</tr>
</thead>
<tbody>
<tr>
<td>$G_{RX}$</td>
<td>Gain Relative to Gain at 1.02 KHz</td>
<td>-30</td>
<td>-25</td>
<td>-23</td>
<td>dB</td>
<td>0 dBm0 Signal input at $V_F\times L$ +</td>
</tr>
<tr>
<td>16.67 Hz</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>50 Hz</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>60 Hz</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>200 Hz</td>
<td></td>
<td>-1.8</td>
<td>-0.125 dB</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>300 Hz</td>
<td></td>
<td>-0.125</td>
<td>+0.125 dB</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3300 Hz</td>
<td></td>
<td>-0.35</td>
<td>+0.03 dB</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3400 Hz</td>
<td></td>
<td>-0.7</td>
<td>-0.10 dB</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4000 Hz</td>
<td></td>
<td>-14</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4600 Hz and Above</td>
<td></td>
<td>-32</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Figure 8. Transmit Channel**

6-61
### RECEIVE CHANNEL TRANSFER CHARACTERISTICS

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Parameter</th>
<th>Min</th>
<th>Typ</th>
<th>Max</th>
<th>Unit</th>
<th>Test Conditions</th>
</tr>
</thead>
<tbody>
<tr>
<td>$G_{RR}$</td>
<td>Gain Relative to Gain at 1.02 KHz</td>
<td>+0.125</td>
<td>dB</td>
<td></td>
<td></td>
<td>0 dBm0 Signal input at $D_R$</td>
</tr>
<tr>
<td></td>
<td>Below 200 Hz</td>
<td>-0.5</td>
<td>dB</td>
<td>+0.125</td>
<td>dB</td>
<td>200 Hz</td>
</tr>
<tr>
<td></td>
<td>300 to 3000 Hz</td>
<td>-0.125</td>
<td>dB</td>
<td>+0.125</td>
<td>dB</td>
<td>3000 Hz</td>
</tr>
<tr>
<td></td>
<td>3400 Hz</td>
<td>-0.35</td>
<td>dB</td>
<td>+0.03</td>
<td>dB</td>
<td>3400 Hz</td>
</tr>
<tr>
<td></td>
<td>4000 Hz</td>
<td>-0.7</td>
<td>dB</td>
<td>-1.4</td>
<td>dB</td>
<td>4000 Hz</td>
</tr>
<tr>
<td></td>
<td>4600 Hz and Above</td>
<td>-30</td>
<td>dB</td>
<td></td>
<td></td>
<td>4600 Hz and Above</td>
</tr>
</tbody>
</table>

*Figure 9. Receive Channel*
A.C. CHARACTERISTICS—TIMING PARAMETERS

CLOCK SECTION

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Parameter</th>
<th>Min</th>
<th>Typ</th>
<th>Max</th>
<th>Unit</th>
<th>Test Conditions</th>
</tr>
</thead>
<tbody>
<tr>
<td>tCY</td>
<td>Clock Period, CLKX, CLKR</td>
<td>488</td>
<td></td>
<td>ns</td>
<td>ns</td>
<td>$f_{CLKX} = f_{CLKR} = 2.048 MHz$</td>
</tr>
<tr>
<td>tCLK</td>
<td>Clock Pulse Width, CLKX, CLKR</td>
<td>220</td>
<td></td>
<td>ns</td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>tDCLK</td>
<td>Data Clock Pulse Width</td>
<td>220</td>
<td></td>
<td>ns</td>
<td>ns</td>
<td>$64 \text{ KHz} \leq f_{DCLK} \leq 2.048 \text{ MHz}$</td>
</tr>
<tr>
<td>tDCDC</td>
<td>Clock Duty Cycle, CLKX, CLKR</td>
<td>45</td>
<td>50</td>
<td>55</td>
<td>%</td>
<td></td>
</tr>
<tr>
<td>t_r, t_f</td>
<td>Clock Rise and Fall Time</td>
<td>5</td>
<td>30</td>
<td></td>
<td>ns</td>
<td></td>
</tr>
</tbody>
</table>

TRANSMIT SECTION, FIXED DATA RATE MODE(1)

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Parameter</th>
<th>Min</th>
<th>Typ</th>
<th>Max</th>
<th>Unit</th>
<th>Test Conditions</th>
</tr>
</thead>
<tbody>
<tr>
<td>tDZX</td>
<td>Data Enabled on TS Entry</td>
<td>0</td>
<td></td>
<td>145</td>
<td>ns</td>
<td>$0 &lt; C_{LOAD} &lt; 100 \text{ pF}$</td>
</tr>
<tr>
<td>tDDX</td>
<td>Data Delay from CLKX</td>
<td>0</td>
<td></td>
<td>145</td>
<td>ns</td>
<td>$0 &lt; C_{LOAD} &lt; 100 \text{ pF}$</td>
</tr>
<tr>
<td>tHZX</td>
<td>Data Float on TS Exit</td>
<td>60</td>
<td></td>
<td>215</td>
<td>ns</td>
<td>$C_{LOAD} = 0$</td>
</tr>
<tr>
<td>tSON</td>
<td>Timeslot X to Enable</td>
<td>0</td>
<td></td>
<td>145</td>
<td>ns</td>
<td>$0 &lt; C_{LOAD} &lt; 100 \text{ pF}$</td>
</tr>
<tr>
<td>tSOFF</td>
<td>Timeslot X to Disable</td>
<td>60</td>
<td></td>
<td>215</td>
<td>ns</td>
<td>$C_{LOAD} = 0$</td>
</tr>
<tr>
<td>tFSO</td>
<td>Frame Sync Delay</td>
<td>100</td>
<td></td>
<td></td>
<td>tCLK</td>
<td>ns</td>
</tr>
<tr>
<td>tSS</td>
<td>Signal Setup Time</td>
<td>0</td>
<td></td>
<td></td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>tSH</td>
<td>Signal Hold Time</td>
<td>0</td>
<td></td>
<td></td>
<td>ns</td>
<td></td>
</tr>
</tbody>
</table>

RECEIVE SECTION, FIXED DATA RATE MODE

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Parameter</th>
<th>Min</th>
<th>Typ</th>
<th>Max</th>
<th>Unit</th>
<th>Test Conditions</th>
</tr>
</thead>
<tbody>
<tr>
<td>tDSR</td>
<td>Receive Data Setup</td>
<td>10</td>
<td></td>
<td></td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>tDHR</td>
<td>Receive Data Hold</td>
<td>60</td>
<td></td>
<td></td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>tFSD</td>
<td>Frame Sync Delay</td>
<td>100</td>
<td></td>
<td>tCLK</td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>tSIGR</td>
<td>SIGR Update</td>
<td>0</td>
<td></td>
<td>2</td>
<td>μs</td>
<td></td>
</tr>
</tbody>
</table>

NOTE:
1. Timing parameters $t_{DZX}$, $t_{HZX}$, and $t_{SOFF}$ are referenced to a high impedance state.
WAVEFORMS

Fixed Data Rate Timing

TRANSMIT TIMING

NOTE:
All timing parameters referenced to $V_{IH}$ and $V_{IL}$ except $t_{DZX}$, $t_{SOFF}$ and $t_{HZX}$ which reference a high impedance state.

RECEIVE TIMING

NOTE:
All timing parameters referenced to $V_{IH}$ and $V_{IL}$.
### WAVEFORMS (Continued)

#### TRANSMIT SECTION, VARIABLE DATA RATE MODE\(^{(1)}\)

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Parameter</th>
<th>Min</th>
<th>Typ</th>
<th>Max</th>
<th>Unit</th>
<th>Test Conditions</th>
</tr>
</thead>
<tbody>
<tr>
<td>t(TSDX)</td>
<td>Timeslot Delay from DCLK(_X)(^{(2)})</td>
<td>140</td>
<td></td>
<td></td>
<td></td>
<td>t(_{DX}) - 140 ns</td>
</tr>
<tr>
<td>t(FSD)</td>
<td>Frame Sync Delay</td>
<td>100</td>
<td></td>
<td></td>
<td></td>
<td>t(_{CY}) - 100 ns</td>
</tr>
<tr>
<td>t(DDX)</td>
<td>Data Delay from DCLK(_X)</td>
<td>0</td>
<td>100</td>
<td></td>
<td>ns</td>
<td>0 &lt; C(_{LOAD}) &lt; 100 pF</td>
</tr>
<tr>
<td>t(DON)</td>
<td>Timeslot to D(_X) Active</td>
<td>0</td>
<td>50</td>
<td></td>
<td>ns</td>
<td>0 &lt; C(_{LOAD}) &lt; 100 pF</td>
</tr>
<tr>
<td>t(DOFF)</td>
<td>Timeslot to D(_X) Inactive</td>
<td>0</td>
<td>80</td>
<td></td>
<td>ns</td>
<td>0 &lt; C(_{LOAD}) &lt; 100 pF</td>
</tr>
<tr>
<td>t(DX)</td>
<td>Data Clock Period</td>
<td>488</td>
<td></td>
<td>15620</td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>t(DFSX)</td>
<td>Data Delay from FS(_X)</td>
<td>0</td>
<td></td>
<td>140</td>
<td>ns</td>
<td></td>
</tr>
</tbody>
</table>

#### RECEIVE SECTION, VARIABLE DATA RATE MODE

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Parameter</th>
<th>Min</th>
<th>Typ</th>
<th>Max</th>
<th>Unit</th>
<th>Test Conditions</th>
</tr>
</thead>
<tbody>
<tr>
<td>t(TSDR)</td>
<td>Timeslot Delay from DCLK(_R)(^{(3)})</td>
<td>140</td>
<td></td>
<td></td>
<td></td>
<td>t(_{DR}) - 140 ns</td>
</tr>
<tr>
<td>t(FSD)</td>
<td>Frame Sync Delay</td>
<td>100</td>
<td></td>
<td></td>
<td></td>
<td>t(_{CY}) - 100 ns</td>
</tr>
<tr>
<td>t(DSR)</td>
<td>Data Setup Time</td>
<td>10</td>
<td></td>
<td></td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>t(DSR)</td>
<td>Data Hold Time</td>
<td>60</td>
<td></td>
<td></td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>t(DR)</td>
<td>Data Clock Period</td>
<td>488</td>
<td></td>
<td>15620</td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>t(SDR)</td>
<td>Timeslot End Receive Time</td>
<td>60</td>
<td></td>
<td></td>
<td>ns</td>
<td></td>
</tr>
</tbody>
</table>

#### 64 KB OPERATION, VARIABLE DATA RATE MODE

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Parameter</th>
<th>Min</th>
<th>Typ</th>
<th>Max</th>
<th>Unit</th>
<th>Test Conditions</th>
</tr>
</thead>
<tbody>
<tr>
<td>t(FSLX)</td>
<td>Transmit Frame Sync Minimum Downtime</td>
<td>488</td>
<td></td>
<td></td>
<td>ns</td>
<td>FS(_X) is TTL high for remainder of frame</td>
</tr>
<tr>
<td>t(FSLR)</td>
<td>Receive Frame Sync Minimum Downtime</td>
<td>1952</td>
<td></td>
<td></td>
<td>ns</td>
<td>FS(_R) is TTL high for remainder of frame</td>
</tr>
<tr>
<td>t(DCLK)</td>
<td>Data Clock Pulse Width</td>
<td>10</td>
<td></td>
<td>10</td>
<td>(\mu)s</td>
<td></td>
</tr>
</tbody>
</table>

**NOTES:**
1. Timing parameters for t\(DON\) and t\(DOFF\) are referenced to a high impedance state.
2. t\(FSLX\) minimum requirements override t\(TSDX\) maximum spec for 64 KHz operation.
3. t\(FSLR\) minimum requirements override t\(TSDR\) maximum spec for 64 KHz operation.
VARIABLE DATA RATE TIMING

TRANSMIT TIMING

RECEIVE TIMING

NOTE:
All timing parameters referenced to VIH and VIL except tDON and tDOFF which reference a high impedance state.

A.C. TESTING INPUT, OUTPUT WAVEFORM

A.C. Testing: Inputs are driven at 2.4V for a Logic “1” and 0.45V for Logic “0”. Timing measurements are made at 2.0V for a Logic “1” and 0.8V for a Logic “0”.

6-66
2916/2917
HMOS COMBINED SINGLE CHIP PCM CODEC AND FILTER

- 2916 μ-Law, 2.048 MHz Master Clock
- 2917 A-Law, 2.048 MHz Master Clock
- New 16-Pin Package for Higher Linecard Density
- AT&T D3/D4 and CCITT Compatible
- Variable Timing Mode for Flexible Digital Interface: Supports Data Rates from 64 KB to 2.048 MB
- Fully Differential Internal Architecture Enhances Noise Immunity
- Fixed Timing Mode for Standard 32-Channel Systems: 2.048 MHz Master Clock
- Low Power HMOS-E Technology
  - 5 mW Typical Power Down
  - 140 mW Typical Operating
- On-Chip Auto Zero, Sample and Hold, and Precision Voltage References
- Compatible with Direct Mode Intel 2910A, 2911A, and 2912A Designs

The Intel 2916 and 2917 are limited feature versions of Intel’s 2913 and 2914 combination codec/filter chips. They are fully integrated PCM codecs with transmit/receive filters fabricated in a highly reliable and proven N-channel HMOS silicon gate technology (HMOS-E). These devices provide the functions that were formerly provided by two complex chips (2910A or 2911A and 2912A). Besides the higher level of integration, the performance of the 2916 and 2917 is superior to that of the separate devices.

The primary applications for the 2916 and 2917 are in telephone systems:
- Switching—Digital PBX’s and Central Office Switching Systems
- Subscriber Instruments—Digital Handsets and Office Workstations

Other possible applications can be found where the wide dynamic range (78 dB) and minimum conversion time (125 μs) are required for analog to digital interface functions:
- High Speed Modems
- Voice Store and Forward
- Secure Communications
- Digital Echo Cancellation

Figure 1. Pin Configuration

270156-1
Figure 2. Block Diagram

Table 1. Pin Names

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>VBB</td>
<td>Power (-5V)</td>
<td>GSX</td>
<td>Transmit Gain Control</td>
</tr>
<tr>
<td>PWRO+</td>
<td>Power Amplifier Outputs</td>
<td>VFX1</td>
<td>Analog Input</td>
</tr>
<tr>
<td>PWRO-</td>
<td>Power Down Select</td>
<td>GRDA</td>
<td>Analog Ground</td>
</tr>
<tr>
<td>PDN</td>
<td>Receive Variable Data Clock</td>
<td>TSX</td>
<td>Timeslot Strobe/Buffer Enable</td>
</tr>
<tr>
<td>DCLKR</td>
<td>Receive PCM Input</td>
<td>DCLKX</td>
<td>Transmit Variable Data Clock</td>
</tr>
<tr>
<td>DR</td>
<td>Receive Frame Synchronization Clock</td>
<td>DX</td>
<td>Transmit PCM Output</td>
</tr>
<tr>
<td>FSX</td>
<td>Synchronization Clock</td>
<td>FSX</td>
<td>Transmit Frame Synchronization Clock</td>
</tr>
<tr>
<td>GRDD</td>
<td>Digital Ground</td>
<td>CLK</td>
<td>Master Clock</td>
</tr>
<tr>
<td>VCC</td>
<td>Power (+5V)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Symbol</td>
<td>Function</td>
<td></td>
<td></td>
</tr>
<tr>
<td>--------</td>
<td>----------</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>V_{BB}</strong></td>
<td>Most negative supply, input voltage is (-5) volts ±5%.</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>PWRO+</strong></td>
<td>Non-inverting output of power amplifier. Can drive transformer hybrids or high impedance loads directly in either a differential or single ended configuration.</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>PWRO−</strong></td>
<td>Inverting output of power amplifier. Functionally identical and complementary to PWRO+.</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>PDN</strong></td>
<td>Power down select. When PDN is TTL high, the device is active. When low, the device is powered down.</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>DCLK_R</strong></td>
<td>Selects the fixed or variable data rate mode. When DCLK_R is connected to V_{BB}, the fixed data rate mode is selected. When DCLK_R is not connected to V_{BB}, the device operates in the variable data rate mode. In this mode DCLK_R becomes the receive data clock which operates at TTL levels from 64 Kb to 2.048 Mb data rates.</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>DR</strong></td>
<td>Receive PCM input. PCM data is clocked in on this lead on eight consecutive negative transitions of the receive data clock; CLK in the fixed data rate mode and DCLK_R in variable data rate mode.</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>FS_R</strong></td>
<td>8 KHz frame synchronization clock input/timeslot enable, receive channel. In variable data rate mode this signal must remain high for the entire length of the timeslot. The receive channel enters the standby state whenever FS_R is TTL low for 300 milliseconds.</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>GRDD</strong></td>
<td>Digital ground for all internal logic circuits. Not internally tied to GRDA.</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>CLK</strong></td>
<td>Master and data clock for the fixed data rate mode; master clock only in variable data rate mode.</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>FS_X</strong></td>
<td>8 KHz frame synchronization clock input/timeslot enable, transmit channel. Operates independently but in an analogous manner to FS_R. The transmit channel enters the standby state whenever FS_X is TTL low for 300 milliseconds.</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>DX</strong></td>
<td>Transmit PCM output. PCM data is clocked out on this lead on eight consecutive positive transitions of the transmit data clock; CLK in fixed data rate mode and DCLK_X in variable data rate mode.</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>TS_X/DCLK_X</strong></td>
<td>Transmit channel timeslot strobe (output) or data clock (input) for the transmit channel. In fixed data rate mode, this pin is an open drain output designed to be used as an enable signal for a three-state buffer. In variable data rate mode, this pin becomes the transmit data clock which operates at TTL levels from 64 Kb to 2.048 Mb data rates.</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>GRDA</strong></td>
<td>Analog ground return for all internal voice circuits. Not internally connected to GRDD.</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>VF_XI−</strong></td>
<td>Inverting analog input to uncommitted transmit operational amplifier.</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>GS_X</strong></td>
<td>Output terminal of on-chip transmit channel input op amp. Internally, this is the voice signal input to the transmit filter.</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>V_{CC}</strong></td>
<td>Most positive supply; input voltage is +5 volts ±5%.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
FUNCTIONAL DESCRIPTION

The 2916 and 2917 provide the analog-to-digital and the digital-to-analog conversions and the transmit and receive filtering necessary to interface a full duplex (4 wires) voice telephone circuit with the PCM highways of a time division multiplexed (TDM) system. They are intended to be used at the analog termination of a PCM line.

The following major functions are provided:

- Bandpass filtering of the analog signals prior to encoding and after decoding
- Encoding and decoding of voice and call progress information
- Encoding and decoding of the signaling and supervision information

GENERAL OPERATION

System Reliability Features

The combochip can be powered up by pulsing FSX and/or FSR while a TTL high voltage is applied to PDN, provided that all clocks and supplies are connected. The 2916 and 2917 have internal resets on power up (or when VBB or VCC are re-applied) in order to ensure validity of the digital outputs and thereby maintain integrity of the PCM highway.

On the transmit channel, digital outputs DX and TSX are held in a high impedance state for approximately four frames (500 μs) after power up or application of VBB or VCC. After this delay, DX and TSX will be functional and will occur in the proper timeslot. The analog circuits on the transmit side require approximately 60 milliseconds to reach their equilibrium value due to the autozero circuit settling time.

Fixed Data Rate Mode

Fixed data rate timing, which is 2910A and 2911A compatible, is selected by connecting DCLKR to VBB. It employs master clock CLK, frame synchronization clocks FSX and FSR, and output TSX.

To enhance system reliability, TSX and DX will be placed in a high impedance state approximately 30 μs after an interruption of CLK.

Power Down and Standby Modes

To minimize power consumption, two power down modes are provided in which most 2916/2917 functions are disabled. Only the power down, clock, and frame sync buffers, which are required to power up the device, are enabled in these modes. As shown in Table 3, the digital outputs on the appropriate channels are placed in a high impedance state until the device returns to the active mode.

The Power Down mode utilizes an external control signal to the PDN pin. In this mode, power consumption is reduced to an average of 5 mW. The device is active when the signal is high and inactive when it is low. In the absence of any signal, the PDN pin floats to TTL high allowing the device to remain active continuously.

The Standby mode leaves the user an option of powering either channel down separately or powering the entire device down by selectively removing FSX and/or FSR. With both channels in the standby state, power consumption is reduced to an average of 12 mW. If transmit only operation is desired, FSX should be applied to the device while FSR is held low. Similarly, if receive only operation is desired, FSR should be applied while FSX is held low.

<table>
<thead>
<tr>
<th>Device Status</th>
<th>Power-Down Method</th>
<th>Typical Power Consumption</th>
<th>Digital Output Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>Power Down Mode</td>
<td>PDN = TTL Low</td>
<td>5 mW</td>
<td>TX and DX are placed in a high impedance state within 10 μs.</td>
</tr>
<tr>
<td>Standby Mode</td>
<td>FSX and FSR are TTL Low</td>
<td>12 mW</td>
<td>TX and DX are placed in a high impedance state within 300 milliseconds.</td>
</tr>
<tr>
<td>Only Transmit is on Standby</td>
<td>FSX is TTL Low</td>
<td>70 mW</td>
<td>TX and DX are placed in a high impedance state within 300 milliseconds.</td>
</tr>
<tr>
<td>Only Receive is on Standby</td>
<td>FSR is TTL Low</td>
<td>110 mW</td>
<td></td>
</tr>
</tbody>
</table>
CLK serves as the master clock to operate the codec and filter sections and as the bit clock to clock the data in and out from the PCM highway. FSX and FS_R are 8 KHz inputs which set the sampling frequency. TSX is a timeslot strobe/buffer enable output which gates the PCM word onto the PCM highway when an external buffer is used to drive the line.

Data is transmitted on the highway at DX on the first eight positive transitions of CLK following the rising edge of FSX. Similarly, on the receive side, data is received on the first eight falling edges of CLK. The frequency of CLK must be 2.048 MHz. No other frequency of operation is allowed in the fixed data rate mode.

Variable Data Rate Mode
Variable data rate timing is selected by connecting DCLK_R to the bit clock for the receive PCM highway rather than to V_BB. It employs master clock CLK, bit clocks DCLK_R and DCLK_X, and frame synchronization clocks FS_R and FS_X.

Variable data rate timing allows for a flexible data frequency. It provides the ability to vary the frequency of the bit clocks, from 64 KHz to 2.048 MHz. The master clock is still restricted to 2.048 MHz.

In this mode, DCLK_R and DCLK_X become the data clocks for the receive and transmit PCM highways. While FSX is high, PCM data from DX is transmitted onto the highway on the next eight consecutive positive transitions of DCLK_X. Similarly, while FS_R is high, each PCM bit from the highway is received by DR on the next eight consecutive negative transitions of DCLK_R.

On the transmit side, the PCM word will be repeated in all remaining timeslots in the 125 μs frame as long as DCLK_X is pulsed and FSX is held high. This feature allows the PCM word to be transmitted to the PCM highway more than once per frame, if desired, and is only available in the variable data rate mode.

Precision Voltage References
No external components are required with the combochip to provide the voltage reference function. Voltage references are generated on-chip and are calibrated during the manufacturing process. The technique uses a difference in sub-surface charge density between two suitably implanted MOS devices to derive a temperature and bias stable reference voltage. These references determine the gain and dynamic range characteristics of the device.

Separate references are supplied to the transmit and receive sections and each is trimmed independently during the manufacturing process. The reference value is then further trimmed in the gain setting op-amps to a final precision value. With this method the combochip can achieve the extremely accurate Digital Milliwatt Responses specified in the transmission parameters, providing the user a significant margin for error in other board components.

TRANSMIT OPERATION

Transmit Filter
The input section provides gain adjustment in the passband by means of an on-chip operational amplifier. This operational amplifier has a common mode range of ±2.17 volts, a maximum DC offset of 25 mV, a minimum open loop voltage gain of 5000, and a unity gain bandwidth of typically 1 MHz. Gain of up to 20 dB can be set without degrading the performance of the filter. The load impedance to ground (GRDA) at the amplifier output (GS_X) must be greater than 10 kilohms in parallel with less than 50 pF. The input signal on lead VF_X can be either AC or DC coupled. The input op amp can only be used in the inverting mode as shown in Figure 3.

A low pass anti-aliasing section is included on-chip. This section typically provides 35 dB attenuation at the sampling frequency. No external components are required to provide the necessary anti-aliasing function for the switched capacitor section of the transmit filter.

![Figure 3. Transmit Filter Gain Adjustment](image)
The passband section provides flatness and stopband attenuation which fulfills the AT&T D3/D4 channel bank transmission specification and CCITT recommendation G.714. The 2916 and 2917 specifications meet or exceed digital class 5 central office switching systems requirements. The transmit filter transfer characteristics and specifications will be within the limits shown in Figure 4.

A high pass section configuration was chosen to reject low frequency noise from 50 and 60 Hz power lines, 17 Hz European electric railroads, ringing frequencies and their harmonics, and other low frequency noise. Even though there is high rejection at these frequencies, the sharpness of the band edge gives low attenuation at 200 Hz. This feature allows the use of low-cost transformer hybrids without external components.

### Encoding

The encoder internally samples the output of the transmit filter and holds each sample on an internal sample and hold capacitor. The encoder then performs an analog to digital conversion on a switched capacitor array. Digital data representing the sample is transmitted on the first eight data clock bits of the next frame.

An on-chip autozero circuit corrects for DC-offset on the input signal to the encoder. This autozero circuit uses the sign bit averaging technique; the sign bit from the encoder output is long term averaged and subtracted from the input to the encoder. In this way, all DC offset is removed from the encoder input waveform.

### RECEIVE OPERATION

#### Decoding

The PCM word at the D_R lead is serially fetched on the first eight data clock bits of the frame. A D/A conversion is performed on the digital word and the corresponding analog sample is held on an internal sample and hold capacitor. This sample is then transferred to the receive filter.

#### Receive Filter

The receive filter provides passband flatness and stopband rejection which fulfills both the AT&T D3/D4 specification and CCITT recommendation G.714. The filter contains the required compensation for the (sin x)/x response of such decoders. The receive filter characteristics and specifications will be within the limits shown in Figure 5.

#### Receive Output Power Amplifiers

A balanced output amplifier is provided in order to allow maximum flexibility in output configuration. Either of the two outputs can be used single ended (referenced to GRDA) to drive single ended loads. Alternatively, the differential output will drive a bridged load directly. The output stage is capable of driving loads as low as 300 ohms single ended or 600 ohms differentially.

Transmission levels are specified relative to the receive channel output under digital milliWatt conditions, that is, when the digital input at D_R is the eight-code sequence specified in CCITT recommendation G.711.

### Table 4. Zero Transmission Level Points

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Parameter Description</th>
<th>Value</th>
<th>Units</th>
<th>Test Conditions</th>
</tr>
</thead>
<tbody>
<tr>
<td>0TLP1x</td>
<td>Zero Transmission Level Point</td>
<td>+2.76</td>
<td>dBm</td>
<td>Referenced to 600Ω</td>
</tr>
<tr>
<td></td>
<td>Transmit Channel (0dBm0) µ-law</td>
<td>+1.00</td>
<td>dBm</td>
<td>Referenced to 900Ω</td>
</tr>
<tr>
<td>0TLP2x</td>
<td>Zero Transmission Level Point</td>
<td>+2.79</td>
<td>dBm</td>
<td>Referenced to 600Ω</td>
</tr>
<tr>
<td></td>
<td>Transmit Channel (0dBm0) A-law</td>
<td>+1.03</td>
<td>dBm</td>
<td>Referenced to 900Ω</td>
</tr>
<tr>
<td>0TLP1r</td>
<td>Zero Transmission Level Point</td>
<td>+5.76</td>
<td>dBm</td>
<td>Referenced to 600Ω</td>
</tr>
<tr>
<td></td>
<td>Receive Channel (0dBm0) µ-law</td>
<td>+4.00</td>
<td>dBm</td>
<td>Referenced to 900Ω</td>
</tr>
<tr>
<td>0TLP2r</td>
<td>Zero Transmission Level Point</td>
<td>+5.79</td>
<td>dBm</td>
<td>Referenced to 600Ω</td>
</tr>
<tr>
<td></td>
<td>Receive Channel (0dBm0) A-law</td>
<td>+4.03</td>
<td>dBm</td>
<td>Referenced to 900Ω</td>
</tr>
</tbody>
</table>
ABSOLUTE MAXIMUM RATINGS

*Notice: Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions above those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

D.C. CHARACTERISTICS

(DA = 0°C to 70°C, VCC = +5V ±5%, VBB = -5V ±5%, GRDA = 0V, GRDD = 0V, unless otherwise specified)

Typical values are for TA = 25°C and nominal power supply values.

**DIGITAL INTERFACE**

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Parameter</th>
<th>Min</th>
<th>Typ</th>
<th>Max</th>
<th>Unit</th>
<th>Test Conditions</th>
</tr>
</thead>
<tbody>
<tr>
<td>IIL</td>
<td>Low Level Input Current</td>
<td>10</td>
<td>μA</td>
<td></td>
<td>GRDD ≤ VIN ≤ VIL (Note 1)</td>
<td></td>
</tr>
<tr>
<td>IIH</td>
<td>High Level Input Current</td>
<td>10</td>
<td>μA</td>
<td></td>
<td>VIH ≤ VIN ≤ VCC</td>
<td></td>
</tr>
<tr>
<td>VIL</td>
<td>Input Low Voltage</td>
<td>0.8</td>
<td>V</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>VIH</td>
<td>Input High Voltage</td>
<td>2.0</td>
<td>V</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>VOL</td>
<td>Output Low Voltage</td>
<td>0.4</td>
<td>V</td>
<td></td>
<td>IOL = 3.2 mA at DX, TX</td>
<td></td>
</tr>
<tr>
<td>VOH</td>
<td>Output High Voltage</td>
<td>2.4</td>
<td>V</td>
<td></td>
<td>IOH = 9.6 mA at DX</td>
<td></td>
</tr>
<tr>
<td>COX</td>
<td>Digital Output Capacitance(2)</td>
<td>5</td>
<td>pF</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CIN</td>
<td>Digital Input Capacitance</td>
<td>5</td>
<td>10</td>
<td>pF</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**POWER DISSIPATION**

All measurements made at fDCLK = 2.048 MHz, outputs unloaded.

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Parameter</th>
<th>Min</th>
<th>Typ</th>
<th>Max</th>
<th>Unit</th>
<th>Test Conditions</th>
</tr>
</thead>
<tbody>
<tr>
<td>ICC1</td>
<td>VCC Operating Current(4)</td>
<td>14</td>
<td>19</td>
<td>mA</td>
<td></td>
<td></td>
</tr>
<tr>
<td>IBB1</td>
<td>VBB Operating Current</td>
<td>-18</td>
<td>-24</td>
<td>mA</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ICC0</td>
<td>VCC Power Down Current</td>
<td>0.5</td>
<td>1.0</td>
<td>mA</td>
<td>PDN ≤ VIL; after 10 μs</td>
<td></td>
</tr>
<tr>
<td>IBB0</td>
<td>VBB Power Down Current</td>
<td>-0.5</td>
<td>-1.0</td>
<td>mA</td>
<td>PDN ≤ VIL; after 10 μs</td>
<td></td>
</tr>
<tr>
<td>ICCS</td>
<td>VCC Standby Current</td>
<td>1.2</td>
<td>2.4</td>
<td>mA</td>
<td>FSX, FSR ≤ VIL; after 300 ms</td>
<td></td>
</tr>
<tr>
<td>IBBS</td>
<td>VBB Standby Current</td>
<td>-1.2</td>
<td>-2.4</td>
<td>mA</td>
<td>FSX, FSR ≤ VIL; after 300 ms</td>
<td></td>
</tr>
<tr>
<td>PD1</td>
<td>Operating Power Dissipation(3)</td>
<td>140</td>
<td>200</td>
<td>mW</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PD0</td>
<td>Power Down Dissipation(3)</td>
<td>5</td>
<td>10</td>
<td>mW</td>
<td>PDN ≤ VIL; after 10 μs</td>
<td></td>
</tr>
<tr>
<td>PST</td>
<td>Standby Power Dissipation(3)</td>
<td>12</td>
<td>25</td>
<td>mW</td>
<td>FSX, FSR ≤ VIL</td>
<td></td>
</tr>
</tbody>
</table>

**NOTES:**

1. VIN is the voltage on any digital pin.
2. Timing parameters are guaranteed based on a 100 pF load capacitance. Up to eight digital outputs may be connected to a common PCM highway without buffering, assuming a board capacitance of 60 pF.
3. With nominal power supply values.
4. VCC applied last or simultaneously with VBB.
### ANALOG INTERFACE, TRANSMIT CHANNEL INPUT STAGE

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Parameter</th>
<th>Min</th>
<th>Typ</th>
<th>Max</th>
<th>Unit</th>
<th>Test Conditions</th>
</tr>
</thead>
<tbody>
<tr>
<td>I_{BX1}</td>
<td>Input Leakage Current, ( V_{FX1}^- )</td>
<td>100</td>
<td>nA</td>
<td></td>
<td></td>
<td>(-2.17V \leq V_{IN} \leq 2.17V)</td>
</tr>
<tr>
<td>R_{IX1}</td>
<td>Input Resistance, ( V_{FX1}^- )</td>
<td>10</td>
<td>(\Omega)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>V_{OX11}</td>
<td>Input Offset Voltage, ( V_{FX1}^- )</td>
<td>25</td>
<td>mV</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A_{VOL}</td>
<td>DC Open Loop Voltage Gain, ( G_{Sx} )</td>
<td>5000</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>f_c</td>
<td>Open Loop Unity Gain Bandwidth, ( G_{Sx} )</td>
<td>1</td>
<td>MHz</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C_{IX1}</td>
<td>Load Capacitance, ( G_{Sx} )</td>
<td>50</td>
<td>pF</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>R_{IX1}</td>
<td>Minimum Load Resistance, ( G_{Sx} )</td>
<td>10</td>
<td>K(\Omega)</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### ANALOG INTERFACE, RECEIVE CHANNEL DRIVER AMPLIFIER STATE

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Parameter</th>
<th>Min</th>
<th>Typ</th>
<th>Max</th>
<th>Unit</th>
<th>Test Conditions</th>
</tr>
</thead>
<tbody>
<tr>
<td>R_{ORA}</td>
<td>Output Resistance, ( PWRO^+ , PWRO^- )</td>
<td>1</td>
<td>(\Omega)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>V_{OSRA}</td>
<td>Single-Ended Output DC Offset, ( PWRO^+ , )</td>
<td>75</td>
<td>mV</td>
<td></td>
<td></td>
<td>Relative to GRDA</td>
</tr>
<tr>
<td></td>
<td>( PWRO^- )</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C_{LRA}</td>
<td>Load Capacitance, ( PWRO^+ , PWRO^- )</td>
<td>100</td>
<td>pF</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### A.C. CHARACTERISTICS—TRANSMISSION PARAMETERS

Unless otherwise noted, the analog input is a 0 dBm0, 1020 Hz sine wave.\(^{(1)}\) Input amplifier is set for unity gain, inverting. The digital input is a PCM bit stream generated by passing a 0 dBm0, 1020 Hz sine wave through an ideal encoder. Receive output is measured single ended. All output levels are \((\sin x)/x\) corrected. Typical values are for \(T_A = 25°C\) and nominal power supply values. \((T_A = 0°C \text{ to } +70°C; \ V_{CC} = +5V \pm 5%; \ V_{BB} = -5V \pm 5%; \ GRDA = 0V; \ GRDD = 0V; \text{ unless otherwise specified})\.

### GAIN AND DYNAMIC RANGE

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Parameter</th>
<th>Min</th>
<th>Typ</th>
<th>Max</th>
<th>Units</th>
<th>Test Conditions</th>
</tr>
</thead>
<tbody>
<tr>
<td>E_{mW}</td>
<td>Encoder Milliwatt Response ((Transmit Gain)</td>
<td>-0.18</td>
<td>±0.04</td>
<td>+0.18</td>
<td>(dBm0)</td>
<td>Signal Input of 1.064 Vrms (\mu)-law \ Signal Input of 1.068 Vrms A-law (T_A = 25°C; \ V_{BB} = -5V, \ V_{CC} = +5V)</td>
</tr>
<tr>
<td>E_{mWTS}</td>
<td>Variation with Temperature and Supplies</td>
<td>-0.07</td>
<td>±0.02</td>
<td>+0.07</td>
<td>(dB)</td>
<td>(\pm 5% \text{ Supplies, 0 to 70°C}\ Relative to Nominal Conditions)</td>
</tr>
<tr>
<td>D_{mW}</td>
<td>Digital Milliwatt Response ((Receive Gain)</td>
<td>-0.18</td>
<td>±0.04</td>
<td>+0.18</td>
<td>(dBm0)</td>
<td>Measure Relative to OTLP. Signal Input per CCITT Recommendation %G.711. Output Signal of 1000 Hz. (R_L = \infty) (T_A = 25°C; \ V_{BB} = -5V, \ V_{CC} = +5V)</td>
</tr>
<tr>
<td>D_{mWTS}</td>
<td>Variation with Temperature and Supplies</td>
<td>-0.07</td>
<td>±0.02</td>
<td>+0.07</td>
<td>(dB)</td>
<td>(\pm 5% \text{ Supplies, 0 to 70°C})</td>
</tr>
</tbody>
</table>

**NOTE:**
1. 0 dBm0 is defined as the zero reference point of the channel under test (OTLP). This corresponds to an analog signal input of 1.064 volts rms or an output of 1.503 volts rms (for \(\mu\)-law).
### GAIN TRACKING

Reference Level = −10 dBmO

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Parameter</th>
<th>2916</th>
<th>2917</th>
<th>Unit</th>
<th>Test Conditions</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Min</td>
<td>Max</td>
<td>Min</td>
<td>Max</td>
</tr>
<tr>
<td>GT1X</td>
<td>Transmit Gain Tracking Error Sinusoidal Input; μ-law</td>
<td>±0.25</td>
<td>±0.5</td>
<td>±1.2</td>
<td>dB</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>dB</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>dB</td>
</tr>
<tr>
<td>GT2X</td>
<td>Transmit Gain Tracking Error Sinusoidal Input; A-law</td>
<td>±0.25</td>
<td>±0.5</td>
<td>±1.2</td>
<td>dB</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>dB</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>dB</td>
</tr>
<tr>
<td>GT1R</td>
<td>Receive Gain Tracking Error Sinusoidal Input; μ-law</td>
<td>±0.25</td>
<td>±0.5</td>
<td>±1.2</td>
<td>dB</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>dB</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>dB</td>
</tr>
<tr>
<td></td>
<td>Measured at PWRO+, RL = 300Ω</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GT2R</td>
<td>Receive Gain Tracking Error Sinusoidal Input; A-law</td>
<td>±0.25</td>
<td>±0.5</td>
<td>±1.2</td>
<td>dB</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>dB</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>dB</td>
</tr>
<tr>
<td></td>
<td>Measured at PWRO+, RL = 300Ω</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### NOISE (All receive channel measurements are single ended)

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Parameter</th>
<th>2916</th>
<th>2917</th>
<th>Unit</th>
<th>Test Conditions</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Min</td>
<td>Typ</td>
<td>Max</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Min</td>
<td>Typ</td>
<td>Max</td>
<td></td>
</tr>
<tr>
<td>NXC1</td>
<td>Transmit Noise, C-Message Weighted</td>
<td>15</td>
<td></td>
<td></td>
<td>dBm0O</td>
</tr>
<tr>
<td>NXP</td>
<td>Transmit Noise, Psophometrically Weighted</td>
<td></td>
<td>−75</td>
<td>dBm0p</td>
<td>Unity Gain</td>
</tr>
<tr>
<td>NRC1</td>
<td>Receive Noise, C-Message Weighted: Quiet Code</td>
<td>11</td>
<td></td>
<td></td>
<td>dBm0O</td>
</tr>
<tr>
<td>NRC2</td>
<td>Receive Noise, C-Message Weighted: Sign Bit Toggle</td>
<td>12</td>
<td></td>
<td></td>
<td>dBm0O</td>
</tr>
<tr>
<td>NRP</td>
<td>Receive Noise, Psophometrically Weighted</td>
<td></td>
<td>−79</td>
<td>dBm0p</td>
<td>D_R = Lowest Positive Decode Level</td>
</tr>
<tr>
<td>NSF</td>
<td>Single Frequency Noise End to End Measurement</td>
<td>−50</td>
<td></td>
<td>−50</td>
<td>dBm0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PSRR1</td>
<td>VCC Power Supply Rejection, Transmit Channel</td>
<td>−30</td>
<td>−30</td>
<td>−30</td>
<td>dB</td>
</tr>
<tr>
<td>PSRR2</td>
<td>VBB Power Supply Rejection, Transmit Channel</td>
<td>−30</td>
<td>−30</td>
<td>−30</td>
<td>dB</td>
</tr>
<tr>
<td>PSRR3</td>
<td>VCC Power Supply Rejection, Receive Channel</td>
<td>−25</td>
<td>−25</td>
<td>−25</td>
<td>dB</td>
</tr>
<tr>
<td>PSRR4</td>
<td>VBB Power Supply Rejection, Receive Channel</td>
<td>−25</td>
<td>−25</td>
<td>−25</td>
<td>dB</td>
</tr>
<tr>
<td>CTTR</td>
<td>Crosstalk, Transmit to Receive</td>
<td>−71</td>
<td>−71</td>
<td>−71</td>
<td>dB</td>
</tr>
<tr>
<td>CTRT</td>
<td>Crosstalk, Receive to Transmit</td>
<td>−71</td>
<td>−71</td>
<td>−71</td>
<td>dB</td>
</tr>
<tr>
<td>Symbol</td>
<td>Parameter</td>
<td>Min</td>
<td>Typ</td>
<td>Max</td>
<td>Unit</td>
</tr>
<tr>
<td>--------</td>
<td>-----------</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>------</td>
</tr>
<tr>
<td>SD1X</td>
<td>Transmit Signal to Distortion, μ-Law Sinusoidal Input; CCITT G.714-Method 2 (2916)</td>
<td>36</td>
<td>30</td>
<td>25</td>
<td>dB</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>dB</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>dB</td>
</tr>
<tr>
<td>SD2X</td>
<td>Transmit Signal to Distortion, A-Law Sinusoidal Input; CCITT G.714-Method 2 (2917)</td>
<td>36</td>
<td>30</td>
<td>25</td>
<td>dB</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>dB</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>dB</td>
</tr>
<tr>
<td>SD1R</td>
<td>Receive Signal to Distortion, μ-Law Sinusoidal Input; CCITT G.714-Method 2 (2916)</td>
<td>36</td>
<td>30</td>
<td>25</td>
<td>dB</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>dB</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>dB</td>
</tr>
<tr>
<td>SD2R</td>
<td>Receive Signal to Distortion, A-Law Sinusoidal Input; CCITT G.714-Method 2 (2917)</td>
<td>36</td>
<td>30</td>
<td>25</td>
<td>dB</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>dB</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>dB</td>
</tr>
<tr>
<td>DPX</td>
<td>Transmit Single Frequency Distortion Products (2916)</td>
<td>46</td>
<td>dBm</td>
<td>−46</td>
<td>dBm0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>dBm0</td>
</tr>
<tr>
<td>DRY</td>
<td>Receive Single Frequency Distortion Products (2916)</td>
<td>46</td>
<td>dBm</td>
<td>−46</td>
<td>dBm0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>dBm0</td>
</tr>
<tr>
<td>IMD1</td>
<td>Intermodulation Distortion, End to End Measurement</td>
<td>−35</td>
<td>dB</td>
<td></td>
<td></td>
</tr>
<tr>
<td>IMD2</td>
<td>Intermodulation Distortion, End to End Measurement</td>
<td>−49</td>
<td>dBm</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SOS</td>
<td>Spurious Out of Band Signals, End to End Measurement</td>
<td>−25</td>
<td>dBm</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SIS</td>
<td>Spurious In Band Signals, End to End Measurement</td>
<td>−40</td>
<td>dBm</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DAX</td>
<td>Transmit Absolute Delay</td>
<td>245</td>
<td>μs</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DDX</td>
<td>Transmit Differential Envelope Delay Relative to DAX</td>
<td>170</td>
<td>μs</td>
<td>95</td>
<td>μs</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>45</td>
<td>μs</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>105</td>
<td>μs</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DAR</td>
<td>Receive Absolute Delay</td>
<td>190</td>
<td>μs</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DDR</td>
<td>Receive Differential Envelope Delay Relative to DAR</td>
<td>45</td>
<td>μs</td>
<td>35</td>
<td>μs</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>85</td>
<td>μs</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>110</td>
<td>μs</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
TRANSMIT CHANNEL TRANSFER CHARACTERISTICS
Input amplifier is set for unity gain, inverting.

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Parameter</th>
<th>Min</th>
<th>Typ</th>
<th>Max</th>
<th>Unit</th>
<th>Test Conditions</th>
</tr>
</thead>
<tbody>
<tr>
<td>G_{RX}</td>
<td>Gain Relative to Gain at 1.02 KHz</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0 dBm0 Signal Input at V_{F\text{X}}-</td>
</tr>
<tr>
<td>16.67 Hz</td>
<td></td>
<td>-30</td>
<td></td>
<td></td>
<td>dB</td>
<td></td>
</tr>
<tr>
<td>50 Hz</td>
<td></td>
<td>-25</td>
<td></td>
<td></td>
<td>dB</td>
<td></td>
</tr>
<tr>
<td>60 Hz</td>
<td></td>
<td>-23</td>
<td></td>
<td></td>
<td>dB</td>
<td></td>
</tr>
<tr>
<td>200 Hz</td>
<td></td>
<td>-1.8</td>
<td></td>
<td>-0.125</td>
<td>dB</td>
<td></td>
</tr>
<tr>
<td>300 to 3000 Hz</td>
<td></td>
<td>-0.125</td>
<td>+0.125</td>
<td>dB</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3300 Hz</td>
<td></td>
<td>-0.35</td>
<td></td>
<td>+0.03</td>
<td>dB</td>
<td></td>
</tr>
<tr>
<td>3400 Hz</td>
<td></td>
<td>-0.7</td>
<td></td>
<td>-0.10</td>
<td>dB</td>
<td></td>
</tr>
<tr>
<td>4000 Hz</td>
<td></td>
<td>-14</td>
<td></td>
<td></td>
<td>dB</td>
<td></td>
</tr>
<tr>
<td>4600 Hz and Above</td>
<td></td>
<td>-32</td>
<td></td>
<td></td>
<td>dB</td>
<td></td>
</tr>
</tbody>
</table>
Figure 4. Transmit Channel
## RECEIVE CHANNEL TRANSFER CHARACTERISTICS

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Parameter</th>
<th>Min</th>
<th>Typ</th>
<th>Max</th>
<th>Unit</th>
<th>Test Conditions</th>
</tr>
</thead>
<tbody>
<tr>
<td>$G_{RR}$</td>
<td>Gain Relative to Gain at 1.02 KHz</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0 dBm0 Signal Input at $D_R$</td>
</tr>
<tr>
<td></td>
<td>Below 200 Hz</td>
<td></td>
<td></td>
<td>+0.125</td>
<td>dB</td>
<td></td>
</tr>
<tr>
<td></td>
<td>200 Hz</td>
<td>−0.5</td>
<td></td>
<td>+0.125</td>
<td>dB</td>
<td></td>
</tr>
<tr>
<td></td>
<td>300 to 3000 Hz</td>
<td>+0.125</td>
<td></td>
<td>+0.125</td>
<td>dB</td>
<td></td>
</tr>
<tr>
<td></td>
<td>3300 Hz</td>
<td>−0.35</td>
<td></td>
<td>+0.03</td>
<td>dB</td>
<td></td>
</tr>
<tr>
<td></td>
<td>3400 Hz</td>
<td>−0.7</td>
<td></td>
<td>−0.1</td>
<td>dB</td>
<td></td>
</tr>
<tr>
<td></td>
<td>4000 Hz</td>
<td></td>
<td></td>
<td>−14</td>
<td>dB</td>
<td></td>
</tr>
<tr>
<td></td>
<td>4600 Hz and Above</td>
<td></td>
<td></td>
<td>−30</td>
<td>dB</td>
<td></td>
</tr>
</tbody>
</table>
Figure 5. Receive Channel
# A.C. Characteristics—Timing Parameters

## Clock Section

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Parameter</th>
<th>Min</th>
<th>Typ</th>
<th>Max</th>
<th>Unit</th>
<th>Test Conditions</th>
</tr>
</thead>
<tbody>
<tr>
<td>tCY</td>
<td>Clock Period, CLK</td>
<td>488</td>
<td></td>
<td></td>
<td>ns</td>
<td>fCLK = 2.048 MHz</td>
</tr>
<tr>
<td>tCLK</td>
<td>Clock Pulse Width, CLK</td>
<td>220</td>
<td></td>
<td></td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>tDCLK</td>
<td>Data Clock Pulse Width</td>
<td>220</td>
<td></td>
<td></td>
<td>ns</td>
<td>64 KHz ≤ fDCLK ≤ 2.048 MHz</td>
</tr>
<tr>
<td>tCDC</td>
<td>Clock Duty Cycle, CLK</td>
<td>45</td>
<td>50</td>
<td>55</td>
<td>%</td>
<td></td>
</tr>
<tr>
<td>tr, tf</td>
<td>Clock Rise and Fall Time</td>
<td>5</td>
<td>30</td>
<td></td>
<td>ns</td>
<td></td>
</tr>
</tbody>
</table>

## Transmit Section, Fixed Data Rate Mode

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Parameter</th>
<th>Min</th>
<th>Typ</th>
<th>Max</th>
<th>Unit</th>
<th>Test Conditions</th>
</tr>
</thead>
<tbody>
<tr>
<td>tDZX</td>
<td>Data Enabled on TS Entry</td>
<td>0</td>
<td></td>
<td>145</td>
<td>ns</td>
<td>0 &lt; CLOAD &lt; 100 pF</td>
</tr>
<tr>
<td>tDDX</td>
<td>Data Delay from CLK</td>
<td>0</td>
<td></td>
<td>145</td>
<td>ns</td>
<td>0 &lt; CLOAD &lt; 100 pF</td>
</tr>
<tr>
<td>tHZX</td>
<td>Data Float on TS Exit</td>
<td>60</td>
<td></td>
<td>215</td>
<td>ns</td>
<td>CLOAD = 0</td>
</tr>
<tr>
<td>tSON</td>
<td>Timeslot X to Enable</td>
<td>0</td>
<td></td>
<td>145</td>
<td>ns</td>
<td>0 &lt; CLOAD &lt; 100 pF</td>
</tr>
<tr>
<td>tSOFN</td>
<td>Timeslot X to Disable</td>
<td>60</td>
<td></td>
<td>215</td>
<td>ns</td>
<td>CLOAD = 0</td>
</tr>
<tr>
<td>tFSD</td>
<td>Frame Sync Delay</td>
<td>100</td>
<td></td>
<td></td>
<td>tCLK</td>
<td>ns</td>
</tr>
</tbody>
</table>

## Receive Section, Fixed Data Rate Mode

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Parameter</th>
<th>Min</th>
<th>Typ</th>
<th>Max</th>
<th>Unit</th>
<th>Test Conditions</th>
</tr>
</thead>
<tbody>
<tr>
<td>tDSR</td>
<td>Receive Data Setup</td>
<td>10</td>
<td></td>
<td></td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>tDHR</td>
<td>Receive Data Hold</td>
<td>60</td>
<td></td>
<td></td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>tFSD</td>
<td>Frame Sync Delay</td>
<td>100</td>
<td></td>
<td>tCLK</td>
<td>ns</td>
<td></td>
</tr>
</tbody>
</table>

**Note:**
1. Timing parameters T_{DZX}, T_{HZX}, and T_{SOFN} are referenced to a high impedance state.
WAVEFORMS

Fixed Data Rate Timing

TRANSMIT TIMING

NOTE:
1. All timing parameters referenced to $V_{IH}$ and $V_{IL}$ except $t_{DZ}, t_{SOFF}$ and $t_{HZ}$ which reference a high impedance state.

RECEIVE TIMING

NOTE:
1. All timing parameters referenced to $V_{IH}$ and $V_{IL}$. 
### TRANSMIT SECTION, VARIABLE DATA RATE MODE

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Parameter</th>
<th>Min</th>
<th>Typ</th>
<th>Max</th>
<th>Unit</th>
<th>Test Conditions</th>
</tr>
</thead>
<tbody>
<tr>
<td>tTSOX</td>
<td>Timeslot Delay from DCLKx(2)</td>
<td>140</td>
<td></td>
<td>tDX−140</td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>tFSD</td>
<td>Frame Sync Delay</td>
<td>100</td>
<td></td>
<td>tCY−100</td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>tDDX</td>
<td>Data Delay from DCLKx</td>
<td>0</td>
<td></td>
<td>100</td>
<td>ns</td>
<td>0 &lt; CLOAD &lt; 100 pF</td>
</tr>
<tr>
<td>tDON</td>
<td>Timeslot to Dx Active</td>
<td>0</td>
<td></td>
<td>50</td>
<td>ns</td>
<td>0 &lt; CLOAD &lt; 100 pF</td>
</tr>
<tr>
<td>tDOFF</td>
<td>Timeslot to Dx Inactive</td>
<td>0</td>
<td></td>
<td>80</td>
<td>ns</td>
<td>0 &lt; CLOAD &lt; 100 pF</td>
</tr>
<tr>
<td>tDX</td>
<td>Data Clock Period</td>
<td>488</td>
<td></td>
<td>15620</td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>tDFSX</td>
<td>Data Delay from FSX</td>
<td>0</td>
<td></td>
<td>140</td>
<td>ns</td>
<td></td>
</tr>
</tbody>
</table>

### RECEIVE SECTION, VARIABLE DATA RATE MODE

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Parameter</th>
<th>Min</th>
<th>Typ</th>
<th>Max</th>
<th>Unit</th>
<th>Test Conditions</th>
</tr>
</thead>
<tbody>
<tr>
<td>tTSOR</td>
<td>Timeslot Delay from DCLKR(3)</td>
<td>140</td>
<td></td>
<td>tDR−140</td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>tFSD</td>
<td>Frame Sync Delay</td>
<td>100</td>
<td></td>
<td>tCY−100</td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>tDSR</td>
<td>Data Setup Time</td>
<td>10</td>
<td></td>
<td></td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>tDHR</td>
<td>Data Hold Time</td>
<td>60</td>
<td></td>
<td></td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>tDR</td>
<td>Data Clock Period</td>
<td>488</td>
<td></td>
<td>15620</td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>tSER</td>
<td>Timeslot End Receive Time</td>
<td>60</td>
<td></td>
<td></td>
<td>ns</td>
<td></td>
</tr>
</tbody>
</table>

### 64 KB OPERATION, VARIABLE DATA RATE MODE

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Parameter</th>
<th>Min</th>
<th>Typ</th>
<th>Max</th>
<th>Unit</th>
<th>Test Conditions</th>
</tr>
</thead>
<tbody>
<tr>
<td>tFSLX</td>
<td>Transmit Frame Sync Minimum</td>
<td>488</td>
<td></td>
<td></td>
<td>ns</td>
<td>FSx is TTL High for Remainder of Frame</td>
</tr>
<tr>
<td>tFSLR</td>
<td>Receive Frame Sync Minimum</td>
<td>1952</td>
<td></td>
<td></td>
<td>ns</td>
<td>FSR is TTL High for Remainder of Frame</td>
</tr>
<tr>
<td>tDCLK</td>
<td>Data Clock Pulse Width</td>
<td>10</td>
<td>µs</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**NOTES:**
1. Timing parameters tDON and tDOFF are referenced to a high impedance state.
2. tFSLX minimum requirements overrides tTSOX maximum spec for 64 KHz operation.
3. tFSLR minimum requirements overrides tTSOR maximum spec for 64 KHz operation.
**VARIABLE DATA RATE TIMING**

**TRANSMIT TIMING**

![Transmit Timing Diagram](image_url)

**RECEIVE TIMING**

![Receive Timing Diagram](image_url)

**NOTE:**
1. All timing parameters referenced to \( V_{IH} \) and \( V_{IL} \) except \( t_{DON} \) and \( t_{OFF} \) which reference a high impedance state.

**A.C. TESTING INPUT, OUTPUT WAVEFORM**

**INPUT/OUTPUT**

![Input/Output Waveform](image_url)

A.C. Testing: Inputs are driven at 2.4V for a Logic "1" and 0.45V for a Logic "0". Timing measurements are made at 2.0V for a Logic "1" and 0.8V for a Logic "0".

6-84
CODEC INTERFACE
The 2912A PCM Filter is designed to directly interface to the 2910A and 2911A Codecs as shown below. The transmit path is completed by connecting the VFxO output of the 2912A to the coupling capacitor associated with the VFx input of the 2910A and 2911A codecs. The receive path is completed by directly connecting the codec output VF_R to receive input of the 2912A VF_R1. The PDN input of the 2912A should be connected to the PDN output of the codec to allow the filter to be put in the power-down standby mode under control of the codec.

CLOCK INTERFACE
To assure proper operation, the CLK input of the 2912A should be connected to the same clock provided to receive bit clock, CLK_R of 2910A or 2911A Codec as shown below. The CLK0 input of the 2912A should be set to the proper voltage depending on the standard clock frequency chosen for the codec and filter.

Figure 1. A Typical 2910A Codec and 2912A Filter Configuration
GROUNDING, DECOUPLING, AND LAYOUT RECOMMENDATIONS

The most important steps in designing a low noise line card are to insure that the layout of the circuit components and traces results in a minimum of cross coupling between analog and digital signals, and to provide well bypassed and clean power supplies, solid ground planes, and minimal lead lengths between components.

1) All power source leads should be bypassed to ground on each printed circuit board (PCB), on which codecs are provided. At least one electrolytic bypass capacitor (at least 50 μF) per board is recommended at the point where all power traces from the codecs and filters join prior to interfacing with the edge connector pins assigned to the power leads.

2) When using two-sided PCBs, use both corresponding pins on opposite sides of the board for the same power lead. Strap them together both on the PCB and on the back of the edge connector.

3) Lay out the traces on codec- and filter-equipped boards such that analog signal and capacitor leads are separated as widely as possible from the digital clock and data leads.

4) Connect the codec sample and hold capacitor with the shortest leads possible. Mount it as close to the codec CAP1X, CAP2X pins as possible. Shield the capacitor traces with analog ground.

5) Do not layout any board traces (especially digital) that pass between or near the leads of the sample and hold capacitor(s) since they are in high impedance circuits which are sensitive to noise coupling.

6) Keep analog voice circuit leads paired on their layouts so that no intervening circuit leads are permitted to run parallel to them and/or between them.

7) Arrange the layout for each duplicated line, trunk or channel circuit in identical form.

8) Line circuits mounted extremely close to adjacent line circuits increase the possibility of inter-channel crosstalk.

9) Avoid assignment of edge connector pins to any analog signal adjacent to any lead carrying digital (periodic) signals or power.

10) The optimum grounding configuration is to maintain separate digital and analog grounds on the circuit boards, and to carry these grounds back to the power supply with a low impedance connection. This keeps the grounds separate over the entire system except at the power supply.

11) The voltage difference between ground leads GRDA and GRDD (analog and digital ground) should not exceed two volts. One method of preventing any substantial voltage difference between leads GRDA and GRDD is to connect two diodes back to back in opposite directions across these two ground leads on each board.

12) Codec-filter pairs should be aligned so that pins 9 through 16 of the filter face pins 1 through 12 of the codec. This minimizes the distance for analog connections between devices and with no crossing analog lines.

13) No digital or high voltage level (such as ringing supply) lines should run under or in parallel with these analog VF connections. If the analog lines are on the top (component side) of the PCB board, then GRDD, GRDA, or power supply leads should be directly under them, on the bottom to prevent analog/digital coupling.

14) Both the codec and filter devices should be shielded from traces on the bottom of the PCB by using ground or power supply leads on the top side directly under the device (like a ground plane).

15) Two +5V power supply leads (VCC) should be used on each PCB, one to the filters, the other to the codecs. These leads should be separately decoupled at the PCB where they then join to a single 5V supply at the backplane connector. Decoupling can be accomplished with either a series resistor/parallel capacitor (RC lowpass) or a series RF choke and parallel capacitor of each 5V lead. The capacitor should be at least 10 μF in parallel with a 0.1 μF ceramic. This filters both high and low frequencies and accommodates large current spikes due to switching.

16) Both grounds and power supply leads must have low resistance and inductance. This should be accomplished by using a ground plane whenever possible. When narrower traces must be used, a minimum width of 4 millimeters should be maintained. Either multiple or extra large plated through holes should be used when passing the ground connections through the PCB.
17) The 2912A PCM filter should have all power supplies bypassed to analog ground (GRDA). The 2910A/2911A Codec +5V power supplies should be bypassed to the digital ground (GRDD). This is appropriate when separate +5V power supply leads are used as suggested in item 15. The −5V and +12V supplies should be bypassed to analog ground (GRDA). Bypass capacitors at each device should be high frequency capacitors of approximately 0.1 to 1.0 μF value. Their lead lengths should be minimized by routing the capacitor leads to the appropriate ground plane under the device (either GRDA or GRDD).

18) Relay operation, ring voltage application, interruptions, and loop current surges can produce enormous transients. Leads carrying such signals must be routed well away from both analog and digital circuits on the line card and in backplanes. Lead pairs carrying current surges should be routed closely together to minimize possible inductive coupling. The microcomputer clock lead is particularly vulnerable, and should be buffered. Care should also be used in the backplane layout to prevent pickup surges. Any other latching components (relay buffers, etc.) should also be protected from surges.

19) When not used, the AUTO pin should float with minimum PC board track area.

---

**ZERO TRANSMISSION LEVEL POINTS**

### 2910A/2912A 0 dBm0

![Diagram of 2910A/2912A 0 dBm0](image)

- **TRANSMIT FILTER**
  - 2.85 dBm
  - 1.06 Vrms

- **ENCODER**
  - A/D

- **DECODER**
  - D/A

- **RECEIVE FILTER**
  - 5.85 dBm
  - 1.52 Vrms

- **POWER AMPLIFIERS**
  - SINGLE ENDED, 600 Ω
  - 5.85 dBm
  - 1.52 Vrms
  - BALANCED, 600 Ω
  - 11.5 dBm
  - 3.05 Vrms

### 2911A/2912A 0 dBm0

![Diagram of 2911A/2912A 0 dBm0](image)

- **TRANSMIT FILTER**
  - 2.88 dBm
  - 1.06 Vrms

- **ENCODER**
  - A/D

- **DECODER**
  - D/A

- **RECEIVE FILTER**
  - 5.88 dBm
  - 1.52 Vrms

- **POWER AMPLIFIERS**
  - SINGLE ENDED, 600 Ω
  - 5.85 dBm
  - 1.52 Vrms
  - BALANCED, 600 Ω
  - 11.8 dBm
  - 3.05 Vrms

---

6-87
Designing Second-Generation Digital Telephony Systems Using the Intel 2913/14 Codec/Filter Combochip

ROBERT E. HOLM
TELECOM TECHNICAL SUPPORT

JOHN HUGGINS
TELECOM DESIGN ENGINEERING
1.0 INTRODUCTION

This application note describes the features and capabilities of the 2913 and 2914 codec/filter combochips, and relates these capabilities to the design and manufacturing of transmission and switching linecards.

1.1 Background

The first generation of per line codecs (Intel 2910A/11A) and filters (Intel 2912A) economically integrated the analog-digital conversion circuits and PCM formatting circuits into one chip and the filtering and gain setting circuits into another chip. These two chips helped to make possible the rapid conversion to digital switching systems that has taken place in the last few years.

The second generation of Intel LSI PCM telephony components, the 2913/14 Combochip, extends the level of integration of the linecard by combining the codec and filter functions for each line on a single LSI chip. In the process of combining both functions, circuit design improvements have also improved performance, reduced external component count, lowered power dissipation, increased reliability, added new features, and maintained architectural transparency.

The 2913 and 2914 data sheet contains a complete description of both parts, including detailed discussions of each feature and specifications for timing and performance levels. This application note, in conjunction with the data sheet, describes in more detail how the new and improved features help in the design of second-generation linecards first by comparing the two generations of components to see where the improvements have been made, and then by discussing specific design considerations.

1.2 Comparison of First- and Second-Generation Component Capabilities

The combochip represents a higher level of component integration than the devices it replaces and, because of the economics of LSI (replacing two chips with one), ultimately will cost significantly less at the component level. But comparison of the combochip block diagram with first-generation single-chip codec and filter reveals few major functional differences. Figure 1 compares the first-generation codec and filter chips to the combochip. Both provide rigidly specified PCM capabilities of voice signal bandlimiting and nonlinear companded A/D and D/A conversion. The first on-chip reference voltage was introduced in the 2910/2911 single-chip codecs and is included in the combochip. The provision of uncommitted buffer amplifiers for flexible transmission level adjustment and enhanced analog output drive was a feature of the now standard 2912 switched-capacitor PCM filter is available on the combochip. Like-

![Figure 1. LSI Partitioning of Codec/Filter Functions](image-url)
wise, independent transmit (A/D) and receive (D/A) analog voice channels which permit the two channels to be timed from independent (asynchronous) clock sources is common to the first- and second-generation devices. Finally, the ability to multiplex signalling bits on a bit-stealing basis from the digital side of the device has been duplicated on the combochip.

Data traffic-conscious systems manufacturers now provide dedicated codec, filter, and subscriber interface functions on a per-subscriber basis, which in turn puts intense cost pressures on these functions. The functional duplication of first-generation components addresses the needs of the system manufacturer who wants to cost reduce existing fixed-architecture system designs. Whereas the bulk of the system development costs (and time) are in the switching machine call processing and diagnostic software, the bulk of the production costs are in the high-volume linecards. The combochip addresses these cost pressures and defers the appetite for new integrated functions to a future generation of PCM components.

Figure 2 contains the block diagram of the 2913/14 combochip which illustrates not only the basic companding and filtering functions but also some of the changes and new features contained in the second-generation devices, such as internal auto zero, separate ADC and DAC for transmit and receive sections, respectively, precision gain setting (RCV section), and input/output registers for both fixed and variable data rates. Table 1 lists many of the features that are important to linecard design and performance. A direct comparison between first- and second-generation products

| Table 1. Comparison between 2913/14 Combochip and the 2910A/11A/12A Single-Chip Codecs and Filters |
|--------------------------------------------------------|-------------------------------|------------------|
| Features                                              | 2910A/11A plus 2912A               | 2913/14          |
| Power                                                  | Operating 280–310 mW            | 140 mW          |
|                                                       | Standby 33 mW                    | 5 mW            |
| Pins                                                   | 38–40                           | 20–24           |
| Board Area Including Interconnects                    | Normalized = 1.0                | 0.33            |
| Data Rates                                             | —Fixed 1.536, 1.544, 2.048 Mbps  | Same            |
|                                                       | —Variable None                   | 64 Kbps → 2.048 Mbps |
| Companding Law                                         | —μ-Law 2910 + 2912              | Strap Selectable|
|                                                       | —A-Law 2911 + 2912              |                 |
| PSRR                                                   | 1 KHz 30 dB                     | > 35 dB         |
|                                                       | > 10 KHz Not Spec’d             | > 35 dB         |
| Gain Setting                                           | Trim Using Pot Necessary        | Precision Resistors |
|                                                       | Eliminate Trim Req.             |                 |
| Operating Modes                                        | Direct Yes                       | Yes             |
|                                                       | Timeslot Assign Yes              | No              |
| On-Chip V_REF                                         | Yes                             | Yes             |
| ICN — Half Channel Improvement                        | 15 dBmcc0 Transmit 11 dBmcc0 Receive | 15 dBmcc0 Transmit 11 dBmcc0 Receive |
| S/D — Half Channel Improvement                        | See Data Sheet                  | See Section 2.0 |
| GT — Half Channel Improvement                         | See Data Sheet                  | See Section 2.0 |
| Power Down (Standby)                                  | PDN Pin                         | Frame Sync Removal or PDN Pin |
| Signalling                                             | 2910-8th Bit                    | 2914-8th Bit    |
| Auto Zero                                              | External                        | Internal        |
| S & H Caps                                             | External Transmit Internal Receive |                |
| Test Modes                                             | None                            | Design Tests   |
|                                                       |                                | Manufacturing Test |
|                                                       |                                | On-Line Operational Tests |
| Encoder Implementation                                 | Resistive Ladder                | Capacitive Charge Redistribution |
|                                                       |                                | Ladder          |
| Filter/Gain Trim                                       | Fuse Blowing ±0.2 dB            | Fuse Blowing ±0.04 dB |
(a) Combochip Block Diagram

- **VBB**: Power (-5V)
- **PWRO+, PWRO-**: Power Amplifier Outputs
- **GSx**: Transmit Gain Control
- **VFxI-, VFxI+**: Analog Inputs
- **GSR**: Receive Gain Control
- **PDN**: No Connect
- **GRDA**: Analog Ground
- **PDN**: NC
- **CLKSEL**: Select
- **LOOP**: Analog Loop Back
- **ASEL**: A-law or µ-law Select
- **SIGNAL**: Receive Signaling Bit Output
- **TSx**: Timeslot Strobe/Buffer Enable
- **DCLKR**: Receive Variable Data Clock
- **DCLKX**: Transmit Variable Data Clock
- **DR**: Receive PCM Input
- **DX**: Transmit PCM Output
- **FSR**: Receive Frame
- **FSX**: Transmit Frame
- **GRDD**: Synchronization Clock
- **CLKX**: Synchronization Clock
- **VCC**: Power (+5V)
- **CLKR**: Transmit Master Clock

(b) Combochip Pin Names

Figure 2. Block Diagram of 2913/14 Combochip
shows the significant improvement in the combochip both in performance levels and system flexibility.

2.0 DESIGN CONSIDERATIONS

The key point with the 2913/14 is that it will result in a linecard that performs better and costs less than any two-chip codec/filter solution. The lower cost results from many factors, as seen in Table 2. Both direct replacement costs and less tangible design and manufacturing time savings combine to yield lower recurring and nonrecurring costs. As an example, the wider margins to transmission specs and the higher power supply rejection ratios of the 2913/14 will both shorten the design time needed to build and test the linecard prototype and reduce the reject rate on the manufacturing line.

Table 2. 2913/14 Factors which Lower the Cost of Linecard Design and Manufacturing

| Lower LSI Cost (2914 vs. 2910/11 + 2912) |
| Fewer External Components |
| Less Board Area |
| Shorter Design/Prototype Cycle |
| Better Yields/Higher Reliability |
| Lower Power/Higher Density |

Part of the recurring cost of linecard production is the efficiency of the manufacturing line in turning out each board. This is measured in both parts cost and time. Average manufacturing time is strongly affected by the line yield, i.e., the reject rate reliability. A linecard using the 2913/14 has many labor-saving features, which also increases the reliability of the manufacturing process. Some of these features are detailed in Table 3.

The combination of fewer parameters to trim (gain, reference voltage, etc.), tolerance to wider power supply variations, and on-chip test modes make the linecard very manufacturable compared to first-generation designs.

Probably the most obvious improvement in linecard design based around the 2913/14 is the reduction in linecard PCB area needed compared to two-chip designs. The combination of the codec and filter into a single package alone reduced the LSI area by one-third. Table 4 shows many of the other ways in which board area is conserved. In general, it reduces to fewer components, more on-chip features, and layout of the chip resulting in an efficient board layout which neatly separates the analog and digital signals both inside the chip and on the board.

Table 3. 2914 Factors which Increase Linecard Manufacturing Yields and Efficiency

| Higher Reliability |
| Fewer connections and components |
| More integrated packaging |
| More margin to specs |
| Lower power |
| NMOS proven process |
| Less sensitive to parameter variations |

| Fewer Manufacturing Steps |
| No gain trimming |
| On chip V_REF |
| Wide power supply tolerance |
| On-chip test modes |
| Wide margins to spec |

Table 4. Design Factors for 2914 which Reduce Linecard PCB Area

| Integrated Packaging |
| 2914 vs. 2910/11 + 2912 = 1/3 board area |
| 2913 takes even less space |

| Fewer Interconnects/Components |
| Codec/filter combined |
| On-chip reference voltage |
| On-chip auto zero |
| On-chip capacitors |
| No gain trim components |
| No voltage regulators |

| Efficient Layout (Facilitates Auto Insertion) |
| Analog/digital sections separated on chip |
| Digital traces can cross under chip |
| Two power supplies only |
| Low power/high density |
Many of the factors discussed—which result in efficient, cost-effective linecard designs—are discussed in more detail both in the 2913/14 data sheet and in the following sections of this note.

### 2.1 Operating and Test Mode Selection

A key to designing with the 2913/14 combo is the wide range of options available in configuring, either with strap options or in real time, the different modes of operation. The 2913 combochip (20 pins) is specifically aimed at synchronous switching systems (remote concentrators, PABXs, central offices) where small package size is especially desirable. The 2914 combochip (24 pins) has additional features which are most suitable for applications requiring 8th-bit signalling, asynchronous operation, and remote testing of transmission paths (e.g., channel banks). Once the specific device is selected, there is a wide range of operating modes to use in the card design, as seen in Table 5. This table lists the optional parameters and the pins which control the operating mode. The result of selecting a mode is listed for both the 2913 and 2914.

The purpose of offering these options is to ensure that the 2913/14 combo will accommodate any existing linecard design with architectural transparency. At the same time, features were designed in to facilitate design and manufacturing testing to reduce overall cost of development and production.

### 2.2 Data Rate Modes

Any rapid conversion scenario presumes that the combochip will fit existing system architectures (retrofit) without significant system timing, control, or software modifications. To this end, two distinct user-selectable timing modes are possible with the combochip. For purposes of discussion, these are designated (a) fixed data rate timing (FDRT) and (b) variable data rate timing (VDRT).

FDRT is identical to the 2910/2911 codec timing in which a single high-speed clock serves both as master clock for the codec/filter internal conversion/filtering functions and as PCM bit clock for the high-speed serial PCM data bus over which the combochip transmits and receives its digitized voice code words. In this mode, PCM bit rates are necessarily confined to one of three distinct frequencies (1.536 MHz, 1.544 MHz, or 2.048 MHz). Many recently designed systems employ this type of timing which is sometimes referred to as burst-mode timing because of the low duty cycle of each timeslot (i.e., channel) on the time division multiplexed PCM bus. It is possible for up to 32 active combochips to share the same serial PCM bus with FDRT.

VDRT (sometimes referred to as shift register timing), by comparison, utilizes one high-speed master clock for the combochip internal conversion/filtering functions and a separate, variable frequency, clock as the PCM bit clock for the serial PCM data bus. Because the serial PCM data rate is independent of internal conversion timing, there is considerable flexibility in the choice of PCM data rate. In this mode the master clock is permitted to be 1.536 MHz, 1.544 MHz, or 2.048 MHz, while the bit clock can be any rate between 64 KHz and 2048 MHz. In this mode it is possible to have a dedicated serial bus for each combochip or to share a single serial PCM bus among as many as 32 active combochips.
Thus, the two predominant timing configurations of present system architectures are served by the same device, allowing, in many cases, linecard redesign without modification of any common system hardware or software. Additional details relating to the design of systems using either mode are found in section 3.0.

2.3 Margin to Performance Specifications

The combochip benefits from design, manufacturing, and test experience with first-generation PCM products on the part of the system manufacturer, component suppliers, and test equipment suppliers. The sub-millivolt PCM measurement levels and tens of microvolts accuracy requirements on the lowest signal measurements often result in tester correlation problems, yield losses, and excess costs for system and PCM component manufacturers alike. Thus additional performance margin built into the PCM components themselves will have its effect on line circuit costs even though the system transmission specifications may not reflect the improved performance margin.

Half channel measurements have been made of the transmission parameters—gain tracking (GT), signal to distortion ratio (S/D), and idle channel noise (ICN).

Gain Tracking—Figure 3 shows the gain tracking data for both the transmit and receive sides of the combo using both sine wave testing (CCITT G712.11 Method 2) and white noise testing (CCITT G712.11 Method 1). The data shows a performance very nearly equal to the theoretically best achievable using both test techniques. End to end measurements, although not spec'd, also show a corresponding good performance with errors less than or equal to the sum of the half channel values.

Signal to Distortion Ratio—This is a measure of the system linearity and the accuracy in implementing the companding codes. Figure 4 shows the excellent perfor-
mance of the 2914 for both the transmit (A/D) and receive (D/A) channels using sine wave and noise testing. The margin is greater than 3 dB above the half channel spec which means that a larger error budget is available to the rest of the channel.

**Statistical Analysis**—A statistical analysis of G.T. and S/D measurements over many devices shows a very tight distribution, as seen in Figure 5. There are several consequences resulting from this highly desirable distribution: (1) the device performance is controllable, resulting in high yields, (2) the device circuit design is tolerant of normal process variations, thereby ensuring predictable production yields and high reliability, and (3) understanding of the circuit design and process fundamentals is clearly demonstrated—largely as a result of previous telephony experience with the Intel NMOS process.

**Idle Channel Noise**—The third transmission parameter is idle channel noise (ICN). Figure 6 gives half channel ICN measurements which show a substantial margin to specification.

**Power Supply Rejection**—Circuit innovation in the internal combochip design has resulted in significant improvements in power supply rejection in the 5 to 50 KHz range (Figure 7), and it is this frequency band which usually contains the bulk of the switching regulator noise. These higher frequencies, outside the audio range as they are, are not objectionable or even detectable in the transmit direction except to the extent that they alias into the audio range as a result of internal sampling processes in the transmit filter and A/D converter. Sampling techniques in the combochip minimize this aliasing. In the receive direction, excess high frequency noise which propagates onto the subscriber loop can interfere with signals in adjacent wires and is thus objectionable even without aliasing. The symmetrical true differential analog outputs of the combochip are an improvement from earlier designs which failed to maintain true power supply symmetry through the output amplifiers. Not only does the differential design improve transmission performance, but it also reduces the need for power supply bypass capacitors, thereby saving component cost on the linecard.
2.4 Power Conservation

Figure 8 illustrates typical power consumption and office equipment dissipation for a resistive line biasing arrangement (with no loop current limiting) and for the per-line PCM components. It can be seen that overall line circuit power consumption and dissipation are strong functions of subscriber loop resistance, and are dominated by line biasing current regardless of loop length. It can also be seen that the combochip achieves significant reductions in PCM component contributions relative to both the 2910A/2912A and 2910/2912. Present residential traffic characteristics are such that the PCM components are active less than 10% of the time, and in its low-power standby state, the combochip power dissipation drops to typically 5 mW as the line current (and dissipation) goes to its background on-hook leakage level of typically a few milliwatts (but for very leaky lines, as much as 50 mW - 500 mW).

The concern for linecard power consumption and dissipation is related both to the cost of providing power and to the system density problem involving convection heat removal from the linecards. Consequently, much recent line circuit development activity centers on elimination of the inefficient resistive line current feed both by current limiting in short loops and by more exotic, and expensive per-line dc-dc converters. For both present-generation designs and cost-reduction redesigns, the typical combochip dissipation of 140 mW active/5 mW standby will allow system board packing density improvements and power supply cost reductions.

A closer look at the effect of loading (duty cycle) on the average power dissipation of a combochip is given in Table 6. Typical loading percents run as low as 5% for very large switching systems (thousands of lines) up to 100% in nonswitching applications such as channel banks. Clearly, the average power dissipation in a typical switching system is below 35 mW which facilitates board packing density and cost of power considerations.

<table>
<thead>
<tr>
<th>Duty Cycle</th>
<th>Power Dissipation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Central Office</td>
<td>5%</td>
</tr>
<tr>
<td>PABX</td>
<td>15%</td>
</tr>
<tr>
<td>Peak Hour C.O.</td>
<td>50%</td>
</tr>
<tr>
<td>Channel Bank</td>
<td>100%</td>
</tr>
</tbody>
</table>

Table 6. Typical Power Dissipation Per Line Using 2914 Combochip
Figure 7. Wideband 2914 Power Supply Rejection Ratio (PSRR)

Figure 8. Line Circuit Power Consumption and Dissipation Curves
2.5 Elimination of Gain Trim in the Line Circuit

Four resistors—R1–R4 of Figure 9—on the transformer side of the PCM components are used to establish appropriate transmission levels at the PCM components and are, at first glance, equivalent in the two cases. However, a significant reduction in linecard manufacturing costs associated with individual line trim (or mop-up) is possible with the combochip. The need for this trim is dictated by system gain contrast specifications which typically require that the line-to-line gain variation shall not exceed 0.5 dB, which translates to 0.25 dB for each (transmit and receive) channel. Table 7 shows that the major portion of this gain variation has previously been in the nominal insertion loss of the PCM filter and in the uncertainty of the reference voltage of the codec. With this cumulative 0.15 dB uncertainty in the PCM components themselves, the system manufacturer had no choice but to resort to the cost and manufacturing complexity of the active trim. The combochip, however, can be trimmed during its manufacture to a nominal tolerance of ± 0.04 dB which includes uncertainties in both the filter and codec voltage reference functions. This leaves 0.21 dB uncertainty to variations in the other line circuit elements and to temperature and supply variations.

The variation in combochip gain with supply and temperature has also been improved to allow as low as
There are no requirements section 2.2 of this note and in the 2913/14 data sheet. The FORT should happen, both Dx and fail-safe mode in able data rate ed by the of the There

In addition, Intel Application Note

3.2 Variable Data Rate Mode

The VDRT mode is described in some detail both in section 2.2 and in the 2913/14 data sheet. This section focuses on two design aspects: (1) the advantage of clocking data on the rising edges of the clock for transmit and receive data, respectively, and (2) making the 2913/14 transparent in previously designed systems (a retrofit, cost reduction redesign).

Clock Timing—The 2913/14 is ideally set up to transmit and receive data, using the same clock, with no race conditions or other marginal timing requirements. This is accomplished by transmitting data on the rising edge of the first clock pulse following the data enable pulse FSX and receiving data on the falling edge of the clock which is directly in the middle of the DX data pulse. Several manufacturers use leading edge timing for both transmit and receive requiring an inversion of the receive clock.

Figure 10 shows the transmit and receive clock and data timing for an entire time slot of data. A closer look at the timing functions is given in Figure 11 which looks specifically at the first clock cycle after the transmit data enable FSX.

According to the 2913/14 data sheet, the frame sync/data enable FSX must precede the clock (DCLKX) by at least T_{txd} or nominally 15 ns for that clock pulse to be recognized as the first clock pulse in the time slot. In actuality, the 2914 will allow FSX to lag up to 80 ns the DCLKX rising edge and recognize it as the first clock pulse in a 2.048 MHz system.

Once FSX has reached V_H of about 2V, the DX output will remain in the tri-state high-impedance mode for

Table 7. Gain Trim Budget for Codec/Filter Functions

<table>
<thead>
<tr>
<th>Device</th>
<th>Manufacturing Uncertainty (Initial)</th>
<th>( \Delta T ) ( \Delta \text{Supplies} )</th>
<th>Total</th>
<th>Variation* Budget for Other Components</th>
</tr>
</thead>
<tbody>
<tr>
<td>2910</td>
<td>( \pm 0.1 )</td>
<td>( \pm 0.1 )</td>
<td>( \pm 0.3 \text{ dB} )</td>
<td>0 dB</td>
</tr>
<tr>
<td>2912</td>
<td>( \pm 0.05 )</td>
<td>( \pm 0.05 )</td>
<td>( \pm 0.12 \text{ dB} )</td>
<td>( \pm 0.13 \text{ dB} )</td>
</tr>
<tr>
<td>2914</td>
<td>( \pm 0.04 )</td>
<td>( \pm 0.08 )</td>
<td>( \pm 0.12 \text{ dB} )</td>
<td>( \pm 0.13 \text{ dB} )</td>
</tr>
</tbody>
</table>

*Assumes 0.5 dB end to end gain contrast specifications.

0.08 dB variation over supplies and temperature so that more than half the system specification could be reserved for transformer, wiring, and resistor uncertainties. This possibility of using fixed precision gain trim components and abandoning the active trim holds the potential for simplification and cost reduction of the line board manufacturing process.

2.6 Power Up/Down Considerations

Power Supply Sequence—There are no requirements for a particular sequence of powering up the combochip. All discussions of power up or power down timing assume that both \( V_{CC} \) and \( V_{BB} \) are present.

Power Up Delay—Upon application of power supplies, or coming out of the standby power down mode, three circuit time constants must be observed: (1) digital signal timing, (2) autozero timing, and (3) filter settling. An internal timing circuit activates SIF, Dx, and Ts approximately two or three frames after power up. Until this time, SIG is held low and the other two signals are in a tri-state mode. During this time, SIGX will have no effect on the PCM output.

Power Down Modes—These modes are described in detail in Table 3 of the 2913/14 data sheet except for a fail-safe mode in case CLKx is interrupted. If this should happen, both DX and TSx go into the tri-state mode until the clock is restored. This ensures the safety of the PCM highway should the interrupted clock be a local problem.

3.0 OPERATING MODES

There are three basic operating modes that are supported by the 2913/14: fixed data rate timing (FDRT), variable data rate timing (VDRT), and on-line testing.

3.1 Fixed Data Rate Mode

The FDRT mode is described in some detail in both section 2.2 of this note and in the 2913/14 data sheet. In addition, Intel Application Note AP-64 (Data Con-
Figure 10. Variable Data Rate Timing for an Entire Time Slot

T_{don} or about 34 ns longer. It then comes out of tristate and will represent some data which is invalid until the valid data is available T_{DDX} or about 75 ns (100 ns worst case) after the clock rising edge. This means there is about 90 ns of invalid data after the tri-state mode. At this point there is valid data on the D_{X} highway that lasts for approximately one full clock cycle.

Since the D_{X} highway is tied directly to the D_{r} highway in digital loopback, the valid data above is now available to the receive channel with some propagation delay. The receiver is only interested in the data for about a 50 ns (110 ns worst case) window centered about the falling edge of the DCLK_{r} clock which occurs about half a clock cycle from the FS_{r} rising edge. The window width is equal to the data set-up time, T_{dss}, plus the clock fall time, T_{f}, plus the data hold time, T_{dhr}. Information at any other time on the D_{r} highway falls into the DON'T CARE category.

Retrofitting the 2913/14—Several switching/transmission systems have been designed using first-generation codecs which operate at data rates from 64 Kbps to 2.048 MBps. In addition, they may have been designed using the rising clock edges for both transmit and receive data.

Other aspects of these older designs could be relative skewing between the sync pulses (Data Enable) and the clock pulses in such a way that the sync pulse occurs after (Lags) the first clock pulse rising edge. All of these conditions can be easily handled using the variable data rate timing mode of the 2913/14 plus some simple external logic. By the addition of this logic, the 2913/14 becomes transparent to the older design thereby allowing an upgrade in performance while having no impact on backplane wiring or on system control hardware/software. In addition, many of the features of the 2913/14 may be incorporated, such as the test modes, which provide additional capabilities beyond those available in the original design and at a lower cost.

The circuit diagram in Figure 12 shows the maximum amount of additional random logic that could be necessary to make the 2913 or 2914 completely transparent at the linecard level (no impact on backplane wiring or timing). The inverter on DCLK_{r} inverts all the receive clocks for each linecard. This inverter is only needed if (1) the transmit and receive clocks are inverted at the system/backplane level (as opposed to the linecard level) and (2) the previous design used only rising (or falling) edges to clock the transmit/receive data.
3.3 On-Line Test Modes

Two modes are available which permit maintenance checking of the linecard up to the SLIC/combochip interface, including the PCM highways and time slot interchanges. Tests include time slot-dependent error checking. The two test modes are called “redundancy testing” and “analog loopback.” These test modes are described in detail in Section 4.3.

4.0 MULTIMODE TEST CAPABILITIES

The 2913/14 was designed with every phase of design, manufacturing, and operation taken into consideration. In particular, several test modes have been implemented within the device with essentially no increase in the package size or pin count. These test modes fall into three categories: design/prototype tests, manufacturing tests, and on-line operation tests; see Table 8.

4.1 Design/Prototype Testing

In the design of a linecard prototype or in the qualification of a device, it is often helpful to have direct access to the internal nodes at key points in the LSI system. Some manufacturers even dedicate pins specifically for this function. The Intel 2913/14 approach was to reduce cost by using multifunction pins and smaller packages to achieve this goal. Measurements through these multipurpose pins will typically yield full device capability against performance specifications, however these measurements are not included in the device specifications. This is done for two reasons: first, to save manufacturing cost by eliminating unnecessary tests and specifications, and, second, more cost effective manufacturing test techniques are available, as discussed in section 4.2.
Table 8. Multimode Testing for Each Level from Design to On-Line Operation

- Design/Prototype Testing
  - Direct access to transmit codec inputs
  - Direct access to the receive filter input and the transmit filter differential outputs

- Manufacturing Tests
  - Standard half channel tests for combined codec/filters
  - Filter response half channel measurements

- Operation On-Line Tests
  - Analog loopback for testing PCM and codec analog highways
  - Redundancy checks with repeatable DX outputs

Table 9 gives the input control pin values and the corresponding functions assigned to the key test pins on the 2914 for the design test modes.

Transmit Coded (Encoder)—The transmit filter can be bypassed by directly accessing the differential input of the transmit encoder with an analog differential drive signal. Table 9 shows the control pin voltages and the input pins for this test. This test mode permits DC testing of the encoder which is otherwise blocked by the AC coupling (low frequency reject filter) of the transmit filter.

Transmit and Receiver Filter—Table 9 shows the control values that permit access to the differential outputs of the transmit filter and the single-ended input to the receive filter. The voltage difference between the transmit filter outputs represents the filtered output that will be encoded. By driving VFXI (single ended or differentially), the transmit filter response is obtained as a differential output. The final stage is the 60 Hz reject filter which is a switched capacitor filter sampled at an 8 KHz rate. When measured digitally (after the encoder), the filter characteristic is obtained directly; however, when measured in analog, a sin(ωT/2)/(ωT/2) correction factor must be included.

Figure 12. Circuit Diagram Showing Connections Needed to Retrofit the 2913/14 into Existing Variable Data Rate Systems
The input to the receive filter first passes through a sample and hold. This is necessary to simulate the sin (oT/2)/oT/2 characteristic that results from the decoder D/A output. The net result is a filter characteristic that can be compared directly to the specifications.

Start-up Procedure for Test Modes—To place the 2913/14 in the test mode it is first necessary to operate the device for a few ms in normal operation. Then VBB can be applied to the control pins to select the desired test access.

### 4.2 Production Testing

While it may be convenient for the designer to have access to both the filter and the codec inputs and outputs during the design or evaluation phase the final product will always use the filter and codec circuits together with all signals passing through both on the way to or from the PCM highways. It therefore makes sense to perform all manufacturing measurements with the device configured in its normal operating mode, i.e., all measurements should be complete filter/codec half channel measurements. This approach not only tests the combo as it will actually be used, but also saves time and money by eliminating separate measurements and correlation exercises to determine the full half channel performance.

Since the transmission specifications of S/D, gain tracking, and ICN all require measurements which are "in-band" or "filter independent," the codec functions can be easily tested using conventional half channel measurement equipment. The apparent difficulty arises in trying to fully measure the filter characteristics beyond the half sampling frequency of 4 KHz. In fact, this is not really a problem with today's computer-based testing plus an understanding of the sampled data process which is discussed under "Filter Testing".

<table>
<thead>
<tr>
<th>Input</th>
<th>Pin Function (24-Pin)</th>
<th>Test Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>PDN</td>
<td>DR</td>
<td>Pin 9 DCLKR</td>
</tr>
<tr>
<td>O-VCC</td>
<td>O-VCC</td>
<td>DCLKR</td>
</tr>
<tr>
<td>VBB</td>
<td>O-VCC</td>
<td>—</td>
</tr>
<tr>
<td>O-VCC</td>
<td>VBB</td>
<td>VFRI</td>
</tr>
</tbody>
</table>

**NOTES:**
The terms used above are defined as:

±VFX = Encoder Input
±VFX0 = XMIT Filter Output
VFRI = RCV Filter Input

**ENCODER/DECODER TESTING**

Transmission specifications are AC-coupled in-band measurements when using either CCITT G.712.11 methods 1 & 2 (white noise testing and sinusoidal testing, respectively) or AT&T Pub 43801 (Sinusoidal Testing). The noise testing uses a narrowband of flat noise from 300 to 500 Hz to drive the filter/codec (either in analog or the equivalent digital sequence for the transmit/receive channels, respectively). The resulting harmonic products are used to determine S/D. Likewise, gain tracking is also determined from this signal input. Sinusoidal testing uses a tone at 1.020 KHz for S/D measurements and gain tracking measurements. Idle channel noise measurements require the combined filter/codec since it has long been shown that separate measurements of filters and codecs are difficult to relate to the combined measurement (usually there is no specific relationship because of the non-linear properties of the encoder/decoder operations). Typically the frequency response of ICN measurements is primarily determined by the weighting filter (either C message or psophometric, which are both AC-coupled, bandpass type filters).

The conclusion is that combined filter/codec testing in no way limits the measurement of half channel transmission parameters of S/D, G.T., or ICN.

**FILTER TESTING**

Testing the filter response, of the transmit and receive channels presents two separate test situations which, in some ways, are mirror images of one another. With the transmit side, signals may be introduced at any frequency to test the filter response. At the output of the filter, the resulting signals are sampled at 8 KHz and digitized resulting in a sequence of PCM words representing the samples of filtered input signal. On the receive side, a digital PCM sequence of samples representing the driving signal is converted to an analog signal by the decoder and can be measured at the filter output in analog form.
Sampling Process—In both cases of testing the filter, the signal eventually is in a sampled form. Since the sampling rate is fixed at 8 KHz, all signals must be represented below 4 KHz (half the sampling frequency). This means that the PCM bit stream can only represent signals at frequencies below 4 KHz (half the sampling frequency). Therefore, if two signals are introduced digitally representing 1 KHz and 2 KHz, there will also be frequency components located at 8 KHz = ±1 KHz and 8 KHz = ±2 KHz, and so on for all multiples of 8 KHz. Thus it is possible to generate frequencies at arbitrary values after sampling by controlling the frequency of each signal within the 4 KHz input band regardless of whether it is in analog or PCM.

Conversely, the sampling process produces replicas (aliasing) of the sampled signal around multiples of the sampling frequency. Therefore, if two signals are introduced in analog to the filter input in such a way that there is no confusion as to where the resulting component will be after sampling (i.e., don’t simultaneously put in 1 KHz and 7 KHz since both of these inputs result in a 1 KHz component in the PCM data). Then, using either technique (FFT or analog) mentioned above, measure the amplitude of the corresponding

When an analog signal is sampled, the frequency components generated are all of the same amplitude as the corresponding input spectral components. Therefore, on the transmit side, measurements made from the PCM data will have a throughput gain of unity except where components are superimposed (e.g., a 4 KHz input signal will have an alias component at 4 KHz which may double the amplitude at 4 KHz when the two components are combined).

When an analog signal is reconstructed from digital samples, it goes through a sample and hold stage which has the effect of imposing a weighting function on the resulting spectral components that is represented by

\[
\text{Sinc} \left( \frac{\omega T}{2} \right) = \sin \left( \frac{\omega T}{2} \right) \quad \frac{\omega T}{2}
\]

where \( \omega \) is the actual spectral component frequency going into the filter, and \( T \) is the width of the hold pulse at the decoder output. For the 2913/14, the analog output is held the full sample period of 125 \( \mu \)s (1/8000 Hz) so that a frequency component at \( f_1 \) will have a weighting of

\[
W = \left( \frac{8000}{\pi f_1} \right) \sin \left( \frac{\pi f_1 T}{8000} \right)
\]

Transmit Filter Test Approach—Two approaches can be used for half channel testing of the transmit filter characteristic: (1) input analog test frequencies and perform an FFT on the corresponding PCM samples that are generated to determine spectral frequencies and amplitudes at the codec output, or (2) use an “ideal” D/A converter on the PCM samples to convert the digital data back to analog so that the spectral amplitudes and frequencies can be determined using analog circuits such as spectrum analyzers or filter banks. In either case, the effects of sampling will be the same. Figure 13 shows two spectral diagrams of amplitude versus frequency. The top diagram represents the locations of nine test frequencies corresponding to the seven specified frequencies in the 2913/14 data sheet plus a component at 7 KHz and one at 10 KHz. The bottom figure shows the “equivalent” spectral component locations when carried in the PCM bit stream. As an example, frequency \#8 is located at 7 KHz. The corresponding PCM frequency is seen in the lower figure at 1 KHz. Note also that the analog component at 9 KHz (see \#8) would also generate the 1 KHz component in the PCM data.

To test the filter, the desired test frequencies are introduced in analog to the filter input in such a way that there is no confusion as to where the resulting component will be after sampling (i.e., don’t simultaneously put in 1 KHz and 7 KHz since both of these inputs result in a 1 KHz component in the PCM data). Then, using either technique (FFT or analog) mentioned above, measure the amplitude of the corresponding

![Figure 13. Spectral Properties of the Filter Test Frequencies in Analog and PCM](image-url)
sampled component. The difference between that amplitude and the input amplitude represents the filter attenuation at the frequency of the input signal. So, if the signal was at 7 KHz, the FFT will determine the amplitude of the corresponding 1 KHz signal. The amplitude change relative to the input will represent the filter attenuation at 7 KHz.

**Receive Filter Test Approach**—In this case, the PCM test signals can be generated directly from digital circuits or by going through an "ideal" A/D (companded) to generate the PCM samples. Since these samples represent frequencies below the half sampling rate, Figure 12(b) now represents the input signals and 12(a) the output, but with one significant difference—a Sinc($\pi f_f/8000$) weighting function is imposed on all the frequency components because of the decoder sample and hold output. At the filter output, the spectral component amplitudes will include the effect of the filter response and the weighting function measured at the actual test frequency. The receive filter includes a compensation network for the weighting function in its passband. Therefore, inside the passband (300 Hz to 3.4 KHz) the measured amplitudes should be compared directly to the data sheet specifications. Frequencies outside the passband must be compensated for the weighting function first to determine the true filter response.

**Summary of Filter Testing**—Table 10 lists the nine test frequencies shown in Figure 12 for both the transmit and receive filter testing. For each filter test, the input frequency (analog or PCM), measurement frequency, and test circuit gain is tabulated corresponding to the desired test frequency. The various weighting values are easily handled by computer-based test equipment since the inverse weighting function can be stored in the computer and applied to each measured amplitude as appropriate.

### 4.3 Operational On-Line Testing

Two test modes are available which facilitate on-line testing to verify operation of both the combochip and the entire switching highway network. The first is simply the capability to duplicate the same Dx transmission in multiple PCM time slots (redundancy checking), and the second is the analog loopback capability which allows the testing of a call completion through the entire PCM voice path including the time slot interchange network.

**Redundancy Checking**—A feature of the 2913/14 is that the same 8-bit PCM word can be put on the Dx highway in multiple time slots simply by holding the frame sync/data enable (FSX) high and continuing to supply clock pulses (CLKX or DCLKX). If the data enable was held high for multiple time slots, each time slot would have identical data in it. By routing this data through the PCM highways, time slot interchanges, etc., and then correlating the data between time slots, it would be possible to detect time slot-dependent data errors. When this test mode is used, no other data will be generated for the transmit highway until the frame sync returns low for at least one full clock cycle.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>200</td>
<td>200</td>
<td>1</td>
<td>200</td>
<td>200</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>300</td>
<td>300</td>
<td>1</td>
<td>300</td>
<td>300</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>3000</td>
<td>3000</td>
<td>1</td>
<td>3000</td>
<td>3000</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>3300</td>
<td>3300</td>
<td>1</td>
<td>3300</td>
<td>3300</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>3400</td>
<td>3400</td>
<td>1</td>
<td>3400</td>
<td>3400</td>
<td>1</td>
</tr>
<tr>
<td>6</td>
<td>4000</td>
<td>4000</td>
<td>0 to 2</td>
<td>4000</td>
<td>4000</td>
<td>0 to 2</td>
</tr>
<tr>
<td>7</td>
<td>4600</td>
<td>4600</td>
<td>1</td>
<td>3400</td>
<td>4600</td>
<td>Sinc $\frac{4600 \pi}{8000}$</td>
</tr>
<tr>
<td>8</td>
<td>7000</td>
<td>1000</td>
<td>1</td>
<td>1000</td>
<td>7000</td>
<td>Sinc $\frac{7000 \pi}{8000}$</td>
</tr>
<tr>
<td>9</td>
<td>10000</td>
<td>2000</td>
<td>1</td>
<td>2000</td>
<td>10000</td>
<td>Sinc $\frac{10000 \pi}{8000}$</td>
</tr>
</tbody>
</table>
**Analog Loopback**—The 2914 (2913 does not have this feature) has the capability to be remotely programmed to disconnect the outside telephone lines and tie the transmit input directly to the receive output to effect analog loopback within the combo chip. This is accomplished by setting the LOOP input to VCC (TTL high). The result is to disconnect VFxi+ and VFxi− from the external circuitry and to connect internally PWRO+ to VFxi+, GSx to PWRO−, and VFxi− to GSx (see Figure 14).

With this test set up, the entire PCM and analog transmission path up to the SLIC can be tested remotely by assigning a PCM word to a time slot that is read by the combo being tested. This data is converted to analog and passed out of the receive channel. It is taken as input by the transmit channel where it is filtered and redigitized (encoded) back to PCM. The PCM word can now be put on the transmit highway and sent back to the remote test facility. By comparing the PCM data (individually or as a series of codes) the health of that particular connection can be verified.

**Figure 14. Simplified Block Diagram of 2914 Combochip in the Analog Loopback Configuration**
**OpenNET™: THE COMPLETE OPEN NETWORK SOLUTION**

The OpenNET family provides the OEM with complete Open Network solutions for an enterprise-wide, multi-vendor network based on international standards.

**FEATURES:**
- Interoperability between the factory, office, and engineering environments
- Complete hardware and software network solutions
- Ongoing customer support through extensive training and application development

**GUIDE TO THE OpenNET™ PRODUCTS**
OpenNET™ OVERVIEW

OpenNET™ MEANS OPEN NETWORKS

Users are placing increasing demands for data communications capabilities on their computing applications. The OpenNET family of networking products supplies those capabilities to let OEMs offer solutions to communications-intensive requirements, based on Intel’s real-time computing products.

- Open to multiple media
  - IEEE 802.3/Ethernet
  - Thin-Wire Ethernet
  - IEEE 802.4
  - X.25

- Open to different Operating Systems
  - IRMX
  - MS-DOS®
  - PC-DOS
  - UNIX SYSTEM V®
  - VAX/VMS®
  - AIX®
  - INIX®

- Open to expansion
  - MULTIBUS®
  - MULTIBUS II®
  - PC XT/AT Bus

- Open to different environments
  - Factory
  - Office
  - Lab
  - Engineering Workstation

- Open to multi-vendor solutions

- Open to future upgrades
COMPLETE OpenNET™ SOLUTION FOR REAL-TIME SYSTEMS

Real-Time computer systems require a real-time operating system. The IRMX operating system from Intel is the world's most popular operating system for real-time systems.

Many real-time applications require network communication. Intel's IRMX-NET Release 3.0 delivers a rich set of networking capabilities and a full range of IRMX platform support:

- Transparent Network File Access
- Transport and Distributed Name Server Software with Programmatic Access
- IRMX System 120 (AT-bus), 320 (MULTIBUS I) and 520 (MULTIBUS II) Connections
- Remote Boot for Diskless Systems

Networked IRMX systems serve in a wide range of real-time application areas including data acquisition, factory automation, financial workstations, military, medical instrumentation, simulation and process control.

TRANSPARENT NETWORK FILE ACCESS

IRMX-NET implements the NPA protocol to provide transparent file access capabilities among IRMX, DOS, VAX/VMS, UNIX, XENIX and INIX systems on the OpenNET network. Remote files are accessed as if they resided on the local IRMX system. IRMX-NET can be configured as a network file consumer, file server, or both, depending on the application's requirements.

The IRMX operating system provides a rich set of human interface commands and system calls for accessing local files. With the addition of IRMX-NET, these commands and system calls are transparently extended to remote access as well. Transparency means that applications using the IRMX Human Interface commands or BIOS system calls do not need to know whether the files they access reside locally or on some remote system.
COMPLETE OpenNET™ SOLUTION FOR REAL-TIME SYSTEMS

Real-Time computer systems require a real-time operating system. The iRMX operating system from Intel is the world's most popular operating system for real-time systems.

Many real-time applications require network communication. Intel's iRMX-NET Release 3.0 delivers a rich set of networking capabilities and a full range of iRMX platform support:

- Transparent Network File Access
- Transport and Distributed Name Server Software with Programmatic Access
- iRMX System 120 (AT-bus), 320 (MULTIBUS I) and 520 (MULTIBUS II) Connections
- Remote Boot for Diskless Systems

Networked iRMX systems serve in a wide range of real-time application areas including data acquisition, factory automation, financial workstations, military, medical instrumentation, simulation and process control.

TRANSPARENT NETWORK FILE ACCESS

iRMX-NET implements the NFA protocol to provide transparent file access capabilities among iRMX, DOS, VAX/VMS, UNIX, XENIX and INDA systems on the OpenNET network. Remote files are accessed as if they resided on the local iRMX system. iRMX-NET can be configured as a network file consumer, file server, or both, depending on the application's requirements.

The iRMX operating system provides a rich set of human interface commands and system calls for accessing local files. With the addition of iRMX-NET, these commands and system calls are transparently extended to remote access as well. Transparency means that applications using the iRMX Human Interface commands or BIOS system calls do not need to know whether the files they access reside locally or on some remote system.
OSI TRANSPORT AND DISTRIBUTED NAME SERVER WITH PROGRAMMATIC INTERFACE

The IRMX-NET R3.0 product includes INA 960 R3 OSI Transport and Network software preconfigured for a variety of Intel Network Interface Adapters.

IRMX-NET R3.0 also includes the IRMX-NET Distributed Name Server software. The Distributed Name Server software maintains and provides access to a network directory database. The database is distributed across the network with each system maintaining its own logical view of the directory. The Distributed Name Server software provides a full set of network directory services and is used to perform such tasks as logical name to network address mapping for establishing network connections between systems.

The combination of transparent network file access with IRMX commands and system calls, plus direct programmatic access to the I NA 960 Transport and IRMX-NET Distributed Name Server software gives the programmer a powerful set of capabilities for developing real-time network applications.

IRMX® SYSTEM 120, 320 AND 520 CONNECTIONS

IRMX-NET R3.0 provides networking support for the full range of Intel real-time Systems, from the low-cost AT-Bus System 120, through the MULTIBUS I System 320, to the high-end multiprocessing MULTIBUS II System 520. IRMX-NET R3.0 also supports IRMX board-level designs built around Intel's family of host CPU boards and Network Interface Adapters. Consistent operating system and networking software interfaces provide for easy development of network applications that span the various IRMX platforms.

REAL-TIME BOARD AND SYSTEM LEVEL SUPPORT

<table>
<thead>
<tr>
<th>IRMX® 86</th>
<th>IRMX® II</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>MULTIBUS® I</strong></td>
<td><strong>MULTIBUS® I</strong></td>
</tr>
<tr>
<td>SYSTEM</td>
<td>AT-BUS</td>
</tr>
<tr>
<td>SYSTEM 310AP</td>
<td></td>
</tr>
<tr>
<td>HOST BOARD</td>
<td></td>
</tr>
<tr>
<td>ISBC 86/30</td>
<td>ISBC 286/10A</td>
</tr>
<tr>
<td>ISBC 86/35</td>
<td>ISBC 286/12, 14, 16</td>
</tr>
<tr>
<td>ISBC 86/C38</td>
<td>ISBC 386/12</td>
</tr>
<tr>
<td>ISBC 286/10(A)</td>
<td>ISBC 386/12</td>
</tr>
<tr>
<td>ISBC 286/12, 14, 16</td>
<td></td>
</tr>
<tr>
<td>NETWORK INTERFACE ADAPTER</td>
<td>PCLINK2</td>
</tr>
<tr>
<td>ISBC 552(A)</td>
<td>ISBC 552(A)</td>
</tr>
<tr>
<td>ISBC 186/51*</td>
<td></td>
</tr>
</tbody>
</table>

*ISBC 186/51 support requires separate purchase of I NA 960 R30

REMOTE BOOT FOR DISKLESS SYSTEMS

IRMX-NET R3.0 supports networked diskless systems by providing network Boot Consumer, Boot Server and File Server capabilities.

PRODUCT CODES

IRMXNET
IRMX-NET Networking Software for the IRMX 86 operating system.

IRMXII
IRMX-NET Networking Software for the IRMX II operating system.
COMPLETE OpenNET™ SOLUTION FOR THE PC

Users of IBM PC AT, PC XT and other compatible computers can access Intel's OpenNET networking system through the OpenNET PC Link2 family of hardware and software products. The hardware connection is provided by an 80186/82586-based intelligent expansion board, the PCLINK2 Network Interface Adapter (PCLINK2NIA). The software package incorporates MS-NET for transparent file access under DOS, INA 961, NetBIOS interface, dynamic name resolution and user-friendly installation software.

The NetBIOS interface provides the flexibility to use the PCLINK2NIA with commercially available NetBIOS compatible applications, such as IBM's PC-LAN program. Optionally, MS-NET networking software is available for the upper layers.

TRANSPARENT NETWORKING FILE ACCESS

OpenNET/PCLINK2 gives users the freedom to network PCs as consumer workstations or as file servers. PCLINK2 with MS-NET implements the NPA Protocol for easy access to files on other operating systems, such as INDX, XENIX, UNIX, IRAI, or VAXA/MS.

REMOTE BOOT FOR DISKLESS SYSTEM

Diskless workstation support for the PC is provided by on-board firmware, an IRAI-NEXT Boot Server and any OpenNET File Server on the network.

PRODUCT CODES

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>sPCLINK2NIA</td>
<td>PC Link2 Network Interface Adapter, hardware only</td>
</tr>
<tr>
<td>sPCLINK2</td>
<td>Seven-layer solution with: sPCLINK2NIA, INA 961, NetBIOS interface, MS-NET</td>
</tr>
<tr>
<td>sPCLINKIBD</td>
<td>Five-layer solution with: sPCLINK2NIA, INA 961, NetBIOS interface</td>
</tr>
<tr>
<td>sPCLINK2TWKIT</td>
<td>Seven-layer thin-wire solution with: sPCLINK2, CNSTXCR, XCVRCBL-5</td>
</tr>
<tr>
<td>sPCLINK2DEVKIT</td>
<td>NetBIOS Developer's Kit with: 2-sPCLINK2CNETKITs, NetBIOS programmer kit</td>
</tr>
<tr>
<td>PCLDOSRBIRO</td>
<td>Request Block Developer's Software with INA 961 for PCLINK2</td>
</tr>
</tbody>
</table>

QSI Layers

<table>
<thead>
<tr>
<th>Layer</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>7 Application</td>
<td>MS-NET or</td>
</tr>
<tr>
<td>6 Presentation</td>
<td>IBM PC LAN</td>
</tr>
<tr>
<td>5 Session</td>
<td>sPCLINK2TWKIT</td>
</tr>
<tr>
<td>4 Transport</td>
<td>sPCLINK2</td>
</tr>
<tr>
<td>3 Network</td>
<td>sPCLINK2NIA</td>
</tr>
<tr>
<td>2 Data Link</td>
<td>INA 961, sPCLINKIBD</td>
</tr>
<tr>
<td>1 Physical</td>
<td>XCVRCBL-5 Thin-wire Transceiver</td>
</tr>
</tbody>
</table>

5-Foot Cable
**COMPLETE OpenNET™ SOLUTION FOR THE VAX**

VAX/VMS Networking software (VMSNET) provides the OpenNET connection for a VAX* or MicroVAX II* system to iRMX, XENIX, MS-DOS, UNIX System V and iNRM systems. VMSNET enables a MicroVAX or VAX system to act as an OpenNET file server system allowing any OpenNET consumer node transparent file access to files on the MicroVAX or VAX. The VMSNET product includes one of two types of Ethernet controller boards: a UNIBUS* board for the VAX or a Q-Hub* board for the MicroVAX.

VMSNET software performs the OpenNET functions via an implementation of the Network File Access (NFA) file server protocols. VMS consumer bidirectional file transfer utilities, and Intel's IIA 960 transport layer software running on the supplied intelligent LAN controllers.

A set of network management utilities provide (Micro)VAX users with information and statistics about VMSNET activities.

**FILE ACCESS**
- Transparent file access between a VAX/VMS server and MS-DOS, IRMx, XENIX, UNIX System V and iNRM systems.
- DECNet compatibility - consumer nodes may access remote files using VMS logical names over DECNet (no file locking or compatibility mode open).

**HOST REQUIREMENTS**
- VAX 750, 780, 782, 785
- VAX 8200, 8250, 8500, 8530, 8600, 8650
- MicroVAX II
- (Micro)VMS versions 4.2-5.0

**HARDWARE FEATURES**
- 80186/82586-based LAN Boards
- Unibus power requirements: +5 VDC at 4.5 amps, 6 amps maximum, -5 VDC at .5 amps, 3 amp surge
- Qbus power requirements: +5 VDC at 6 amps, 6 amps maximum
- Internal cables, mounting hardware and user manuals are included.

Hardware installation and service contracts should be arranged with Digital Equipment Service Personnel.

---

**PRODUCT CODES**

**VMSNET**
- VAX/VMS Networking Software for VAX family
- VAX/VMS Networking Software for MicroVAX II
COMPLETE OpenNET™ SOLUTION FOR UNIX SYSTEM V

SV-OpenNET connects Intel SYSTEM V/386 systems with all the OpenNET nodes. SV-OpenNET is available for MULTIBUS I and MULTIBUS II. The product includes a complete solution: communications board, Mail, VT, print spooling, nameserver interface library (NSI), and network management.

SV-OpenNET allows application interfacing through the UNIX TLI library. Applications may also access SV-OpenNET via the higher-level NSI library. SV-OpenNET can also coexist with the UNIX network, NFS.

FEATURES

• Network File Access (NFA) based
• Core, Extended, and Intel protocols supported
• Both Server and Consumer functionality supported
• Remote Batch Execution (RBE) through "rexe"

NETWORK ADMINISTRATION AND MANAGEMENT

• Compatible with XENIX-Net
• File-based Nameserver compatible with XENIX-Net /net/data files

MAIL

• Supports VMDF (1.2BSD UNIX mail)
• Interoperates with XENIX-Net

VIRTUAL TERMINAL (VT)

• OpenNET/MS-Net VT protocols supported
• Both Server and Consumer functionality supported

PRINT SPOOLING

• Interface through "rprint"
• Supports Core printer spooling protocol

UNIX STANDARD INTERFACE

• Interface via AT&T supplied TLI (Streams) library, allowing all TLI applications to interoperate with SV-OpenNET
• SV-OpenNET provides a library, NSI, for high-level Virtual Circuit (VC) creation and name to address translation. The NSI then communicates directly with the UNIX TLI

HOST REQUIREMENTS

Intel SYSTEM V.3.1 UNIX Operating System on MULTIBUS I or MULTIBUS II

PRODUCT CODES

SVNET552A    SV-OpenNET with ISBC 552A on MULTIBUS I
SVNET530     SV-OpenNET with ISBC 186/530 on MULTIBUS II
**MAP/TOP OpenNET™ NETWORKING SOFTWARE**

**ISO/OSI CONFORMANT NETWORK SOFTWARE**

Intel's MAPNET™ provides all seven layers of the industry-standard ISO/OSI specification for both Broadband (IEEE 802.4) and Ethernet (IEEE 802.3) environments.

The MAPNET software comes preconfigured or configurable to allow the OEM to change parameters as necessary. In addition, MAPNET provides multiple implementation methods (MAP on Broadband, MAP on Ethernet, and the coexistence of Broadband and Ethernet) to get started with MAP. The open software architecture allows an easy port to other operating systems and hardware.

**PRECONFIGURED MAP21SX M**

The preconfigured form (MAP21SX M) provides ISO/OSI Layers 3 through 7 of the MAP2.1 specification. It is preconfigured with iNA 960 to run on Intel's iSBC-554 IEEE 802.4 Token Bus MAP board to provide a seven layer solution. The preconfigured MAP21SX M software product is supplied with iRMX device drivers, user interface utilities, and the conformance tested MAPNET software.

**CONFIGURABLE MAPNET21**

The configurable MAPNET21 implements layers 5 through 7 of the MAP2.1 specification. MAPNET21 is designed to interface with iNA 960 and the iSBC-554 to provide a complete seven layer configurable MAP solution for OEMs.

**FEATURES**

The MAPNET products provide session services, directory services, network management, FTAM, and CASE as specified in the MAP2.1 specification.

Using the services of MAPNET, users can initiate communications with other users on a MAP network, access information regarding resources available on the network, transfer files across the network, and address others by logical names rather than numbered addresses.

The Manufacturing Messaging Specification (RS-511 or MMS) for MAPNET on iRMX-86 is also available from independent software vendors.

**PRODUCT CODES**

MAPNET21  Configurable ISO/OSI Layers 5 through 7 of the MAP2.1

MAP21SX MRO  Preconfigured ISO/OSI Layers 3 through 7 of the MAP2.1
FULLY COMPLIANT ISO/OSI TRANSPORT AND NETWORK

INA 960 is a complete Network and Transport (ISO/OSI) Layers 3 and 4 software system plus a comprehensive set of network management functions, Data Link (OSI Layer 2) drivers for IEEE 802.3 Ethernet and IEEE 802.4 Token Bus (MAP), and system environment features.

FLEXIBLE AND HIGHLY CONFIGURABLE

INA 960 is a mature, flexible, and ready-to-use software building block for OEM suppliers of networked systems for both manufacturing and office applications (e.g., MAP and TOP).

This software is highly configurable for designs based on the 82586 and 82586 LAN controllers, 82301 and 82302 Ethernet serial interface and transceiver, and the 8086 family of microprocessors.

CONFIGURABLE AT THE OBJECT CODE LEVEL

Consisting of linkable object modules, the INA 960 software can be configured to implement a range of capabilities and interface protocols. INA 960 has a large installed base and has been used reliably in a variety of systems from IBM PC XT/AT to 80286 to IBM mainframes.

BASED ON INTERNATIONAL STANDARDS

Based on the ISO/OSI seven layer model for network communications, INA 960 implements ISO 8073 Transport Class 4 providing reliable full-duplex message delivery service on top of the internet capabilities offered by the network layers. The INA 960 network layer is an implementation of the ISO 8473 Network Class 3 Connectionless Network Protocol and supports ISO 9542 End System to Intermediate System Network Dynamic Routing. INA 960 also supports ISO 8602 Connectionless Transport Protocol (Datagram).

PRECONFIGURED INA 961

INA 960 contains the preconfigured INA 961 which includes support for the ISBE 552 A, ISBC 186/530, and the ISBC 534.

REMOTE BOOT SERVER SUPPORT

INA 960 provides basic boot server capabilities that will transmit predefined images to diskless workstations that request them.

MULTI-SERVER/CONSUMER SUPPORT

INA 960 supports the powerful MULTIBUS II feature of multiple host and communications boards. Ideal for LAN load balancing and redundant networks for fault-tolerant systems.

FEATURES

- Certified ISO/OSI Transport and Network Layer Software
- ISO 8072/8073 Transport Class 4
- ISO 8602 Connectionless Transport
- ISO 8348/8473 Connectionless Network
- ISO 9542 End System to Intermediate System Dynamic Routing
- Comprehensive Network Management Functions
- Remote Boot Server for diskless workstations
- Data Link Drivers for ISBE 552 A, ISBC 586, ISBC 186/530, ISBC 554, and ISBC 186/51

PRODUCT CODES

INA960J includes INA 961 on RAM diskette format
MULTIBUS® OpenNET™ NETWORKING HARDWARE

**ISBC® 186/51 MULTIBUS I**
**IEEE 802.3/ETHERNET COMMUNICATION COMPUTER**

- 80186 LAN coprocessor for Ethernet/IEEE 802.3 communication
- Two serial interfaces, RS232C and RS422/485 compatible
- 6 MHz 80186 microprocessor
- 128K bytes of dual-port RAM expandable on-board to 256K bytes
- Sockets for up to 192K bytes of JEDEC 28-pin standard memory devices
- Product Code: sSBC18651

**ISBC®/ISXM® 552A MULTIBUS I**
**IEEE 802.3/ETHERNET NETWORK INTERFACE ADAPTER**

- High Performance: IEEE 802.3/Ethernet compatible network front-end processor
- Resident network software can be downloaded over the bus or the LAN
- On-board diagnostic and boot firmware
- ISXM® 552A version is a preconfigured controller for executing INA 961 (ISO 8073 Transport and ISO 8473 Network software) in System 310 and 320 family products
- Product Code: pSBC552A, pSX552A
MULTIBUS®II OpenNET™ NETWORKING HARDWARE

ISBC® 186/530 MULTIBUS®II IEEE 802.3 ETHERNET NETWORK INTERFACE ADAPTER

- Provides Ethernet® (IEEE 802.3) compatible networking capability for all MULTIBUS®II systems
- MULTIBUS®I IPSB (Parallel System Bus) interface with full Message Passing capability
- Resident firmware to support Built-in Self-Test (BIST) power-up diagnostics, and host-to-controller software download
- Four 28-pin JEDEC sites, expandable to 8 sites with ISBC® 341 MULTIMODULE® for a maximum of 512k bytes of EPROM
- Provides one RS232C serial port for use in debug and testing
- Product Code: pSBC186530

PC BUS OpenNET™ NETWORKING HARDWARE

PC LINK2 NETWORK INTERFACE ADAPTER (PC LINK2 NIA)

- Intelligent high performance hardware with on-board microprocessor, 16K bytes EPROM and 256k bytes RAM.
- Full slot PC AT, PC XT (or compatible computer system) board
- 80186 microprocessor, 82586 LAN coprocessor, 8 MHz zero-wait-state memory access
- RAM shared by the PC host and PC Link2 board via an 8K memory window.
- Jumper selection for Ethernet or IEEE 802.3
- Effective self diagnostics.
- Product Code: pSCLINK2NIA
**OpenNET™ Networking Accessories**

**ISBX™ 586 Data Link Engine Multimodule™ Board**

- Provides an IEEE 802.3/Ethernet compatible connection for 8086 and 80186-based host boards over a 16-bit ISBX™ interface
- Single-wide ISBX™ Multimodule™
- LAN Coprocessor: 82586 (8 MHz)
- RAM (Bytes): 16K (dual-port)
- Software Support: INA 960/961
- Power Requirements:
  - +5V: 2.0 A
  - +12V: 1.0 A
- Compatible with INA 960/961 ISO 8073 Transport and ISO 8473 Network software
- Provides an IEEE 802.3 to IEEE 802.4 Router capability when used with the ISBX 744 IEEE 802.4 LAN controller
- Product Code: SS1A5386

**IDCM 911-1 Intellink™ Fan-Out Unit**

- Connects up to nine Ethernet compatible workstations without the need for transceivers or coaxial cable
- Connects directly to the Ethernet coaxial cable through a standard transceiver cable

**Ethernet/IEEE 802.3 Thin-Wire Transceiver**

- Die-cast metal case for protection, reduced EMI, and efficient heat dissipation
- Low rush current at power-up, auto shutdown when low-input voltage occurs, and surge protection
- IEEE 802.3-compliant, Ethernet V1.0A2.0 compatible

<table>
<thead>
<tr>
<th>Size</th>
<th>14 in W x 7.8 in H x 5.5 in D</th>
</tr>
</thead>
<tbody>
<tr>
<td>Power Requirements</td>
<td>100/120/220/240 VAC, 47-64 Hz</td>
</tr>
</tbody>
</table>

- Three LEDs monitor power status, packet collisions and signal quality
- Removable BNC type cable tap
- User-configurable for use with or without heatshrink
- Product Code: CNETVAR

---

7-13
# OpenNET™ NETWORKING ORDERING INFORMATION

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IRMX-NET</td>
<td>IRMX-NET for IRMX 86 operating system</td>
</tr>
<tr>
<td>IRMX-NET</td>
<td>IRMX-NET for IRMX II operating system</td>
</tr>
</tbody>
</table>

## PClINK2 OpenNET PRODUCTS

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PClINK2NIA</td>
<td>PC Link2 Network Interface Adapter: Hardware only</td>
</tr>
<tr>
<td>PClINK2</td>
<td>Seven-layer Solution with PClINK2NIA, iNA961, NetBIOS interface, MS-NET</td>
</tr>
<tr>
<td>PClINK2HBD</td>
<td>Five-layer Solution with PClINK2NIA, iNA961, NetBIOS interface</td>
</tr>
<tr>
<td>PClINK2TWK</td>
<td>Seven-layer Thin-wire Solution with PClINK2, CNETXCVR, XCVRCHL-5</td>
</tr>
<tr>
<td>PClINK2DEV</td>
<td>NetBIOS Developer's Kit with 2-PClINK2CNRTKITS, NetBIOS programmer kit</td>
</tr>
<tr>
<td>PCLDOSRBI</td>
<td>Request Block Developer's Software with iNA 961 for PClink2</td>
</tr>
<tr>
<td>PCLDOSRBI</td>
<td>Royalty fee for PCLDOSRBI</td>
</tr>
<tr>
<td>PCLINKSWUP</td>
<td>iNA961 K1 to K3 migration software for PC Link and R1 to R3 for PCLink2</td>
</tr>
<tr>
<td>PCLINKSWUP</td>
<td>Royalty fee for PCLINKSWUP</td>
</tr>
</tbody>
</table>

## VAX/VMS OpenNET PRODUCTS

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>VMSNET</td>
<td>Networking Software for VAX family</td>
</tr>
<tr>
<td>MVMNSET</td>
<td>VAX/VMS Networking Software for MicroVAX II</td>
</tr>
</tbody>
</table>

## UNIX SYSTEM V OpenNET PRODUCTS

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SVNETS52A</td>
<td>UNIX SV-OpenNET with ISBC 552A on MULTIBUS I</td>
</tr>
<tr>
<td>SVNETS30</td>
<td>UNIX SV-OpenNET with ISBC 186/530 on MULTIBUS II</td>
</tr>
</tbody>
</table>

## MAP/TOP OpenNET PRODUCTS

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MAPNET21</td>
<td>Configurable ISO OSI Layers 5 through 7 of the MAP2.1</td>
</tr>
<tr>
<td>MAPNET21RF</td>
<td>Royalty fee for MAPNET21</td>
</tr>
<tr>
<td>MAP21SXMRO</td>
<td>Preconfigured ISO OSI Layers 3 through 7 of the MAP2.1, includes license</td>
</tr>
<tr>
<td>MAP21SXMRF</td>
<td>Royalty fee for MAP21SXM</td>
</tr>
<tr>
<td>ISBC554-1</td>
<td>ISBC554-1 MULTIBUS I MAP Communications Engine: Xmit: CH 3', 4' Rev: CH P, Q</td>
</tr>
<tr>
<td>ISBC554-3</td>
<td>ISBC554-3 MULTIBUS I MAP Communications Engine: Xmit: CH6', FM1' Rev: CH T, U</td>
</tr>
</tbody>
</table>

## INA 960 OpenNET ISO/OSI PRODUCTS

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>INA960J</td>
<td>ISO/OSI Transport and Network layers, includes INA961</td>
</tr>
<tr>
<td>INA960RF</td>
<td>Royalty fee for INA960</td>
</tr>
</tbody>
</table>

## MULTIBUS I AND MULTIBUS II IEEE 802.3/ETHERNET PRODUCTS

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>eSBC18651</td>
<td>ISBC 186/51 MULTIBUS I IEEE 802.3/Ethernet Communication Computer</td>
</tr>
<tr>
<td>pSBC552A</td>
<td>ISBC 552A MULTIBUS II IEEE 802.3/Ethernet Network Interface Adapter</td>
</tr>
<tr>
<td>pSBC552A</td>
<td>ISBC 552A preconfigured for Intel System 310 and 320, includes INA 961 royalty</td>
</tr>
<tr>
<td>pSBC186530</td>
<td>ISBC 186/530 MULTIBUS II IEEE 802.3/Ethernet Network Interface Adapter</td>
</tr>
</tbody>
</table>

## OpenNET NETWORKING ACCESSORIES

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>eSBCX586</td>
<td>ISBC 586 MULTIMODULE, IEEE 802.3/Ethernet Data Link board</td>
</tr>
<tr>
<td>pDCM911</td>
<td>IDCM 911-1 Intellink Fan-out Unit</td>
</tr>
<tr>
<td>CNETXCVR</td>
<td>Thin-wire transceiver. Requires transceiver cable (XCVRCHL-5)</td>
</tr>
<tr>
<td>XCVRCHL-5</td>
<td>Five-foot transceiver cable</td>
</tr>
<tr>
<td>Code</td>
<td>Description</td>
</tr>
<tr>
<td>------------</td>
<td>------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>IRMX-NET OpenNET PRODUCTS</td>
<td></td>
</tr>
<tr>
<td>462040-001</td>
<td>iRMA-NET Software Release 3.0 Installation and Configuration Guide</td>
</tr>
<tr>
<td>462041-001</td>
<td>iRMA-NET Software Release 3.0 User's Guide</td>
</tr>
<tr>
<td>PCLINK2 OpenNET PRODUCTS</td>
<td></td>
</tr>
<tr>
<td>460665-001</td>
<td>PCLINK2 User's Guide</td>
</tr>
<tr>
<td>450772-001</td>
<td>PCLINK2 Hardware Reference Manual</td>
</tr>
<tr>
<td>462305-001</td>
<td>PCLINK2 NIA Hardware Installation Guide</td>
</tr>
<tr>
<td>462311-001</td>
<td>PCLINK2 Software Developer's Manual</td>
</tr>
<tr>
<td>462308-001</td>
<td>PCLINK R3 0 Software For DOS—Installation Guide</td>
</tr>
<tr>
<td>VAX/VMS OpenNET PRODUCTS</td>
<td></td>
</tr>
<tr>
<td>480071-001</td>
<td>VAX/VMS OpenNET User's Manual</td>
</tr>
<tr>
<td>UNIX SYSTEM V OpenNET PRODUCTS</td>
<td></td>
</tr>
<tr>
<td>462740-001</td>
<td>SV-OpenNET User’s Manual</td>
</tr>
<tr>
<td>462741-001</td>
<td>SV-OpenNET Installation and Administration Guide</td>
</tr>
<tr>
<td>MAP/TOP OpenNET PRODUCTS</td>
<td></td>
</tr>
<tr>
<td>461298-001</td>
<td>MAPNET User's Guide</td>
</tr>
<tr>
<td>454209-001</td>
<td>ISPC 554 Network Interface Adapter Hardware Reference Manual</td>
</tr>
<tr>
<td>460432-001</td>
<td>MAP Broadband Starter Kit Guide</td>
</tr>
<tr>
<td>INA 960 OpenNET ISO/OSI PRODUCTS</td>
<td></td>
</tr>
<tr>
<td>462252-001</td>
<td>INA 960 K3 Installation and Configuration Guide</td>
</tr>
<tr>
<td>MULTIBUS I AND MULTIBUS II IEEE 802.3/ETHERNET PRODUCTS</td>
<td></td>
</tr>
<tr>
<td>122330-001</td>
<td>ISBC 18651 COMMputer Board Hardware Reference Manual</td>
</tr>
<tr>
<td>149228-001</td>
<td>ISBC 552A IEEE 802.3 Communications Controller User’s Guide</td>
</tr>
<tr>
<td>149226-002</td>
<td>ISBC 186530 Network Interface Adapter User’s Guide</td>
</tr>
<tr>
<td>OpenNET NETWORKING ACCESSORIES</td>
<td></td>
</tr>
<tr>
<td>122290-001</td>
<td>ISBA 586 MULTIMODULE Ethernet Communication Controller Hardware Reference Manual</td>
</tr>
<tr>
<td>122074-002</td>
<td>IDCM 91.1-1 Intellink Cluster Module Reference Manual</td>
</tr>
<tr>
<td>250865-001</td>
<td>Ethernet/IEEE 802.3 Thin-wire Transceiver Factsheet</td>
</tr>
</tbody>
</table>

*UNIA is a registered trademark of AT&T  
MS-DOS, XENIX, are trademarks of Microsoft  
DECnet, VAX/VMS, MicroVAX, UNIBUS are trademarks of Digital Equipment Corporation
The iSBC 88/45 Advanced Data Communications Processor (ADCP) Board adds 8 MHz, 8088 (8088-2) 8-bit microprocessor-based communications flexibility to the Intel line of OEM microcomputer systems. The iSBC 88/45 ADCP board offers asynchronous, synchronous, SDLC, and HDLC serial interfaces for gateway networking or general purpose solutions. The iSBC 88/45 ADCP board provides the CPU, system clock, EPROM/RAM, serial I/O ports, priority interrupt logic, and programmable timers to facilitate higher-level application solutions.
FUNCTIONAL DESCRIPTION

Three Communication Channels

Three programmable HDLC/SDLC serial interfaces are provided on the iSBC 88/45 ADCP board. The SDLC interface is familiar to IBM system and terminal equipment users. The HDLC interface is known by users of CCITT’s X.25 packet switching interface.

One channel utilizes an Intel 8273 controller to manage the serial data transfers. Accepting the 8-bit data bytes from the local bus, the 8273 controller translates the data into the HDLC/SDLC format. The channel operates in half/full-duplex mode.

In addition to the synchronous mode, the 8273 controller operates asynchronously with NRZI encoded data which is found in systems such as the IBM 3650 Retail Store System. An SDLC loop configuration using iSBX 352 and iSBC 88/45 products is shown in Figure 1.

The two additional channels utilize the Intel 8274 Multi-Protocol Serial Controller (MPSC). The MPSC provides two independent half/full-duplex serial channels which provide asynchronous, synchronous, HDLC or SDLC protocol operations. The sync and async protocol operations are commonly used to communicate with inexpensive terminals and systems.

The three serial channels of the iSBC 88/45 ADCP board offer communications capability to manage a gateway application. The gateway application, as shown in Figure 1, manages diverse protocol requirements for data movement between channels. Typical protocol management software layers implemented by the user include SNA terminal interfaces to IBM systems.

On-Board DMA

For high-speed communications, one MPSC channel has a DMA capacity to support an 800K baud rate. The second channel attached to the MPSC is capable of simultaneous 800K baud operation when configured with DMA capability, but is connected to an RS232C interface which is defined as 20K baud maximum. Figure 2 shows an RS422A/449 multi-drop application which supports high-speed operation.

Interfaces Supported

The iSBC 88/45 ADCP board provides an excellent foundation to support these electrical and diverse software drivers protocol interfaces. The control lines, serial data lines, and signal ground lines are brought out to the three double-edge connectors. Figure 3 shows the cable to connector construction. Two connectors are pre-configured for RS422A/449. All three channels are configurable for RS232C/CCITT V.24 interfaces as shown in Table 1.

Table 1. iSBC® 88/45 Supported Configurations

<table>
<thead>
<tr>
<th>Connection</th>
<th>Synchronous</th>
<th>Asynchronous</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Modem</td>
<td>Direct</td>
</tr>
<tr>
<td>Point-to-Point</td>
<td>X**</td>
<td>X</td>
</tr>
<tr>
<td>Multidrop</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Loop</td>
<td>N.A.</td>
<td>N.A.</td>
</tr>
</tbody>
</table>

*Modem should not respond to break. **Channels A, B, and C denoted by X.

Figure 1. iSBC® 88/45 Gateway Processor Example
NOTE:
The last slave device in the system must contain termination resistors on all signal lines received by the slave board. The master device contains bias resistors on all signal lines.

Figure 2. Synchronous Multidrop Network Configuration Example—RS422A

Figure 3. Cable Construction and Installation for RS232C and RS422A/449 Interface
Self Clocking Point-to-Point Interface

The ISBC 88/45 ADCP board is used in an asynchronous mode interface when configured as shown in Figure 4. The point-to-point RS232C example uses the self-clocking mode interface for NRZI encoding/decoding of data. The digital phase-lock loop allows operation of the interface in either half-duplex or full-duplex implementation with or without modems.

![Diagram of Self-Clocking or Asynchronous Point-to-Point Modem Interface Configuration Example—RS232C](image)

Synchronous Point-to-Point Interface

Figure 5 shows a synchronous point-to-point mode of operation for the ISBC 88/45 ADCP board. This RS232C example uses a modem to generate the receive clock for coordination of the data transfer. The ISBC 88/45 ADCP board generates the transmit synchronizing clock for synchronous transmission.

![Diagram of Synchronous Point-to-Point Modem Interface Configuration Example—RS232C](image)

Central Processing Unit

The central processor for the ISBC 88/45 Advanced Data Communications Processor board is Intel's iAPX 8088 microprocessor operating at 8 MHz. The microprocessor interface to other functions is illustrated in Figure 6. The microprocessor architecture is designed to effectively execute the application and networking software written in higher-level languages.

This architectural support includes four 16-bit byte addressable data registers, two 16-bit memory base pointer registers and two 16-bit index registers. These registers are addressable through 24 different operand addressing modes for comprehensive memory addressing and for high-level language data structure manipulation.

The stack-oriented architecture readily supports Intel's iRMX executives and iMMX multiprocessing software. Both software packages are designed for modular application programming. Facilitating the fast inter-module communications, the 4-byte instruction queue supports program constructs needed for real-time systems.

Since programs are segmented between pure procedure and data, four segment registers (code, stack, data, extra) are available for addressing 1 megabyte of memory space. These registers contain the offset values used to address a 64K byte segment. The registers are controlled explicitly through program control or implicitly by high-level language functions and instructions.

The real-time system software can also utilize the programmable timers as shown in Table 2 and various interrupt control modes available on the ADCP board to have responsive and effective application solutions.

<table>
<thead>
<tr>
<th>Table 2. Programmable Timer Functions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Function</td>
</tr>
<tr>
<td>Interrupt on Terminal Count</td>
</tr>
<tr>
<td>Rate Generator</td>
</tr>
<tr>
<td>Square Wave Generator</td>
</tr>
<tr>
<td>Software Triggered Strobe</td>
</tr>
</tbody>
</table>
**Numeric Data Processor Extension**

The 8088 instruction set includes 8-bit and 16-bit signed and unsigned arithmetic operators for binary, BCD, and unpacked ASCII data. For enhanced numerics processing capability, the ISBC 337 MULTI-MODULE Numeric Data Processor extends the 8088 architecture and data set.

The extended numerics capability includes over 60 numeric instructions offering arithmetic, trigonometric, transcendental, logarithmic, and exponential instructions. Many math-oriented applications utilize the 16-, 32-, and 64-bit integer, 32- and 64-bit floating point, 18-digit packed BCD, and 80-bit temporary data types.

**16K Bytes Static Ram**

The ISBC 88/45 ADCP board contains 16K bytes of high-speed static RAM, with 12K bytes dual-ported which is addressable from other MULTIBUS devices. When coupled with the high-speed DMA capability of the ISBC 88/45 ADCP board, the dual-ported memory provides effective data communication buffers. The dual-ported memory is useful for interprocessor message transfers.

**Interrupt Capability**

The ISBC 88/45 ADCP board provides nine vectored interrupt levels. The highest level is the NMI (Non-Maskable Interrupt) line. The additional eight interrupt levels are vectored via the Intel 8259A Programmable Interrupt Controller (PIC). As shown in Table 3, four priority processing modes are available to match interrupt servicing requirements. These modes and priority assignments are dynamically configurable by the system software.

<table>
<thead>
<tr>
<th>Mode</th>
<th>Operation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Nested</td>
<td>Interrupt request line priorities fixed; interrupt 0 is the highest and 7 is the lowest.</td>
</tr>
<tr>
<td>Auto-Rotating</td>
<td>The interrupt priority rotates; once an interrupt is serviced it becomes the lowest priority.</td>
</tr>
<tr>
<td>Specific Priority</td>
<td>System software assigns lowest level priority. The other levels are sequenced based on the level assigned.</td>
</tr>
<tr>
<td>Polled</td>
<td>System software examines priority interrupt via interrupt status register.</td>
</tr>
</tbody>
</table>

**Table 3. Programmable Interrupt Modes**

**Figure 6. Block Diagram of the ISBC® 88/45 ADCP Board**
Interrupt Request Generation

Listed in Table 4 are the devices and functions supported by interrupts on the iSBC 88/45 ADCP board. All interrupt signals are brought to the interrupt jump­er matrix. Any of the 23 interrupt sources are strapped to the appropriate 8259A PIC request level. The PIC resolves requests according to the software selected mode and, if the interrupt is unmasked, issues an interrupt to the CPU.

EPROM/RAM Expansion

In addition to the on-board RAM, the iSBC 88/45 ADCP board provides four 28-pin JEDEC sockets for EPROM expansion. By using 2764 EPROMs, the board has 32K bytes of program storage. Three of the JEDEC standard sockets also support byte-wide static RAMs or iRAMs; using 8K x 8 static RAMs provides an additional 24K bytes of RAM.

Inserting the optional iSBC 341 MULTIMODULE EPROM expansion board onto the iSBC 88/45 ADCP board provides four additional 28-pin JEDEC sites. This expansion doubles the available program storage or extends the RAM capability by 32K bytes.

ISBX™ MULTIMODULE™ Expansion

Two 8-bit ISBX MULTIMODULE connectors are provided on the iSBC 88/45 microcomputer. Through these connectors, additional ISBX functions extend the I/O capability of the microcomputer. The ISBX connectors provide the necessary signals to interface to the local bus.

In addition to specialized or custom designed ISBX boards, the customer has a broad range of Intel ISBC MULTIMODU­Les available, including parallel I/O, analog I/O, IEEE 488 GPIB, floppy disk, magnetic bubbles, video, and serial I/O boards.

The serial I/O MULTIMODULE boards include the iSBC 351 (one ASYNC/SYNC serial channel) the iSBC 352 (one HDLC/SDL­C serial channel) and the iSBC 354 (two SYNC/ASYNC, HDLC/SDL­C serial channels) boards. Adding two iSBC 352 MULTIMODU­Les to the iSBC 88/45 ADCP provides a total of five HDLC/SDL­C channels.

MULTIBUS® Multimaster Capabilities

OVERVIEW

The MULTIBUS system is Intel's industry standard microcomputer bus structure. Both 8- and 16-bit single board computers are supported on the MULTIBUS structure with 24 address and 16 data lines. In addition to expanding functions contained on a single board computer (e.g., memory and digital I/O), the MULTIBUS structure allows very powerful distributed processing configurations with multiple processors, intelligent slaves, and peripheral boards.

Multimaster Capability

The iSBC 88/45 ADCP board provides full MULTIBUS arbitration control logic. This control

<table>
<thead>
<tr>
<th>Device</th>
<th>Function</th>
<th>No. of Interrupts</th>
</tr>
</thead>
<tbody>
<tr>
<td>MULTIBUS Interface</td>
<td>Select 1 interrupt from MULTIBUS resident peripherals or other CPU boards.</td>
<td>8</td>
</tr>
<tr>
<td>8273 HDLC/SDL­C Controller</td>
<td>Transmit buffer empty and receive buffer full</td>
<td>2</td>
</tr>
<tr>
<td>8274 HDLC/SDL­C SYNC/ASYNC Controller</td>
<td>Software examines register for status of communication operation</td>
<td>1</td>
</tr>
<tr>
<td>8254-Timer</td>
<td>Counter 2 of both PIT devices</td>
<td>2</td>
</tr>
<tr>
<td>ISBX Connectors</td>
<td>Function determined by ISBX MULTIMODULE Board (2 interrupts per socket)</td>
<td>4</td>
</tr>
<tr>
<td>Bus Fail Safe Timer</td>
<td>Indicates MULTIBUS addressed device has not responded to command within 4 msec</td>
<td>1</td>
</tr>
<tr>
<td>Power Line Clock</td>
<td>Source of 60 MHz signal from power supply</td>
<td>1</td>
</tr>
<tr>
<td>Bus Flag Interrupt</td>
<td>Flag interrupt in byte location 1000H signals board reset or data handling request</td>
<td>2</td>
</tr>
<tr>
<td>iSBC 337A Board</td>
<td>Numeric Data Processor generated status information</td>
<td>1</td>
</tr>
<tr>
<td>8237A-5</td>
<td>Signals end of 8237 DMA operation</td>
<td>1</td>
</tr>
</tbody>
</table>
logic allows up to three iSBC 88/45 ADCP boards or other bus masters, including iSBC 286, iSBC 86 and iSBC 86 family boards to share the system bus using a serial (daisy chain) priority scheme. By using an external parallel priority decoder, the MULTIBUS system bus could be shared among sixteen masters.

The Intel standard MULTIBUS Interprocessor Protocol (MIP) software, implemented as the Intel iMMX 800 package for iRMX 86 and iRMX 88 Real-Time Executives, fully supports multiple 8- and 16-bit distributed processor functions. The software manages the message passing protocol between microprocessors.

**System Development Capabilities**

The application development cycle for an iSBC 88/45 ADCP board is reduced and simplified through the usage of several Intel tools. The tools include the Intellec Series Microcomputer Development System, the ICE-88 In-Circuit Emulator, the iSDM 86 debug monitor software, and the iRMX 86 and iRMX 88 run-time support packages.

The Intellec Series Microcomputer Development System offers a complete development environment for the iSBC 88/45 software. In addition to the operating system, assembler, utilities and application debugger features provided with the system, the user optionally can utilize higher-level languages like PL/M, PASCAL, and FORTRAN.

The ICE-88 In-Circuit Emulator provides a link between the Intellec system and the target iSBC 88/45-based system for code loading and execution. The ICE-88 package assists the developer with the debugging and system integrating processes.

**Run-Time Building Blocks**

Intel offers run-time foundation software to support applications which range from general purpose to high-performance solutions. The iRMX 88 Real-time Multitasking Executive provides a multitasking structure which includes task scheduling, task management, intertask communications, and interrupt servicing for high-performance applications. The highly configurable modules make the system tailoring job easier whether one uses the compact executive or the complete executive with its variety of peripheral devices supported.

The iRMX 86 Operating System provides a very rich set of features and options to support sophisticated applications solutions. In addition to supporting real-time requirements, the iRMX 86 Operating System has a powerful, but easy-to-use human interface. When added to the sophisticated I/O system, the iRMX 86 Operating System is readily extended to support assembler, PL/M, PASCAL, and FORTRAN software development environments. The modular building block software lends itself well to customized application solutions.

**SPECIFICATIONS**

**Word Size**

Instruction: 8, 16, 24, or 32 bits
Data: 8 or 16 bits

**System Clock**

8 MHz: ±0.1%

**NOTE:**
Jumper selectable for 4 MHz operation with iSBC 337 Numeric Data Processor module or ICE-88 product.

**Cycle Time**

Basic Instruction Cycle at 8.00 MHz: 1.25 μs, 250 ns (assumes instruction in the queue)

**NOTE:**
Basic instruction cycle is defined as the fastest instruction time (i.e., two clock cycles).

**Memory Cycle Time**

RAM: 500 ns (no wait states)
EPROM: jumper selectable from 500 ns to 625 ns.

**On-Board RAM**

<table>
<thead>
<tr>
<th>K Bytes</th>
<th>Hex Address Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>16 (total)</td>
<td>0000–3FFF</td>
</tr>
<tr>
<td>12 (dual-ported)</td>
<td>1000–3FFF</td>
</tr>
</tbody>
</table>

*Four iSBC 88/45 EPROM sockets support JEDEC 24/28-pin standard EPROMs and RAMs (3 sockets); iSBC 341 (4 sockets)*

**Environmental Characteristics**

Temperature: 0°C to +55°C, free moving air across the base board and MULTIMODULE board

Humidity: 90%, non-condensing

**Physical Characteristics**

Width: 30.48 cm (12.00 in)
Length: 17.15 cm (6.75 in)
Height: 1.50 cm (0.59 in)
Weight: 6.20 gm (22 oz)
Memory Capacity/Addressing

**On-Board EPROM**

<table>
<thead>
<tr>
<th>Device</th>
<th>Total K Bytes</th>
<th>Hex Address Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>2716</td>
<td>8</td>
<td>FE000–FFFFF</td>
</tr>
<tr>
<td>2732A</td>
<td>16</td>
<td>FC000–FFFFF</td>
</tr>
<tr>
<td>2764</td>
<td>32</td>
<td>F8000–FFFFF</td>
</tr>
<tr>
<td>27128</td>
<td>64</td>
<td>F0000–FFFFF</td>
</tr>
</tbody>
</table>

**With optional ISBC 341 MULTIMODULE™ EPROM**

<table>
<thead>
<tr>
<th>Device</th>
<th>Total K Bytes</th>
<th>Hex Address Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>2716</td>
<td>16</td>
<td>FC000–FFFFF</td>
</tr>
<tr>
<td>2732A</td>
<td>32</td>
<td>F8000–FFFFF</td>
</tr>
<tr>
<td>2764</td>
<td>64</td>
<td>F0000–FFFFF</td>
</tr>
<tr>
<td>27128</td>
<td>128</td>
<td>E0000–FFFFF</td>
</tr>
</tbody>
</table>

*Four ISBC 88/45 EPROM sockets support JEDEC 24/28-pin standard EPROMs and RAMs (static and RAM, 3 sockets); ISBC 341 sockets also support EPROMs and RAMs.

**Timer Input Frequency**—8.00 MHz ± 0.1%

**Interfaces**

iSBX™ Bus—All signals TTL compatible

Serial RS232C Signals—

- CTS: CLEAR TO SEND
- DSR: DATA SET READY
- DTE: TRANSMIT CLOCK
- DTR: DATA TERMINAL READY
- FG: FRAME GROUND
- RTS: REQUEST TO SEND
- RXC: RECEIVE CLOCK
- RXD: RECEIVE DATA
- SG: SIGNAL GROUND
- TXD: TRANSMIT DATA

Serial RS422A/449 Signals—

- CS: CLEAR TO SEND
- DM: DATA MODE
- RC: RECEIVE COMMON
- RD: RECEIVE DATA
- RS: REQUEST TO SEND
- RT: RECEIVE TIMING
- SC: SEND COMMON
- SD: SEND DATA
- SG: SIGNAL GROUND
- TR: TERMINAL READY
- TT: TERMINAL TIMING

**Electrical Characteristics**

**DC Power Dissipation**—28.3 Watts

**DC Power Requirements**

<table>
<thead>
<tr>
<th>Configuration</th>
<th>Current Requirements (All Voltages ± 5%)</th>
<th>+5V</th>
<th>+12V</th>
<th>-12V</th>
</tr>
</thead>
<tbody>
<tr>
<td>Without EPROM(1)</td>
<td>5.1A  20 mA  20 mA</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>With 8K EPROM (Using 2716)</td>
<td>+0.14A — —</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>With 16K EPROM (Using 2732A)</td>
<td>+0.20A — —</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>With 32K EPROM (Using 2764)</td>
<td>+0.24A — —</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>With 64K EPROM (Using 27128)</td>
<td>+0.24A — —</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**NOTE:**

1. AS SHIPPED—no EPROMs in sockets, no ISBC 341 module. Configuration includes terminators for two RS422A/449 and one RS232C channels.

**Serial Communication Characteristics**

<table>
<thead>
<tr>
<th>Channel</th>
<th>Device</th>
<th>Supported Interface</th>
<th>Max. Baud Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>8274(1)</td>
<td>RS442A/449</td>
<td>800K SDLC/HDLC</td>
</tr>
<tr>
<td></td>
<td></td>
<td>RS232C</td>
<td>125K Synchronous</td>
</tr>
<tr>
<td></td>
<td></td>
<td>CCITT V.24</td>
<td>50K Asynchronous</td>
</tr>
<tr>
<td>B</td>
<td>8274</td>
<td>RS232C</td>
<td>125K Synchronous(2)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>CCITT V.24</td>
<td>50K Asynchronous</td>
</tr>
<tr>
<td>C</td>
<td>8273(3)</td>
<td>RS442A/449</td>
<td>64K SDLC/HDLC(3)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>RS232C</td>
<td>9.6K SELF CLOCKING</td>
</tr>
<tr>
<td></td>
<td></td>
<td>CCITT V.24</td>
<td></td>
</tr>
</tbody>
</table>

**NOTES:**

1. 8274 supports HDLC/SDLC/SYNC/ASYNC multiprotocol
2. Exceed RS232C/CCITT V.24 rating of 20K baud
3. 8273 supports HDLC/SDLC

**BAUD RATE EXAMPLES (Hz)**

<table>
<thead>
<tr>
<th>8254 Timer Divide Count N</th>
<th>Synchronous K Baud</th>
<th>Asynchronous K Baud</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>16 32 64</td>
</tr>
<tr>
<td>10</td>
<td>800</td>
<td>50.0 25.0 12.5</td>
</tr>
<tr>
<td>26</td>
<td>300</td>
<td>19.2 9.6 4.8</td>
</tr>
<tr>
<td>31</td>
<td>256</td>
<td>16.1 8.0 4.03</td>
</tr>
<tr>
<td>52</td>
<td>154</td>
<td>9.6 4.8 2.4</td>
</tr>
<tr>
<td>104</td>
<td>76.8</td>
<td>4.8 2.4 1.2</td>
</tr>
<tr>
<td>125</td>
<td>64</td>
<td>4.0 2.0 1.0</td>
</tr>
<tr>
<td>143</td>
<td>56</td>
<td>3.5 1.7 0.87</td>
</tr>
<tr>
<td>167</td>
<td>48</td>
<td>3.0 1.5 0.75</td>
</tr>
<tr>
<td>417</td>
<td>19.2</td>
<td>— — —</td>
</tr>
<tr>
<td>833</td>
<td>9.6</td>
<td>— — —</td>
</tr>
<tr>
<td>EQUATION</td>
<td>8,000,000</td>
<td>500K 250K 125K</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>8,000,000</th>
</tr>
</thead>
<tbody>
<tr>
<td>N</td>
<td>N</td>
</tr>
</tbody>
</table>
SERIAL INTERFACE CONNECTORS

<table>
<thead>
<tr>
<th>Interface</th>
<th>Mode(1)</th>
<th>MULTIMODULE™ Edge Connector</th>
<th>Cable</th>
<th>Connector</th>
</tr>
</thead>
<tbody>
<tr>
<td>RS232C</td>
<td>DTE</td>
<td>26-pin(4), 3M-3462-0001</td>
<td>3M(2)-3349/25</td>
<td>25-pin(6), 3M-3482-1000</td>
</tr>
<tr>
<td>RS449</td>
<td>DTE</td>
<td>40-pin(5), 3M-3464-0001</td>
<td>3M(3)-3349/37</td>
<td>37-pin(7), 3M-3502-1000</td>
</tr>
<tr>
<td>RS449</td>
<td>DCE</td>
<td>40-pin(5), 3M-3464-0001</td>
<td>3M(3)-3349/37</td>
<td>37-pin(7), 3M-3503-1000</td>
</tr>
</tbody>
</table>

NOTES:
1. DTE—Data Terminal Equipment Mode (male connector); DCE—Data Circuit Equipment mode (female connector) requires line swaps.
2. Cable is tapered at one end to fit the 3M-3462 connector.
3. Cable is tapered to fit 3M-3464 connector.
4. Pin 26 of the edge connector is not connected to the flat cable.
5. Pins 36, 39, and 40 of the edge connector are not connected to the flat cable.
6. May be used with the cable housing 3M-3485-1000.
7. Cable housing 3M-3485-4000 may be used with the connector.

Line Drivers (Supplied)

<table>
<thead>
<tr>
<th>Device</th>
<th>Characteristic</th>
<th>Qty</th>
<th>Installed</th>
</tr>
</thead>
<tbody>
<tr>
<td>1488</td>
<td>RS232C</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>1489</td>
<td>RS232C</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>3486</td>
<td>RS422A</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>3487</td>
<td>RS422A</td>
<td>2</td>
<td>2</td>
</tr>
</tbody>
</table>

Reference Manual

143824—ISBC 88/45 Advanced Data Communications Processor Board Hardware Reference Manual (not supplied).

Reference manuals may be ordered from any Intel sales representative, distributor office or from Intel Literature Department, 3065 Bowers Avenue, Santa Clara, CA 95051.

ORDERING INFORMATION

<table>
<thead>
<tr>
<th>Part Number</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SBC 88/45</td>
<td>8-bit 8088-based Single Board Computer with 3 HDLC/SDLC serial channels</td>
</tr>
</tbody>
</table>
The iSBC® 188/56 Advanced Communicating Computer (COMMputer™) is an intelligent 8-channel single board computer. This iSBC board adds the 8 MHz 80188 microprocessor-based communications flexibility to the Intel line of OEM microcomputer systems. Acting as a stand-alone CPU or intelligent slave for communication expansion, this board provides a high performance, low-cost solution for multi-user systems. The features of the iSBC 188/56 board are uniquely suited to manage higher-layer protocol requirements needed in today's data communications applications. This single board computer takes full advantage of Intel's VLSI technology to provide state-of-the-art, economic, computer based solutions for OEM communications-oriented applications.
Operating Environment

The iSBC 188/56 COMMMputer™ features have been designed to meet the needs of numerous communications applications. Typical applications include:

1. Terminal/cluster controller
2. Front-end processor
3. Stand-alone communicating computer

Terminal/Cluster Controller

A terminal/cluster controller concentrates communications in a central area of a system. Efficient handling of messages coming in or going out of the system requires sufficient buffer space to store messages and high-speed I/O channels to transmit messages. More sophisticated applications, such as cluster controllers, also require character and format conversion capabilities to allow different types of terminals to be attached.

The iSBC 188/56 Advanced Communicating Computer is well suited for multi-terminal systems (see Figure 1). Up to 12 serial channels can be serviced in multi-user or cluster applications by adding two iSBX 354 MULTIMODULE boards. The dual-port RAM provides a large on-board buffer to handle incoming and outgoing messages at data rates up to 19.2K baud. Two channels are supported for continuous data rates greater than 19.2K baud. Each serial channel can be individually programmed for different baud rates to allow system configurations with differing terminal types. The firmware supplied on the iSBC 188/56 board supports up to 12 asynchronous RS232C serial channels, provides modem control and performs power-up diagnostics. The high performance of the on-board CPU provides intelligence to handle protocols and character handling typically assigned to the system CPU. The distribution of intelligence results in optimizing system performance by releasing the system CPU of routine tasks.

Front-End Processor

A front-end processor off-loads a system’s central processor of tasks such as data manipulation and text editing of characters collected from the attached terminals. A variety of terminals require flexible terminal interfaces. Program code is often dynamically downloaded to the front-end processor from the system CPU. Downloading code requires sufficient memory space for protocol handling and program code. Flow control and efficient handling of interrupts require an efficient operating system to manage the hardware and software resources.

Figure 1. Terminal/Cluster Controller Application
The iSBC 188/56 board features are designed to provide a high performance solution for front-end processor applications (see Figure 2). A large amount of random access memory is provided for dynamic storage of program code. In addition, local memory sites are available for storing routine programs such as X.25, SNA or bisync protocol software. The serial channels can be configured for links to mainframe systems, point-to-point terminals, modems or multidrop configurations.

Stand-Alone COMMputer™ Application

A stand-alone communication computer is a complete computer system. The CPU is capable of managing the resources required to meet the needs of multi-terminal, multi-protocol applications. These applications typically require multi-terminal support, floppy disk control, local memory allocation, and program execution and storage.

To support stand-alone applications, the iSBC 188/56 COMMputer board uses the computational capabilities of an on-board CPU to provide a high-speed system solution controlling 8 to 12 channels of serial I/O (see Figure 3). The local memory available is large enough to handle special purpose code, execution code and routine protocol software.

The MULTIBUS interface can be used to access additional system functions. Floppy disk control and graphics capability can be added to the iSBC stand-alone computer through the ISBX connectors.

ARCHITECTURE

The four major functional areas are Serial I/O, CPU, Memory and DMA. These areas are illustrated in Figure 4.

Serial I/O

Eight HDLC/SDLC serial interfaces are provided on the ISBC 188/56 board. The serial interface can be expanded to 12 channels by adding 2 ISBX 354 MULTIMODULE boards. The HDLC/SDLC interface is compatible with IBM® system and terminal equipment and with CCITT’s X.25 packet switching interface.

Four 82530 Serial Communications Controllers (SCC) provide eight channels of half/full duplex serial I/O. Six channels support RS232C interfaces. Two channels are RS232C/422/449 configurable and can be tri-stated to allow multidrop networks. The 82530 component is designed to satisfy several serial communications requirements; asynchronous,
byte-oriented synchronous (HDLC/SDLC) modes. The increased capability at the serial controller point results in off-loading the CPU of tasks formerly assigned to the CPU or its associated hardware. Configurability of the 82530 allows the user to configure it to handle all asynchronous data formats regardless of data size, number of start or stop bits, or parity requirements. An on-chip baud rate generator allows independent baud rates on each channel.

The clock can be generated either internally with the SCC chip, with an external clock or via the NRZ1 clock encoding mechanism.

All eight channels can be configured as Data Terminal Equipment (DTE) or Data Communications Equipment (DCE). Table 1 lists the interfaces supported.

**Table 1. ISBC® 188/56 Interface Support**

<table>
<thead>
<tr>
<th>Connection</th>
<th>Synchronous Modem to Direct</th>
<th>Asynchronous Modem to Direct</th>
</tr>
</thead>
<tbody>
<tr>
<td>Point-to-Point</td>
<td>X** Channels</td>
<td>X Channels</td>
</tr>
<tr>
<td>Multidrop</td>
<td>0 and 1</td>
<td>0 and 1</td>
</tr>
<tr>
<td>Loop</td>
<td>X</td>
<td>N/A</td>
</tr>
</tbody>
</table>

**All 8 channels are denoted by X.**

**Central CPU**

The 80188 central processor component provides high performance, flexibility and powerful processing. The 80188 component is a highly integrated microprocessor with an 8-bit data bus interface and a 16-bit internal architecture to give high performance. The 80188 is upward compatible with 86 and 186 software.

The 80188/82530 combination with on-board PROM/EPROM sites, and dual-port RAM provide the intelligence and speed to manage multi-user, multi-protocol communication operations.

**Memory**

There are two areas of memory on-board: dual-port RAM and universal site memory. The ISBC 188/56 board contains 256K bytes of dual-port RAM that is addressable by the 80188 on-board. The dual-port memory is configurable anywhere in a 16M byte address space on 64K byte boundaries as addressed from the MULTIBUS port. Not all of the 256K bytes are visible from the MULTIBUS bus side. The amount of dual-port memory visible to the
MULTIBUS side can be set (with jumpers) to none, 16K bytes, or 48K bytes. In a multiprocessor system these features provide local memory for each processor and shared system memory configurations where the total system memory size can exceed one megabyte without addressing conflicts.

The second area of memory is universal site memory providing flexible memory expansion. Two 28-pin JEDEC sockets are provided. One of these sockets is used for the resident firmware as described in the Firmware section.

The default configuration of the boards supports 16K byte EPROM devices such as the Intel 27128 component. However, these sockets can contain ROM, EPROM, Static RAM, or EEPROM. Both sockets must contain the same type of component (i.e. as the first socket contains an EPROM for the resident firmware, the second must also contain an EPROM with the same pinout). Up to 32K bytes can be addressed per socket giving a maximum universal site memory size of 64K bytes. By using the ISBC 341 MULTIMODULE board, a maximum of 192K bytes of universal site memory is available. This provides sufficient memory space for on-board network or resource management software.

On-Board DMA

Seven channels of Direct Memory Access (DMA) are provided between serial I/O and on-board dual port RAM by two 8237-5 components. Each of channels 0, 1, 2, 3, 5, 6, and 7 is supported by their own DMA line. Serial channels 0 and 1 are configurable for full duplex DMA. Configuring the full duplex DMA option for Channels 0 and 1 would require Channels 2 and 3 to be interrupt driven or polled. Channel 4 is interrupt driven or polled only.

Two DMA channels are integrated in the 80188 processor. These additional channels can be connected to the ISBX interfaces to provide DMA capability to ISBX MULTIMODULE boards such as the ISBX 218A Floppy Disk Controller MULTIMODULE board.

OPERATING SYSTEM SUPPORT

Intel offers run-time foundation software to support applications that range from general purpose to high-performance solutions.
Release 6 of the iRMX 86 Operating System provides a rich set of features and options to support sophisticated stand-alone communications applications on the ISBC 188/56 Advanced Communicating Computer. In addition to supporting real-time requirements, the iRMX 86 Operating System Release 6 has a powerful, yet easy to use human interface. Services provided by the iRMX 86 Operating System include facilities for executing programs concurrently, sharing resources and information, servicing asynchronous events and interactively controlling system resources and utilities. The iRMX 86 Operating System is readily extended to support assembler, PL/M, PASCAL, and FORTRAN software development environments. The modular building block software lends itself well to customized application solutions. If the ISBC 188/56 board is acting as an intelligent slave in a system environment, an iRMX 86 driver resident in the host CPU can be written by following the examples in the manual "Guide to Writing Device Driven for iRMX 86 and iRMX 88 I/O Systems".

The ISDM™ 86 System Debug Monitor supports target system debugging for the ISBC 188/56 Advanced Communicating Computer board. The monitor contains the necessary hardware, software and documentation required to interface the ISBC 188/56 target system to an Intel microcomputer development system for debugging application software.

The XENIX® 286 Operating System, Release 3, is a fully licensed adaptation of the Bell Laboratories System III UNIX® Operating System. The XENIX system is an interactive, protected, multi-user, multi-tasking operating system with a powerful, flexible human interface. Release 3 of XENIX 286 includes a software driver for the ISBC 188/56 board (and up to two ISBX 354 MULTIMODULE Boards) acting as an intelligent slave for multi-user applications requiring multiple persons running independent, terminal-oriented jobs. Example applications include distributed data processing, business data processing, software development and engineering or scientific data analysis. XENIX 286 Release 3 Operating System services include device independent I/O, tree-structured file directory and task hierarchies, re-entrant/shared code and system accounting and security access protection.

### Table 2. Features of the ISBC® 188/56 Firmware

<table>
<thead>
<tr>
<th>Feature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Asynchronous Serial Channel Support</td>
<td>Supports the serial channels in asynchronous ASCII mode. Parameters such as baud rate, parity generation, parity checking and character length can be programmed independently for each channel.</td>
</tr>
<tr>
<td>Block Data Transfer (On Output)</td>
<td>Relieves the host CPU of character-at-a-time interrupt processing. The ISBC 188/56 board accepts blocks of data for transmission and interrupts the processor only when the entire block is transmitted.</td>
</tr>
<tr>
<td>Limited Modem Control</td>
<td>Provides software control of the Data Terminal Ready (DTR) line on all channels. Transitions on the Carrier Detect (CD) line are sensed and reported to the host CPU.</td>
</tr>
<tr>
<td>Tandem Modem Support</td>
<td>Transmits an XOFF character when the number of characters in its receive buffer exceeds a threshold value and transmits an XON character when the buffer drains below some other threshold.</td>
</tr>
<tr>
<td>Download and Execute Capability</td>
<td>Provides a capability for the host CPU to load code anywhere in the address space of the ISBC 188/56 board and to start executing at any address in its address space.</td>
</tr>
<tr>
<td>Power Up Confidence Tests</td>
<td>On board reset, the firmware executes a series of simple tests to establish that crucial components on the board are functional.</td>
</tr>
</tbody>
</table>
The iSBC 188/56 Communicating COMMputer board is supplied with resident firmware that supports up to 12 RS232C asynchronous serial channels. In addition, the firmware provides a facility for a host CPU to download and execute code on the iSBC 188/56 board. Simple power-up confidence tests are also included to provide a quick diagnostic service. The firmware converts the iSBC 188/56 COMMputer board to a slave communications controller. As a slave communications controller, it requires a separate MULTIBUS host CPU board and requires the use of MULTIBUS interrupt line to signal the host processor. Table 2 summarizes the features of the firmware.

### INTERRUPT CAPABILITY

The iSBC 188/56 board has two programmable interrupt controllers (PICs). One is integrated into the 80188 processor and the other in the 80130 component. The two controllers are configured with the 80130 controller as the master and the 80188 controller as the slave. Two of the 80130 interrupt inputs are connected to the 82530 serial controller components to provide vector interrupt capabilities by the serial controllers. The iSBC 188/56 board provides 22 interrupt levels. The highest level is the NMI (Non-Maskable Interrupt) line which is directly tied to the 80188 CPU. This interrupt is typically used for signaling catastrophic events (e.g. power failure). There are 5 levels of interrupts internal to the 80188 processor. Another 8 levels of interrupts are available from the 80130 component. Of these 8, one is tied to the programmable interrupt controller (PIC) of the 80188 CPU. An additional 8 levels of interrupts are available at the MULTIBUS interface. The iSBC 188/56 board does not support bus vectored interrupts. Table 3 lists the possible interrupt sources.

**Table 3. Interrupt Request Sources**

<table>
<thead>
<tr>
<th>Device</th>
<th>Function</th>
<th>Number of Interrupts</th>
</tr>
</thead>
<tbody>
<tr>
<td>MULTIBUS Interface INT0–INT7</td>
<td>Requests from MULTIBUS resident peripherals or other CPU boards.</td>
<td>8</td>
</tr>
<tr>
<td>82530 Serial Controllers</td>
<td>Transmit buffer empty, receive buffer full and channel errors 1 and external status.</td>
<td>8 per 82530 Total = 32</td>
</tr>
<tr>
<td>Internal 80188 Timer and DMA</td>
<td>Timer 0, 1, 2 outputs and 2 DMA channel interrupts.</td>
<td>5</td>
</tr>
<tr>
<td>80130 Timer Outputs</td>
<td>Timer 0, 1, 2 outputs of 80130.</td>
<td>3</td>
</tr>
<tr>
<td>Interrupt from Flag Byte Logic</td>
<td>Flag byte interrupt set by MULTIBUS master (through MULTIBUS® I/O Write).</td>
<td>1</td>
</tr>
<tr>
<td>Bus Flag Interrupt</td>
<td>Interrupt to MULTIBUS® (Selectable for INT0 to INT7) generated from on-board 80188 I/O Write.</td>
<td>1</td>
</tr>
<tr>
<td>iSBX Connectors</td>
<td>Function determined by iSBX MULTIMODULE board.</td>
<td>4 (Two per Connector)</td>
</tr>
<tr>
<td>iSBX DMA</td>
<td>DMA interrupt from iSBX (TDMA).</td>
<td>2</td>
</tr>
<tr>
<td>Bus Fail-Safe Timeout Interrupt</td>
<td>Indicates iSBC 188/48 board timed out either waiting for MULTIBUS access or timed out from no acknowledge while on MULTIBUS System Bus.</td>
<td>1</td>
</tr>
<tr>
<td>Latched Interrupt</td>
<td>Converts pulsed event to a level interrupt. Example: 8237A-5 EOP.</td>
<td>1</td>
</tr>
<tr>
<td>OR-Gate Matrix</td>
<td>Concentrates up to 4 interrupts to 1 interrupt (selectable by stake pins).</td>
<td>1</td>
</tr>
<tr>
<td>Ring Indicator Interrupt</td>
<td>Latches a ring indicator event from serial channels 4, 5, 6, or 7.</td>
<td>1</td>
</tr>
<tr>
<td>NOR-Gate Matrix</td>
<td>Inverts up to 2 interrupts into 1 (selectable by stake pins).</td>
<td>1</td>
</tr>
</tbody>
</table>
SUPPORT FOR THE 80130 COMPONENT

Intel does not support the direct processor execution of the IRMX nucleus primitives from the 80130 component. The 80130 component provides timers and interrupt controllers.

EXPANSION

EPROM Expansion

Memory may be expanded by adding Intel compatible memory expansion boards. The universal site memory can be expanded to six sockets by adding the ISBC 341 MULTIMODULE board for a maximum total of 192K bytes of universal site memory.

**iSBX™ MULTIMODULE™ Expansion Module**

Two 8-bit ISBX MULTIMODULE connectors are provided on the ISBC 188/56 board. Using ISBX modules additional functions can be added to extend the I/O capability of the board. In addition to specialized or custom designed ISBX boards, there is a broad range of ISBX MULTIMODULE boards from the Intel including parallel I/O, analog I/O, IEEE 488 GPIB, floppy disk, magnetic bubbles, video and serial I/O boards.

The serial I/O MULTIMODULE boards available include the ISBC 354 Dual Channel Expansion MULTIMODULE board. Each ISBX 354 MULTIMODULE board adds two channels of serial I/O to the ISBC 188/56 board for a maximum of twelve serial channels. The 82530 serial communications controller on the MULTIMODULE board handles a large variety of serial communications protocols. This is the same serial controller as is used on the ISBC 188/56 board to offer directly compatible expansion capability for the ISBC 188/56 COMPUTER board.

**MULTIBUS® INTERFACE**

The ISBC 188/56 Advanced COMPUTER board can be a MULTIBUS master or intelligent slave in a multimeter system. The ISBC 188/56 board incorporates a flag byte signalling mechanism for use in multiprocessor environments where the ISBC 188/56 board is acting as an intelligent slave. The mechanism provides an interrupt handshake from the MULTIBUS System Bus to the on-board-processor and vice-versa.

The Multimaster capabilities of the ISBC 188/56 board offers easy expansion of processing capacity and the benefits of multiprocessing. Memory and I/O capacity may be expanded and additional functions added using Intel MULTIBUS compatible expansion boards.

**SPECIFICATIONS**

**Word Size**

Instruction—8, 16, 24 or 32 bits
Data Path—8 bits

**Processor Clock**

<table>
<thead>
<tr>
<th>ISBC® 188/56 Board Using:</th>
<th>Size (K bytes)</th>
<th>On Board Capacity</th>
<th>Address Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>2732</td>
<td>4K</td>
<td>8K bytes</td>
<td>FE000–FFFFFH</td>
</tr>
<tr>
<td>2764</td>
<td>8K</td>
<td>16K bytes</td>
<td>FC000–FFFFFH</td>
</tr>
<tr>
<td>27128</td>
<td>16K</td>
<td>32K bytes</td>
<td>F8000–FFFFFH</td>
</tr>
<tr>
<td>27256</td>
<td>32K</td>
<td>64K bytes</td>
<td>F0000–FFFFFH</td>
</tr>
<tr>
<td>27512</td>
<td>64K</td>
<td>128K bytes</td>
<td>E0000–FFFFFH</td>
</tr>
</tbody>
</table>

**Memory Expansion**

**EPROM with ISBC® 341 Board Using:**

<table>
<thead>
<tr>
<th>ISBC® 188/56 Board Using:</th>
<th>Capacity (K bytes)</th>
<th>Address Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>2732</td>
<td>24K</td>
<td>F8000–FFFFFH</td>
</tr>
<tr>
<td>2764</td>
<td>48K</td>
<td>F0000–FFFFFH</td>
</tr>
<tr>
<td>27128</td>
<td>96K</td>
<td>E0000–FFFFFH</td>
</tr>
<tr>
<td>27256</td>
<td>192K</td>
<td>C0000–FFFFFH</td>
</tr>
</tbody>
</table>

**I/O Capacity**

Serial—8 programmable lines using four 82530 components

ISBX MULTIMODULE—2 ISBX single-wide boards
Serial Communications Characteristics

Synchronous—Internal or external character synchronization on one or two synchronous characters

Asynchronous—5–8 bits and 1, 1½, or 2 stop bits per character; programmable clock factor; break detection and generation; parity, overrun, and framing error detection.

Baud Rates

<table>
<thead>
<tr>
<th>Baud Rate</th>
<th>82530 Count Value (Decimal)</th>
</tr>
</thead>
<tbody>
<tr>
<td>64000</td>
<td>36</td>
</tr>
<tr>
<td>48000</td>
<td>49</td>
</tr>
<tr>
<td>19200</td>
<td>126</td>
</tr>
<tr>
<td>9600</td>
<td>254</td>
</tr>
<tr>
<td>4800</td>
<td>510</td>
</tr>
<tr>
<td>2400</td>
<td>1022</td>
</tr>
<tr>
<td>1800</td>
<td>1363</td>
</tr>
<tr>
<td>1200</td>
<td>2046</td>
</tr>
<tr>
<td>300</td>
<td>8190</td>
</tr>
</tbody>
</table>

Asynchronous

<table>
<thead>
<tr>
<th>Baud Rate</th>
<th>82530 Count Value (Decimal)</th>
</tr>
</thead>
<tbody>
<tr>
<td>19200</td>
<td>6</td>
</tr>
<tr>
<td>9600</td>
<td>14</td>
</tr>
<tr>
<td>4800</td>
<td>30</td>
</tr>
<tr>
<td>2400</td>
<td>62</td>
</tr>
<tr>
<td>1800</td>
<td>83</td>
</tr>
<tr>
<td>1200</td>
<td>126</td>
</tr>
<tr>
<td>300</td>
<td>510</td>
</tr>
<tr>
<td>110</td>
<td>1394</td>
</tr>
</tbody>
</table>

SERIAL RS232C SIGNALS

CD  — Carrier
CTS — Clear to Send
DSR — Data Set Ready
DTE TXC — Transmit Clock
DTR — Data Terminal Ready
RTS — Request to Send
RXC — Receive Clock
RXD — Receive Data
SG — Signal Ground
TXD — Transmit Data
RI — Ring Indicator

RS422A/449 SIGNALS

RC — Receive Common
RD — Receive Data
RT — Receive Timing
SD — Send Data
TT — Terminal Timing

Environmental Characteristics

Temperature: 0 to 55°C at 200 Linear Feet/Min. (LFM) Air Velocity
Humidity: to 90%, non-condensing (25°C to 70°C)

Physical Characteristics

Width: 30.48 cm (12.00 in)
Length: 17.15 cm (6.75 in)
Height: 1.04 cm (0.41 in)
Weight: 595 gm (21 oz)

Electrical Characteristics

The power required per voltage for the iSSC 188/56 board is shown below. These numbers do not include the current required by universal memory sites or expansion modules.

<table>
<thead>
<tr>
<th>Voltage (Volts)</th>
<th>Current (Amps) typ.</th>
<th>Power (Watts) typ.</th>
</tr>
</thead>
<tbody>
<tr>
<td>+ 5</td>
<td>4.56A</td>
<td>22.8W</td>
</tr>
<tr>
<td>+ 12</td>
<td>0.12A</td>
<td>1.5W</td>
</tr>
<tr>
<td>− 12</td>
<td>0.11A</td>
<td>1.3W</td>
</tr>
</tbody>
</table>

Reference Manual

iSSC 188/56 Advanced Data Communications Computer Reference Manual Order Number 148209-001.

ORDERING INFORMATION

Part Number  Description
iSSC 188/56  8-Serial Channel Advanced Communicating Computer
The iSBC® 534 Four Channel Communication Expansion Board is a member of Intel’s complete line of memory and I/O expansion boards. The iSBC® 534 interfaces directly to any single board computer via the MULTIBUS to provide expansion of system serial communications capability. Four fully programmable synchronous and asynchronous serial channels with RS232C buffering and provision for 20 mA optically isolated current loop buffering are provided. Baud rates, data formats, and interrupt priorities for each channel are individually software selectable. In addition to the extensive complement of EIA Standard RS232C signals provided, the iSBC® 534 provides 16 lines of RS232C buffered programmable parallel I/O. This interface is configured to be directly compatible with the Bell Model 801 automatic calling unit. These capabilities provide a flexible and easy means for interfacing Intel iSBC based systems to RS232C and optically isolated current loop compatible terminals, cassettes, asynchronous and synchronous modems, and distributed processing networks.
FUNCTIONAL DESCRIPTION

Communications Interface

Four programmable communications interfaces using Intel's 8251A Universal Synchronous/Asynchronous Receiver/Transmitter (USART) are contained on the board.* Each USART can be programmed by the system software to individually select the desired asynchronous or synchronous serial data transmission technique (including IBM Bisync). The mode of operation (i.e., synchronous or asynchronous), data format, control character format, parity, and baud rate are all under program control. Each 8251A provides full duplex, double buffered transmit and receive capability. Parity, overrun, and framing error detection are all incorporated in each USART. Each set of RS232C command lines, serial data lines, and signal ground lines are brought out to 26-pin edge connectors that mate with RS232C flat or round cables.

16-Bit Interval Timers

The iSBC 534 provides six fully programmable and independent BCD and binary 16-bit interval timers utilizing two Intel 8253 programmable interval timers.* Four timers are available to the systems designer to generate baud rates for the USARTs under software control. Routing for the outputs from the other two counters is jumper selectable. Each may be independently routed to the programmable interrupt controller to provide real time clocking or to the USARTs (for applications requiring different transmit and receive baud rates). In utilizing the iSBC 534, the systems designer simply configures, via software, each timer independently to meet system requirements. Whenever a given baud rate or time delay is needed, software commands to the programmable timers select the desired function. Three functions of these timers are supported on the iSBC 534, as shown in Table 1. The contents of each counter may be read at any time during system operation.

<table>
<thead>
<tr>
<th>Table 1. Programmable Timer Functions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Function</td>
</tr>
<tr>
<td>-------------------------</td>
</tr>
<tr>
<td>Interrupt on terminal count</td>
</tr>
<tr>
<td>Rate generator</td>
</tr>
<tr>
<td>Square wave rate generator</td>
</tr>
</tbody>
</table>

Interrupt Request Lines

Two independent Intel 8259A programmable interrupt controllers (PIC's) provide vectoring for 16 interrupt levels.* As shown in Table 2, a selection of three priority processing algorithms is available to the system designer. The manner in which requests are serviced may thus be configured to match system requirements. Priority assignments may be reconfigured dynamically via software at any time during system operation. Any combination of interrupt levels may be masked through storage, via software, of a single byte to the interrupt mask register of each PIC. Each PIC's interrupt request output line may be jumper selected to drive any of the nine interrupt lines on the MULTIBUS.

<table>
<thead>
<tr>
<th>Table 2. Interrupt Priority Options</th>
</tr>
</thead>
<tbody>
<tr>
<td>Algorithm</td>
</tr>
<tr>
<td>---------------------</td>
</tr>
<tr>
<td>Fully nested</td>
</tr>
<tr>
<td>Auto-rotating</td>
</tr>
<tr>
<td>Specific priority</td>
</tr>
</tbody>
</table>
Figure 1. ISBC® 534 Four Channel Communications Expansion Board Block Diagram
Interrupt Request Generation—As shown in Table 3, interrupt requests may originate from 16 sources. Two jumper selectable interrupt requests (8 total) can be automatically generated by each USART when a character is ready to be transferred to the MULTIBUS system bus (i.e., receive buffer is full) or a character has been transmitted (transmit buffer is empty). Jumper selectable requests can be generated by two of the programmable timers (PITs), and six lines are routed directly from peripherals to accept carrier detect (4 lines), ring indicator, and the Bell 801 present next digit request lines.

Systems Compatibility
The iSBC 534 provides 16 RS232C buffered parallel I/O lines implemented utilizing an Intel 8255A programmable peripheral interface (PPI) configured to operate in mode 0.* These lines are configured to be directly compatible with the Bell 801 automatic calling unit (ACU). This capability allows the iSBC 534 to interface to Bell 801 type ACUs and up to four modems or other serial communications devices. For systems not requiring interface to an ACU, the parallel I/O lines may also be used as general purpose RS232C compatible control lines in system implementation.

*NOTE:
Complete operational details on the Intel 8251A USART, the Intel 8253 Programmable Interval Timer, the Intel 8255A Programmable Peripheral Interface, and the Intel 8259A Programmable Interrupt Controller are contained in the Intel Component Data Catalog.

Sample Baud Rates(1)

<table>
<thead>
<tr>
<th>Frequency(2) (kHz, Software Selectable)</th>
<th>Synchronous</th>
<th>Asynchronous</th>
</tr>
</thead>
<tbody>
<tr>
<td>153.6</td>
<td>—</td>
<td>9600</td>
</tr>
<tr>
<td>76.8</td>
<td>—</td>
<td>4800</td>
</tr>
<tr>
<td>38.4</td>
<td>38400</td>
<td>2400</td>
</tr>
<tr>
<td>19.2</td>
<td>19200</td>
<td>1200</td>
</tr>
<tr>
<td>9.6</td>
<td>9600</td>
<td>600</td>
</tr>
<tr>
<td>4.8</td>
<td>4800</td>
<td>300</td>
</tr>
<tr>
<td>6.98</td>
<td>6980</td>
<td>—</td>
</tr>
</tbody>
</table>

NOTES:
1. Baud rates shown here are only a sample subset of possible software programmable rates available. Any frequency from 18.75 Hz to 614.4 kHz may be generated utilizing on-board crystal oscillator and 16-bit programmable interval timer (used here as frequency divider).
2. Frequency selected by I/O writes of appropriate 16-bit frequency factor to Baud Rate Register.

Interval Timer and Baud Rate Generator Frequencies

Input Frequency (On-Board Crystal Oscillator)—1.2288 MHz ± 0.1% (0.813 μs period, nominal)

<table>
<thead>
<tr>
<th>Function</th>
<th>Single Timer</th>
<th>Dual/Timer Counter (Two Timers Cascaded)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Min</td>
<td>Max</td>
</tr>
<tr>
<td>Real-Time interrupt interval</td>
<td>1.63 μs</td>
<td>53.3 ms</td>
</tr>
<tr>
<td>Rate generator (Frequency)</td>
<td>18.75 Hz</td>
<td>614.4 kHz</td>
</tr>
</tbody>
</table>

SPECIFICATIONS

Serial Communications Characteristics
Synchronous—5-8 bit characters; internal or external character synchronization; automatic sync insertion.
Interfaces—RS232C Interfaces

EIA Standard RS232C Signals provided and supported:
- Carrier detect
- Clear to send
- Data set ready
- Data terminal ready
- Request to send
- Receive clock
- Receive data
- Ring indicator
- Secondary receive data
- Secondary transmit data
- Transmit clock
- Transmit data
- Ring indicator
- Secondary receive data
- Secondary transmit data
- Transmit clock
- Transmit data

Parallel I/O—8 input lines, 8 output lines, all signals RS232C compatible

Bus—All signals MULTIBUS system bus compatible

I/O Addressing

The USART, interval timer, interrupt controller, and parallel interface registers of the ISBC 534 are configured as a block of 16 I/O address locations. The location of this block is jumper selectable to begin at any 16-byte I/O address boundary (i.e., 00H, 10H, 20H, etc.).

I/O Access Time

400 ns  USART registers
400 ns  Parallel I/O registers
400 ns  Interval timer registers
400 ns  Interrupt controller registers

Physical Characteristics

Width: 12.00 in. (30.48 cm)
Height: 6.75 in. (17.15 cm)
Depth: 0.50 in. (1.27 cm)
Weight: 14 oz. (398 gm)

Electrical Characteristics

Average DC Current

<table>
<thead>
<tr>
<th>Voltage</th>
<th>Without Opto-Isolators</th>
<th>With Opto-Isolators(1)</th>
</tr>
</thead>
<tbody>
<tr>
<td>VCC = +5V</td>
<td>1.9 A, max</td>
<td>1.9 A, max</td>
</tr>
<tr>
<td>VDD = +12V</td>
<td>275 mA, max</td>
<td>420 mA, max</td>
</tr>
<tr>
<td>VAA = -12V</td>
<td>250 mA, max</td>
<td>400 mA, max</td>
</tr>
</tbody>
</table>

NOTE: 1. With four 4N33 and four 4N37 opto-isolator packages installed in sockets provided to implement four 20 mA current loop interfaces.

Environmental Characteristics

Operating Temperature: 0°C to +55°C

Reference Manual

502140-002—ISBC 534 Hardware Reference Manual (NOT SUPPLIED)

Reference manuals are shipped with each product only if designated SUPPLIED (see above). Manuals may be ordered from any Intel sales representative, distributor office or from Intel Literature Department, 3065 Bowers Avenue, Santa Clara, California 95051.

ORDERING INFORMATION

<table>
<thead>
<tr>
<th>Part Number</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SBC 534</td>
<td>Four Channel Communication Expansion Board</td>
</tr>
</tbody>
</table>
iSBC® 544
INTELLIGENT COMMUNICATIONS CONTROLLER

- iSBC® Communications Controller
  Acting as a Single Board
  Communications Computer or an
  Intelligent Slave for Communications
  Expansion

- Ten Programmable Parallel I/O Lines
  Compatible with Bell 801 Automatic
  Calling Unit

- On-Board Dedicated 8085A
  Microcontroller Providing
  Communications Control and Buffer
  Management for Four Programmable
  Synchronous/Asynchronous Channels

- Individual Software Programmable
  Baud Rate Generation for Each Serial
  I/O Channel

- Sockets for Up To 8K Bytes of EPROM

- Twelve Levels of Programmable
  Interrupt Control

- 16K Bytes of Dual Port Dynamic Read/
  Write Memory with On-Board Refresh

- Three Independent Programmable
  Interval Timer/Counters

- Extended MULTIBUS® Addressing
  Permits ISBC 544 Board Partitioning
  into 16K-Byte Segments In a
  1-Megabyte Address Space

- Interface Control for Auto Answer and
  Auto Originate Modem

The ISBC 544 Intelligent Communications Controller is a member of Intel's family of single-board computers, memory, I/O, and peripheral controller boards. The ISBC 544 board is a complete communications controller on a single 6.75 x 12.00 inch printed circuit card. The on-board 8085A CPU may perform local communications processing by directly interfacing with on-board read/write memory, nonvolatile read only memory, four synchronous/asynchronous serial I/O ports, RS232/RS366 compatible parallel I/O, programmable timers, and programmable interrupts.
FUNCTIONAL DESCRIPTION

Intelligent Communications Controller

Two Mode Operation — The iSBC 544 board is capable of operating in one of two modes: 1) as a single board communications computer with all computer and communications interface hardware on a single board; 2) as an “intelligent bus slave” that can perform communications related tasks as a peripheral processor to one or more bus masters. The iSBC 544 may be configured to operate as a standalone single board communications computer with all MPU, memory and I/O elements on a single board. In this mode of operation, the iSBC 544 may also interface with expansion memory and I/O boards (but no additional bus masters). The iSBC 544 performs as an intelligent slave to the bus master by performing all communications related tasks. Complete synchronous and asynchronous I/O and data management are controlled by the on-board 8085A CPU to coordinate up to four serial channels. Using the iSBC 544 as an intelligent slave, multi-channel serial transfers can be managed entirely on-board, freeing the bus master to perform other system functions.

Architecture — The iSBC 544 board is functionally partitioned into three major sections: I/O, central computer, and shared dual port RAM memory (Figure 1). The I/O hardware is centered around the four Intel 8251A USART devices providing fully programmable serial interfacing. Included here as well is a 10-bit parallel interface compatible with the Bell 801 automatic calling unit, or equivalent. The I/O is under full control of the on-board CPU and is protected from access by system bus masters. The second major segment of the intelligent communications controller is a central computer, with an 8085A CPU providing powerful processing capability. The 8085A together with on-board EPROM/ROM, static RAM, programmable timers/counters, and programmable

---

Figure 1. ISBC® 544 Intelligent Communications Controller Block Diagram
interrupt control provide the intelligence to manage sophisticated communications operations on-board the iSBC 544 board. The timer/counters and interrupt control are also common to the I/O area providing programmable baud rates to the USARTs and prioritizing interrupts generated from the USARTs. The central computer functions are protected for access only by the on-board 8085A. Likewise, the on-board 8085A may not gain access to the system bus when being used as an intelligent slave. When the iSBC 544 is used as a bus master, the on-board 8085A CPU controls complete system operation accessing on-board functions as well as memory and I/O expansion. The third major segment, dual port RAM memory, is the key link between the iSBC 544 intelligent slave and bus masters managing the system functions. The dual port concept allows a common block of dynamic memory to be accessed by the on-board 8085A CPU and off-board bus masters. The system program can, therefore, utilize the shared dual port RAM to pass command and status information between the bus masters and on-board CPU. In addition, the dual port concept permits blocks of data transmitted or received to accumulate in the on-board shared RAM, minimizing the need for a dedicated memory board.

Serial I/O

Four programmable communications interfaces using Intel's 8251A Universal Synchronous/Asynchronous Receiver/Transmitter (USART) are contained on the board and controlled by the on-board CPU in combination with the on-board interval timer/counter to provide all common communication frequencies. Each USART can be programmed by the system software to individually select the desired asynchronous or synchronous serial data transmission technique (including IBM Bisync). The mode of operation (i.e., synchronous or asynchronous), data format, control character format, parity, and baud rate are all under program control. Each 8251A provides full duplex, double-buffered, transmit and receive capability. Parity, overrun, and framing error detection are all incorporated in each USART. Each channel is fully buffered to provide a direct interface to RS232C compatible terminals, peripherals, or synchronous/asynchronous modems. Each channel of RS232C command lines, serial data lines, and signal ground lines are brought out to 26-pin edge connectors that mate with RS232C flat or round cable.

Parallel I/O Port

The iSBC 544 provides a 10-bit parallel I/O interface controlled by an Intel 8155 Programmable Interface (PPI) chip. The parallel I/O port is directly compatible with an Automatic Calling Unit (ACU) such as the Bell Model 801, or equivalent, and can also be used for auxiliary functions. All signals are RS232C compatible, and the interface cable signed assignments meet RS366 specifications. For systems not requiring an ACU interface, the parallel I/O port can be used for any general purpose interface requiring RS232C compatibility.

Central Processing Unit

Intel's powerful 8-bit microprocessor 8085A CPU, fabricated on a single LSI chip, is the central processor for the iSBC 544. The 8085A CPU is directly software compatible with the Intel 8080A CPU. The 8085A contains six 8-bit general purpose registers and an accumulator. The six general purpose registers may be addressed individually or in pairs, providing both single and double precision operators. The minimum instruction execution time is 1.45 microseconds. The 8085A CPU has a 16-bit program counter. An external stack, located within any portion of iSBC 544 read/write memory, may be used as a last-in/first-out storage area for the contents of the program counter, flags, accumulator, and all of the six general purpose registers. A 16-bit stack pointer controls the addressing of this external stack. This stack provides subroutine nesting bounded only by memory size.

EPROM/ROM Capacity

Sockets for up to 8K bytes of nonvolatile read only memory are provided on the iSBC 544 board. Read only memory may be added in 2K byte increments up to a maximum of 4K bytes using Intel 2716 EPROMs or masked ROMs; or in 4K byte increments up to 8K bytes maximum using Intel 2732 EPROMs. All on-board EPROM/ROM operations are performed at maximum processor speed.

RAM Capacity

The iSBC 544 contains 16K bytes of dynamic read/write memory using Intel 2117 RAMs. Power for the on-board RAM may be provided on an auxiliary power bus, and memory protect logic is included for RAM battery backup requirements. The iSBC 544 contains a dual port controller, which provides dual port capability for the on-board RAM memory. RAM accesses may occur from either the on-board 8085A CPU or from another bus master, when used as an intelligent slave. Since on-board RAM accesses do not require the MULTIBUS, the bus is available for concurrent bus master use. Dynamic RAM refresh is accomplished automatically by the iSBC 544 for accesses originating from either the CPU or from the MULTIBUS.
Addressing — On board RAM, as seen by the on-board 8085A CPU, resides at address 8000H–BFFFFH. On-board RAM, as seen by an off-board CPU, may be placed on any 4K byte address boundary. The iSBC 544 provides extended addressing jumpers to allow the on-board RAM to reside within a one megabyte address space when accessed via the MULTIBUS. In addition, jumper options are provided which allow the user to protect 8K or 12K bytes on-board RAM for use by the on-board 8085 CPU only. This reserved RAM space is not accessible via the MULTIBUS and does not occupy any system address space.

Static RAM — The iSBC 544 board also has 256 bytes of static RAM located on the Intel 8155 PPI. This memory is only accessible to the on-board 8085A CPU and is located at address 7F00H–7FFFH.

Programmable Timers

The iSBC 544 board provides seven fully programmable and independent interval timer/counters utilizing two Intel 8253 Programmable Interval Timers (PIT), and the Intel 8155. The two Intel 8253 PITs provide six independent BCD or binary 16-bit interval timer/counters and the 8155 provides one 14-bit binary timer/counter. Four of the PIT timers (BGDO–4) are dedicated to the USARTs providing fully independent programmable baud rates.

Three General Use Timers — The fifth timer (BGD4) may be used as an auxiliary baud rate to any of the four USARTs or may alternatively be cascaded with timer six to provide extended interrupt intervals. The sixth PIT timer/counter (TINT1) can be used to generate interrupt intervals to the on-board 8085A. In addition to the timer/counters on the 8253 PITs, the iSBC 544 has a 14-bit timer available on the 8155 PPI providing a third general use timer/counter (TINT0). This timer output is jumper selectable to the interrupt structure of the on-board 8085A CPU to provide additional timer/counter capability.

Timer Functions — In utilizing the iSBC 544 board, the systems designer simply configures, via software, each timer independently to meet system requirements. Whenever a given baud rate or interrupt interval is needed, software commands to the programmable timers select the desired function. The on-board PITs together with the 8155 provide a total of seven timer/counters and six operating modes. Mode 3 of the 8253 is the primary operating mode of the four dedicated USART baud rate generators. The timer/counters and useful modes of operation for the general use timer/counters are shown in Table 1.

Interrupt Capability

The iSBC 544 board provides interrupt service for up to 21 interrupt sources. Any of the 21 sources may interrupt the intelligent controller, and all are brought through the interrupt logic to 12 interrupt levels. Four interrupt levels are handled directly by the interrupt processing capability of the 8085A CPU and eight levels are serviced from an Intel 8259A Programmable Interrupt Controller (PIC) routing an interrupt request output to the INTR input of the 8085A (see Table 2).

<table>
<thead>
<tr>
<th>Function</th>
<th>Operation</th>
<th>Counter</th>
</tr>
</thead>
<tbody>
<tr>
<td>Interrupt on Terminal Count (Mode 0)</td>
<td>When terminal count is reached, an interrupt request is generated. This function is useful for generation of real-time clocks.</td>
<td>8253 TINT1</td>
</tr>
<tr>
<td>Rate Generator (Mode 2)</td>
<td>Divide by N counter. The output will go low for one input clock cycle and high for N – 1 input clock periods.</td>
<td>8253 BGD4*</td>
</tr>
<tr>
<td>Square-Wave Rate Generator (Mode 3)</td>
<td>Output will remain high until one-half the TC has been completed, and go low for the other half of the count. This is the primary operating mode used for generating a Baud rate clocked to the USARTs.</td>
<td>8253 BGD0–4 TINT1</td>
</tr>
<tr>
<td>Software Triggered Strobe (Mode 4)</td>
<td>When the TC is loaded, the counter will begin. On TC the output will go low for one input clock period.</td>
<td>8253 BGD4* TINT1</td>
</tr>
<tr>
<td>Single Pulse</td>
<td>Single pulse when TC reached.</td>
<td>8155 TINT0</td>
</tr>
<tr>
<td>Repetitive Single Pulse</td>
<td>Repetitive single pulse each time TC is reached until a new command is loaded.</td>
<td>8155 TINT0</td>
</tr>
</tbody>
</table>

* BGD4 is jumper selectable as an auxiliary baud rate generator to the USARTs or as a cascaded output to TINT1. BGD4 may be used in modes 2 and 4 only when configured as a cascaded output.
Table 2. Interrupt Vector Memory Locations

<table>
<thead>
<tr>
<th>Interrupt Source</th>
<th>Vector Location</th>
<th>Interrupt Level</th>
</tr>
</thead>
<tbody>
<tr>
<td>Power Fail</td>
<td>TRAP</td>
<td>24H</td>
</tr>
<tr>
<td>8253 TINT1</td>
<td>RST 7.5</td>
<td>2</td>
</tr>
<tr>
<td>8155 TINT0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Ring Indicator(1)</td>
<td>RST 6.5</td>
<td>34H</td>
</tr>
<tr>
<td>Carrier Detect</td>
<td></td>
<td>3</td>
</tr>
<tr>
<td>Flag Interrupt</td>
<td>RST 5.5</td>
<td>2CH</td>
</tr>
<tr>
<td>INTO/-INT7/ (1 of 8)</td>
<td></td>
<td>4</td>
</tr>
<tr>
<td>RXRDY0</td>
<td>INTR</td>
<td>Programmable</td>
</tr>
<tr>
<td>TXRDY0</td>
<td></td>
<td>5–12</td>
</tr>
<tr>
<td>TXRDY1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TXRDY2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TXRDY3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RXRDY2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RXRDY3</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

NOTE:
1. Four ring indicator interrupts and four carrier detect interrupts are summed to the RST 6.5 input. The 8155 may be interrogated to inspect any one of the eight signals.

Interrupt Sources — The 22 interrupt sources originate from both on-board communications functions and the MULTIBUS. Two interrupts are routed from each of the four USARTs (8 interrupts total) to indicate that the transmitter and receiver are ready to move a data byte to or from the on-board CPU. The PIC is dedicated to accepting these 8 interrupts to optimize USART service request. One of eight interrupt request lines are jumper selectable for direct interface from a bus master via the system bus. Two auxiliary timers (TINT0 from 8155 and TINT1 from 8253) are jumper selectable to provide general purpose counter/timer interrupts. A jumper selectable Flag Interrupt is generated to allow any bus master to interrupt the iSBC 544 by writing into the base address of the shared dual port memory accessible to the system. The Flag Interrupt is then cleared by the iSBC 544 when the on-board processor reads the base address. This interrupt provides an interrupt link between a bus master and intelligent slave (see System Programming). Eight inputs from the serial ports are monitored to detect a ring indicator and carrier detect from each of the four channels. These eight interrupt sources are summed to a single interrupt level of the 8085A CPU. If one of these eight interrupts occur, the 8155 PPI can then be interrogated to determine which port caused the interrupt. Finally, a jumper selectable Power Fail Interrupt is available from the MULTIBUS to detect a power down condition.

8085 Interrupt — Thirteen of the twenty-two interrupt sources are available directly to four interrupt inputs of the on-board 8085A CPU. Requests routed to the 8085A interrupt inputs, TRAP, RST 7.5, RST 6.5 and RST 5.5 have a unique vector memory address. An 8085A jump instruction at each of these addresses then provides software linkage to interrupt service routines located independently anywhere in the Memory. All interrupt inputs with the exception of the TRAP may be masked via software.

8259A Interrupts — Eight interrupt sources signaling transmitter and receiver ready from the four USARTs are channeled directly to the Intel 8259A PIC. The PIC then provides vectoring for the next eight interrupt levels. Operating mode and priority assignments may be reconfigured dynamically via software at any time during system operation. The PIC accepts transmitter and receiver interrupts from the four USARTs. It then determines which of the incoming requests is of highest priority, determines whether this request is of higher priority than the level currently being serviced, and, if appropriate, issues an interrupt to the CPU. The output of the PIC is applied directly to the INTR input of the 8085A. Any combination of interrupt levels may be masked, via software, by storing a single byte in the interrupt mask register of the PIC. When the 8085A responds to a PIC interrupt, the PIC will generate a CALL instruction for each interrupt level. These addresses are equally spaced at intervals of 4 or 8 (software selectable) bytes. Interrupt response to the PIC is software programmable to a 32- or 64-byte block of memory. Interrupt sequences may be expanded from this block with a single 8085A jump instruction at each of these addresses.

Interrupt Output — In addition, the iSBC 544 board may be jumper selected to generate an interrupt from the on-board serial output data (SOD) of the 8085A. The SOD signal may be jumpered to any one of the 8 MULTIBUS interrupt lines (INT0/-INT7/) to provide an interrupt signal directly to a bus master.

Power-Fail Control

Control logic is also included to accept a power-fail interrupt in conjunction with the AC-low signal from the iSBC 635 Power Supply or equivalent.

Expansion Capabilities

When the iSBC 544 board is used as a single board communications controller, memory and I/O capacity may be expanded and additional functions added using Intel MULTIBUS™ compatible expansion boards. In this mode, no other bus masters may be configured in the system. Memory may be expanded to a 65K byte capacity by adding user specified combinations of RAM boards, EPROM boards, or combination boards. Input/output capacity may be increased by adding digital I/O and analog I/O expan-
sion boards. Furthermore, multiple iSBC 544 boards may be included in an expanded system using one iSBC 544 board as a single board communications computer and additional controllers as intelligent slaves.

System Programming

In the system programming environment, the iSBC 544 board appears as an additional RAM memory module when used as an intelligent slave. The master CPU communicates with the iSBC 544 board as if it were just an extension of system memory. Because the iSBC 544 board is treated as memory by the system, the user is able to program into it a command structure which will allow the iSBC 544 board to control its own I/O and memory operation. To enhance the programming of the iSBC 544 board, the user has been given some specific tools. The tools are: 1) the flag interrupt, 2) an on-board RAM memory area that is accessible to both an off-board CPU and the on-board 8085A through which a communications path can exist, and 3) access to the bus interrupt line.

Flag Interrupt — The Flag Interrupt is generated anytime a write command is performed by an off-board CPU to the base address of the iSBC 544 board’s RAM. This interrupt provides a means for the master CPU to notify the iSBC 544 board that it wishes to establish a communications sequence. In systems with more than one intelligent slave, the flag interrupt provides a unique interrupt to each slave outside the normal eight MULTIBUS interrupt lines (INT0/−INT7).

On-Board RAM — The on-board 16K byte RAM area that is accessible to both an off-board CPU and the on-board 8085A can be located on any 4K boundary in the system. The selected base address of the iSBC 544 RAM will cause an interrupt when written into by an off-board CPU.

Bus Access — The third tool to improve system operation as an intelligent slave is access to the MULTIBUS interrupt lines. The iSBC 544 board can both respond to interrupt signals from an off-board CPU, and generate an interrupt to the off-board CPU via the MULTIBUS.

System Development Capability

The development cycle of iSBC 544 board based products may be significantly reduced using the Intellec series microcomputer development systems. The Intellec resident macroassembler, text editor, and system monitor greatly simplify the design, development and debug of iSBC 544 system software. An optional ISIS-II diskette operating system provides a linker, object code locator, and library manager. A unique in-circuit emulator (ICE-85) option provides the capability of developing and debugging software directly on the iSBC 544 board.

SPECIFICATIONS

Serial Communications Characteristics

Synchronous — 5–8 bit characters; automatic sync insertion; parity.

Asynchronous — 5–8 bit characters; break character generation; 1, 1½, or 2 stop bits; false start bit detection; break character detection.

<table>
<thead>
<tr>
<th>Frequency (KHz)(1)</th>
<th>Baud Rate (Hz)(2)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(Software Selectable)</td>
<td>Synchronous</td>
</tr>
<tr>
<td>153.6</td>
<td>—</td>
</tr>
<tr>
<td>76.8</td>
<td>—</td>
</tr>
<tr>
<td>38.4</td>
<td>38400</td>
</tr>
<tr>
<td>19.2</td>
<td>19200</td>
</tr>
<tr>
<td>9.6</td>
<td>9600</td>
</tr>
<tr>
<td>4.8</td>
<td>4800</td>
</tr>
<tr>
<td>6.98</td>
<td>6980</td>
</tr>
</tbody>
</table>

NOTES:
1. Frequency selected by I/O writes of appropriate 16-bit frequency factor to Baud Rate Register.
2. Baud rates shown here are only a sample subset of possible software programmable rates available. Any frequency from 18.75 Hz to 614.4 KHz may be generated utilizing on-board crystal oscillator and 16-bit Programmable Interval Timer (used here as a frequency divider).

8085A CPU

Word Size — 8, 16 or 24 bits/instruction; 8 bits of data

Cycle Time — 1.45/μs ±0.01% for fastest executable instruction; i.e., four clock cycles.

Clock Rate — 2.76 MHz ±0.1%

System Access Time

Dual port memory — 740 ns

NOTE: Assumes no refresh contention.
Memory Capacity

On-Board ROM/PROM — 4K, or 8K bytes of user installed ROM or EPROM

On-Board Static RAM — 256 bytes on 8155

On-Board Dynamic RAM (on-board access) — 16K bytes. Integrity maintained during power failure with user-furnished batteries (optional)

On-Board Dynamic RAM (MULTIBUS access) — 4K, 8K, or 16K bytes available to bus by switch selection

Memory Addressing

On-Board ROM/PROM — 0–0FFF (using 2716 EPROMs or masked ROMs); 0–1FFF (using 2732A EPROMs)

On-Board Static RAM — 256 bytes: 7F00–7FFF

On-Board Dynamic RAM (on-board access) — 16K bytes: 8000–BFFF.

On-Board Dynamic RAM (MULTIBUS® access) — any 4K increment 00000–FF000 which is switch and jumper selectable. 4K, 8K or 16K bytes can be made available to the bus by switch selection.

I/O Capacity

Serial — 4 programmable channels using four 8251A USARTs

Parallel — 10 programmable lines available for Bell 801 ACU, or equivalent use. Two auxiliary jumper selectable signals

I/O Addressing

<table>
<thead>
<tr>
<th>On-Board Programmable I/O</th>
<th>Port</th>
<th>Data</th>
<th>Control</th>
</tr>
</thead>
<tbody>
<tr>
<td>USART 0</td>
<td>D0</td>
<td></td>
<td>D1</td>
</tr>
<tr>
<td>USART 1</td>
<td>D2</td>
<td>D3</td>
<td></td>
</tr>
<tr>
<td>USART 2</td>
<td>D4</td>
<td>D5</td>
<td></td>
</tr>
<tr>
<td>USART 3</td>
<td>D6</td>
<td>D7</td>
<td></td>
</tr>
<tr>
<td>8155 PPI</td>
<td>E9 (Port A)</td>
<td>E8</td>
<td></td>
</tr>
<tr>
<td></td>
<td>EA (Port B)</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>EB (Port C)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Interrupts

Address for 8259A Registers (Hex notation, I/O address space)

<table>
<thead>
<tr>
<th>Interrupt</th>
<th>Register Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>E6</td>
<td>Interrupt request register</td>
</tr>
<tr>
<td>E6</td>
<td>In-service register</td>
</tr>
<tr>
<td>E7</td>
<td>Mask register</td>
</tr>
<tr>
<td>E7</td>
<td>Command register</td>
</tr>
<tr>
<td>E7</td>
<td>Block address register</td>
</tr>
<tr>
<td>E6</td>
<td>Status (polling register)</td>
</tr>
</tbody>
</table>

NOTE:
Several registers have the same physical address: Sequence of access and one data bit of the control word determines which register will respond.

Interrupt levels routed to the 8085 CPU automatically vector the processor to unique memory locations:

24 TRAP
3C RST 7.5
34 RST 6.5
2C RST 5.5

Timers

Addresses for 8253 Registers (Hex notation, I/O address space)

<table>
<thead>
<tr>
<th>Programmable Interrupt Timer One</th>
</tr>
</thead>
<tbody>
<tr>
<td>D8</td>
</tr>
<tr>
<td>D9</td>
</tr>
<tr>
<td>DA</td>
</tr>
<tr>
<td>DB</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Programmable Interrupt Timer Two</th>
</tr>
</thead>
<tbody>
<tr>
<td>DC</td>
</tr>
<tr>
<td>DD</td>
</tr>
<tr>
<td>DE</td>
</tr>
<tr>
<td>DF</td>
</tr>
</tbody>
</table>

Address for 8155 Programmable Timer

<table>
<thead>
<tr>
<th>Control</th>
</tr>
</thead>
<tbody>
<tr>
<td>E8</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>

Input Frequencies — Jumper selectable reference 1.2288 MHz ± 0.1% (0.814 μs period nominal) or 1.843 MHz ± 0.1% crystal (0.542 μs period, nominal)
### Output Frequencies (at 1.2288 MHz)

<table>
<thead>
<tr>
<th>Function</th>
<th>Single Timer/Counter</th>
<th>Dual Timer/Counter (two timers cascaded)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Min</td>
<td>Max</td>
</tr>
<tr>
<td>Real-Time Interrupt Interval</td>
<td>1.63 μs</td>
<td>53.3 μs</td>
</tr>
<tr>
<td>Rate Generator (frequency)</td>
<td>18.75 Hz</td>
<td>614.4 KHz</td>
</tr>
</tbody>
</table>

### Interfaces

**Serial I/O** — EIA Standard RS232C signals provided and supported:
- Carrier Detect
- Clear to Send
- Data Set Ready
- Data Terminal Ready
- Request to Send
- Receive Clock
- Receiver Data
- Ring Indicator
- Secondary Receive Data
- Secondary Transmit Data
- Transmit Clock
- Transmit Data
- DTE Transmit clock

* Optional if parallel I/O port is not used as Automatic Calling Unit.

**Parallel I/O** — Four inputs and eight outputs (includes two jumper selectable auxiliary outputs). All signals compatible with EIA Standard RS232C. Directly compatible with Bell Model 801 Automatic Calling Unit, or equivalent.

**MULTIBUS** — Compatible with iSBC MULTIBUS.

### Connectors

<table>
<thead>
<tr>
<th>Interface</th>
<th>Pins (qty)</th>
<th>Centers (In.)</th>
<th>Mating Connectors</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bus</td>
<td>86</td>
<td>0.156</td>
<td>Viking 2KH43/9AMK12</td>
</tr>
<tr>
<td>Parallel I/O</td>
<td>50</td>
<td>0.1</td>
<td>3M 3415-000 or AMP 88083-1</td>
</tr>
<tr>
<td>Serial I/O</td>
<td>26</td>
<td>0.1</td>
<td>3M 3462-000 or AMP 88373-5</td>
</tr>
</tbody>
</table>

### Memory Protect

An active-low TTL compatible memory protect signal is brought out on the auxiliary connector which, when asserted, disables read/write access to RAM memory on the board. This input is provided for the protection of RAM contents during the system power-down sequences.

### On-Board Addressing

All communications to the parallel and serial I/O ports, to the timers, and to the interrupt controller, are via read and write commands from the on-board 8085A CPU.

### Auxiliary Power

An auxiliary power bus is provided to allow separate power to RAM for systems requiring battery backup of read/write memory. Selection of this auxiliary RAM power bus is made via jumpers on the board.

### Bus Drivers

<table>
<thead>
<tr>
<th>Function</th>
<th>Characteristic</th>
<th>Sink Current (mA)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data</td>
<td>Tri-state</td>
<td>50</td>
</tr>
<tr>
<td>Address</td>
<td>Tri-state</td>
<td>15</td>
</tr>
<tr>
<td>Commands</td>
<td>Tri-state</td>
<td>32</td>
</tr>
</tbody>
</table>

**NOTE:**
- Used as a master in the single board communications computer mode.

### Physical Characteristics

<table>
<thead>
<tr>
<th>Characteristic</th>
<th>Measurement</th>
</tr>
</thead>
<tbody>
<tr>
<td>Width</td>
<td>30.48 cm (12.00 inches)</td>
</tr>
<tr>
<td>Depth</td>
<td>17.15 cm (6.75 inches)</td>
</tr>
<tr>
<td>Thickness</td>
<td>1.27 cm (0.50 inch)</td>
</tr>
<tr>
<td>Weight</td>
<td>3.97 gm (14 ounces)</td>
</tr>
</tbody>
</table>
Electrical Characteristics

DC Power Requirements

<table>
<thead>
<tr>
<th>Configuration</th>
<th>( V_{CC} = +5V \pm 5% ) (max)</th>
<th>( V_{DD} = \pm 12V \pm 5% ) (max)</th>
<th>( V_{BB} = -5V(3) \pm 5% ) (max)</th>
<th>( V_{AA} = -12V \pm 5% ) (max)</th>
</tr>
</thead>
<tbody>
<tr>
<td>With 4K EPROM</td>
<td>( I_{CC} = 3.4A ) max</td>
<td>( I_{DD} = 350 mA ) max</td>
<td>( I_{BB} = 5 mA ) max</td>
<td>( I_{AA} = 200 mA ) max</td>
</tr>
<tr>
<td>Without EPROM</td>
<td>3.3A max</td>
<td>350 mA max</td>
<td>5 mA max</td>
<td>200 mA max</td>
</tr>
<tr>
<td>RAM only(1)</td>
<td>390 mA max</td>
<td>176 mA max</td>
<td>5 mA max</td>
<td></td>
</tr>
<tr>
<td>RAM(2) refresh only</td>
<td>390 mA max</td>
<td>20 mA max</td>
<td>5 mA max</td>
<td></td>
</tr>
</tbody>
</table>

NOTES:
1. For operational RAM only, for AUX power supply rating.
2. For RAM refresh only. Used for battery backup requirements. No RAM accessed.
3. V_{BB} is normally derived on-board from V_{AA}, eliminating the need for a V_{BB} supply. If it is desired to supply V_{BB} from the bus, the current requirement is as shown.

Environmental Characteristics

Operating Temperature: 0°C to 55°C (32°F to 131°F)
Relative Humidity: To 90% without condensation

Reference Manual

502160 — iSBC 544 Intelligent Communications Controller Board Hardware Reference Manual (NOT SUPPLIED)

ORDERING INFORMATION

<table>
<thead>
<tr>
<th>Part Number</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>iSBC 544</td>
<td>Intelligent Communications Controller</td>
</tr>
</tbody>
</table>
The ISBC 548 and ISBC 549 are intelligent terminal controllers for MULTIBUS 1 applications. The ISBC 548 provides basic multiuser support with 8 channels of RS 232 Asynchronous interface. The ISBC 549 combines 4 serial channels with a real-time clock and a line printer interface. Acting as intelligent slaves for communication expansion, these boards provide high performance, low cost solutions for multi-user systems.

FEATURES

ISBC 548 FEATURES
- Supports eight channels asynchronous RS232 interface

ISBC 549 FEATURES
- Supports four channels asynchronous RS232 interface
- Line printer interface
- Real-time clock/calendar with battery backup

STANDARD ISBC 548/549 FEATURES
- 8 MHz 80186 Microprocessor
- Supports transfer rates up to 19.2K Baud
- 128K Bytes Zero Wait State DRAM (32K Dual Port)
- Supports Full Duplex Asynchronous Transmissions
- Jumper selectable memory mapping, I/O mapping and MULTIBUS Interrupts
FEATURES

ASYNCHRONOUS RS232 INTERFACE SUPPORT
The ISBC 548/549 Asynchronous RS232 Internal support is presented in DTE Configuration. 82330 Serial Communications Controllers (SCCs) provide channels of half/full duplex serial I/O. Configurability of the 82330 allows handling all asynchronous data formats regardless of data size, number of start or stop bits, or parity requirements. The synchronous transmission features of the 82330 are not supported. An on-chip baud rate generator allows independent baud rates on each channel. The serial lines can be brought to the back-panel via 40-pin connectors and ribbon cable.

LINE PRINTER INTERFACE
The ISBC 549 incorporates a standard line printer interface compatible with IBM* or Centronics* line printers. Intelligent buffering on the ISBC 549 allows the CPU to offload printing tasks and return to higher priority jobs.

REAL-TIME CLOCK/CALENDAR
Multibus systems will benefit from the real-time clock present on the ISBC 549 in applications requiring time stamp operations. unattended boots and other calendar requirements. The clock/calendar circuit is backed up by a non-rechargeable battery, which keeps the clock/calendar operating for six months with all other power off.

8 MHZ 80186 MICROPROCESSOR
The 80186 central processor component provides high-performance, flexibility, and powerful processing. The 80186/82330 combination with on-board PROM/EPROM sites, and dual-port RAM provides the intelligence and speed to manage multi-user communications.

TRANSFER RATES UP TO 19.2K BAUD
Collectively, each board has dual-port RAM providing an onboard buffer to handle incoming and outgoing messages at data rates up to 19.2K baud. The resident firmware supports asynchronous RS232 serial channels, provides modem control and performs power-up diagnostics. Each serial channel can be individually programmed to different baud rates to allow system configurations with differing terminal types.

MEMORY
The ISBC 548/549 have three areas of memory on-board: dual-port RAM, private RAM, and EPROM. Each board contains 128K bytes of on-board RAM, 32K bytes of dual-port RAM can be addressed by other MULTIBUS boards. The dual port memory is configurable in a 16M byte address space on 32K byte boundaries as addressed from the MULTIBUS port. The starting address is jumper selectable.

The second area of memory is 96K bytes of private RAM which is addressable by the 80186 on-board.

The third area of memory is EPROM memory expansion. Two 28-pin JEDEC sockets are provided. These sockets come populated with two EPROMs which contain the controller firmware. The boards can support 2764, 27128 and 27256 EPROMS, giving a total capacity of 64K bytes. The EPROM runs with zero wait states if EPROMs of access times 250 ns or less are used. No jumper changes are needed to access different size EPROMs.

WORLDWIDE SERVICE AND SUPPORT
Intel provides support for board repair or on-site service. Development options include phone support, subscription service, on-site consulting, and customer training.

QUALITY AND RELIABILITY
The ISBC 548 and ISBC 549 are designed and manufactured in accordance with Intel's high quality standards. We then verify quality through rigorous testing in our state-of-the-art Environmental Test Laboratory.

*IBM is a trademark of International Business Machines
*Centronics is a registered trademark of Centronics, Inc
FEATURES

Figure 1: Terminal/Cluster Controller Application

Figure 2: ISBC 548/549 Boards Block Diagram
**SPECIFICATIONS**

**SERIAL COMMUNICATIONS CHARACTERISTICS**

- Asynchronous only
- 6-8 bit character length
- 1, 1½, or 2 stop bits per character
- Parity
  - Programmable clock
  - Break Generation
  - Framing error detection

**Baud Rates**

The on-board firmware can automatically detect and set baud rates of 150, 300, 600, 1200, 4800, 9600 and 19200. Other baud rates can be set by the host.

**Serial RS232C Signals Supported**

- CD Carrier Detect
- RXD Receive Data
- TXD Transmit Data
- DTR Data Terminal Ready
- SG Signal Ground
- DSR Data Set Ready
- RTS Ready to Send
- CTS Clear to Send
- RI Ringer Indicator

These signals are supported by the iSBC 548/549 Controller and on-board firmware. All signals may not be supported by the host operating system.

**MEMORY**

- On-Board RAM - 128K bytes total
- Private RAM - 96K bytes
- Dual Port RAM - 32K bytes, can be addressed from MULTIBUS interface at any 32K boundary between 90000H and F80000H or between F80000H and FF0000H.

**EPROM Options**

<table>
<thead>
<tr>
<th>Component</th>
<th>On-Board Capacity</th>
<th>Start Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>2764</td>
<td>16K</td>
<td>F0000H</td>
</tr>
<tr>
<td>27128</td>
<td>32K</td>
<td>F80000H</td>
</tr>
<tr>
<td>27236</td>
<td>64K</td>
<td>F0000H</td>
</tr>
</tbody>
</table>

**MULTIBUS SYSTEM BUS INTERFACE**

The iSBC 548/549 boards meet MULTIBUS (IEEE 796) bus specification D16 M24 116 V0 E.

**DEVICE DRIVERS**

Check the latest release of the following operating systems for details:

- IRMA 86
- IRMA II

**ENVIRONMENTAL CHARACTERISTICS**

- Temperature: 0 to 55°C at 200 Linear Feet/Minute (LFM)
- Air Velocity
- Humidity: 5% to 90% non-condensing (25 to 70°C)

**PHYSICAL CHARACTERISTICS**

<table>
<thead>
<tr>
<th>iSBC 548</th>
<th>iSBC 549</th>
</tr>
</thead>
<tbody>
<tr>
<td>Width</td>
<td>30.34cm (12.00 in)</td>
</tr>
<tr>
<td>Length</td>
<td>16.87cm (6.75 in)</td>
</tr>
<tr>
<td>Height</td>
<td>1.27 cm (.5 in)</td>
</tr>
<tr>
<td>Weight</td>
<td>400 gm (14 oz)</td>
</tr>
</tbody>
</table>

**POWER REQUIREMENTS**

<table>
<thead>
<tr>
<th>Voltage (Volts)</th>
<th>Current (Amps)</th>
<th>Power (Watts)</th>
</tr>
</thead>
<tbody>
<tr>
<td>iSBC 548</td>
<td></td>
<td></td>
</tr>
<tr>
<td>+ 5</td>
<td>3.49</td>
<td>17.5</td>
</tr>
<tr>
<td>+ 12</td>
<td>.14</td>
<td>1.7</td>
</tr>
<tr>
<td>- 12</td>
<td>.11</td>
<td>1.3</td>
</tr>
<tr>
<td>iSBC 549</td>
<td></td>
<td></td>
</tr>
<tr>
<td>+ 5</td>
<td>3.26</td>
<td>16.3</td>
</tr>
<tr>
<td>+ 12</td>
<td>.07</td>
<td>.8</td>
</tr>
<tr>
<td>- 12</td>
<td>.06</td>
<td>.7</td>
</tr>
</tbody>
</table>

**ORDERING INFORMATION**

<table>
<thead>
<tr>
<th>Part Number</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>iSBC 548</td>
<td>8 Channel High Performance Terminal Controller</td>
</tr>
<tr>
<td>iSBC 549</td>
<td>4 Channel High Performance Terminal Controller with Line Printer/Clock</td>
</tr>
</tbody>
</table>

**REFERENCE MANUALS**

iSBC 546/547/548/549 High Performance Terminal Controller Hardware Reference Manual - Order Number 122704-002

For more information or the number of your nearest Intel sales office, call 800-548-4725 (good in the U.S. and Canada).
Intel iSBC® 561
SOEMI (Serial OEM Interface) CONTROLLER BOARD

- Dedicated I/O Controller Provides a Direct Connection of MULTIBUS®- Based Systems to an IBM 9370 or 4361 Mainframe Host or to any IBM System/370 via an IBM 3174 Subsystem Control Unit via IBM’s SOEMI (Serial OEM Interface) Protocol
- Includes a SMC-to-BNC Cable Assembly to Attach into the IBM 3270 Information Display System
- On-Board Diagnostic Capability Provides Operational Status of Board Function and Link with the Host
- Supported by a Complete Family of Single Board Computers, Memory, Digital and Analog I/O, Peripheral and Graphics Controllers’ Packaging and Software

The Intel iSBC 561 SOEMI (Serial OEM Interface) Controller Board is a member of Intel’s family of single board computers, memory, I/O, peripheral and graphics controller boards. It is a dedicated intelligent I/O controller on a MULTIBUS form-factor printed circuit card. The board allows OEMs of MULTIBUS-based systems a direct, standard link to an IBM 9370 Information System, to an IBM System 4361, or to any IBM System/370 attached to an IBM 3174 Subsystem Control Unit via the SOEMI (Serial OEM Interface). The ISBC 561 Controller also provides IBM System/370 users access to the broad range of applications supported by hundreds of MULTIBUS vendors.

The SOEMI interface is comprised of an IBM System/370 programming interface and an IBM 3270 coax interface. It is a flexible, high speed, point-to-point serial interface offered as a feature on the IBM 9370 and 4361 processor families and on the 3174 Subsystem Control Unit. The ISBC 561 SOEMI Controller Board contains two processors and provides the necessary intelligence for conversion, control functions, and buffer management between the IBM mainframe and the MULTIBUS system. This board allows an IBM user to distribute control and information to MULTIBUS compatible systems for a variety of applications including factory automation, data acquisition, measurement, control, robotics, process control, communications, local area networking, medical instrumentation, and laboratory automation.

*IBM is a trademark of International Business Machines Corp.
SOEMI INTERFACE OVERVIEW

The Serial OEM Interface (SOEMI) is a new means of connecting Original Equipment Manufacturer (OEM) MULTIBUS-based systems and subsystems to an IBM System/370 mainframe. Previously, the only low-cost way to attach non-IBM equipment into the IBM mainframe environment was to use 3270 emulation software and hardware adaptors. This type of interface is low-speed (approx. 19.6K bits/sec.) and not very flexible as to the type and format of data that can be transferred. The 3270 emulators must mimic the device formats of the displays and printers that are typically attached on this interface; stripping out command characters, carriage return and line feed characters, etc. The SOEMI interface is available on the IBM 9370, the IBM 4361, and the 3174 Subsystem Control Unit model 1L. The SOEMI Protocol is much faster and more flexible, in that any type of raw data or formatted data may be sent across the connecting coax cable.

The SOEMI attachment into the MULTIBUS system architecture, via the ISBC 561 SOEMI Controller Board, extends the attachment capabilities of the IBM 9370, 4361 and 3174 to a variety of systems, boards, and I/O devices provided by other manufacturers. Figure 1 is an example of the variety achievable on Intel's MULTIBUS (IEEE 796) system architecture.

The SOEMI interface utilizes the System/370 Programming Interface on the IBM 9370, 4361 and 3174 to create the protocols and formats required by a given application for connection to and communication with virtually any type of OEM device.

Figure 1. IBM 4361-to-MULTIBUS® Attachment Capability Block Diagram

*XENIX is a trademark of MICROSOFT Corporation
6 SOEMI ports. This can be increased to 32 ports using optional terminal multiplexers. The connection to the IBM 3174 model 1L is made via IBM dual-purpose connectors (DPC) which can connect up to 4 SOEMI ports. This can be increased to 32 ports using terminal multiplexer adapters. A typical configuration can support an aggregate data rate of approximately 45K Bytes/second (approx. 360K bits/second).

OPERATING ENVIRONMENT

The iSBC board functions as a slave to the host mainframe, reacting and executing under System/370 program control as a mainframe resource. In addition, it has a full multimaster MULTIBUS interface that allows the board to arbitrate for bus ownership, generate bus clocks, respond to and generate interrupts, etc. With the iSBC 561 controller connected to the mainframe, all MULTIBUS system resources are available to the IBM host program/controller. From the IBM side, the mainframe is capable of accessing the entire 16 MBytes of MULTIBUS system memory, 64K Bytes of I/O space, and all on-board resources of the iSBC 561 board. Other intelligent MULTIBUS boards access iSBC 561 controller services through normal interrupt mechanisms.

Using the SOEMI interface in a relatively low-level application may simply require the user to write System/370 application control programs that reside in the IBM mainframe. A more elaborate implementation would also involve application programs that reside in the MULTIBUS system under its "native" op-

Figure 2. ISBC® 561 SOEMI (Serial OEM Interface) Controller Board Functional Block Diagram

8-39
erating environment (i.e., iRMX or XENIX operating systems) and an end-to-end protocol that ties both sets of application programs together.

ARCHITECTURE

The iSBC 561 board is functionally partitioned into three major sections: the front-end section, the common section, and the back-end section (see Figure 2).

Front-End Processor Section: IBM Host Interface

The front-end section of the iSBC 561 Controller board interfaces with the IBM mainframe via the IBM 3270 Information Display System, and consists of an 8X305 Signetics microcontroller, the 8X305 instruction memory, and the coaxial interface. The 8X305 executes the coax commands and places the structured field's instructions in shared memory buffers for subsequent execution by the back-end processor. The front-end instruction memory consists of three 2K x 8-bit PROMs which provide the instruction code for the 8X305 processor and the information needed to generate the various control signals required by the coax to elicit system functions. The information contained in each PROM is not modifiable by the user. The coaxial interface is based on a DP8340 transmitter component that converts 8-bit parallel data received from the front-end processor to a 12-bit serial stream, and a DP8341 receiver component, that converts a 12-bit serial stream of data from the mainframe to parallel data with separated command and parity bits.

Common Section: Shared Memory Buffer

The common section of the iSBC 561 board consists of two 8-bit, bi-directional message registers and a 16K x 8-bit static RAM shared buffer. This shared memory buffer between the front-end processor and the back-end processor is the resource for transferring information and control messages between the IBM host and the MULTIBUS system.

Back-End Processor Section: MULTIBUS® Interface

The back-end section of the board provides an intelligent interface to the MULTIBUS system bus, and consists of the 8086-2 microprocessor, local memory, bus interface circuitry, and memory-mapped logic. The 8086 processor is capable of either retrieving information the 8X305 placed in the shared buffer, or placing information in the shared buffer, depending on the direction of the transfer and type of operation or task to be performed. The information is stored in the shared buffer as a set(s) of structured fields. The back-end processor transfers this information by performing 8- or 16-bit data transfers to or from the MULTIBUS system bus, the shared buffer, and the local memory.

The control program for this high-speed, back-end processor is resident in two local ROM sites. The processor also has access to 16K bytes of static RAM for local data storage.

The back-end section interfaces to other MULTIBUS boards through two bus controllers, a bus arbiter, and the address, data, and command buffers for access over the 24 address lines and 16 data lines of the MULTIBUS system bus.

OPERATION FLOW

The commands and information passed along the coax by the IBM host to the iSBC 561 controller represent what is known as a "structured field." The iSBC 561 front-end processor strips out the 12-bit protocol header deposits the remaining structured field(s) in the shared memory buffer, and notifies the back-end processor. The back-end processor then processes these structured fields in order to access the proper MULTIBUS memory space and I/O ports. It then deposits the information or task in the space and notifies the MULTIBUS subsystem master that a transfer has occurred and is awaiting service.

When requiring service, the MULTIBUS system application sends an interrupt to the iSBC 561 board. The board then issues an attention to the mainframe. At this point, the mainframe is under no obligation or time constraint to service the interrupt, and its response is application dependent.

The mainframe issues commands to service the interrupt. The information concerned with the interrupt is then passed through the shared memory and serialized by the iSBC 561 board before being sent to the mainframe. The exact communications protocol used for this end-to-end transfer is defined by the user application programs running in both operating environments.

Interface Connector/Cable Assembly

The cable assembly used to connect the iSBC 561 SOEMI Controller Board to the IBM mainframe or 3174 control unit cable assembly consists of RG180 type cable having an SMC connector on one end (which mates to the iSBC 561 board right angle SMC connector) and a BNC connector on the other end (which mates to the IBM cable assembly connector).
SPECIFICATIONS

Operational Characteristics
Back-end processor—Intel 8086-2/5 MHz
- 20-bit address path; 8/16 bit data path
Front-end processor—Signetics 8X305/8 MHz
- 16-bit instruction path; 8-bit data path
Serial Transfer Rate—2.3587 Mbits/second (max. bit rate)
- 360K bits/second (approx. aggregate throughput)
Serial Transfer Rate—Binary dipulse (with 12-bit serial stream)
Memory Capacity — All iSBC 561 controller board memory is available to onboard firmware only.
Common memory — 16K Bytes of Shared Buffer memory (SRAM @ 0 wait state access)
8086-2 memory — 16K Bytes of EPROM;
- 16K Bytes of SRAM
8X305 memory — 4K Bytes of Instruction memory (EPROM)
- 2K Bytes of Control memory (EPROM)

Physical Characteristics
Width: 30.48 cm (12.00 in)
Height: 17.15 cm (6.75 in)
Depth: 1.78 cm (0.70 in)
Weight: 510 gm (18 oz)

Electrical Characteristics
DC Power Requirements:
Voltage—+5V
Current (Max)—6.28A
Current (Typ)—5.46A
Power Dissipation (Max)—35.5VA

Cable Characteristics
Impedance: coax connector—50 ohms (nominal)
- external cable (user furnished)—95 ohms (nominal)
Capacitance: 35 pF/ft
Propagation: 1.6 ns/ft

Environmental Characteristics
Operating Temperature: 0° to 55°C at 200 LFM air velocity
Operating Humidity: 10 to 85% non-condensing (0° to 55°C)
Non-Operating Temperature: −40°C to 75°C
Shock: 30G for a duration of 11 ms with ½ sinewave shape.
Vibration: 0 to 55 Hz with 0.0 to 0.010 inches peak to peak excursion.

Reference Manuals
147048-001 — iSBC 561 SOEMI (Serial OEM Interface) Controller Board Hardware Reference Manual (NOT SUPPLIED)

Reference manual may be ordered from any Intel sales representative, distributor office, or from Intel Literature Department, 3065 Bowers Avenue, Santa Clara, California 95051.
GA33-1585-0 (File No. S370-03—IBM Serial OEM Interface (SOEMI) Reference Manual (NOT SUPPLIED)

Reference manual may be ordered from IBM Advanced Technical Systems; Dept. 3291, 7030-16; Schoenaicherstr. 220; 7030 Boeblingen. Federal Republic of Germany.
The Intel iSBX 351 Serial I/O MULTIMODULE board is a member of Intel’s new line of iSBX bus compatible MULTIMODULE products. The iSBX MULTIMODULE board plugs directly into any iSBX bus compatible host board offering incremental on-board I/O expansion. The iSBX 351 module provides one RS232C or RS449/422 programmable synchronous/asynchronous communications channel with software selectable baud rates. Two general purpose programmable 16-bit BCD or binary timers/event counters are available to the host board to generate accurate time intervals under software control. The iSBX board is closely coupled to the host board through the iSBX bus, and as such, offers maximum on-board performance and frees MULTIBUS system traffic for other system resources. In addition, incremental power dissipation is minimal requiring only 3.0 watts (assumes RS232C interface).
FUNCTIONAL DESCRIPTION

Communications Interface
The iSBX 351 module uses the Intel 8251A Universal Synchronous/Asynchronous Receiver/Transmitter (USART) providing one programmable communications channel. The USART can be programmed by the system software to individually select the desired asynchronous or synchronous serial data transmission technique (including IBM Bi-Sync). The mode of operation (i.e., synchronous or asynchronous), data format, control character format, parity, and baud rate are all under program control. The 8251A provides full duplex, double buffered transmit and receive capability. Parity, overrun, and framing error detection are all incorporated in the USART. The command lines, serial data lines, and signal ground lines are brought out to a double edge connector configurable for either an RS232C or RS449/422 interface (see Figure 3). In addition, the iSBX 351 module is jumper configurable for either point-to-point or multidrop network connection.

16-Bit Interval Timers
The iSBX 351 module uses an Intel 8253 Programmable Interval Timer (PIT) providing 3 fully programmable and independent BCD and binary 16-bit interval timers. One timer is available to the system designer to generate baud rates for the USART under software control. Routing for the outputs from the other two counters is jumper selectable to the host board. In utilizing the iSBX 351 module, the systems designer simply configures, via software, each timer independently to meet system requirements. Whenever a given baud rate or time delay is needed, software commands the programmable timers to select the desired function. The functions of the timers are shown in Table 1. The contents of each counter may be read at any time during system operation.

Interrupt Request Lines
Interrupt requests may originate from four sources. Two interrupt requests can be automatically generated by the USART when a character is ready to be transferred to the host board (i.e., receive buffer is full) or a character has been transmitted (i.e., transmit buffer is empty). In addition, two jumper selectable requests can be generated by the programmable timers.

Installation
The iSBX 351 module plugs directly into the female iSBX connector on the host board. The module is then secured at one additional point with nylon hardware to insure the mechanical security of the assembly (see Figures 1 and 2).

<table>
<thead>
<tr>
<th>Function</th>
<th>Operation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Interrupt on</td>
<td>When terminal count is reached, an interrupt</td>
</tr>
<tr>
<td>Terminal Count</td>
<td>request is generated. This function is</td>
</tr>
<tr>
<td></td>
<td>useful for generation of real-time clocks.</td>
</tr>
<tr>
<td>Programmable</td>
<td>Output goes low upon receipt of an external</td>
</tr>
<tr>
<td>One-Shot</td>
<td>trigger edge and returns high when</td>
</tr>
<tr>
<td></td>
<td>terminal count is reached. This function is</td>
</tr>
<tr>
<td></td>
<td>retriggerable.</td>
</tr>
<tr>
<td>Rate Generator</td>
<td>Divide by N counter. The output will go low</td>
</tr>
<tr>
<td></td>
<td>for one input clock cycle, and the period from</td>
</tr>
<tr>
<td></td>
<td>one low going pulse to the next is N times</td>
</tr>
<tr>
<td></td>
<td>the input clock period.</td>
</tr>
<tr>
<td>Square-Wave</td>
<td>Output will remain high until one-half the</td>
</tr>
<tr>
<td>Rate Generator</td>
<td>count has been completed, and go low for the</td>
</tr>
<tr>
<td></td>
<td>other half of the count.</td>
</tr>
<tr>
<td>Software Triggered Strobe</td>
<td>Output remains high until software loads count</td>
</tr>
<tr>
<td></td>
<td>(N). N counts after count is loaded, output</td>
</tr>
<tr>
<td></td>
<td>goes low for one input clock period.</td>
</tr>
<tr>
<td>Hardware Triggered Strobe</td>
<td>Output goes low for one clock</td>
</tr>
<tr>
<td></td>
<td>period N counts after rising edge counter</td>
</tr>
<tr>
<td></td>
<td>trigger input. The counter is retriggerable.</td>
</tr>
<tr>
<td>Event Counter</td>
<td>On a jumper selectable basis, the clock input</td>
</tr>
<tr>
<td></td>
<td>becomes an input from the external system. CPU</td>
</tr>
<tr>
<td></td>
<td>may read the number of events occurring after</td>
</tr>
<tr>
<td></td>
<td>the counting &quot;window&quot; has been enabled or an</td>
</tr>
<tr>
<td></td>
<td>interrupt may be generated after N events</td>
</tr>
<tr>
<td></td>
<td>occur in the system.</td>
</tr>
</tbody>
</table>
Figure 1. Installation of ISBC® 351 Module on a Host Board

Figure 2. Mounting Clearances (Inches)
**Figure 3. Cable Construction and Installation for RS232C and RS449/422 Interface**

## SPECIFICATIONS

### I/O Addressing

<table>
<thead>
<tr>
<th>I/O Address for an 8-Bit Host</th>
<th>I/O Address for a 16-Bit Host</th>
<th>Chip Select</th>
<th>Function</th>
</tr>
</thead>
</table>
| X0, X2, X4 or X6              | Y0, Y4, Y8 or YC              | 8251A USART | Write: Data  
|                               |                               |             | Read: Data |
| X1, X3, X5 or X7              | Y2, Y6, YA or YE              | MCS0/Activated (True) | Write: Mode or Command  
|                               |                               |             | Read: Status |
| X8 or XC                      | Z0 or Z8                      | 8253 PIT    | Write: Counter 0  
|                               |                               |             | Load: Count (N) |
|                               |                               |             | Read: Counter 0 |
| X9 or XD                      | Z2 or ZA                      | MSC1/Activated (True) | Write: Counter 1  
|                               |                               |             | Load: Count N |
|                               |                               |             | Read: Counter 1 |
|XA or XE                       | Z4 or ZC                      |             | Write: Counter 2  
|                               |                               |             | Load: Count (N) |
|                               |                               |             | Read: Counter 2 |
|XB or XF                       | Z6 or ZE                      |             | Write: Control  
|                               |                               |             | Read: None |

**NOTE:**
X = The iSBX base address that activates MCS0 & MSC1 for an 8-bit host.  
Y = The iSBX base address that activates MCS0 for a 16-bit host.  
Z = The iSBX base address that activates MCS1 for a 16-bit host.  
The first digit, X, Y or Z, is always a variable, since it will depend on the type of host microcomputer used. Refer to the Hardware Reference Manual for your host microcomputer to determine the first digit of the I/O base address.  
The first digit of each port I/O address is listed as "X" since it will change depending on the type of host iSBX microcomputer used. Refer to the Hardware Reference Manual for your host iSBX microcomputer to determine the first digit of the I/O address.
Word Size
Data—8 bits

Access Time
Read—250 ns max
Write—300 ns max

NOTE:
Actual transfer speed is dependent upon the cycle time of the host microcomputer.

Serial Communications
Synchronous—5–8-bit characters; internal character synchronization; automatic sync insertion; even, odd or no parity generation/detection.

Asynchronous—5–8-bit characters; break character generation and detection; 1, 1½, or 2 stop bits; false start bit detection; even, odd or no parity generation/detection.

Interval Timer and Baud Rate Generator

Input Frequency (selectable):
1.23 MHz ± 0.1% (0.813 μs period nominal)
153.6 kHz ± 0.1% (6.5 μs period nominal)

Sample Baud Rate

<table>
<thead>
<tr>
<th>8253 PIT(1) Frequency (kHz, Software Selectable)</th>
<th>8251 USART Baud Rate (Hz)(2)</th>
</tr>
</thead>
<tbody>
<tr>
<td>307.2</td>
<td>38400</td>
</tr>
<tr>
<td>153.6</td>
<td>19200</td>
</tr>
<tr>
<td>76.8</td>
<td>9600</td>
</tr>
<tr>
<td>38.4</td>
<td>4800</td>
</tr>
<tr>
<td>19.2</td>
<td>2400</td>
</tr>
<tr>
<td>9.6</td>
<td>1200</td>
</tr>
<tr>
<td>4.8</td>
<td>600</td>
</tr>
<tr>
<td>2.4</td>
<td>300</td>
</tr>
<tr>
<td>1.76</td>
<td>150</td>
</tr>
</tbody>
</table>

NOTES:
1. Frequency selected by I/O writes of appropriate 16-bit frequency factor to Baud Rate Register.
2. Baud rates shown here are only a sample subset of possible software-programmable rates available. Any frequency from 18.75 Hz to 614.4 kHz may be generated utilizing on-board crystal oscillator and 16-bit Programmable Interval Timer (used here as frequency divider).

Output Frequency

<table>
<thead>
<tr>
<th>Rate Generator (Frequency)</th>
<th>Real-Time Interrupt (Interval)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Single Timer(1)</td>
<td>18.75 Hz</td>
</tr>
<tr>
<td>Single Timer(2)</td>
<td>2.34 Hz</td>
</tr>
<tr>
<td>Dual Timer(3) (Counters 0 and 1 in Series)</td>
<td>0.000286 Hz</td>
</tr>
<tr>
<td>Dual Timer(4) (Counters 0 and 1 in Series)</td>
<td>0.0000358 Hz</td>
</tr>
</tbody>
</table>

NOTES:
1. Assuming 1.23 MHz clock input.
2. Assuming 153.6 kHz clock input.
3. Assuming Counter 0 has 1.23 MHz clock input.
4. Assuming Counter 0 has 153.6 kHz clock input.
Interrupts

Interrupt requests may originate from the USART (2) or the programmable timer (2).

Interfaces

iSBX Bus—all signals TTTL compatible.

Serial—configurable of EIA Standards RS232C or RS449/422

EIA Standard RS232C signals provided and supported.

Clear to Send (CTS)
Data Set Ready (DSR)
Data Terminal Ready (DTR)
Request to Send (RTS)
Receive Clock (RXC)
Receive Data (RXD)
Transmit Clock (DTE TXC)
Transmit Data (TXD)

Serial Interface Connectors

<table>
<thead>
<tr>
<th>Configuration</th>
<th>Mode(2)</th>
<th>MULTIMODULE™ Edge Connector</th>
<th>Cable</th>
<th>Connector(8)</th>
</tr>
</thead>
<tbody>
<tr>
<td>RS449</td>
<td>DTE</td>
<td>40-pin(6), 3M-3464-0001</td>
<td>3M(4)-3349/37</td>
<td>37-pin(1), 3M-3502-1000</td>
</tr>
<tr>
<td>RS449</td>
<td>DCE</td>
<td>40-pin(6), 3M-3464-0001</td>
<td>3M(4)-3349/37</td>
<td>37-pin(1), 3M-3503-1000</td>
</tr>
</tbody>
</table>

NOTES:
1. Cable housing 3M-3485-4000 may be used with the connector.
2. DTE—Data Terminal mode (male connector), DCE—Data Set mode (female connector).
3. Cable is tapered at one end to fit the 3M-3462 connector.
4. Cable is tapered to fit 3M-3464 connector.
5. Pin 26 of the edge connector is not connected to the flat cable.
6. Pins 37, 39, and 40 of the edge connector are not connected to the flat cable.
7. May be used with cable housing 3M-3485-1000.
8. Connectors compatible with those listed may also be used.

Electrical Characteristics

DC Power Requirements

<table>
<thead>
<tr>
<th>Mode</th>
<th>Voltage</th>
<th>Amps (Max)</th>
</tr>
</thead>
<tbody>
<tr>
<td>RS232C</td>
<td>+5V ±0.25V</td>
<td>460 mA</td>
</tr>
<tr>
<td></td>
<td>+12V ±0.6V</td>
<td>30 mA</td>
</tr>
<tr>
<td></td>
<td>−12V ±0.6V</td>
<td>30 mA</td>
</tr>
<tr>
<td>RS449/422</td>
<td>+5V ±0.25V</td>
<td>530 mA</td>
</tr>
</tbody>
</table>

Environmental Characteristics

Temperature: 0°C–55°C, free moving air across the base board and MULTIMODULE board.

Reference Manual

9803190-01—ISBX 351 Serial I/O MULTIMODULE Manual (NOT SUPPLIED)

Reference Manuals may be ordered from any Intel sales representative, distributor office or from Intel Literature Department, 3065 Bowers Ave., Santa Clara, California, 95051.

ORDERING INFORMATION

Part Number  Description
SBX 351   Serial I/O MULTIMODULE Board
The Intel iSBX 354 Serial I/O MULTIMODULE board is a member of Intel's line of iSBX compatible MULTIMODULE products. The iSBX MULTIMODULE board plugs directly into any iSBX bus compatible host board offering incremental on-board I/O expansion. Utilizing Intel's 82530 Serial Communications Controller component, the iSBX 354 module provides two RS232C or RS422A/449 programmable synchronous/asynchronous communications channels. The 82530 component provides two independent full duplex serial channels, on-chip crystal oscillator, baud-rate generator and digital phase locked loop capability for each channel. The iSBX board connects to the host board through the iSBX bus. This offers maximum on-board performance and frees the MULTIBUS® System bus for use by other system resources.
**FUNCTIONAL DESCRIPTION**

**Communications Interface**

The iSBX 354 module uses the Intel 82530 Serial Communications Controller (SCC) component providing two independent full duplex serial channels. The 82530 is a multi-protocol data communications peripheral designed to interface high speed communications lines using Asynchronous, Byte-Synchronous and Bit-Synchronous protocols to Intel's microprocessor based board and system level products. The mode of operation (i.e. asynchronous or synchronous), data format, control character format, and baud-rate generation are all under program control. The 82530 SCC component can generate and check CRC codes in any Synchronous mode and can be programmed to check data integrity in various modes. The command lines, serial data lines, and signal ground lines are brought out to a double edge connector.

The iSBX 354 module provides a low cost means to add two serial channels to ISBC® boards with 8 or 16 bit MULTIMODULE interfaces. In the factory default configuration, the iSBX 354 module will support two RS232C interfaces. With user supplied drivers and termination resistors, the iSBX 354 module can be reconfigured to support RS422A/449 communication interfaces with support on Channel A only for multidrop control from the base board. Both channels can be configured as DTE or DCE with RS232C interfaces.

**Interrupt Request Line**

The 82530 SCC component provides one interrupt to the MINTRO signal of the iSBX interface. There are six sources of interrupts in the SCC component (Transmit, Receive and External/Status interrupts in both channels). Each type of interrupt is enabled under program control with Channel A having higher priority than Channel B, and with Receive, Transmit

![Figure 1. Installation of 2 iSBX™ 354 MULTIMODULE™ Boards on an ISBC® Board](image1)

![Figure 2. Mounting Technique](image2)
and External/Status interrupts prioritized in that order within each channel.

**Installation**

The iSBX 354 module plugs directly into the female iSBX connector on the host board. The module is then secured at one additional point with nylon hardware to insure the mechanical security of the assembly. Figures 1 and 2 demonstrate the installation of the iSBX 354 MULTIMODULE board on a Host Board. Figures 3 and 4 provide cabling diagrams.

**Programming Considerations**

The Intel 82530 SCC component contains several registers that must be programmed to initialize and control the two channels. Intel's iSBX 354 Module Hardware Reference Manual (Order #146531-001) describes these registers in detail.

---

**RS232C DB-25 CONNECTORS**

![RS232C Cable Construction](image1)

**RS422A/449 DB-37 CONNECTORS**

![RS422A/449 Cable Construction](image2)
SPECIFICATIONS

Word Size

Data—8 bits

Clock Frequency

4.9152 MHz

Serial Communications

Synchronous—Internal or external character synchronization on one or two synchronous characters

Asynchronous—5–8 bits and 1, 1½ or 2 stop bits per character; programmable clock factor; break detection and generation; parity, overrun, and framing error detection

Sample Baud Rate:

<table>
<thead>
<tr>
<th>Baud Rate</th>
<th>82530 Count Value (Decimal)</th>
</tr>
</thead>
<tbody>
<tr>
<td>64000</td>
<td>36</td>
</tr>
<tr>
<td>48000</td>
<td>49</td>
</tr>
<tr>
<td>19200</td>
<td>126</td>
</tr>
<tr>
<td>9600</td>
<td>254</td>
</tr>
<tr>
<td>4800</td>
<td>510</td>
</tr>
<tr>
<td>2400</td>
<td>1022</td>
</tr>
<tr>
<td>1800</td>
<td>1363</td>
</tr>
<tr>
<td>1200</td>
<td>2046</td>
</tr>
<tr>
<td>300</td>
<td>8190</td>
</tr>
</tbody>
</table>

Asynchronous X16 Clock

<table>
<thead>
<tr>
<th>Baud Rate</th>
<th>82530 Count Value (Decimal)</th>
</tr>
</thead>
<tbody>
<tr>
<td>19200</td>
<td>6</td>
</tr>
<tr>
<td>9600</td>
<td>14</td>
</tr>
<tr>
<td>4800</td>
<td>30</td>
</tr>
<tr>
<td>2400</td>
<td>62</td>
</tr>
<tr>
<td>1800</td>
<td>83</td>
</tr>
<tr>
<td>1200</td>
<td>126</td>
</tr>
<tr>
<td>300</td>
<td>510</td>
</tr>
<tr>
<td>110</td>
<td>1394</td>
</tr>
</tbody>
</table>

Signals Provided

RS232C DTE
- Transmit Data
- Receive Data
- Request to Send
- Clear to Send
- Data Set Ready
- Signal Ground
- Carrier Detect
- Transmit Clock (2)
- Receive Clock
- Data Terminal Ready
- Ring Indicator

RS232C DCE
- Transmit Data
- Receive Data
- Clear to Send
- Data Set Ready
- Signal Ground
- Carrier Detect
- Transmit Clock (2)
- Receive Clock
- Ring Indicator

RS422A/449
- Send Data
- Receive Timing
- Receive Data
- Terminal Timing
- Receive Common

I/O Port Addresses

<table>
<thead>
<tr>
<th>Port Address</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>X0</td>
<td>Read Status Channel B</td>
</tr>
<tr>
<td></td>
<td>Write Command Channel B</td>
</tr>
<tr>
<td>X2</td>
<td>Read Data Channel B</td>
</tr>
<tr>
<td></td>
<td>Write Data Channel B</td>
</tr>
<tr>
<td>X4</td>
<td>Read Status Channel A</td>
</tr>
<tr>
<td></td>
<td>Write Command Channel A</td>
</tr>
<tr>
<td>X6</td>
<td>Read Data Channel A</td>
</tr>
<tr>
<td></td>
<td>Write Data Channel A</td>
</tr>
<tr>
<td>Y0</td>
<td>Read Disable RS422A/449 Buffer</td>
</tr>
<tr>
<td></td>
<td>Write Enable RS422A/449 Buffer</td>
</tr>
</tbody>
</table>

NOTES:
1. The "X" and "Y" values depend on the address of the ISBX interface as viewed by the base board.
2. "X" corresponds with Activation of the MCS0/interface signal; "Y" corresponds with Activation of the MCS1/interface signal.

Power Requirements

+5V at 0.5A
+12V at 50 mA
-12V at 50 mA

Physical Characteristics

Width: 2.85 inches
Length: 3.70 inches
Height: 0.8 inches
Weight: 85 grams
ENVIRONMENTAL CHARACTERISTICS
Temperature: 0°C to 55°C operating at 200 linear feet per minute across baseboard and MULTIMODULE board
Humidity: To 90%, without condensation

ORDERING INFORMATION
Part Number Description
iSBX 354 Dual Channel I/O MULTIMODULE

REFERENCE MANUAL
146531-001—iSBX 354 Channel Serial I/O Board Hardware Reference Manual
Reference manuals may be ordered from any Intel sales representative, distributor office, or from Intel Literature Department, 3065 Bowers Avenue, Santa Clara, CA 95051.
iSBC® 186/410 MULTIBUS® II SERIAL COMMUNICATIONS COMPUTER

- Six Serial Communication Channels on a Single MULTIBUS® II Board, Expandable to 10 Channels via iSBXTM Bus Connectors
- High Integration 8 MHz 80186 Microprocessor
- 82258 Advanced DMA Controller Provides 4 Independent High Performance DMA Channels
- Supports RS232C-Only on 4 Channels, RS422A or RS232C Interface Configurable on 2 Channels
- 512K Bytes DRAM Provided
- MULTIBUS® II IPSB (Parallel System Bus) Interface with Full Message Passing Capability
- Four 28-Pin JEDEC Sites, Expandable to 8 Sites with iSBC® 341 MULTIMODULE™ for a Maximum of 512K Bytes EPROM
- Two iSBXTM Connectors for Low Cost I/O Expansion
- MULTIBUS® II Interconnect Space for Software Configurability and Diagnostics
- Resident Firmware to Support Host-to-Controller Download Capability and Built-In-Self-Test (BIST) Diagnostics

The iSBC 186/410 MULTIBUS II Serial Communications Computer is an intelligent 6-channel communications processor implementing the full, high performance message passing interface of the MULTIBUS II (IPSB) Parallel System Bus. This iSBC board combines an 8 MHz 80186 16-bit microprocessor, with six serial channels (expandable to 10 serial channels on-board via iSBX connectors), up to 512K bytes of DRAM, four 28-pin JEDEC sites, two iSBX connectors, and an 82258 ADMA controller on a single 220 mm x 233 mm (8.7 in. x 9.2 in.) Eurocard printed circuit board. The iSBC 186/410 board supports asynchronous, byte synchronous, and bit-synchronous (HDLC/SDLC) communications protocols on the two full/half duplex RS232C/RS422A channels, and asynchronous-only on the four full/half duplex RS232C-only channels. Acting as a terminal controller or front-end processor, this board adds significant data communications flexibility to an OEM's MULTIBUS II design.
OPERATING ENVIRONMENT

The iSBC 186/410 MULTIBUS II Serial Communications Computer is a powerful data communications sub-system specifically designed to operate in and support the message-based, multi-processor system configurations being implemented on the MULTIBUS II architecture. The board's on-board CPU, an 8 MHz 80186 microprocessor, provides significant intelligence to off-load and distribute the serial communications functions away from one or all of a system's processor boards.

The iSBC 186/410 board was designed with a set of features to address several communications application areas: terminal/cluster controller, or front-end processor.

Terminal/Cluster Controller

A terminal/cluster controller concentrates communications in a central area of a system. Efficient handling of messages coming in or going out of the system requires sufficient buffer space to store messages along with high speed I/O channels to transmit and receive those messages. Sophisticated cluster controller applications also require character and format conversion capabilities to allow attachment of different types of terminals.

The iSBC 186/410 MULTIBUS II Serial Communications Computer is well suited for multi-terminal system applications (see Figure 1). Up to 10 serial channels can be serviced in multi-user or cluster configurations by adding two iSBX 354 Dual Serial Channel MULTIMODULE boards. The on-board 512K byte (expandable to 512K bytes) DRAM array is the buffer area designed to handle incoming and outgoing messages at data rates up to 19.2K baud (asynch). Each serial channel can be individually programmed for different baud rates to allow system configurations with differing terminal types. The on-board 80186 CPU handles the protocols and character manipulation tasks traditionally performed by a system host.

Front-end Processor

A front-end processor off-loads a system's central processor of bandwidth-draining tasks such as data manipulation and text editing of characters collected from the attached serial I/O devices. Since most ter-

Figure 1. Terminal/Cluster Controller Application
minal and serial I/O devices require flexible interfaces, program code is often dynamically downloaded to the front-end processor from a system CPU. Downloading code requires sufficient memory space for protocol handling and program code. Flow control and interrupt handling requirements need an efficient real-time operating system environment to manage the hardware and software resources on the board.

The iSBC 186/410 board features are designed to provide a high performance solution for front-end processor applications (see Figure 2). A large amount of memory is provided for dynamic storage of program code. Two serial channels (as well as four ISBX expansion serial channels) can be configured for links to mainframe systems, point-to-point terminals, modems or multi-drop designs and four serial channels are for terminal communication, asynchronous RS232C operation only.

**ARCHITECTURE**

The iSBC 186/410 MULTIBUS II Serial Communications Computer consists of six major subsystem areas: Processor, Serial I/O, Memory, General I/O, iPSB bus interface, and Interconnect (see Figure 3).

**Processor Subsystem**

**80186 PROCESSOR**

The central processor unit on the iSBC 186/410 board is Intel’s 16-bit 8 MHz 80186 microprocessor. The highly integrated 80186 CPU combines several system components onto a single chip (i.e., two Direct Memory Access lines, three Interval Timers, Clock Generator, and Programmable Interrupt Controller). The 80186 instruction set is a superset of the 8086 and maintains object code compatibility while adding additional instructions.

This high performance component manages the board’s multi-user, multi-protocol communications operations. Refer to the Microsystem Components Handbook, Order Number 230843-00X, for more detailed information on the hardware operation and requirements of the 80186 microprocessor component.
Figure 3. ISBC® 186/410 Board Functional Block Diagram
DIRECT MEMORY ACCESS (DMA) FUNCTION

The ISBC 186/410 board provides 13 channels of DMA to support serial I/O, iPSB interface, and/or ISBX bus transfer operations. The 80186 microprocessor provides two DMA channels, the 82258 Advanced (ADMA) controller supports three "direct" channels of DMA, and the ADMA multiplexer circuit uses the fourth 82258 ADMA channel providing eight additional multiplexed DMA channels. The allocation of the board's DMA channels to on-board resources is listed in Table 1.

SERIAL I/O SUBSYSTEM

Six serial interfaces are provided on the ISBC 186/410 board: two interfaces support full asynchronous, byte-synchronous, and bit-synchronous (HDLC/SDLC) communication and four interfaces support asynchronous-only communication. The two RS422A configurable ports can also be tri-stated to allow multi-drop networks. The board's serial capability can be expanded to 10 channels by adding two ISBX 354 Dual Channel Serial I/O MULTIMODULE boards. Each added ISBX 354 board uses an

<table>
<thead>
<tr>
<th>Channel Count</th>
<th>Channel Number</th>
<th>DMA Configuration Local Bus Resource</th>
</tr>
</thead>
<tbody>
<tr>
<td>80186</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1 DMA Channel</td>
<td>0</td>
<td>Half-Duplex High Speed Serial Interface (SCC1 Channel A) (High Density 15-Pin Connector)</td>
</tr>
<tr>
<td>2 DMA Channel</td>
<td>1</td>
<td>Full-Duplex Serial Interface (SCC1 Channel A) or SBX1 DMA Request</td>
</tr>
<tr>
<td>82258 ADMA</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3 DMA Channel</td>
<td>0</td>
<td>Input DMA from MPC (Message Passing Coprocessor)</td>
</tr>
<tr>
<td>4 DMA Channel</td>
<td>1</td>
<td>Output DMA to MPC</td>
</tr>
<tr>
<td>5 DMA Channel</td>
<td>2</td>
<td>Half-Duplex High Speed Serial Interface (SCC1 Channel B) (High Density 15-Pin Connector) or SBX1 DMA REQ</td>
</tr>
<tr>
<td>DMA Channel</td>
<td>3</td>
<td>Full-Duplex High Speed Serial Interface (SCC1 Channel B) or INT2 DMA REQ from DMA Multiplexer</td>
</tr>
<tr>
<td>DMA Multiplexer*</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6 DMA Channel</td>
<td>0</td>
<td>Half-Duplex Serial Interface (SCC2 Chan. A, 9-pin conn.)</td>
</tr>
<tr>
<td>7 DMA Channel</td>
<td>1</td>
<td>Full-Duplex Serial Interface (SCC2 Chan. A)</td>
</tr>
<tr>
<td>8 DMA Channel</td>
<td>2</td>
<td>Half-Duplex Serial Interface (SCC2 Chan. B, 9-pin conn.)</td>
</tr>
<tr>
<td>9 DMA Channel</td>
<td>3</td>
<td>Full-Duplex Serial Interface (SCC2 Chan. B) or SBX1 DMA Request or Half-Duplex SCC1 Channel B.</td>
</tr>
<tr>
<td>10 DMA Channel</td>
<td>4</td>
<td>Half-Duplex Serial Interface (SCC3 Chan. A, 9-pin conn.)</td>
</tr>
<tr>
<td>11 DMA Channel</td>
<td>5</td>
<td>Full-Duplex Serial Interface (SCC3 Chan. A) or SBX2 DMA Request</td>
</tr>
<tr>
<td>12 DMA Channel</td>
<td>6</td>
<td>Half-Duplex Serial Interface (SCC3 Chan. B, 9-pin conn.)</td>
</tr>
<tr>
<td>13 DMA Channel</td>
<td>7</td>
<td>Full-Duplex Serial Interface (SCC3 Chan. B) or INT1 SBX1 for SBX344</td>
</tr>
</tbody>
</table>

NOTE:
*ADMA Channel 3 is used to add the DMA Multiplexer.
82530 SCC component to provide two independent full duplex serial channels configurable as either RS232C or RS422A interfaces. It also supports both asynchronous or programmable byte and bit synchronous (HDLC/SDLC) protocols. The HDLC/SDLC interface is compatible with IBM system and terminal equipment and with CCITT's X.25 packet switching interface.

Three 82530 Serial Communications Controllers (SCCs) provide six channels of half/full serial I/O. Two channels are configurable as either RS232C or RS422 on two high density 15-pin female D-shell connectors. Four more channels are RS232C-only using IBM standard 9-pin male D-shell connectors. All six channels directly support the Data Terminal Equipment (DTE) configuration, with the Data Communication Equipment (DCE) pin-out supported by changes in the cable wiring.

The 82530 component is designed to satisfy several serial communications requirements; asynchronous, byte-synchronous, and bit-synchronous (HDLC/SDLC) modes. The increased capability at the serial controller point results in off-loading a CPU of tasks normally assigned to the CPU or its associated hardware. Configurability of the 82530 allows the user to configure it to handle all asynchronous data formats regardless of data size, number of start or stop bits, or parity requirements. An on-chip baud rate generator allows independent baud rates on each channel.

Memory Subsystem

The ISBC 186/410 board's on-board memory subsystem consists of a large DRAM array and a set of universal memory sites. Access to the on-board memory subsystem resources, as well as off-board IPSB bus access, is accomplished by observing the ISBC 186/410 board memory map (see Figure 4). The mapping occurs within the 1 megabyte memory space of the 80186 microprocessor, and is split into three main areas: DRAM reserved, IPSB window, and EPROM reserved. The first 0 to 512K bytes is always reserved for local DRAM, the next 128K or

![Figure 4. ISBC® 186/410 Board Memory Map Diagram](image-url)
256K bytes (or up to 768K) is the iSPB window, and the remaining 384K or 256K byte area is reserved for local EPROM. The iSPB window maps a 128K or 256K byte local memory area into the 4 gigabyte global physical address range of the MULTIBUS II iPSB bus. This window is programmable and allows the 80186 processor to access the complete 4 gigabyte memory space of the iPSB bus.

The board's memory map also supports a 64K byte access window for I/O space between local and iPSB bus access. The 64K bytes of local I/O space is mapped 1-to-1 to the iPSB bus' 64K byte I/O space and is not programmable. The upper 32K bytes access the iPSB bus I/O space, and the lower 32K bytes are reserved for local on-board I/O.

**DRAM CAPABILITIES**

The iSBC 186/410 board comes standard with a 512K byte DRAM memory array on-board.

**EPROM MEMORY**

A total of four 28-pin JEDEC universal sites reside on the iSBC 186/410 board. These sockets support addition of byte-wide ROM and EPROM devices in densities from 8K bytes (2764) to 64K bytes (27512) per device. Two of the four sockets contain a pair of 27812 EPROM devices installed at the factory\(^1\). These devices contain 128K bytes of firmware providing both the Host-to-controller download routine and the Built-In-Self-Test (BIST) power-up diagnostics routine. The remaining two sockets allow the user to add either two additional devices or an iSBC 341 EPROM MULTIMODULE for a maximum of 512K bytes.

**NOTE:**

(1) These devices may be removed by the user for access to the two 28-pin sites.

**General I/O Subsystem**

The I/O subsystem provides timers, interrupt control and two IEEE P959 iSBX connectors for I/O expansion or customization.

**PROGRAMMABLE TIMERS AND INTERRUPT CONTROL**

The 80186 microprocessor on the iSBC 186/410 board provides three independent, fully programmable 16-bit interval timers/event counters for use by the systems designer to generate accurate time intervals under software control. The outputs may be independently routed to a PIC to count external events. The system software configures each timer independently and can read the contents of each counter at any time during system operation.

In a MULTIBUS II system, external interrupts (interrupts originating from off-board) are interrupt type messages over the iPSB bus rather than signals on individual lines. Interrupt type messages are handled by the bus interface logic, the MPC Message Passing Coprocessor chip. The MPC component interrupts the 80186 processor via an 8259A Programmable Interrupt Controller (PIC) indicating a message has been received. This means that 1 Interrupt line can handle interrupts from up to 255 sources.

Two on-board 8259A PICs are used in a master-slave configuration for processing on-board interrupts. One of the interrupt lines handles the interrupt messages received from the iPSB bus. Table 2 includes a list of devices and functions supported.

**iSBX™ BUS I/O EXPANSION**

Two 8/16-bit iSBX bus (IEEE P959) connectors are provided for modular, low-cost I/O expansion. The iSBC 186/410 board supports both 8-bit and 16-bit iSBX MULTIMODULEs through these mating, gas-tight pins and socket connectors. DMA is also supported to the iSBX connectors and can be configured by programming the DMA multiplexor attached to the 82258 ADMA component. The iSBX connectors on the iSBC 186/410 board support a wide variety of standard iSBX compatible boards from Intel and other independent vendors providing add-on functions such as, floppy control, ¼” tape control, bubble memory, parallel/serial I/O, BITBUS™ interface, math, graphics, IEEE 488, and analog I/O. Custom iSBX module designs are also supported as per the IEEE P959 iSBX bus specification.

**iPSB Bus Interface Subsystem**

This subsystem's main component is the Message Passing Coprocessor chip. Subsystem services provided by the MPC bus interface component include full message passing support and memory, I/O, and interconnect access to the iPSB bus by the 80186 processor. The single-chip Message Passing Coprocessor is a highly integrated CMOS device implementing the full message passing protocol and performing all the arbitration, transfer, and exception cycle protocols specified in the MULTIBUS II Architecture Specification Rev. C., Order Number 146077.
Table 2. ISBC® 186/410 Board Interrupt Devices and Functions

<table>
<thead>
<tr>
<th>Device</th>
<th>Function</th>
<th>Number of Interrupts</th>
</tr>
</thead>
<tbody>
<tr>
<td>iPSB Bus Interface (MPC)</td>
<td>Message-Based Interrupt Requests from the iPSB bus via MPC Message Passing Coprocessor</td>
<td>1 interrupt for up to 255 sources</td>
</tr>
<tr>
<td>8751 Interconnect Controller</td>
<td>Interconnect Space</td>
<td>1</td>
</tr>
<tr>
<td>80186 Timers &amp; Interrupt</td>
<td>Timers 0 and 1 and Interrupt Acknowledge</td>
<td>3</td>
</tr>
<tr>
<td>82530 SCCs (3 devices)</td>
<td>SCC #1 and SCC #2 or SCC #3 for Transmit Buffer Empty, Receive Buffer Full, and Channel Errors</td>
<td>2</td>
</tr>
<tr>
<td>iPSB Bus Interface (MPC)</td>
<td>Indicates Transmission Error on iPSB Bus</td>
<td>1</td>
</tr>
<tr>
<td>82258 ADMA</td>
<td>DMA Transfer Complete</td>
<td>1</td>
</tr>
<tr>
<td>IEEE P959 iSBX Bus Connectors (2)</td>
<td>Functions Determined by iSBX Bus MULTIMODULE Boards</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(2/connector)</td>
</tr>
<tr>
<td>IEEE P959 iSBX Bus Connectors (2)</td>
<td>DMA Interrupt from iSBX (TDMA)</td>
<td>2</td>
</tr>
</tbody>
</table>

Interconnect Subsystem

MULTIBUS II interconnect space is a standardized set of software configurable registers designed to hold and control board configuration information as well as system and board level diagnostics and testing information. Interconnect space is implemented with the 8751 microcontroller and the MPC silicon resident on the ISBC 186/410 board.

The read-only registers store information such as board type, vendor I.D., firmware rev. level, etc. The software configurable registers are used for auto-software configurability and remote/local diagnostics and testing.

Firmware Capability

HOST/CONTROLLER SOFTWARE DOWNLOAD ROUTINE

Resident in ROM on this controller is a host-to-controller software download routine to support the downloading of communication firmware into the ISBC 186/410 Serial Communication Computer. This loader adheres to the MULTIBUS II Download Protocol and responds to commands issued by software running on a host CPU board. The host CPU passes these commands to the loader via registers defined in the board's interconnect space. A download function, a commence execution function, and an examine local memory function are all provided in the routine. Data transfers are supported by both shared memory systems and message based systems. The top 1K of DRAM on the board is reserved for the exclusive use of the download program. Host CPUs must not overwrite this area with download commands.

Software on the host is responsible for accessing the ISBC 186/410 board's firmware on disk or from ROM visible to the host and translating it into linear sequences of bytes suitable for downloading (see Figure 5). After downloading the firmware, the host issues a command for the loader routine on the controller to begin execution of the downloaded software.

BUILT-IN SELF-TEST DIAGNOSTICS

On-board built-in self-test (BIST) diagnostics provide a customer confidence test of the various functional areas on the ISBC 186/410 board. The initialization checks are performed by the 8751 microcontroller, while the BIST package is executed by the 80186 microprocessor. On-board tests included in the BIST package are: DRAM, EPROM, 80186, 82530 SCCs, and the MPC.

Additional activities performed include initialization at power-up using the Initialization and Diagnostics Executive and a program table check, a feature allowing users to add custom code in EPROM while still maintaining full use of factory supplied BISTs. Immediately after power-up and initialization of the 8751 microcontroller, the 80186 microprocessor begins its own initialization and on-board diagnostics. Upon successful completion of these activities, the Initialization and Diagnostics Executive invokes the user-defined program table. A check is made of the program table which then executes user-defined custom programs.
The BIST package provides a valuable testing, error reporting and recovery capability on MULTIBUS II boards enabling the OEM to reduce manufacturing and maintenance costs. An LED on the board's front panel indicates the status of power-up diagnostics. It is on when BIST diagnostics start running and is turned off upon successful completion of the BISTs.

**SPECIFICATIONS**

**Word Size**
Instruction: 8-, 16-, 24-, 32-, 40-, or 48-bits  
Data: 8- or 16-bits

**System Clock**
CPU: 8.0 MHz

**Cycle Time**
Basic Instruction: 8.0 MHz—500 ns

---

**NOTE:**  
Basic instruction cycle is defined as the fastest instruction time (i.e., 4 clock cycles).

**Memory Capacity**

**Local Memory**
DRAM—512K bytes on-board (64K x 4-bit devices); 8 sockets provided to support additional 256K bytes

EPROM—Number of sockets—four 28-pin JEDEC sites

<table>
<thead>
<tr>
<th>EPROM</th>
<th>Device Size (Bytes)</th>
<th>Max. Memory Capacity</th>
</tr>
</thead>
<tbody>
<tr>
<td>2764</td>
<td>8K</td>
<td>32K bytes</td>
</tr>
<tr>
<td>27128</td>
<td>16K</td>
<td>64K bytes</td>
</tr>
<tr>
<td>27256</td>
<td>32K</td>
<td>128K bytes</td>
</tr>
<tr>
<td>27512</td>
<td>64K</td>
<td>256K bytes</td>
</tr>
</tbody>
</table>

**NOTE:**  
**EPROM** Expansion to up to a maximum of 512K bytes is achieved via attachment of the iSBC 341 EPROM (256K byte) MULTIMODULE board.

**I/O Capability**

Serial—Six programmable serial channels using three 82530 Serial Communications Controller components.
I/O Expansion—Two 8/16-bit IEEE P959 ISBX connectors (DMA supported). (The board supports either two single wide or one double-wide form factor ISBX module(s).)

Timers—Three programmable timers on the 80186 microprocessor.

Input Frequencies—Frequencies supplied by the internal 80186 16 MHz crystal; 82530 SCCs: crystal driven at 9.8304 MHz div. by two; ISBX Connector: crystal driven at 9.8304 MHz.

**Serial Communications Characteristics**

Synchronous—Internal or external character synchronization on one or two synchronous characters.

Asynchronous—5—8 data bits and 1, 1½ or 2 stop bits per character; programmable clock factor; break detection and generation; parity, overrun, and framing error detection.

### Baud Rates

#### Synchronous X1 Clock

(Channels 0, 1)

<table>
<thead>
<tr>
<th>Baud Rate</th>
<th>82530 Count Value (Decimal)</th>
</tr>
</thead>
<tbody>
<tr>
<td>64000</td>
<td>36</td>
</tr>
<tr>
<td>48000</td>
<td>49</td>
</tr>
<tr>
<td>19200</td>
<td>126</td>
</tr>
<tr>
<td>9600</td>
<td>254</td>
</tr>
<tr>
<td>4800</td>
<td>510</td>
</tr>
<tr>
<td>2400</td>
<td>1022</td>
</tr>
<tr>
<td>1800</td>
<td>1363</td>
</tr>
<tr>
<td>1200</td>
<td>2046</td>
</tr>
<tr>
<td>300</td>
<td>8190</td>
</tr>
</tbody>
</table>

#### Asynchronous X16 Clock

(Channels 0–5)

<table>
<thead>
<tr>
<th>Baud Rate</th>
<th>82530 Count Value (Decimal)</th>
</tr>
</thead>
<tbody>
<tr>
<td>19200</td>
<td>6</td>
</tr>
<tr>
<td>9600</td>
<td>14</td>
</tr>
<tr>
<td>4800</td>
<td>30</td>
</tr>
<tr>
<td>2400</td>
<td>62</td>
</tr>
<tr>
<td>1800</td>
<td>83</td>
</tr>
<tr>
<td>1200</td>
<td>126</td>
</tr>
<tr>
<td>300</td>
<td>510</td>
</tr>
<tr>
<td>110</td>
<td>1394</td>
</tr>
</tbody>
</table>

### Serial Signals/Pin-Outs

#### RS232C Interface Pin Assignment for High Density 15-Pin Connectors

<table>
<thead>
<tr>
<th>J2 Pin</th>
<th>RS-232C Pin Number</th>
<th>RS-232C Signal Name</th>
<th>RS-232C Signal Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>TXD</td>
<td>Transmit Data</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>RTS</td>
<td>Request To Send</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>RXD</td>
<td>Receive Data</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>CTS</td>
<td>Clear To Send</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>RXC</td>
<td>Receive Clock</td>
</tr>
<tr>
<td>6</td>
<td>6</td>
<td>DSS</td>
<td>Data Signal Select</td>
</tr>
<tr>
<td>7</td>
<td>7</td>
<td>DTR</td>
<td>Data Terminal Ready</td>
</tr>
<tr>
<td>8</td>
<td>8</td>
<td>DSR</td>
<td>Data Set Ready</td>
</tr>
<tr>
<td>9</td>
<td>9</td>
<td>DCD</td>
<td>Carrier Detect</td>
</tr>
<tr>
<td>10</td>
<td>10</td>
<td>STXC</td>
<td>Transmit Clock</td>
</tr>
<tr>
<td>11</td>
<td>11</td>
<td>SGD</td>
<td>Signal Ground</td>
</tr>
<tr>
<td>12</td>
<td>12</td>
<td>LCLPBK</td>
<td>Local Loopback</td>
</tr>
<tr>
<td>13</td>
<td>13</td>
<td>RMLPBK</td>
<td>Remote Loopback</td>
</tr>
<tr>
<td>14</td>
<td>14</td>
<td>TSTMD</td>
<td>Test Mode Indicator</td>
</tr>
<tr>
<td>15</td>
<td>15</td>
<td>RNG</td>
<td>Not Supported</td>
</tr>
</tbody>
</table>
RS422A Interface Pin Assignment for High Density 15-Pin Connectors

<table>
<thead>
<tr>
<th>J1 Pin</th>
<th>Signal Name On Board</th>
<th>RS-422A Signal Name</th>
<th>RS-422A Signal Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>RS42211</td>
<td>TR (a)</td>
<td>Transmit Data</td>
</tr>
<tr>
<td>2</td>
<td></td>
<td>(a)</td>
<td>Control</td>
</tr>
<tr>
<td>3</td>
<td>RS4229</td>
<td>RD (a)</td>
<td>Receive Data</td>
</tr>
<tr>
<td>4</td>
<td></td>
<td>(a)</td>
<td>Indication</td>
</tr>
<tr>
<td>5</td>
<td></td>
<td>(a)</td>
<td>Signal Timing</td>
</tr>
<tr>
<td>6</td>
<td>RS42212</td>
<td>TR (b)</td>
<td>Transmit Data</td>
</tr>
<tr>
<td>7</td>
<td></td>
<td>(b)</td>
<td>Control</td>
</tr>
<tr>
<td>8</td>
<td>RS42290</td>
<td>RD (b)</td>
<td>Receive Data</td>
</tr>
<tr>
<td>9</td>
<td></td>
<td>(b)</td>
<td>Indication</td>
</tr>
<tr>
<td>10</td>
<td></td>
<td>(b)</td>
<td>Signal Timing</td>
</tr>
<tr>
<td>11</td>
<td></td>
<td>Not Used</td>
<td>Signal Ground</td>
</tr>
<tr>
<td>12</td>
<td></td>
<td>Not Used</td>
<td>Not Used</td>
</tr>
<tr>
<td>13</td>
<td></td>
<td>Not Used</td>
<td>Not Used</td>
</tr>
<tr>
<td>14</td>
<td></td>
<td></td>
<td>Chassis Ground</td>
</tr>
<tr>
<td>15</td>
<td></td>
<td>Signal Timing</td>
<td></td>
</tr>
</tbody>
</table>

**NOTE:**
The iSBC® 186/40 board does not support the unused signals.

RS232C Interface Pin Assignment for IBM® Compatible 9-Pin Connectors

<table>
<thead>
<tr>
<th>Pin Number</th>
<th>Signal Name</th>
<th>Function</th>
<th>In/Out</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>CD</td>
<td>Carrier Detect</td>
<td>In</td>
</tr>
<tr>
<td>2</td>
<td>RXD</td>
<td>Received Data</td>
<td>In</td>
</tr>
<tr>
<td>3</td>
<td>TXD</td>
<td>Transmit Data</td>
<td>Out</td>
</tr>
<tr>
<td>4</td>
<td>DTR</td>
<td>Data Terminal Ready</td>
<td>Out</td>
</tr>
<tr>
<td>5</td>
<td>SG</td>
<td>Signal Ground</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>DSR</td>
<td>Data Set Ready</td>
<td>In</td>
</tr>
<tr>
<td>7</td>
<td>RTS</td>
<td>Request To Send</td>
<td>Out</td>
</tr>
<tr>
<td>8</td>
<td>CTS</td>
<td>Clear To Send</td>
<td>In</td>
</tr>
<tr>
<td>9</td>
<td>RI</td>
<td>Ring Indicator</td>
<td>Not Supported</td>
</tr>
</tbody>
</table>

**Interrupt Capability**

Potential Interrupt Sources from iPSB Bus—255 individual and 1 Broadcast

Interrupt Levels—12 vectored requests using two 8259As and 1 input to the master PIC from the slave PIC

Interrupt Requests—All levels TTL compatible

**Interfaces**

iPSB Bus—Compliance Level RQA/RPA D16M32

iSBX Bus—Compliance Level D8/16 DMA

Serial I/O—2 ch. RS232C or RS422A compatible, configured DTE only; 4 ch. RS232C IBM compatible only, configured DTE only.

**Connectors**

<table>
<thead>
<tr>
<th>Interface</th>
<th>Connector</th>
<th>Part #</th>
</tr>
</thead>
<tbody>
<tr>
<td>iPSB bus (P1)</td>
<td>96-pin DIN, right angle female</td>
<td>603-2-IEC-C096-F</td>
</tr>
<tr>
<td>RS232C/RS422A</td>
<td>15-pin high density, D type, right angle female (see note)</td>
<td></td>
</tr>
<tr>
<td>RS232C-only</td>
<td>9-pin IBM compatible, D type, right angle male (see note)</td>
<td></td>
</tr>
</tbody>
</table>

**NOTE:**
The manufacturers below provide connectors which will mate with the connectors supplied on the iSBC 186/410 board front-panel.
iSBC® 186/410

Mating Connectors, Shells and Cables

<table>
<thead>
<tr>
<th>Connectors and Shells</th>
<th>Manufacturer</th>
<th>Pins</th>
<th>Part No.</th>
</tr>
</thead>
<tbody>
<tr>
<td>High Density D-type Plug (male)</td>
<td>AMP</td>
<td>15</td>
<td>204501-1</td>
</tr>
<tr>
<td>High Density D-type Plug (male)</td>
<td>Positronic</td>
<td>15</td>
<td>DD-15M</td>
</tr>
<tr>
<td>D-type Receptacle (female)</td>
<td>AMP</td>
<td>9</td>
<td>205203-3</td>
</tr>
<tr>
<td>D-type Receptacle (female)</td>
<td>ITT-Cannon</td>
<td>9</td>
<td>DE-9S</td>
</tr>
<tr>
<td>Connector Shells</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>High Density D-type Plug (male)</td>
<td>AMP</td>
<td></td>
<td></td>
</tr>
<tr>
<td>D-type Receptacle (female)</td>
<td>ITT-Cannon</td>
<td></td>
<td></td>
</tr>
<tr>
<td>9-pin connect.</td>
<td>3M,</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Cable Description</th>
<th>Manufacturer</th>
<th>Part No.</th>
</tr>
</thead>
<tbody>
<tr>
<td>15 Conductor—Shield, Round</td>
<td>Alpha</td>
<td>5120/15</td>
</tr>
<tr>
<td>15 Conductor—Shield, Round</td>
<td>Beldon</td>
<td>9541</td>
</tr>
<tr>
<td>10 Conductor—Shield, Round</td>
<td>Alpha</td>
<td>5120/10</td>
</tr>
<tr>
<td>9 Conductor—Shield, Round</td>
<td>Beldon</td>
<td>9539</td>
</tr>
</tbody>
</table>

NOTE: All cable referenced is available in 100 ft. minimum lengths.

PHYSICAL DIMENSIONS
The iSBC 186/410 board meets all MULTIBUS II mechanical specifications as presented in the MULTIBUS II Architecture Specification Handbook (#146077, Rev. C)

Eurocard Form Factor
Depth: 220 mm (8.7 inches)
Height: 233 mm (9.2 inches)
Front Panel Width: 20 mm (0.76 inches)
Weight: 822 gm (29 ounces)

ENVIRONMENTAL CHARACTERISTICS

Temperature
Inlet air at 200 LFM airflow over all boards
Non-operating: −40°C to +75°C
Operating: 0°C to +55°C

Humidity
Non-operating—95% Relative Humidity @ +55°C, non-condensing
Operating—90% Relative Humidity @ +55°C, non-condensing

ELECTRICAL CHARACTERISTICS
The maximum power required per voltage is shown below.

<table>
<thead>
<tr>
<th>Voltage</th>
<th>Max. Current</th>
<th>Max. Power</th>
</tr>
</thead>
<tbody>
<tr>
<td>+5V</td>
<td>8.22A</td>
<td>43.16W</td>
</tr>
<tr>
<td>+12V</td>
<td>150 mA</td>
<td>1.89W</td>
</tr>
<tr>
<td>−12V</td>
<td>150 mA</td>
<td>1.89W</td>
</tr>
</tbody>
</table>

REFERENCE MANUALS
iSBC 186/410 Serial Communications Computer User's Guide (#148941-001)
Intel MULTIBUS II Architecture Specification Handbook (#146077)

Manuals may be ordered from any Intel Sales Representative, Distribution Office, or from the Intel Literature Department, 3065 Bowers Avenue, Santa Clara, CA 95051.

ORDERING INFORMATION

<table>
<thead>
<tr>
<th>Part Number</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>iSBC 186/410</td>
<td>MULTIBUS II Serial Communica-tions Computer</td>
</tr>
</tbody>
</table>
DOMESTIC SALES OFFICES

ALABAMA
- Hotel Corp., 5015 Brandon Dr., Suite 314, Huntsville 35801, (205) 830-4010
- ARIZONA
- Hotel Corp., 1225 N. 26th Dr., Suite 336, Phoenix 85029, (602) 260-9880
- Hotel Corp., 1581 N. 5th Place, Suite 101, Phoenix 85014, (602) 269-8115
- CALIFORNIA
- Hotel Corp., 2911 Hollywood Street, Suite 118, Sunnyvale 94086, (408) 796-6000
- Hotel Corp., 1510 Arden Way, Suite 101, Sacramento 95815, (916) 260-8066
- Hotel Corp., 4300 Executive Drive Suite 128, San Diego 92121, (619) 462-0600
- Hotel Corp.* 900 N. Figueroa Street, Suite 400, Santa Monica 90404, (310) 969-1114
- Hotel Corp.* 4020 W. Olympic Expressway, 2nd Floor, Santa Monica 90406, (310) 968-3065, FAX: (310) 727-0260
- COLORADO
- Hotel Corp., 4445 Northpark Drive, Suite 750, Denver 80217, (303) 720-0600
- Hotel Corp.* 650 S. Cherry St., Suite 915, Denver 80206, (303) 965-4119, TWX: 910-931-2289
- CONNECTICUT
- Hotel Corp., 255 E. Main Road, 2nd Floor, New London 06320, (860) 758-1100, TWX: 910-758-1100
- FLORIDA
- Hotel Corp., 1350 S. Lee Blvd., Suite 340, Delray Beach 33445, (561) 243-8000, FAX: 561-243-8267
- Hotel Corp., 11500 S. 2nd Street North, Suite 175, St. Petersburg 33718, (813) 925-8000, FAX: 813-738-1507
- GEORGIA
- Hotel Corp., 5280 Peachtree Parkway, Suite 300, Norcross 30092, (770) 448-0041
- ILLINOIS
- Hotel Corp.* 580 N. Martindale Road, Suite 400, Oak Brook 60523, (312) 725-9401, FAX: 312-725-9782
- INDIANA
- Hotel Corp., 6777 Putnam Road, Suite 115, Indianapolis 46288, (317) 876-9823
- IOWA
- Hotel Corp., 1800 S. Andrews Drive, Suite 2nd Floor, Des Moines 50309, (515) 329-1010
- KANSAS
- Hotel Corp., 16460 C. S. Blvd., Suite 140, Blue Springs 64014, (816) 241-0066
- MARYLAND
- Hotel Corp.* 7251 Parkway Drive South, Suite C, Hanover 21717, (410) 799-7700, TWX: 710-962-1944
- Hotel Corp., 7255 Walker Drive, Suite 2nd Floor, Greenbelt 20770, (301) 471-1020
- MASSACHUSETTS
- Hotel Corp.* 5601 Washington Drive, Suite 3rd Floor, Southfield 02494, (617) 734-9223
- MICHIGAN
- Hotel Corp., 7201 Orchard Lake Road, Suite 100, West Bloomfield 48323, (248) 581-8066
- MINNESOTA
- Hotel Corp., 5210 W. 60th St., Suite 300, Bloomington 55431, (612) 826-6725, TWX: 651-756-2257
- MISSOURI
- Hotel Corp., 4500 City Expressway, Suite 131, Earth City 63045, (573) 281-1900
- NEW JERSEY
- Hotel Corp., Parkway 100 Office Center, 320 New Jersey 310 North, Raritan 08879, (908) 747-2220
- Hotel Corp., 564 Corporate Center 1000 Corporate Center 1st Floor, Secaucus 07094, (201) 740-9100, FAX: 201-740-9129
- NEW MEXICO
- Hotel Corp., 5000 Menaul Boulevard N.E., Suite 8 7th Floor, Albuquerque 87112, (505) 592-5666
- NEW YORK
- Hotel Corp.* 127 Main Street, Bridgehampton 11932, (607) 775-0957, TWX: 807-272-6973
- Hotel Corp.* 2600 Expressway Dr., South Suite 130, Cape Coral 33720, (727) 951-6300, FAX: 914-867-3135
- Hotel Corp.* 5800 Executive Center Drive, Suite 100, Charlotte 28212, (704) 398-0466, FAX: 704-336-2536
- Hotel Corp., 7200 Wofford Road, Suite 100, Raleigh 27607, (919) 781-8022
- OHIO
- Hotel Corp.* 3401 Park Center Drive Suite 500, Dayton 45414, (513) 965-8300, TWX: 810-400-2568
- Hotel Corp.* 8120 Summit Park Drive Suite 100, Cincinnati 45242, (513) 464-9738, TWX: 810-405-9998
- OKLAHOMA
- Hotel Corp., 6801 N. Broadway, Oklahoma City 73162, (405) 846-0466
- OREGON
- Hotel Corp., 8120 SW Airport Way, Suite 100, Portland 97219, (503) 594-9700
- PENNSYLVANIA
- Hotel Corp.* 400 Penn Center Blvd., Suite 810, Pittsburgh 15222, (412) 823-4970
- PUERTO RICO
- Hotel Corp., 584 Corporate Center 1st Floor, South Industrial Park, P.O. Box 92, Las Piedras 00961, (787) 733-9915
- TEXAS
- Hotel Corp., 513 S. Anderson Lane Suite 314, Austin 78738, (512) 454-2628
- Hotel Corp.* 12000 Ford Road Suite 400, Dallas 75244, (214) 841-2697, FAX: 214-349-1180
- Hotel Corp.* 7950 S. Freeway Suite 1490, Houston 77084, (713) 968-8686, TWX: 713-981-0490
- UTAH
- Hotel Corp., 428 East 6400 South Suite 104, Murray 84107, (801) 253-3021
- VIRGINIA
- Hotel Corp., 1504 Santa Rosa Road Suite 106 Richmond 23286, (804) 282-8668
- WASHINGTON
- Hotel Corp., 408 N. Mulvan Road Suite 102, Bellevue 98004, (425) 929-6068
- WISCONSIN
- Hotel Corp., 330 S. Executive Dr., Suite 102, Brookfield 53005, (262) 796-2000, FAX: (414) 796-2115
- CANADA
- BRITISH COLUMBIA
  Intel Semiconductor of Canada, Ltd., 4558 Canada Way, Suite 232, Burnaby V5A 4L6, (604) 296-2697, FAX: (604) 296-2534
- ONTARIO
  Intel Semiconductor of Canada, Ltd., 333 Webb Drive, Regent MSW B8, (613) 926-2107, TWX: 506-836-1151
- QUEBEC
  Intel Semiconductor of Canada, Ltd., 505 Avenue Harris Pl., (711) 926-6068, TWX: 514-926-4298
DOMESTIC DISTRIBUTORS

ALABAMA
Arrow Electronics, Inc.  
1015 Henderson Road  
Huntsville 35806  
Tel: (205) 697-8655

Santella Solutions, Inc.  
1940 Henderson Drive  
Talladega 35166  
Tel: (205) 697-2160

Pioneer/Technologies Group, Inc.  
4255 University Square  
Huntsville 35805  
Tel: (205) 697-9300

ARIZONA
Arrow Electronics, Inc.  
1015 Handerson Road  
Huntsville 35805  
Tel: (205) 837-6956

ARIZONA (Con't.)
Hamilton/Avnet Electronics  
1030 Center Parkway  
Scottsdale 85255  
Tel: (602) 990-8300

CALIFORNIA
Arrow Electronics, Inc.  
13818 West 190th Street  
Gardena 90248  
Tel: (213) 217-5200

CALIFORNIA (Con't.)
Hamilton/Avnet Electronics  
1921 Center Parkway  
Santa Clara 95050  
Tel: (408) 244-8000

COLORADO
Arrow Electronics, Inc.  
11500 South Park Hill Drive  
Orland Park 60467  
Tel: (708) 690-3333

CONNECTICUT
Hamilton/Avnet Electronics  
2398 Main Street  
New Haven 06511  
Tel: (203) 482-7422

ENGLISH
Hamilton/Avnet Electronics  
3200 18th Street  
Santa Monica 90404  
Tel: (310) 451-9995

FLORIDA
Arrow Electronics, Inc.  
8781 University Boulevard  
Winter Park 32792  
Tel: (407) 602-5800

GEORGIA
Hamilton/Avnet Electronics  
10532 Old U.S. Route 1  
Fayetteville 30214  
Tel: (770) 462-5810

ILLINOIS
Arrow Electronics, Inc.  
1148 W. Thome  
Itasca 60143  
Tel: (630) 565-0050

INDIANA
Arrow Electronics, Inc.  
15124 Kennedy Road  
New Haven 06511  
Tel: (203) 482-7410

IOWA
Hamilton/Avnet Electronics  
810 W. 31st Avenue, S  
Cedar Rapids 52404  
Tel: (319) 362-4737

KANSAS
Arrow Electronics, Inc.  
4800 Sand Creek Drive  
Leawood 66211  
Tel: (913) 853-5151

KANSAS (Con't.)
Hamilton/Avnet Electronics  
7675 Golden Triangle Dr.  
Suite B  
Eden Prairie 55344  
Tel: (612) 944-3235

KENTUCKY
Hamilton/Avnet Electronics  
1680 North State  
Lexington 40503  
Tel: (606) 258-1745

MARYLAND
Hamilton/Avnet Electronics  
1300 Quail Drive  
Suite K  
Columbia 21046  
Tel: (301) 955-5003

MICHIGAN
Hamilton/Avnet Electronics  
6925 Oak Hall Lane  
Columbia 21046  
Tel: (301) 955-5000

MINNESOTA
Hamilton/Avnet Electronics  
8010 5th Street NW  
St. Cloud 56301  
Tel: (320) 467-8888

MISSOURI
Hamilton/Avnet Electronics  
13743 Shoreline Court  
Earth City 63045  
Tel: (636) 332-5000

NEW HAMPSHIRE
1Arrow Electronics, Inc.  
1053 Broadmoor Blvd.  
Manchester 03103  
Tel: (603) 866-9609

NEW JERSEY
1Arrow Electronics, Inc.  
2584 Boulevard Dr. E  
Burlington 08016  
Tel: (609) 234-0762

NEW MEXICO
Hamilton/Avnet Electronics  
5700 Falls Ridge Lane  
Albuquerque 87111  
Tel: (505) 990-2220

NEW YORK
1Arrow Electronics, Inc.  
3255 Brighton Terrace Road  
Rochester 14623  
Tel: (716) 260-5200

Pioneer Electronics  
3255 Brighton Terrace Rd.  
Rochester 14623  
Tel: (716) 260-5200

NEW YORK (Con't.)
Hamilton/Avnet Electronics  
1600 British Way  
Nyack 10960  
Tel: (914) 928-0300

1Microcomputer System Technical Distributor Center

CQ/SALE/111008
DOMESTIC DISTRIBUTORS (Cont’d.)

NEW YORK (Cont’d.)
1st Microcomputer System Technical Distributor Center
500 Amersil Drive
Valencia, CA 91355
Tel: (805) 385-8080

DOMESTIC DISTRIBUTORS
(Cont’d.)
NEW YORK
[Text continues with distributor information for various states including New York, Oklahoma, Texas, and Wisconsin.]
EUROPEAN SALES OFFICES

EUROPEAN DISTRIBUTORS/REPRESENTATIVES

AUSTRIA
Bacher Elektronik GmbH
Bachauer Strasse 34
91472 Regensburg
Tel: +43 (0) 744 61 88

BELGIUM
Imeco Belgium S.A.
Av. des Clics de Guerra 94
1130 Brussels
Tel: +32 (0) 2 58 88 00

DENMARK
IT-Multicomponent A/S
Hørbyland 30
2900 Gribskov
Tel: +45 (0) 64 88 68

FINLAND
Leinonen Oy
Sorcikatu 4
00210 Helsinki
Tel: +358 (0) 9 36 16 00

FRANCE
Gérin
Z.A. de Courrèges
Av. de la Réserve-SP 88
91450 Le Pecq Cedex
Tel: +33 (0) 1 69 78 00

FRANCE
Tour d’Ampere
4, av. de L’Avenir
92000 Saint-Priest Cedex
Tel: +33 (0) 1 46 34 78 00

ITALY
Intel
Divisione ITT Industries GmbH
Via Milano 28
20156 Milan
Tel: +39 (0) 2 4736

NETHERLANDS
Koning en Hartman
1105 AB Amsterdam
Tel: +31 (0) 20 394 00 90

UNITED KINGDOM
Intel
1304-8500
Manchester Road
Salford M60 8QY
Tel: +44 (0) 161 460 00 00

UNITED KINGDOM
Intel
1105 AB Amsterdam
Tel: +31 (0) 20 394 00 90

UNITED KINGDOM
Intel
1304-8500
Manchester Road
Salford M60 8QY
Tel: +44 (0) 161 460 00 00

UNITED KINGDOM
Intel
1105 AB Amsterdam
Tel: +31 (0) 20 394 00 90

UNITED KINGDOM
INTERNATIONAL SALES OFFICES

AUSTRALIA
Intel Australia Pty. Ltd.*
Suite 10, 20 Pacific Hwy, Level 6
Chatswood NSW 2067
Tel: (02) 987-2744
TLX: AX 29967
FAX: (02) 823-2892

BRAZIL
Intel Semi-condutor do Brasil LTDA
Av. Rebouças, 1316 - 13º Andar
Rio de Janeiro - RJ
Tel: (021) 323-5999
TLX: 1151-1241574 BR
FAX: 65-1-112-17631

CHINA/HONG KONG
Intel Corporation
15/F. Office 1, Civic Bldg.,
9/F. Civic Bldg.,
180 Wanchai Road,
Kowloon
Tel: (852) 443-9191
TLX: 850699 SHYLINK HK
FAX: 65-901-966

INTERNATIONAL DISTRIBUTORS/REPRESENTATIVES

ARGENTINA
DAFSYS S.R.L.
Chaussée de Francia 714 PBIO
1086 Buenos Aires
Tel: 54-1-334-7728
TLX: S4T72

AUSTRALIA
Total Electronics
16-17 Horn Street
Huntingdale, 3166
Victoria, Australia
Tel: 61-3-943-4044
FAX: 61-3-943-4641

BRAZIL
Elektron Mecanicos
R. Garofalo Fusca Gomem, 78
S. Andre
54675 - Sao Paulo - S.P.
Tel: 55-11-1344-2627
TLX: 96112513 ELBRI BR
FAX: 55-11-534-9624

CHILE
DIN Instruments
Sdedorico 2393
Casilla 6056, Correo 22
Santiago
Tel: 56-2-295-8198
TLX: 44692 RUDY CZ

CHINA/HONG KONG
Novel Precision Machinery Co., Ltd.
Pak B, 390 Kowloon Rd. Bldg.
Room 1304-1305, 13/F.
North Race Course Rd.
Kowloon
Tel: 852-829-3232
TLX: 3911 JMMK HK
FAX: 852-801-852

INDIA
Micron Devices
No. 62, D.V. Road
Bengaluru 560 004
Tel: 91-812-921-0353
FAX: 91-812-921-0353

IRAN
Syntel Tech-U.S.A.
5090 Main St.
Cypress, CA 90630
Tel: 714-572-3321
FAX: 714-572-3321

JAPAN
Intel Japan K.K.
Saito Takanaka Bldg.,
1-1-1 Nihonbashi,
Kotonakacho, Tokyo 103
Tel: 03-326-9782
FAX: 03-326-9782

JAPAN (Cont'd.)
Intel Japan K.K.
5-1-1 Nihonbashi,
Kotonakacho, Tokyo 103
Tel: 03-326-9782
FAX: 03-326-9782

KOREA
Intel Technology Asia, Ltd.
Business Center 18th Floor
11, Yoido-Dong, Young Deung Po-Ku
Seoul 110
Tel: (02) 784-8166, 8258, 8186
TLX: 559999 INTEK KO
FAX: (02) 784-8169

BANGKOK
Intel Electronics Technologies, Ltd.
3rd Floor, Tower A
Bangkok, Thailand
Tel: (662) 258-1200
TLX: 856110 INTEB TH
FAX: 662-258-1200

TAIWAN
Intel Technology (Far East) Ltd.
Taiwan Branch
2F No. 50, Tun Hua Rd. No.
Taipei 103 T.C.
Tel: 02-718-9860
TLX: 28924 TAIWAN TN
FAX: 886-2-271-2455

LISBOA
C.G. Sales Electronica Lda.
R. da Trajano 1
4150 Lisboa
Tel: 351-1-748-3896
TLX: 22077 ELBRI PT
FAX: 351-1-748-3896

MEXICO
OASIS
7041 Millenia Pkwy
Suite 140
La Jolla, CA 92037
Tel: 619-453-2577
TLX: 79574 OASIS MX
FAX: 619-453-2577

NEW ZEALAND
Switch Enterprises
722 King Street
Wellington
Tel: 64-4-931-1555
FAX: 64-4-931-1555

SOUTH AFRICA
Multiverse Distribution, Christianity
P.O. Box 4228
247 Corinna St.
Durban
Tel: 031-513-4884
FAX: 031-513-4884

VENEZUELA
Distribuidora de Publicaciones
Av. Amado Nolasco 698
Caracas, Distrito Nacional
Tel: 58-2-867-8079
FAX: 58-2-867-8079

*Field Application Location

CO/SAL/110188
UNITED STATES
Intel Corporation
3065 Bowers Avenue
Santa Clara, CA 95051

JAPAN
Intel Japan K.K.
5-6 Tokodai, Tsukuba-shi
Ibaraki, 300-26

FRANCE
Intel Corporation S.A.R.L.
1, Rue Edison, BP 303
78054 Saint-Quentin-en-Yvelines Cedex

UNITED KINGDOM
Intel Corporation (U.K.) Ltd.
Pipers Way
Swindon
Wiltshire, England SN3 1RJ

WEST GERMANY
Intel Semiconductor GmbH
Dornacher Strasse 1
8016 Feldkirchen bei Muenchen

HONG KONG
Intel Semiconductor Ltd.
10/F East Tower
Bond Center
Queensway, Central

CANADA
Intel Semiconductor of Canada, Ltd.
190 Attwell Drive, Suite 500
Rexdale, Ontario M9W 6H8