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 7641
Mt. Prospect, IL 60056-7641

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. All handbooks can be ordered individually, and most are available in a pre-packaged set in the U.S. and Canada.

<table>
<thead>
<tr>
<th>TITLE</th>
<th>INTEL ORDER NUMBER</th>
<th>ISBN</th>
</tr>
</thead>
<tbody>
<tr>
<td>SET OF THIRTEEN HANDBOOKS</td>
<td>231003</td>
<td>N/A</td>
</tr>
<tr>
<td>(Available in U.S. and Canada)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

CONTENTS LISTED BELOW FOR INDIVIDUAL ORDERING:

<table>
<thead>
<tr>
<th>TITLE</th>
<th>INTEL ORDER NUMBER</th>
<th>ISBN</th>
</tr>
</thead>
<tbody>
<tr>
<td>COMPONENTS QUALITY/RELIABILITY</td>
<td>210997</td>
<td>1-55512-132-2</td>
</tr>
<tr>
<td>EMBEDDED APPLICATIONS</td>
<td>270648</td>
<td>1-55512-123-3</td>
</tr>
<tr>
<td>8-BIT EMBEDDED CONTROLLERS</td>
<td>270645</td>
<td>1-55512-121-7</td>
</tr>
<tr>
<td>16-BIT EMBEDDED CONTROLLERS</td>
<td>270646</td>
<td>1-55512-120-9</td>
</tr>
<tr>
<td>16/32-BIT EMBEDDED PROCESSORS</td>
<td>270647</td>
<td>1-55512-122-5</td>
</tr>
<tr>
<td>MEMORY PRODUCTS</td>
<td>210830</td>
<td>1-55512-117-9</td>
</tr>
<tr>
<td>MICROCOMMUNICATIONS</td>
<td>231658</td>
<td>1-55512-119-5</td>
</tr>
<tr>
<td>MICROCOMPUTER PRODUCTS</td>
<td>280407</td>
<td>1-55512-118-7</td>
</tr>
<tr>
<td>MICROPROCESSORS</td>
<td>230843</td>
<td>1-55512-115-2</td>
</tr>
<tr>
<td>PACKAGING</td>
<td>240800</td>
<td>1-55512-128-4</td>
</tr>
<tr>
<td>PERIPHERAL COMPONENTS</td>
<td>296467</td>
<td>1-55512-127-6</td>
</tr>
<tr>
<td>PRODUCT GUIDE</td>
<td>210846</td>
<td>1-55512-116-0</td>
</tr>
<tr>
<td>(Overview of Intel's complete product lines)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PROGRAMMABLE LOGIC</td>
<td>296083</td>
<td>1-55512-124-1</td>
</tr>
</tbody>
</table>

ADDITIONAL LITERATURE:
(Not included in handbook set)

<table>
<thead>
<tr>
<th>TITLE</th>
<th>INTEL ORDER NUMBER</th>
<th>ISBN</th>
</tr>
</thead>
<tbody>
<tr>
<td>AUTOMOTIVE HANDBOOK</td>
<td>231792</td>
<td>1-55512-125-x</td>
</tr>
<tr>
<td>INTERNATIONAL LITERATURE GUIDE</td>
<td>F00029</td>
<td>N/A</td>
</tr>
<tr>
<td>(Available in Europe only)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CUSTOMER LITERATURE GUIDE</td>
<td>210620</td>
<td>N/A</td>
</tr>
<tr>
<td>MILITARY HANDBOOK</td>
<td>210461</td>
<td>1-55512-126-8</td>
</tr>
<tr>
<td>(2 volume set)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SYSTEMS QUALITY/RELIABILITY</td>
<td>231°62</td>
<td>1-55512-046-6</td>
</tr>
</tbody>
</table>
U.S. and CANADA LITERATURE ORDER FORM

NAME: ________________________________________________ 

COMPANY: _________________________________________________

ADDRESS: ____________________________________________ 

CITY: ___________________________ STATE: _____ ZIP: ______

COUNTRY: ____________________________________________ 

PHONE NO.: ( ____ ) ______________________________

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

Include postage: Must Add Your Local Sales Tax

Postage

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 7641
Mt. Prospect, IL 60056-7641

International Customers outside the U.S. and Canada should use the International order form on the next page 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 12/31/91.

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>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>1</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 Corporation is a leading supplier of microcomputer components, modules and systems. When Intel invented the microprocessor in 1971, it created the era of the microcomputer. Today, Intel architectures are considered world standards. Whether used in embedded applications such as automobiles, printers and microwave ovens, or as the CPU in personal computers, client servers or supercomputers, Intel delivers leading-edge technology.

MICROCOMMUNICATIONS

1991

About Our Cover:

Thinkers, inventors, and artists throughout history have breathed life into their ideas by converting them into rough working sketches, models, and products. This series of covers shows a few of these creations, along with the applications and products created by Intel customers.
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:

287, 376, 386, 387, 486, 4-SITE, Above, ACE51, ACE96, ACE186, ACE196, ACE960, ActionMedia, BITBUS, COMMputer, CREDIT, Data Pipeline, DVI, ETOX, FaxBACK, Genius, i, i', i486, i750, i860, ICE, iCEL, ICEVIEW, iCS, iDBP, iDIS, i²ICE, iLBX, iMDDX, iMMX, Inboard, Insite, Intel, inte'l, Intel386, inteiBOS, Intel Certified, Intelevision, inteiIgent Identifier, inteiIgent Programming, Intellec, Intellink, iOSP, iPAT, iPDS, iPSC, iRMK, iRMX, iSBC, iSBX, iSDM, iSXM, Library Manager, MAPNET, MCS, Megachassis, MICROMAINFRAME, MULTICHANNEL, MULTIMODULE, MultiSERVER, ONCE, OpenNET, OTP, Pro750, PROMPT, Promware, QUEST, QueX, Quick-Erase, Quick-Pulse Programming, READY LAN, RMX/80, RUPI, Seamless, SLD, SugarCube, ToolTALK, UPI, VAPI, Visual Edge, VLSICEL, and ZapCode, and the combination of ICE, iCS, iRMX, iSBC, iSBX, iSXM, MCS, or UPI and a numerical suffix.

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.

CHMOS and HMOS are patented processes of Intel Corp.

PAL is a registered trademark of Advanced Micro Devices.

GAL is a registered trademark of Lattice Semiconductor, Inc.

Intel Corporation and Intel's FASTPATH are not affiliated with Kinetics, a division of Excelan, Inc. or its FASTPATH trademark or products.

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

Intel Corporation
Literature Sales
P.O. Box 7641
Mt. Prospect, IL 60056-7641

© INTEL CORPORATION 1990
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. Such 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 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 to 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 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).

NETWORK SERVICE AND SUPPORT

Today's broad spectrum of powerful networking capabilities are only as good as the customer support provided by the vendor. Intel offers network services and support structured to meet a wide variety of end-user computing needs. From a ground up design of your network's physical and logical design to implementation, installation and network wide maintenance. From software products to turn-key system solutions; Intel offers the customer a complete networked solution. With over 10 years of network experience in both the commercial and Government arena; network products, services and support from Intel provide you the most optimized network offering in the industry.

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.
DATA SHEET DESIGNATIONS

Intel uses various data sheet markings to designate each phase of the document as it relates to the product. The marking appears in the upper, right-hand corner of the data sheet. The following is the definition of these markings:

<table>
<thead>
<tr>
<th>Data Sheet Marking</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Product Preview</td>
<td>Contains information on products in the design phase of development. Do not finalize a design with this information. Revised information will be published when the product becomes available.</td>
</tr>
<tr>
<td>Advanced Information</td>
<td>Contains information on products being sampled or in the initial production phase of development.*</td>
</tr>
<tr>
<td>Preliminary</td>
<td>Contains preliminary information on new products in production.*</td>
</tr>
<tr>
<td>No Marking</td>
<td>Contains information on products in full production.*</td>
</tr>
</tbody>
</table>

*Specifications within these data sheets are subject to change without notice. Verify with your local Intel sales office that you have the latest data sheet before finalizing a design.
Table of Contents

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

SECTION ONE—DATA COMMUNICATIONS COMPONENTS

CHAPTER 1
Local Area Networks
CSMA/CD Access Method

DATA SHEETS
82586 IEEE 802.3 Ethernet LAN Coprocessor .................................. 1-1
82596CA High-Performance 32-Bit Local Area Network Coprocessor ...... 1-38
82596DX and 82596SX High-Performance 32-Bit Local Area Network Coprocessor ................................................................. 1-110
82588 High Integration Local Area Network Controller ......................... 1-186
82590 Advanced CSMA/CD LAN Controller with 8-Bit Data Path .......... 1-212
82C501AD Ethernet Serial Interface .................................................. 1-245
82521TB Twisted Pair Ethernet Serial Supercomponent ....................... 1-263
82504TA Transceiver Serial Interface (TSI) ......................................... 1-283
82505TA Multiport Repeater Controller (MPR) .................................... 1-300
82506TB Twisted Pair Medium Attachment Unit (TP MAU) ................... 1-321

APPLICATION NOTES
AP-235 An 82586 Data Link Driver .................................................. 1-337
AP-236 Implementing StarLAN with the Intel 82588 ............................... 1-418
AP-344 Interfacing Intel 82596 LAN Coprocessors with M68000 Family Microprocessors .......................................................... 1-494
AP-345 Implementing 10BASE-T Networks with Intel's Twisted Pair Ethernet Components and Supercomponents .......................... 1-570

CSMA/CD Access Method Evaluation Tools
PC586E CSMA/CD LAN Evaluation Board ........................................ 1-594

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
AP-401 Designing With the 82510 Asynchronous Serial Controller .......... 2-182
AP-310 High Performance Driver for 82510 ........................................ 2-263
AP-36 Using the 8273 SDLC/HDLC Protocol Controller ....................... 2-295
AP-134 Asynchronous Communication with the 8274 Multiple-Protocol Serial Controller ..................................................... 2-348
AP-145 Synchronous Communication with the 8274 Multiple Protocol Serial Controller ................................................ ..... 2-387
AP-222 Asynchronous and SDL/C Communications with 82530 ............ 2-426

CHAPTER 3
Other Components

DATA SHEETS
8291A GPIB Talker/Listener .......................................................... 3-1
8292 GPIB Controller ......................................................................... 3-33
Table of Contents (Continued)

8294A Data Encryption/Decryption Unit .................................................. 3-52
APPLICATION NOTES
AP-166 Using the 8291A GPIB Talker/Listener .......................................... 3-65
AP-66 Using the 8292 GPIB Controller ..................................................... 3-96

SECTION TWO—TELECOMMUNICATION COMPONENTS

CHAPTER 4
Modem Products
DATA SHEETS
- 89024 2400 BPS Intelligent Modem Chip Set ........................................ 4-1
- 89C024LT Error Correcting Laptop Modem Chip Set ................................ 4-23
- 89C024FT V.42/42bis Modem Chip Set .................................................. 4-46
EVALUATION KITS
- MD024LT Modem Demo Board ............................................................... 4-69
- MD024FT Modem Demo Board ............................................................... 4-70
- 89024 MEK2 89024 Enhanced Modem Evaluation Kit ............................... 4-71
- MEK3 Modem Evaluation Kit .................................................................. 4-73
APPLICATION BRIEF
- AB-24 89024 Modem Customization for V.23 Data Transmission ............ 4-75

CHAPTER 5
PCM Codec/Filter and Combo
DATA SHEETS
- 2910A PCM Codec-micro Law 8-Bit Companded A/D and D/A Converter ...... 5-1
- 2911A-1 PCM Codec-A Law 8-Bit Companded A/D and D/A Converter ....... 5-19
- 2912A PCM Transmit/Receive Filter ....................................................... 5-34
- 2913 and 2914 Combined Single-Chip PCM Codec and Filter ................. 5-46
- 2916/2917 HMOS Combined Single Chip PCM Codec and Filter ............... 5-67
APPLICATION NOTES
- Applications Information 2910A/2911A/2912A ..................................... 5-85
- AP-142 Designing Second-Generation Digital Telephony Systems Using the Intel
  2913/14 Codec/Filter Combochip ......................................................... 5-88
Alphanumeric Index

2910A PCM Codec-micro Law 8-Bit Companded A/D and D/A Converter .......................... 5-1
2911A-1 PCM Codec-A Law 8-Bit Companded A/D and D/A Converter ............................. 5-19
2912A PCM Transmit/Receive Filter .................................................................................... 5-34
2913 and 2914 Combined Single-Chip PCM Codec and Filter ............................................. 5-46
2916/2917 HMOS Combined Single Chip PCM Codec and Filter ........................................... 5-67
82050 Asynchronous Communications Controller ............................................................. 2-26
82504TA Transceiver Serial Interface (TSI) ........................................................................ 1-283
82505TA Multiport Repeater Controller (MPR) ................................................................ 1-300
82506TB Twisted Pair Medium Attachment Unit (TP MAU) ................................................. 1-321
2913/14 Combined ............................................................................................................. 3-33
82510 Asynchronous Serial Controller ............................................................................... 2-40
8251A Programmable Communication Interface ............................................................... 2-1
82521TB Twisted Pair Ethernet Serial Supercomponent ....................................................... 1-263
82530/82530-6 Serial Communications Controller (SCC) .................................................. 2-150
82586 IEEE 802.3 Ethernet LAN Coprocessor ................................................................. 1-1
82588 High Integration Local Area Network Controller ....................................................... 1-186
82590 Advanced CSMA/CD LAN Controller with 8-Bit Data Path .................................... 1-212
82596CA High-Performance 32-Bit Local Area Network Coprocessor ..................................... 1-38
82596DX and 82596SX High-Performance 32-Bit Local Area Network Coprocessor ......... 1-110
8273 Programmable HDLC/SDLC Protocol Controller ....................................................... 2-80
8274 Multi-Protocol Serial Controller (MPSC) .................................................................... 2-112
8291A GPIB Talker/Listener .............................................................................................. 3-1
8292 GPIB Controller ........................................................................................................... 3-33
8294A Data Encryption/Decryption Unit ............................................................................... 3-52
82C501AD Ethernet Serial Interface .................................................................................. 1-245
89024 2400 BPS Intelligent Modem Chip Set ...................................................................... 4-1
89024 MEK2 89024 Enhanced Modem Evaluation Kit ......................................................... 4-71
89C024FT V.42/42bis Modem Chip Set .............................................................................. 4-46
89C024LT Error Correcting Laptop Modem Chip-Set ......................................................... 4-23
AB-24 89024 Modern Customization for V.23 Data Transmission ....................................... 4-75
AP-134 Asynchronous Communication with the 8274 Multi-Protocol Serial Controller .... 2-348
AP-142 Designing Second-Generation Digital Telephony Systems Using the Intel 2913/14 5-88
Codec/Filter Combochip ....................................................................................................... 5-88
AP-145 Synchronous Communication with the 8274 Multi-Protocol Serial Controller .. 2-387
AP-166 Using the 8291A GPIB Talker/Listener ................................................................... 3-65
AP-222 Asynchronous and SDLC Communications with 82530 ......................................... 2-426
AP-235 An 82586 Data Link Driver .................................................................................... 1-337
AP-236 Implementing StarLAN with the Intel 82588 ............................................................ 1-418
AP-302 Microcommunications Overview .......................................................................... xii
AP-310 High Performance Driver for 82510 ...................................................................... 2-263
AP-344 Interfacing Intel 82596 LAN Coprocessors with M68000 Family Microprocessors 1-494
AP-345 Implementing 10BASE-T Networks with Intel's Twisted Pair Ethernet Components
and Supercomponents ........................................................................................................ 1-570
AP-36 Using the 8273 SDLC/HDLC Protocol Controller .................................................... 2-295
AP-401 Designing With the 82510 Asynchronous Serial Controller .................................. 2-182
AP-66 Using the 8292 GPIB Controller ............................................................................... 3-96
Applications Information 2910A/2911A/2912A ................................................................. 5-85
MD024FT Modem Demo Board ......................................................................................... 4-70
MD024LT Modem Demo Board ......................................................................................... 4-69
MEK3 Modem Evaluation Kit ............................................................................................. 4-73
PC586E CSMA/CD LAN Evaluation Board ........................................................................ 1-594
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. Interoperability 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—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
IEEE 802.3 ETHERNET LAN 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)
  - IEEE 802.3/StarLAN (1BASE5)
  - Proposed 10BASE-T
  - Proposed 10BASE-F
  - Proprietary CSMA/CD Networks up to 10 Mb/s
- 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

Supports Minimum Component Systems
- Shared Bus Configuration
- Interface to 80186 and 80188 Microprocessors Without Glue
Supports High-Performance Systems
- Bus Master, with On-Chip DMA
- 5-MB/s Bus Bandwidth
- Compatible with Dual-Port Memory
- Back-to-Back Frame Reception at 10 Mb/s
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 Corporation.
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 Medium 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 four 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 MB/s respectively. Memory address space is 16 megabytes 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 Mb/s. 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 CRCs are supported.

The 82586 is fabricated in Intel's reliable HMOS II 5-V 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>Type</th>
<th>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>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>System Ground.</td>
</tr>
<tr>
<td>RESET</td>
<td>34</td>
<td>13</td>
<td>TTL</td>
</tr>
<tr>
<td>TxD</td>
<td>27</td>
<td>22</td>
<td>0 TTL</td>
</tr>
<tr>
<td>TxC</td>
<td>26</td>
<td>23</td>
<td>*</td>
</tr>
<tr>
<td>RxD</td>
<td>25</td>
<td>24</td>
<td>1 TTL</td>
</tr>
<tr>
<td>RxC</td>
<td>23</td>
<td>28</td>
<td>*</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</th>
<th>Level</th>
<th>Name and Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>RTS</td>
<td>28</td>
<td>21</td>
<td>0</td>
<td>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</td>
<td>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</td>
<td>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</td>
<td>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</td>
<td>TTL</td>
<td>Active HIGH Interrupt request signal.</td>
</tr>
<tr>
<td>CLK</td>
<td>32</td>
<td>15</td>
<td>I</td>
<td>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</td>
<td>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</td>
<td>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, A20–A23</td>
<td>1, 3–5, 45–47</td>
<td>55–57, 59, 63–65</td>
<td>0</td>
<td>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>A19/S6</td>
<td>2</td>
<td>58</td>
<td>0</td>
<td>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, S1</td>
<td>40, 41</td>
<td>4, 3</td>
<td>0 TTL</td>
<td>Maximum mode only. These status pins define the type of DMA transfer during the current memory cycle. They are encoded as follows: S1 S0 0 0 Not Used 0 1 Read Memory 1 0 Write Memory 1 1 Passive 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 TTL</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 TTL</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 a system RESET and when the bus is not acquired.</td>
</tr>
<tr>
<td>ALE</td>
<td>39</td>
<td>5</td>
<td>0 TTL</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 (&quot;clock low&quot;) of any bus cycle. Note that ALE is never floated.</td>
</tr>
<tr>
<td>DEN</td>
<td>40</td>
<td>4</td>
<td>0 TTL</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/ R</td>
<td>41</td>
<td>3</td>
<td>0 TTL</td>
<td>Used in minimum mode only. DT/R 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/R is equivalent to S7. 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, (0FFFF6H), 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.
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.

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 received. (Figure 9.)

The 82586, once enabled, checks each passing frame for an address match. The 82586 will recognize its own unique address, one or more multicast addresses or the broadcast address. If a match occurs, 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 efficient 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 reclaims 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.

---

**Figure 10. System Control Block (SCB) Format**

<table>
<thead>
<tr>
<th>EVEN BYTE</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>SCB (STATUS)</td>
<td>SCB + 2 (COMMAND)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SCB + 4</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SCB + 6</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SCB + 8</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SCB + 10</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SCB + 12</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SCB + 14</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ODD BYTE</th>
<th>15</th>
<th>ODD BYTE</th>
<th>15</th>
</tr>
</thead>
<tbody>
<tr>
<td>STAT</td>
<td>0</td>
<td>CUS</td>
<td>0</td>
</tr>
<tr>
<td>ACK</td>
<td>CUC</td>
<td>RES</td>
<td>RUC</td>
</tr>
</tbody>
</table>

- CUBL OFFSET
- RFA OFFSET
- CRCERRS
- ALNERRS
- RSCERRS
- OVRNERRS

231246–9
• 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 mail-box 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</th>
<th>Description</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.</td>
</tr>
<tr>
<td>RUS</td>
<td>(3 bits) this field contains the status of the Receive Unit.</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</th>
<th>Description</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.</td>
</tr>
<tr>
<td>CUS</td>
<td>0  • NOP (doesn’t affect current state of the unit).</td>
</tr>
<tr>
<td>CUS</td>
<td>1  • Start execution of the first command on the CBL.</td>
</tr>
<tr>
<td>CUS</td>
<td>2  • Resume the operation of the command unit by executing the next command.</td>
</tr>
<tr>
<td>CUS</td>
<td>3  • Suspend execution of commands on CBL after current command is complete.</td>
</tr>
<tr>
<td>CUS</td>
<td>4  • Abort execution of commands immediately.</td>
</tr>
<tr>
<td>CUS</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.</td>
</tr>
<tr>
<td>RUS</td>
<td>0  • NCP (does not alter current state of unit).</td>
</tr>
<tr>
<td>RUS</td>
<td>1  • Start reception of frames. If a frame is being received, then complete</td>
</tr>
<tr>
<td>RUS</td>
<td>2  • Resume frame receiving (only when in suspended state.)</td>
</tr>
<tr>
<td>RUS</td>
<td>3  • Suspend frame receiving. If a frame is being received, then complete its</td>
</tr>
<tr>
<td>RUS</td>
<td>4  • Abort receiver operation immediately.</td>
</tr>
<tr>
<td>RUS</td>
<td>5-7  • Reserved, illegal for use.</td>
</tr>
<tr>
<td>RUS</td>
<td>(Bit 4-6)</td>
</tr>
<tr>
<td>RUS</td>
<td>(Bit 8-10)</td>
</tr>
<tr>
<td>RUS</td>
<td>(Bit 4-6)</td>
</tr>
<tr>
<td>RUS</td>
<td>(Bit 7)</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 (C)</th>
<th>Bit 14 (B)</th>
<th>Bit 13 (OK)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Command Completed</td>
<td>Busy Executing Command</td>
<td>Error Free Completion</td>
</tr>
</tbody>
</table>

**COMMAND word:**

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

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

![Figure 11. The NOP Command Block](image)
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:

![Figure 12. The IA-SETUP Command Block](image)

<table>
<thead>
<tr>
<th>STATUS word (written by 82586):</th>
</tr>
</thead>
<tbody>
<tr>
<td>C (Bit 15)</td>
</tr>
<tr>
<td>B (Bit 14)</td>
</tr>
<tr>
<td>OK (Bit 13)</td>
</tr>
<tr>
<td>A (Bit 12)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>COMMAND word:</th>
</tr>
</thead>
<tbody>
<tr>
<td>EL (Bit 15)</td>
</tr>
<tr>
<td>S (Bit 14)</td>
</tr>
<tr>
<td>I (Bit 13)</td>
</tr>
<tr>
<td>CMD (Bits 0–2)</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.

![Figure 12. The IA-SETUP Command Block](image)

<table>
<thead>
<tr>
<th>STATUS word (written by 82586):</th>
</tr>
</thead>
<tbody>
<tr>
<td>C (Bit 15)</td>
</tr>
<tr>
<td>B (Bit 14)</td>
</tr>
<tr>
<td>OK (Bit 13)</td>
</tr>
<tr>
<td>A (Bit 12)</td>
</tr>
</tbody>
</table>

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

LINK OFFSET: Address of next Command Block

Byte 6–7:

| BYTE CNT (Bits 0–3) | 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. |
Table 13. The CONFIGURE Command Block

<table>
<thead>
<tr>
<th>FIFO-LIM</th>
<th>(Bits 8–11)</th>
<th>• Value of FIFO Threshold.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Byte 8–9:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SRDY/ARDY</td>
<td>(Bit 6)</td>
<td>• SRDY/ARDY pin operates as ARDY (internal synchronization).</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>• SRDY/ARDY pin operates as SRDY (external synchronization).</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>SAV-BF</td>
<td>(Bit 7)</td>
<td>• Received bad frames are not saved in memory.</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>• Received bad frames are saved in memory.</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>ADD-LEN</td>
<td>(Bits 8–10)</td>
<td>• Number of address bytes. <strong>NOTE:</strong> 7 is interpreted as 0.</td>
</tr>
<tr>
<td>AL-LOC</td>
<td>(Bit 11)</td>
<td>• Address and Length Fields separated from data and associated with Transmit Command Block or Receive Frame Descriptor. For transmitted Frame, Source Address is inserted by the 82586.</td>
</tr>
</tbody>
</table>

|            | 1           | • Address and Length Fields are part of the Transmit/Receive data buffers, including Source Address (which is not inserted by the 82586). |
| PREAM-LEN  | (Bits 12–13) | • Preamble Length including Beginning of Frame indicator: |
|            |             | 00 - 2 bytes |
|            |             | 01 - 4 bytes |
|            |             | 10 - 8 bytes |
|            |             | 11 - 16 bytes |
|            |             |                           |
| INT-LPBCK  | (Bit 14)    | • Internal Loopback |
| EXT-LPBCK  | (Bit 15)    | • External Loopback. **NOTE:** Bits 14 and 15 configured to 1, cause Internal Loopback. |

|            | 0-2         | • Linear Priority |
| LIN-PRIO   | (Bits 0–2)  | • Accelerated Contention Resolution (Exponential Priority) |
| ACR        | (Bits 4–6)  | • Exponential Backoff Method |
| BOF-MET    | (Bit 7)     | 0 - IEEE 802.3/Ethernet |
|            |             | 1 - Alternate Method |
### INTERFRAME SPACING

- **Bits 8–15**
  - Number indicating the Interframe Spacing in TxC period units.

### Byte 12–13:

- **SLOT-TIME (L)**
  - (Bits 0–7)
  - Slot Time Number, Low Byte
  - Slot Time Number, High Bits

- **SLT-TM (H)**
  - (Bits 8–10)
  - Maximum Number of Transmission Retries on Collisions

### Byte 14–15:

- **PRM**
  - (Bit 0)
  - Promiscuous Mode

- **BC-DIS**
  - (Bit 1)
  - Broadcast Disable

- **MANCH/NRZ**
  - (Bit 2)
  - Manchester or NRZ Encoding/Decoding

- **TONO-CRS**
  - (Bit 3)
  - Transmit on No Carrier Sense

- **NCRC-INS**
  - (Bit 4)
  - No CRC Insertion

- **CRC-16**
  - (Bit 5)
  - 32 bit Autodin II CRC Polynomial

- **CRC-15**
  - (Bit 6)
  - 16 bit CCITT CRC Polynomial

- **BT-STF**
  - (Bit 7)
  - HDLC like Bitstuffing Mode

- **PAD**
  - (Bit 8–15)
  - Padding

- **CRSF**
  - (Bits 8–9)
  - Carrier Sense Filter in Bit Times

- **CRS-SRC**
  - (Bit 11)
  - Carrier Sense Source

### CDTF

- **Bits 12–14**
  - Collision Detect Filter in Bit Times

### CDT-SRC

- **Bit 15**
  - Collision Detect Source

- **0**
  - External

- **1**
  - Internal

### Byte 16:

- **MIN-FRM**
  - (Bits 0–7)
  - Minimum Number of Bytes in a Frame

### 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>CRS Filter</td>
<td>0</td>
</tr>
<tr>
<td>Internal CDT</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</th>
<th>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>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>MC-SETUP = 3</td>
</tr>
</tbody>
</table>

**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>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>
<tr>
<td>S10</td>
<td>No Carrier Sense signal during transmission</td>
</tr>
<tr>
<td>S9</td>
<td>Transmission unsuccessful (stopped) due to loss of Clear-to-Send signal.</td>
</tr>
<tr>
<td>S8</td>
<td>Transmission unsuccessful (stopped) due to DMA underrun, i.e. data not supplied from the system for transmission.</td>
</tr>
<tr>
<td>S7</td>
<td>Transmission had to Defer to traffic on the link.</td>
</tr>
</tbody>
</table>

**COMMAND word:**

<table>
<thead>
<tr>
<th>Bit</th>
<th>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>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.
er 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>Bit</th>
<th>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>
</tbody>
</table>

**COMMAND word:**

<table>
<thead>
<tr>
<th>Bit</th>
<th>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>TDR = 5</td>
</tr>
</tbody>
</table>

**Figure 16. The Transmit Buffer Description**

**Figure 17. The TDR Command Block**
**LINK OFFSET:** Address of next Command Block

**RESULT word:**

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

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

| C | (Bit 15) | • Command Completed |
| B | (Bit 14) | • Busy Executing Command |
| OK | (Bit 13) | • Error Free Completion |

**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) | • DUMP = 6 |

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

**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**
**Bytes 14H to 17H**: Content of the Transmit CRC generator. TXCRCR0 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 TXCRCR0 and TXCRCR1 are valid.

---

<table>
<thead>
<tr>
<th>Byte</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>TXCRCR0</td>
</tr>
<tr>
<td>1</td>
<td>TXCRCR1</td>
</tr>
<tr>
<td>2</td>
<td>TXCRCR3</td>
</tr>
<tr>
<td>3</td>
<td>TXCRCR5</td>
</tr>
</tbody>
</table>

---

**Figure 19. The DUMP Area**

1-22
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 - 00F0H
  2. For CRC-Autodin-II - C704D7BH

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-AD-L: 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-AD-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-TB-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.

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:

| 0 | The 82586 writes zero in this location. |
| 1 | The 82586 writes one in this location. |
| X | The 82586 writes zero or one in this location. |
| /// | The 82586 copies this location from the corresponding position in the memory structure. |

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):

| C (Bit 15) | • Command Completed |
| B (Bit 14) | • Busy Executing Command |
| OK (Bit 13) | • Error Free Completion |
| FAIL (Bit 11) | • Indicates that the Self Test Procedure Failed |

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

<table>
<thead>
<tr>
<th>OK</th>
<th>(Bit 13)</th>
<th>• 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.</th>
</tr>
</thead>
<tbody>
<tr>
<td>S11</td>
<td>(Bit 11)</td>
<td>• Received Frame Experienced CRC Error.</td>
</tr>
<tr>
<td>S10</td>
<td>(Bit 10)</td>
<td>• Received Frame Experienced an Alignment Error.</td>
</tr>
<tr>
<td>S9</td>
<td>(Bit 9)</td>
<td>• RU ran out of resources during reception of this frame.</td>
</tr>
<tr>
<td>S8</td>
<td>(Bit 8)</td>
<td>• RCV-DMA Overrun.</td>
</tr>
<tr>
<td>S7</td>
<td>(Bit 7)</td>
<td>• Received frame had fewer bits than configured Minimum Frame Length.</td>
</tr>
<tr>
<td>S6</td>
<td>(Bit 6)</td>
<td>• No EOF flag detected (only when configured to Bit stuffing).</td>
</tr>
</tbody>
</table>

**COMMAND word:**

<table>
<thead>
<tr>
<th>EL</th>
<th>(Bit 15)</th>
<th>• Last FD in the List.</th>
</tr>
</thead>
<tbody>
<tr>
<td>S</td>
<td>(Bit 14)</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.

231246–23
Figure 23. The Receive Buffer Descriptor (RBD) Format

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

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>EOF (Bit 15)  • Last buffer in received frame.</td>
</tr>
<tr>
<td>14</td>
<td>F (Bit 14)  • ACT COUNT field is valid.</td>
</tr>
<tr>
<td>0-13</td>
<td>ACT COUNT (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.

**NEXT RBD OFFSET:** Address of next BD in list of BD’s.

**EL/SIZE:**

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>EL (BIT 15)  • Last BD in list.</td>
</tr>
<tr>
<td>0-13</td>
<td>SIZE (Bits 0-13)  • Number of bytes the buffer is capable of holding.</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 .......... −1.0V to +7V
Power Dissipation ........................... 3.0 Watts

NOTICE: This is a production data sheet. The specifications are subject to change without notice.

WARNING: Stressing the device beyond the "Absolute Maximum Ratings" may cause permanent damage. These are stress ratings only. Operation beyond the "Operating Conditions" is not recommended and extended exposure beyond the "Operating Conditions" may affect device reliability.

D.C. CHARACTERISTICS

\[ T_A = 0^\circ C \text{ to } 70^\circ C, \quad T_C = 0^\circ C \text{ to } 105^\circ C, \quad V_{CC} = 5V \pm 10\%, \quad \text{CLK has MOS levels (See } V_{MIL}, V_{MIH}, V_{MOL}, V_{MOH}) \text{. TxC and RxC have 82C501 compatible levels (} V_{MIL}, V_{TIH}, V_{RIH}) \text{. All other signals have TTL levels (see } V_{IL}, V_{IH}, V_{OL}, V_{OH}). \]

<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></td>
<td>0.45</td>
<td>V</td>
<td>IOL = 2.5 mA</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>VIML</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>VTIH</td>
<td>Input High Voltage (TxC)</td>
<td>3.3</td>
<td>VCC + 0.5</td>
<td>V</td>
<td></td>
</tr>
<tr>
<td>VRIH</td>
<td>Input High Voltage (RxC)</td>
<td>3.0</td>
<td>VCC + 0.5</td>
<td>V</td>
<td></td>
</tr>
<tr>
<td>VMOL</td>
<td>Output Low Voltage (MOS)</td>
<td>0.45</td>
<td>V</td>
<td>IOL 2.5 mA</td>
<td></td>
</tr>
<tr>
<td>VMOH</td>
<td>Output High Voltage (MOS)</td>
<td>VCC − 0.5</td>
<td>V</td>
<td>IOH = 400 μA</td>
<td></td>
</tr>
<tr>
<td>IIL</td>
<td>Input Leakage Current</td>
<td>±10</td>
<td>μA</td>
<td>0 ≤ VIN ≤ VCC</td>
<td></td>
</tr>
<tr>
<td>IOL</td>
<td>Output Leakage Current</td>
<td>±10</td>
<td>μA</td>
<td>0.45 ≤ VOUT ≤ VCC</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 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>550</td>
<td>mA</td>
<td>TA = 0°C</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>450</td>
<td></td>
<td>TA = 70°C</td>
<td></td>
</tr>
</tbody>
</table>

1-28
SYSTEM INTERFACE A.C. TIMING CHARACTERISTICS

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

INPUT AND OUTPUT WAVEFORMS FOR A.C. TESTS

Figures 24 and 25 define how the measurements should be done. 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 (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>1000</td>
<td>55</td>
<td>1000</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></td>
<td></td>
<td>15</td>
<td>15</td>
</tr>
<tr>
<td>T7</td>
<td>CLK Fall Time</td>
<td></td>
<td></td>
<td>15</td>
<td>15</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>20</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 T1</td>
<td>1 T1</td>
<td>1 T1</td>
<td></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></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></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>T2-10</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>0</td>
<td></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: S0, 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. S1, 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 CHARACTERISTICS**

**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:

\[ T_{51}, T_{52} = \frac{1}{2f} \pm 5\% \]

**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>T48</td>
<td>TxC Cycle</td>
<td>100</td>
<td>1000</td>
<td>Notes 14, 2</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>
</tbody>
</table>

**TRANSMIT DATA PARAMETERS**

| T53    | TxD Rise Time | 10   |      | Notes 5, 13    |
| T54    | TxD Fall Time | -10  |      | Notes 5, 13    |
| T55    | TxD Transition-Transition | Min (T51, T52) - 7 | Notes 2, 5    |
| T56    | TxC Low to TxD Valid | 40  |      | Notes 3, 5    |
| T57    | TxC Low to TxD Transition | 30  |      | Notes 2, 5    |
| T58    | TxC High to TxD Transition | 30  |      | Notes 2, 5    |
| T59    | TxC Low to TxD High at the Transmission End | 40  |      | Note 5        |

**REQUEST TO SEND/CLEAR TO SEND PARAMETERS**

| T60    | TxC Low to RTS Low, Time to Activate RTS | 40  |      | Note 6        |
| T61    | CTS Valid to TxC Low, CTS Setup Time | 45  |      |               |
| T62    | TxC Low to CTS Invalid, CTS Hold Time | 20  |      | Note 7        |
| T63    | TxC Low to RTS High, time to Deactivate RTS | 40  |      | Note 6        |

**RECEIVE CLOCK PARAMETERS**

| T64    | RxC Clock Cycle | 100  |      | Notes 15, 3    |
| T65    | RxC Rise Time | 5    |      | Note 15        |
| T66    | RxC Fall Time | 5    |      | Note 15        |
| T67    | RxC High Time @ 2.7V | 36  | 1000 | Note 15        |
| T68    | RxC Low Time @0.9V | 40  |      | Note 15        |

*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><strong>RECEIVE DATA PARAMETERS</strong></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><strong>CARRIER SENSE/COLLISION DETECT PARAMETERS</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T73</td>
<td>CDT Valid to TxC High Ext. Collision Detect Setup</td>
<td>30</td>
<td></td>
<td>Note 12</td>
</tr>
<tr>
<td></td>
<td>Time</td>
<td></td>
<td></td>
<td></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></td>
<td><strong>INTERFRAME SPACING PARAMETER</strong></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:
   \[ T75 = \text{NCDF} \times T48 + (12.5 \text{ to } 23.5) \times T48 \] if collision occurs after preamble
   \[ \text{NCDF} = \text{The collision detection filter configuration value} \]
9. Programmable value:
   \[ T78 = \text{NCSF} \times T48 + (12.5 \text{ to } 23.5) \times T48 \]
   \[ \text{NCSF} = \text{The carrier sense filter configuration value} \]
   \[ \text{TBD is a function of internal/external carrier sense bit} \]
   \[ T79 = 32 \times T48 \]
10. Programmable value:
    \[ T82 = \text{NIFS} \times T48 \]
    \[ \text{NIFS} = \text{the IFS configuration value} \]
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
82596CA
HIGH-PERFORMANCE 32-BIT LOCAL AREA NETWORK COPROCESSOR

- Performs Complete CSMA/CD Medium Access Control (MAC) Functions—Independently of CPU
  - IEEE 802.3 (EOC) Frame Delimiting
  - HDLC Frame Delimiting
- Supports Industry Standard LANs
  - IEEE TYPE 10BASE5 (Ethernet*), IEEE TYPE 10BASE2 (Cheapernet), IEEE TYPE 1BASE5 (StarLAN), and the Proposed Standards TYPE 10BASE-T and 10BASE-F
  - Proprietary CSMA/CD Networks Up to 20 Mb/s
- On-Chip Memory Management
  - Automatic Buffer Chaining
  - Buffer Reclamation after Receipt of Bad Frames; Optional Save Bad Frames
  - 32-Bit Segmented or Linear (Flat) Memory Addressing Formats
- Network Management and Diagnostics
  - Monitor Mode
  - 32-Bit Statistical Counters
- 82586 Software Compatible

- Optimized CPU Interface
  - Optimized Bus Interface to Intel's i486™ and 80960CA Processors
  - Supports Big Endian and Little Endian Byte Ordering
- 32-Bit Bus Master Interface
  - 106 MB/s Bus Bandwidth
  - Burst Bus Transfers
  - Bus Throttle Timers
  - Transfers Data at 100% of Serial Bandwidth
  - 128-Byte Receive FIFO, 64-Byte Transmit FIFO
- Self-Test Diagnostics
- Configurable Initialization Root for Data Structures
- High-Speed, 5V, CHMOS** IV Technology
- 132-Pin Plastic Quad Flat Pack (PQFP) and PGA Package
  (See Packaging Spec Order No. 231369)
i486 is a trademark of Intel Corporation.
*Ethernet is a registered trademark of Xerox Corporation.
**CHMOS is a patented process of Intel Corporation.

Figure 1. 82596CA Block Diagram
# 82596CA High-Performance 32-Bit Local Area Network Coprocessor

## CONTENTS

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>INTRODUCTION</td>
<td>1-40</td>
</tr>
<tr>
<td>PIN DESCRIPTIONS</td>
<td>1-44</td>
</tr>
<tr>
<td>82596 AND HOST CPU INTERACTION</td>
<td>1-48</td>
</tr>
<tr>
<td>82596 BUS INTERFACE</td>
<td>1-48</td>
</tr>
<tr>
<td>82596 MEMORY ADDRESSING</td>
<td>1-48</td>
</tr>
<tr>
<td>82596 SYSTEM MEMORY STRUCTURE</td>
<td>1-50</td>
</tr>
<tr>
<td>TRANSMIT AND RECEIVE MEMORY STRUCTURES</td>
<td>1-51</td>
</tr>
<tr>
<td>TRANSMITTING FRAMES</td>
<td>1-54</td>
</tr>
<tr>
<td>RECEIVING FRAMES</td>
<td>1-55</td>
</tr>
<tr>
<td>82596 NETWORK MANAGEMENT AND DIAGNOSTICS</td>
<td>1-55</td>
</tr>
<tr>
<td>NETWORK PLANNING AND MAINTENANCE</td>
<td>1-57</td>
</tr>
<tr>
<td>STATION DIAGNOSTICS AND SELF-TEST</td>
<td>1-58</td>
</tr>
<tr>
<td>82586 SOFTWARE COMPATIBILITY</td>
<td>1-58</td>
</tr>
<tr>
<td>INITIALIZING THE 82596</td>
<td>1-58</td>
</tr>
<tr>
<td>SYSTEM CONFIGURATION POINTER (SCP)</td>
<td>1-58</td>
</tr>
<tr>
<td>Writing the Sysbus</td>
<td>1-59</td>
</tr>
<tr>
<td>INTERMEDIATE SYSTEM CONFIGURATION POINTER (ISCP)</td>
<td>1-60</td>
</tr>
<tr>
<td>INITIALIZATION PROCESS</td>
<td>1-60</td>
</tr>
<tr>
<td>CONTROLLING THE 82596CA</td>
<td>1-61</td>
</tr>
<tr>
<td>82596 CPU ACCESS INTERFACE (PORT)</td>
<td>1-61</td>
</tr>
<tr>
<td>MEMORY ADDRESSING FORMATS</td>
<td>1-62</td>
</tr>
<tr>
<td>LITTLE ENDIAN AND BIG ENDIAN BYTE ORDERING</td>
<td>1-62</td>
</tr>
<tr>
<td>COMMAND UNIT (CU)</td>
<td>1-62</td>
</tr>
<tr>
<td>RECEIVE UNIT (RU)</td>
<td>1-63</td>
</tr>
<tr>
<td>SYSTEM CONTROL BLOCK (SCB)</td>
<td>1-63</td>
</tr>
<tr>
<td>SCB OFFSET ADDRESSES</td>
<td>1-66</td>
</tr>
<tr>
<td>CBL Offset (Address)</td>
<td>1-66</td>
</tr>
<tr>
<td>RFA Offset (Address)</td>
<td>1-66</td>
</tr>
<tr>
<td>SCB STATISTICAL COUNTERS</td>
<td>1-67</td>
</tr>
<tr>
<td>Statistical Counter Operation</td>
<td>1-67</td>
</tr>
<tr>
<td>ACTION COMMANDS AND OPERATING MODES</td>
<td>1-68</td>
</tr>
<tr>
<td>NOP</td>
<td>1-69</td>
</tr>
<tr>
<td>Individual Address Setup</td>
<td>1-69</td>
</tr>
<tr>
<td>Configure</td>
<td>1-70</td>
</tr>
<tr>
<td>Multicast-Setup</td>
<td>1-76</td>
</tr>
<tr>
<td>Transmit</td>
<td>1-77</td>
</tr>
<tr>
<td>Jamming Rules</td>
<td>1-79</td>
</tr>
<tr>
<td>TDR</td>
<td>1-80</td>
</tr>
<tr>
<td>Dump</td>
<td>1-82</td>
</tr>
<tr>
<td>Diagnose</td>
<td>1-85</td>
</tr>
<tr>
<td>RECEIVE FRAME DESCRIPTOR</td>
<td>1-86</td>
</tr>
<tr>
<td>Simplified Memory Structure</td>
<td>1-86</td>
</tr>
<tr>
<td>Flexible Memory Structure</td>
<td>1-87</td>
</tr>
<tr>
<td>Receive Buffer Descriptor (RBD)</td>
<td>1-88</td>
</tr>
<tr>
<td>ELECTRICAL AND TIMING CHARACTERISTICS</td>
<td>1-93</td>
</tr>
<tr>
<td>DC Characteristics</td>
<td>1-93</td>
</tr>
<tr>
<td>AC Characteristics</td>
<td>1-94</td>
</tr>
<tr>
<td>82596CA Input/Output System Timings</td>
<td>1-94</td>
</tr>
<tr>
<td>Transmit/Receive Clock Parameters</td>
<td>1-96</td>
</tr>
<tr>
<td>82596CA BUS Operation</td>
<td>1-99</td>
</tr>
<tr>
<td>System Interface AC Timing Characteristics</td>
<td>1-100</td>
</tr>
<tr>
<td>Input Waveforms</td>
<td>1-101</td>
</tr>
<tr>
<td>Serial AC Timing Characteristics</td>
<td>1-103</td>
</tr>
<tr>
<td>OUTLINE DIAGRAM</td>
<td>1-105</td>
</tr>
<tr>
<td>REVISION HISTORY</td>
<td>1-109</td>
</tr>
</tbody>
</table>

1-39
INTRODUCTION

The 82596CA is an intelligent, high-performance 32-bit Local Area Network coprocessor. The 82596CA implements the CSMA/CD access method and can be configured to support all existing IEEE 802.3 standards—TYPEs 10BASE5, 10BASE2, 1BASE5, and 10BROAD36. It can also be used to implement the proposed standards TYPE 10BASE-T and 10BASE-F. The 82596CA performs high-level commands, command chaining, and interprocessor communications via shared memory, thus relieving the host CPU of many tasks associated with network control. All time-critical functions are performed independently of the CPU, this increases network performance and efficiency. The 82596CA bus interfaces is optimized for Intel's i486™, 80960CA, and 80960KB processors.

The 82596CA implements all IEEE 802.3 Medium Access Control and channel interface functions, these include framing, preamble generation and stripping, source address generation, destination address checking, short-frame detection, and automatic length-field handling. Data rates up to 20 Mb/s are supported.

The 82596CA provides a powerful host system interface. It manages memory structures automatically, with command chaining and bidirectional data chaining. An on-chip DMA controller manages four channels, this allows autonomous transfer of data blocks (buffers and frames) and relieves the CPU of byte transfer overhead. Buffers containing errored or collided frames can be automatically recovered without CPU intervention. The 82596CA provides an upgrade path for existing 82586 software drivers by providing an 82586-software-compatible mode that supports the current 82586 memory structure. The 82586CA also has a Flexible memory structure and a Simplified memory structure. The 82596CA can address up to 4 gigabytes of memory. The 82596CA supports Little Endian and Big Endian byte ordering.

The 82596CA bus interface can achieve a burst transfer rate of 106 MB/s at 33 MHz. The bus interface employs bus throttle timers to regulate 82596CA bus use. Two large, independent FIFOs—128 bytes for Receive and 64 bytes for Transmit—tolerate long bus latencies and provide programmable thresholds that allow the user to optimize bus overhead for any worst-case bus latency. The high-performance bus is capable of back-to-back transmission and reception during the IEEE 802.3 9.6-μs Interframe Spacing (IFS) period.

The 82596CA provides a wide range of diagnostics and network management functions, these include internal and external loopback, exception condition tallies, channel activity indicators, optional capture of all frames regardless of destination address (promiscuous mode), optional capture of errored or collided frames, and time domain reflectometry for locating fault points on the network cable. The statistical counters, in 32-bit segmented and linear modes, are 32-bits each and include CRC errors, alignment errors, overrun errors, resource errors, short frames, and received collisions. The 82596CA also features a monitor mode for network analysis. In this mode the 82596CA can capture status bytes, and update statistical counters, of frames monitored on the link without transferring the contents of the frames to memory. This can be done concurrently while transmitting and receiving frames destined for that station.

The 82596CA can be used in both baseband and broadband networks. It can be configured for maximum network efficiency (minimum contention overhead) with networks of any length. Its highly flexible CSMA/CD unit supports address field lengths of zero through six bytes—configurable to either IEEE 802.3/Ethernet or HDLC frame delimitation. It also supports 16- or 32-bit cyclic redundancy checks. The CRC can be transferred directly to memory for receive operations, or dynamically inserted for transmit operations. The CSMA/CD unit can also be configured for full duplex operation for high throughput in point-to-point connections.

The 82596CA is fabricated with Intel's reliable, 5-V, CHMOS IV technology. It is available in a 132-pin PQFP or PGA package.
Figure 2. 82596CA PQFP Pin Configuration
Figure 3. 82596CA PGA Pinout
# 82596CA PGA Cross Reference by Pin Name

<table>
<thead>
<tr>
<th>Address</th>
<th>Data</th>
<th>Control</th>
<th>Serial Interface</th>
<th>VCC</th>
<th>VSS</th>
</tr>
</thead>
<tbody>
<tr>
<td>A2</td>
<td>N9</td>
<td>D0</td>
<td>J2</td>
<td>ADS</td>
<td>M5</td>
</tr>
<tr>
<td>A3</td>
<td>M9</td>
<td>D1</td>
<td>H3</td>
<td>AHOLD</td>
<td>M6</td>
</tr>
<tr>
<td>A4</td>
<td>M10</td>
<td>D2</td>
<td>G2</td>
<td>BE0</td>
<td>M7</td>
</tr>
<tr>
<td>A5</td>
<td>P11</td>
<td>D3</td>
<td>G3</td>
<td>BE1</td>
<td>P5</td>
</tr>
<tr>
<td>A6</td>
<td>N11</td>
<td>D4</td>
<td>G1</td>
<td>BE2</td>
<td>M8</td>
</tr>
<tr>
<td>A7</td>
<td>P12</td>
<td>D5</td>
<td>D1</td>
<td>BE3</td>
<td>P9</td>
</tr>
<tr>
<td>A8</td>
<td>M11</td>
<td>D6</td>
<td>C1</td>
<td>BLAST</td>
<td>N2</td>
</tr>
<tr>
<td>A9</td>
<td>N12</td>
<td>D7</td>
<td>F3</td>
<td>BOFF</td>
<td>N6</td>
</tr>
<tr>
<td>A10</td>
<td>M12</td>
<td>D8</td>
<td>D2</td>
<td>BIRDY</td>
<td>M1</td>
</tr>
<tr>
<td>A11</td>
<td>P13</td>
<td>D9</td>
<td>C2</td>
<td>BREQ</td>
<td>P4</td>
</tr>
<tr>
<td>A12</td>
<td>L12</td>
<td>D10</td>
<td>E3</td>
<td>BST6</td>
<td>N1</td>
</tr>
<tr>
<td>A13</td>
<td>N13</td>
<td>D11</td>
<td>D3</td>
<td>CA</td>
<td>P3</td>
</tr>
<tr>
<td>A14</td>
<td>M13</td>
<td>D12</td>
<td>B2</td>
<td>CLK</td>
<td>J3</td>
</tr>
<tr>
<td>A15</td>
<td>P14</td>
<td>D13</td>
<td>B1</td>
<td>DP0</td>
<td>L2</td>
</tr>
<tr>
<td>A16</td>
<td>K12</td>
<td>D14</td>
<td>C3</td>
<td>DP1</td>
<td>L3</td>
</tr>
<tr>
<td>A17</td>
<td>N14</td>
<td>D15</td>
<td>A1</td>
<td>DP2</td>
<td>L1</td>
</tr>
<tr>
<td>A18</td>
<td>J12</td>
<td>D16</td>
<td>B3</td>
<td>DP3</td>
<td>K3</td>
</tr>
<tr>
<td>A19</td>
<td>K13</td>
<td>D17</td>
<td>C4</td>
<td>HLDA</td>
<td>M6</td>
</tr>
<tr>
<td>A20</td>
<td>M14</td>
<td>D18</td>
<td>A2</td>
<td>HOLD</td>
<td>P2</td>
</tr>
<tr>
<td>A21</td>
<td>H12</td>
<td>D19</td>
<td>C5</td>
<td>INT/INT</td>
<td>N3</td>
</tr>
<tr>
<td>A22</td>
<td>K14</td>
<td>D20</td>
<td>A3</td>
<td>LE/BE</td>
<td>B14</td>
</tr>
<tr>
<td>A23</td>
<td>G12</td>
<td>D21</td>
<td>B4</td>
<td>LOCK</td>
<td>M4</td>
</tr>
<tr>
<td>A24</td>
<td>F14</td>
<td>D22</td>
<td>A4</td>
<td>PCHK</td>
<td>P1</td>
</tr>
<tr>
<td>A25</td>
<td>F12</td>
<td>D23</td>
<td>C6</td>
<td>PORT</td>
<td>M2</td>
</tr>
<tr>
<td>A26</td>
<td>F13</td>
<td>D24</td>
<td>B5</td>
<td>READY</td>
<td>M3</td>
</tr>
<tr>
<td>A27</td>
<td>D14</td>
<td>D25</td>
<td>C7</td>
<td>RESET</td>
<td>B13</td>
</tr>
<tr>
<td>A28</td>
<td>E12</td>
<td>D26</td>
<td>A5</td>
<td>W/R</td>
<td>N4</td>
</tr>
<tr>
<td>A29</td>
<td>D13</td>
<td>D27</td>
<td>B8</td>
<td></td>
<td></td>
</tr>
<tr>
<td>A30</td>
<td>D12</td>
<td>D28</td>
<td>C8</td>
<td></td>
<td></td>
</tr>
<tr>
<td>A31</td>
<td>C14</td>
<td>D29</td>
<td>A9</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
## PIN DESCRIPTIONS

<table>
<thead>
<tr>
<th>Symbol</th>
<th>PQFP Pin No.</th>
<th>Type</th>
<th>Name and Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLK</td>
<td>9</td>
<td>I</td>
<td><strong>CLOCK.</strong> The system clock input provides the fundamental timing for the 82596. It is a 1X CLK input used to generate the 82596 clock and requires TTL levels. All external timing parameters are specified in reference to the rising edge of CLK.</td>
</tr>
<tr>
<td>D0–D31</td>
<td>14–53</td>
<td>I/O</td>
<td><strong>DATA BUS.</strong> The 32 Data Bus lines are bidirectional, tri-state lines that provide the general purpose data path between the 82596 and memory. With the 82596 the bus can be either 16 or 32 bits wide; this is determined by the BST6 signal. The 82596 always drives all 32 data lines during Write operations, even with a 16-bit bus. D31 – D0 are floated after a Reset or when the bus is not acquired. These lines are inputs during a CPU Port access; in this mode the CPU writes the next address to the 82596 through the data lines. During PORT commands (Relocatable SCP, Self-Test, Reset and Dump) the address must be aligned to a 16-byte boundary. This frees the D3 – D0 lines so they can be used to distinguish the commands. The following is a summary of the decoding data.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td><img src="data_table.png" alt="Decoding Data" /></td>
</tr>
<tr>
<td>DP0–DP3</td>
<td>4–7</td>
<td>I/O</td>
<td><strong>DATA PARITY.</strong> These are tri-stated data parity pins. There is one parity line for each byte of the data bus. The 82596 drives them with even-parity information during write operations having the same timing as data writes. Likewise, even-parity information, with the same timing as read information, must be driven back to the 82596 over these pins to ensure that the correct parity check status is indicated by the 82596.</td>
</tr>
<tr>
<td>PCHK</td>
<td>127</td>
<td>O</td>
<td><strong>PARITY CHECK.</strong> This pin is driven high one clock after RDY to inform Read operations of the parity status of data sampled at the end of the previous clock cycle. When driven low it indicates that incorrect parity data has been sampled. It only checks the parity status of enabled bytes, which are indicated by the Byte Enable and Bus Size signals. PCHK is only valid for one clock time after data read is returned to the 82596; i.e., it is inactive (high) at all other times.</td>
</tr>
<tr>
<td>A31-A2</td>
<td>7008</td>
<td>O</td>
<td><strong>ADDRESS LINES.</strong> These 30 tri-stated Address lines output the address bits required for memory operation. These lines are floated after a Reset or when the bus is not acquired.</td>
</tr>
<tr>
<td>BE3–BE0</td>
<td>109–114</td>
<td>O</td>
<td><strong>BYTE ENABLE.</strong> These tri-stated signals are used to indicate which bytes are involved with the current memory access. The number of Byte Enable signals asserted indicates the physical size of the data being transferred (1, 2, 3, or 4 bytes).</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- BE0 indicates D7 – D0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- BE1 indicates D15 – D8</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- BE2 indicates D23 – D16</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- BE3 indicates D31 – D24</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>These lines are floated after a Reset or when the bus is not acquired.</td>
</tr>
<tr>
<td>W/R</td>
<td>120</td>
<td>O</td>
<td><strong>WRITE/READ.</strong> This dual function pin is used to distinguish Write and Read cycles. This line is floated after a Reset or when the bus is not acquired.</td>
</tr>
</tbody>
</table>
### PIN DESCRIPTIONS (Continued)

<table>
<thead>
<tr>
<th>Symbol</th>
<th>PQFP Pin No.</th>
<th>Type</th>
<th>Name and Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADS</td>
<td>124</td>
<td>O</td>
<td>ADDRESS STATUS. The 82596 uses this tri-state pin to indicate to indicate that a valid bus cycle has begun and that A31–A2, BE3–BE0, and W/R are being driven. It is asserted during t1 bus states. This line is floated after a Reset or when the bus is not acquired.</td>
</tr>
<tr>
<td>RDY</td>
<td>130</td>
<td>I</td>
<td>READY. Active low. This signal is the acknowledgment from addressed memory that the transfer cycle can be completed. When high, it causes wait states to be inserted. It is ignored at the end of the first clock of the bus cycle’s data cycle. This active-low signal does not have an internal pull-up resistor. This signal must meet the setup and hold times to operate correctly.</td>
</tr>
<tr>
<td>BRDY</td>
<td>2</td>
<td>I</td>
<td>BURST READY. Active low. Burst Ready, like RDY, indicates that the external system has presented valid data on the data pins in response to a Read, or that the external system has accepted the 82596 data in response to a Write request. Also, like RDY, this signal is ignored at the end of the first clock in a bus cycle. If the 82596 can still receive data from the previous cycle, ADS will not be asserted in the next clock cycle; however, Address and Byte Enable will change to reflect the next data item expected by the 82596. BRDY will be sampled during each succeeding clock and if active, the data on the pins will be strobed to the 82596 or to external memory (read/write). BRDY operates exactly like READY during the last data cycle of a burst sequence and during nonburstable cycles.</td>
</tr>
<tr>
<td>BLAST</td>
<td>128</td>
<td>O</td>
<td>BURST LAST. A signal (active low) on this tri-state pin indicates that the burst cycle is finished and when BRDY is next returned it will be treated as a normal ready; i.e., another set of addresses will be driven with ADS or the bus will go idle. BLAST is not asserted if the bus is not acquired.</td>
</tr>
<tr>
<td>A HOLD</td>
<td>117</td>
<td>I</td>
<td>ADDRESS HOLD. This hold signal is active high, it allows another bus master to access the 82596 address bus. In a system where an 82596 and an i486 processor share the local bus, A HOLD allows the cache controller to make a cache invalidation cycle while the 82596 holds the address lines. In response to a signal on this pin, the 82596 immediately (i.e., during the next clock) stops driving the entire address bus (A31–A2); the rest of the bus can remain active. For example, data can be returned for a previously specified bus cycle during Address Hold. The 82596 will not begin another bus cycle while A HOLD is active.</td>
</tr>
<tr>
<td>BOFF</td>
<td>116</td>
<td>I</td>
<td>BACKOFF. This signal is active low, it informs the 82596 that another bus master requires access to the bus before the 82596 bus cycle completes. The 82596 immediately (i.e., during the next clock) floats its bus. Any data returned to the 82596 while BOFF is asserted is ignored. BOFF has higher priority than RDY or BRDY; if two such signals are returned in the same clock period, BOFF is given preference. The 82596 remains in Hold until BOFF goes high, then the 82596 resumes its bus cycle by driving out the address and status, and asserting ADS.</td>
</tr>
<tr>
<td>LOCK</td>
<td>126</td>
<td>O</td>
<td>LOCK. This tri-state pin is used to distinguish locked and unlocked bus cycles. LOCK generates a semaphore handshake to the CPU. LOCK can be active for several memory cycles, it goes active during the first locked memory cycle (t1) and goes inactive at the last locked cycle (t2). This line is floated after a Reset or when the bus is not acquired. LOCK can be disabled via the sysbus byte in software.</td>
</tr>
</tbody>
</table>
### PIN DESCRIPTIONS (Continued)

<table>
<thead>
<tr>
<th>Symbol</th>
<th>PQFP Pin No.</th>
<th>Type</th>
<th>Name and Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>BS16</td>
<td>129</td>
<td>I</td>
<td><strong>BUS SIZE.</strong> This signal allows the 82596CA to work with either 16- or 32-bit bytes. Inserting BS16 low causes the 82596 to perform two 16-bit memory accesses when transferring 32-bit data. In little endian mode the D15–D0 lines are driven when BS16 is inserted, in Big Endian mode the D31–D16 lines are driven.</td>
</tr>
<tr>
<td>HOLD</td>
<td>123</td>
<td>O</td>
<td><strong>HOLD.</strong> The HOLD signal is active high, the 82596 uses it to request local bus mastership. In normal operation HOLD goes inactive before HLDA. The 82596 can be forced off the bus by deasserting HLDA or if the bus throttle timers expire.</td>
</tr>
<tr>
<td>HLDA</td>
<td>118</td>
<td>I</td>
<td><strong>HOLD ACKNOWLEDGE.</strong> The HLDA signal is active high, it indicates that bus mastership has been given to the 82596. HLDA is internally synchronized; after HOLD is detected low, the CPU drives HLDA low.</td>
</tr>
<tr>
<td>BREQ</td>
<td>115</td>
<td>I</td>
<td><strong>BUS REQUEST.</strong> This signal, when configured to an externally activated mode, is used to trigger the bus throttle timers.</td>
</tr>
<tr>
<td>PORT</td>
<td>3</td>
<td>I</td>
<td><strong>PORT.</strong> When this signal is received, the 82596 latches the data on the data bus into an internal 32-bit register. When the CPU is asserting this signal it can write into the 82596 (via the data bus). This pin must be activated twice during all CPU Port access commands.</td>
</tr>
<tr>
<td>RESET</td>
<td>69</td>
<td>I</td>
<td><strong>RESET.</strong> This active high, internally synchronized signal causes the 82596 to terminate current activity. The signal must be high for at least five system clock cycles. After five system clock cycles and four TxC clock cycles the 82596 will execute a Reset when it receives a high RESET signal. When RESET returns to low the 82596 waits for the first CA signal and then begins the initialization sequence.</td>
</tr>
<tr>
<td>LE/BE</td>
<td>65</td>
<td>I</td>
<td><strong>LITTLE ENDIAN/BIG ENDIAN.</strong> This dual-function pin is used to select byte ordering. When LE/BE is high, little endian byte ordering is used; when low, big endian byte ordering is used for data in frames (bytes) and for control (SCB, RFD, CBL, etc).</td>
</tr>
<tr>
<td>CA</td>
<td>119</td>
<td>I</td>
<td><strong>CHANNEL ATTENTION.</strong> The CPU uses this pin to force the 82596 to begin executing memory resident Command blocks. The CA signal is internally synchronized. The signal must be high for at least one system clock. It is latched internally on the high to low edge and then detected by the 82596. The first CA after a Reset forces the 82596 into the initialization sequence beginning at location 00FFFFFF6h or an SCP address written to the 82596 using CPU Port access. All subsequent CA signals cause the 82596 to begin executing new command sequences from the SCB.</td>
</tr>
<tr>
<td>INT/INT</td>
<td>125</td>
<td>O</td>
<td><strong>INTERRUPT.</strong> A high signal on this pin notifies the CPU that the 82596 is requesting an interrupt. This signal is an edge triggered interrupt signal, and can be configured to be active high or low.</td>
</tr>
</tbody>
</table>
# PIN DESCRIPTIONS (Continued)

<table>
<thead>
<tr>
<th>Symbol</th>
<th>PQFP Pin No.</th>
<th>Type</th>
<th>Name and Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>VCC</td>
<td>18 Pins</td>
<td></td>
<td>POWER. +5 V ± 10%.</td>
</tr>
<tr>
<td>VSS</td>
<td>18 Pins</td>
<td></td>
<td>GROUND. 0 V.</td>
</tr>
<tr>
<td>TxD</td>
<td>54</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>TxC</td>
<td>64</td>
<td>I</td>
<td>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. Transmit clock must always be running for proper device operation.</td>
</tr>
<tr>
<td>LPBK</td>
<td>58</td>
<td>O</td>
<td>LOOPBACK. This TTL-level control signal enables the loopback mode. In this mode serial data on the TxD input is routed through the 82C501 internal circuits and back to the RxD output without driving the transceiver cable. To enable this signal, both internal and external loopback need to be set with the Configure command.</td>
</tr>
<tr>
<td>RxD</td>
<td>60</td>
<td>I</td>
<td>RECEIVE DATA. This pin receives NRZ serial data only. It must be high when not receiving.</td>
</tr>
<tr>
<td>RXC</td>
<td>59</td>
<td>I</td>
<td>RECEIVE CLOCK. This signal provides timing information to the internal shifting logic. For NRZ data the state of the RxD pin is sampled on the high to low transition of the clock.</td>
</tr>
<tr>
<td>RTS</td>
<td>57</td>
<td>O</td>
<td>REQUEST TO SEND. When this signal is low the 82596 informs the external interface that it has data to transmit. It is forced high after a Reset or when transmission is stopped.</td>
</tr>
<tr>
<td>CTS</td>
<td>62</td>
<td>I</td>
<td>CLEAR TO SEND. An active-low signal that enables the 82596 to send data. It is normally used as an interface handshake to RTS. Asserting CTS high stops transmission. CTS is internally synchronized. If CTS goes inactive, meeting the setup time to the TxC negative edge, the transmission will stop and RTS will go inactive within, at most, two TxC cycles.</td>
</tr>
<tr>
<td>CRS</td>
<td>63</td>
<td>I</td>
<td>CARRIER SENSE. This signal is active low, it is used to notify the 82596 that traffic is on the serial link. It is only used if the 82596 is configured for external Carrier Sense. In this configuration external circuitry is required for detecting traffic on the serial link. CRS is internally synchronized. To be accepted, the signal must remain active for at least two serial clock cycles (for CRSF = 0).</td>
</tr>
<tr>
<td>CDT</td>
<td>61</td>
<td>I</td>
<td>COLLISION DETECT. This active-low signal informs the 82596 that a collision has occurred. It is only used if the 82596 is configured for external Collision Detect. External circuitry is required for collision detection. CDT is internally synchronized. To be accepted, the signal must remain active for at least two serial clock cycles (for CDTF = 0).</td>
</tr>
</tbody>
</table>
82596 AND HOST CPU INTERACTION

The 82596CA and the host CPU communicate through shared memory. Because of its on-chip DMA capability, the 82596 can make data block transfers (buffers and frames) independently of the CPU; this greatly reduces the CPU byte transfer overhead.

The 82596 is a multitasking coprocessor that comprises two independent logical units—the Command Unit (CU) and the Receive Unit (RU). The CU executes commands from shared memory. The RU handles all activities related to frame reception. The independence of the CU and RU enables the 82596 to engage in both activities simultaneously—the CU can fetch and execute commands from memory while the RU is storing received frames in memory. The CPU is only involved with this process after the CU has executed a sequence of commands or the RU has finished storing a sequence of frames.

The CPU and the 82596 use the hardware signals Interrupt (INT) and Channel Attention (CA) to initiate communication with the System Control Block (SCB), see Figure 4. The 82596 uses INT to alert the CPU of a change in the contents of the SCB, the CPU uses CA to alert the 82596.

The 82596 has a CPU Port Access state that allows the CPU to execute certain functions without accessing memory. The 82596 PORT pin and data bus pins are used to enable this feature. The CPU can directly activate four operations when the 82596 is in this state.

- Write an alternative System Configuration Pointer (SCP). This can be used when the 82596 cannot use the default SCP address space.
- Write a different Dump Command Pointer and execute Dump. This can be used for troubleshooting No Response problems.
- The CPU can reset the 82596 via software without disturbing the rest of the system.
- A self-test can be used for board testing; the 82596 will execute a self-test and write the results to memory.

82596 BUS INTERFACE

The 82596CA has bus interface timings and pin definitions that are compatible with Intel's 32-bit i486 microprocessor. This eliminates the need for additional bus interface logic. Operating at 33 MHz, the 82596's bus bandwidth can be as high as 106 MB/s. Since Ethernet only requires 1.25 MB/s, this leaves a considerable amount of bandwidth for the CPU. The 82596 also has a bus throttle to regulate its use of the bus. Two timers can be programmed through the SCB: one controls the maximum time the 82596 can remain on the bus, the other controls the time the 82596 must stay off the bus (see Figure 5). The bus throttle can be programmed to trigger internally with HLDA or externally with BREQ. These timers can restrict the 82596 HOLD activation time and improve bus utilization.

82596 MEMORY ADDRESSING

The 82596 has a 32-bit memory address range, which allows addressing up to four gigabytes of memory. The 82596 has three memory addressing modes (see Table 1).

- **82586 Mode.** The 82596 has a 24-bit memory address range. The System Control Block, Command List, Receive Descriptor List, and Buffer Descriptors must reside in one 64-KB memory segment. Transmit and Receive buffers can reside in a 24-bit address space.
- **32-Bit Segmented Mode.** The 82596 has a 32-bit memory address range. The System Control Block, Command List, Receive Descriptor List, and Buffer Descriptors must reside in one 64-KB memory segment. Transmit and Receive buffers can reside in a 32-bit address space.
- **Linear Mode.** The 82596 has a 32-bit memory address range. Any memory structure can reside anywhere within the 32-bit memory address range.
Figure 4. 82596 and Host CPU Intervention

Figure 5. Bus Throttle Timers

Table 1. 82596 Memory Addressing Formats

<table>
<thead>
<tr>
<th>Pointer or Offset</th>
<th>82586</th>
<th>32-Bit Segmented</th>
<th>Linear</th>
</tr>
</thead>
<tbody>
<tr>
<td>ISCP Address</td>
<td>24-Bit Linear</td>
<td>32-Bit Linear</td>
<td>32-Bit Linear</td>
</tr>
<tr>
<td>SCB Address</td>
<td>Base (24) + Offset (16)</td>
<td>Base (32) + Offset (16)</td>
<td>32-Bit Linear</td>
</tr>
<tr>
<td>Command Block Pointers</td>
<td>Base (24) + Offset (16)</td>
<td>Base (32) + Offset (16)</td>
<td>32-Bit Linear</td>
</tr>
<tr>
<td>Rx Frame Descriptors</td>
<td>Base (24) + Offset (16)</td>
<td>Base (32) + Offset (16)</td>
<td>32-Bit Linear</td>
</tr>
<tr>
<td>Tx Frame Descriptors</td>
<td>Base (24) + Offset (16)</td>
<td>Base (32) + Offset (16)</td>
<td>32-Bit Linear</td>
</tr>
<tr>
<td>Rx Buffer Descriptors</td>
<td>Base (24) + Offset (16)</td>
<td>Base (32) + Offset (16)</td>
<td>32-Bit Linear</td>
</tr>
<tr>
<td>Tx Buffer Descriptors</td>
<td>Base (24) + Offset (16)</td>
<td>Base (32) + Offset (16)</td>
<td>32-Bit Linear</td>
</tr>
<tr>
<td>Rx Buffers</td>
<td>24-Bit Linear</td>
<td>32-Bit Linear</td>
<td>32-Bit Linear</td>
</tr>
<tr>
<td>Tx Buffers</td>
<td>24-Bit Linear</td>
<td>32-Bit Linear</td>
<td>32-Bit Linear</td>
</tr>
</tbody>
</table>
82596 SYSTEM MEMORY STRUCTURE

The Shared Memory structure consists of four parts: the Initialization Root, the System Control Block, the Command List, and the Receive Frame Area (see Figure 6).

The Initialization Root is in an established location known to the host CPU and the 82596 (00FFFFF6h). However, the CPU can establish the Initialization Root in another location by using the CPU Port access. This root is accessed during initialization, and points to the System Control Block.

Figure 6. 82596 Shared Memory Structure

The System Control Block serves as a bidirectional mail drop for the host CPU and the 82596 CU and RU. It is the central point through which the CPU and the 82596 exchange control and status information. The SCB has two areas. The first contains instructions from the CPU to the 82596. These include: control of the CU and RU (Start, Abort, Suspend, and Resume), a pointer to the list of CU commands, a pointer to the Receive Frame Area, a set of Interrupt Acknowledge bits, and the T-ON and T-OFF timers for the bus throttle. The second area contains status information the 82596 is sending to the CPU. Such as, the CU and RU states (Idle, Active
Ready, Suspended, No Receive Resources, etc.), interrupt bits (Command Completed, Frame Received, CU Not Ready, and RU Not Ready), and statistical counters.

The Command List functions as a program for the CU; individual commands are placed in memory units called Command Blocks (CBs). These CBs contain the parameters and status of specific high-level commands called Action Commands; e.g., Transmit or Configure.

Transmit causes the 82596 to transmit a frame. The Transmit CB contains the destination address, the length field, and a pointer to a list of linked buffers holding the frame that is to be constructed from several buffers scattered throughout memory. The Command Unit operates without CPU intervention; the DMA for each buffer, and the prefetching of references to new buffers, is performed in parallel. The CPU is notified only after a transmission is complete.

The Receive Frame Area is a list of Free Frame Descriptors (descriptors not yet used) and a list of user-prepared buffers. Frames arrive at the 82596 unsolicited; the 82596 must always be ready to receive and store them in the Free Frame Area. The Receive Unit fills the buffers when it receives frames, and reformats the Free Buffer List into received-frame structures. The frame structure is, for all practical purposes, identical to the format of the frame to be transmitted. The first Frame descriptor is referenced by the SCB. Unless the 82596 is configured to Save Bad Frames, the frame descriptor, and the associated buffer descriptor, which is wasted when a bad frame is received, are automatically reclaimed and returned to the Free Buffer List.

Receive buffer chaining (storing incoming frames in a linked buffer list) significantly improves memory utilization. Without buffer chaining, the user must allocate consecutive blocks of memory, each capable of containing a maximum frame (for Ethernet, 1518 bytes). Since an average frame is about 200 bytes, this is very inefficient. With buffer chaining, the user can allocate small buffers and the 82596 will only use those that are needed.

Figure 7 A–D illustrates how the 82596 uses the Receive Frame Area. Figure 7A shows an unused Receive Frame Area composed of Free Frame Descriptors and Free Receive Buffers prepared by the user. The SCB points to the first Frame Descriptor of the Frame Descriptor List. Figure 7B shows the same Receive Frame Area after receiving one frame. This first frame occupies two Receive Buffers and one Frame Descriptor—a valid received frame will only occupy one Frame Descriptor. After receiving this frame the 82596 sets the next Free Frame Descriptor RBD pointer to the next Free RBD. Figure 7C shows the RFA after receiving a second frame. In this example the second frame occupies only one Receive Buffer and one RFD. The 82596 again sets the RBD pointer. This process is repeated again in Figure 7D, showing the reception of another frame using one Receive Buffer; in this example there is an extra Frame Descriptor.

TRANSMIT AND RECEIVE MEMORY STRUCTURES

There are three memory structures for reception and transmission. The 82586 memory structure, the Flexible memory structure, and the Simplified memory structure. The 82586 mode is selected by configuring the 82596 during initialization. In this mode all the 82596 memory structures are compatible with the 82586 memory structures.

When the 82596 is not configured to the 82586 mode, the other two memory structures, Simplified and Flexible, are available for transmitting and receiving. These structures can be selected on a frame-by-frame basis by setting the S/F bit in the Transmit Command and the Receive Frame Descriptor (see Figures 29, 30, 41, and 42). The Simplified memory structure offers a simple structure for ease of programming (see Figure 8). All information about a frame is contained in one structure; for example, during reception the RFD and data field are contained in one structure.

The Flexible memory structure (see Figure 9) has a control field that allows the programmer to specify the amount of receive data the RFD will contain for receive operations and the amount of transmit data the Transmit Command Block will contain for transmit operations. For example, when the control field in the RFD is set to 20 bytes during a reception, the first 20 bytes of the data field are stored in the RFD (6 bytes of destination address, 6 bytes of source address, 2 bytes of length field, and 6 bytes of data) and the remainder of the data field is stored in the Receive Data Buffers. This is useful for capturing frame headers when header information is contained in the data field. The header information can then be automatically stored in the RFD partitioned from the Receive Data Buffer.

The control field can also be used for the Transmit Command when the Flexible memory structure is used. The quantity of data field bytes to be transmitted from the Transmit Command Block is specified by the variable control field.
Figure 7. Frame Reception in the RFA
Figure 8. Simplified Memory Structure

Figure 9. Flexible Memory Structure
TRANSMITTING FRAMES

The 82596 executes high-level Action Commands from the Command List in system memory. Action Commands are fetched and executed in parallel with the host CPU operation, thereby significantly improving system performance. The format of the Action Commands is shown in Figure 10. Figure 28 shows the 82586 mode, and Figures 29 and 30 show the command formats of the Linear and 32-bit Segmented modes.

A single Transmit command contains, as part of the command-specific parameters, the destination address and length field of the transmitted frame and a pointer to buffer area in memory containing the data portion of the frame. The data field is contained in a memory data structure consisting of a buffer descriptor (BD) and a data buffer—or a linked list of buffer descriptors and buffers—as shown in Figure 11.

Multiple data buffers can be chained together using the BDs. Thus, a frame with a long data field can be transmitted using several (shorter) data buffers chained together. This chaining technique allows the system designer to develop efficient buffer management.

The 82596 automatically generates the preamble (alternating 1s and 0s) 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 specified by the Transmit command, and computes and appends the CRC to the end of the frame (see Figure 12). In the Linear and 32-bit Segmented mode the CRC can be optionally inserted on a frame-by-frame basis by setting the NC bit in the Transmit Command Block (see Figures 29 and 30).

The 82596 can be configured to generate two types of start and end frame delimiters—End of Carrier (EOC) or HDLC. In EOC mode the start frame delimiter is 10101011 and the end frame delimiter is indicated by the lack of a signal after the last bit of the frame check sequence field has been transmitted. In EOC mode the 82596 can be configured to extend short frames by adding pad bytes (7Eh) during transmission, according to the length field. In HDLC mode the 82596 will generate the 01111110 flag for the start and end frame delimiters, and do standard bit stuffing and stripping. Furthermore, the 82596 can be configured to pad frames shorter than the specified minimum frame length by appending the appropriate number of flags to the end of the frame.

When a collision occurs, the 82596 manages the jam, random wait, and retry processes, 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 larger than the maximum frame size (1518 bytes for Ethernet).

The 82596 can be configured to send messages larger than the maximum frame size by appending the appropriate number of flags to the end of the frame.
RECEIVING FRAMES

To reduce CPU overhead, the 82596 is designed to receive frames without CPU supervision. The host CPU first sets aside an adequate receive buffer space and then enables the 82596 Receive Unit. Once enabled, the RU watches for arriving frames and automatically stores them in the Receive Frame Area (RFA). The RFA contains Receive Frame Descriptors, Receive Buffer Descriptors, and Data Buffers (see Figure 13). The individual Receive Frame Descriptors make up a Receive Descriptor List (RDL) used by the 82596 to store the destination and source addresses, the length field, and the status of each frame received (see Figure 14).

Once enabled, the 82596 checks each passing frame for an address match. The 82596 will recognize its own unique address, one or more multicast addresses, or the broadcast address. If a match is found the 82596 stores the destination and source addresses and the length field in the next available RFD. It then begins filling the next available Data Buffer on the FBL, which is pointed to by the current RFD, with the data portion of the incoming frame. As one Data Buffer is filled, the 82596 automatically fetches the next DB on the FBL until the entire frame is received. This buffer chaining technique is particularly memory efficient because it allows the system designer to set aside buffers to fit frames much shorter than the maximum allowable frame length. If AL-LOC = 1, or if the flexible memory structure is used, the addresses and length field can be placed in the Receive Buffer.

Once the entire frame is received without error, the 82596 does the following housekeeping tasks.

- The actual count field of the last Buffer Descriptor used to hold the frame just received is updated with the number of bytes stored in the associated Data Buffer.
- The next available Receive Frame Descriptor is fetched.
- The address of the next available Buffer Descriptor is written to the next available Receive Frame Descriptor.
- A frame received interrupt status bit is posted in the SCB.
- An interrupt is sent to the CPU.

If a frame error occurs, for example a CRC error, the 82596 automatically reinitializes its DMA pointers and reclaims any data buffers containing the bad frame. The 82596 will continue to receive frames without CPU help as long as Receive Frame Descriptors and Data Buffers are available.

82596 NETWORK MANAGEMENT AND DIAGNOSTICS

The behavior of data communication networks is normally very complex because of their distributed and asynchronous nature. It is particularly difficult to pinpoint a failure when it occurs. The 82596 has extensive diagnostic and network management functions that help improve reliability and testability. The 82596 reports on the following events after each frame is transmitted.

- Transmission successful.
- Transmission unsuccessful. Lost Carrier Sense.
- Transmission unsuccessful. Lost Clear to Send.
- Transmission unsuccessful. A DMA underrun occurred because the system bus did not keep up with the transmission.
- Transmission unsuccessful. The number of collisions exceeded the maximum allowed.
- Number of Collisions. The number of collisions experienced during the frame.
- Heartbeat Indicator. This indicates the presence of a heartbeat during the last Interframe Spacing (IFS) after transmission.

When configured to Save Bad Frames the 82596 checks each incoming frame and reports the following errors.

- CRC error. Incorrect CRC in a properly aligned frame.
- Alignment error. Incorrect CRC in a misaligned frame.
- Frame too short. The frame is shorter than the value configured for minimum frame length.
- Overrun. Part of the frame was not placed in memory because the system bus did not keep up with incoming data.
- Out of buffer. Part of the frame was discarded because of insufficient memory storage space.
- Receive collision. A collision was detected during reception.
- Length error. A frame not matching the frame length parameter was detected.
Figure 13. Receive Frame Area Diagram

Figure 14. Receive Frame Descriptor
NETWORK PLANNING AND MAINTENANCE

To properly plan, operate, and maintain a communication network, the network management entity must accumulate information on network behavior. The 82596 provides a rich set of network-wide diagnostics that can serve as the basis for a network management entity.

Information on network activity is provided in the status of each frame transmitted. The 82596 reports the following activity indicators after each frame:

- Number of collisions. The number of collisions the 82596 experienced while attempting to transmit the frame.
- Deferred transmission. During the first transmission attempt the 82596 had to defer to traffic on the link.

The 82596 updates its 32-bit statistical counters after each received frame that both passes address filtering and is longer than the Minimum Frame Length configuration parameter. The 82596 reports the following statistics:

- CRC errors. The number of well-aligned frames that experienced a CRC error.
- Alignment errors. The number of misaligned frames that experienced a CRC error.
- No resources. The number of frames that were discarded because of insufficient resources for reception.
- Overrun errors. The number of frames that were not completely stored in memory because the system bus did not keep up with incoming data.
- Receive Collision counter. The number of collisions detected during receive.
- Short Frame counter. The number of frames that were discarded because they were shorter than the configured minimum frame length.

The 82596 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 when implementing a monitoring station to capture all frames for analysis.

A useful method of capturing frame headers is to use the Simplified memory mode, configure the 82596 to Save Bad Frames, and configure the 82596 to Promiscuous mode with space in the RFD allocated for specific number of receive data bytes.

The 82596 will receive all frames and put them in the RFD. Frames that exceed the available space in the RFD will be truncated, the status will be updated, and the 82596 will retrieve the next RFD. This allows the user to capture the initial data bytes of each frame (for instance, the header) and discard the remainder of the frame.

The 82596 also has a monitor mode for network analysis. During normal operation the receive function enables the 82596 to receive frames that pass address filtering. These frames must have the Start of Frame Delimiter (SFD) field and must be longer than the absolute minimum frame length of 5 bytes (6 bytes in case of Multicast address filtering). Contents and status of the received frames are transferred to memory. The monitor function enables the 82596 to simply evaluate the incoming frames. The 82596 can monitor the frames that pass or do not pass the address filtering. It can also monitor frames which do not have the SFD fields. The 82596 can be configured to only keep statistical information about monitor frames. Three options are available in the Monitor mode. These options are selected by the two monitor mode configuration bits available in the configuration command.

When the first option is selected, the 82596 receives good frames that pass address filtering and transfers them to memory while monitoring frames that do not pass address filtering or are shorter than the minimum frame size (these frames are not transferred to memory). When this option is used the 82596 updates six counters: CRC errors, alignment errors, no resource errors, overrun errors, short frames and total good frames received.

When the second option is selected, the receive function is completely disabled. The 82596 monitors only those frames that pass address filterings and meet the minimum frame length requirement. When this option is used the 82596 updates six counters: CRC errors, alignment errors, total frames (good and bad), short frames, collisions detected and total good frames.

When the third option is selected, the receive function is completely disabled. The 82596 monitors all frames, including frames that do not have a Start Frame Delimiter. When this option is used the 82596 updates six counters: CRC errors, alignment errors, total frames (good and bad), short frames, collisions detected and total good frames.
STATION DIAGNOSTICS AND SELF-TEST

The 82596 provides a large set of diagnostic and network management functions. These include internal and external loopback and time domain reflectometry for locating fault points in the network cable. The 82596 ensures software reliability by dumping the contents of the 82596 internal registers into system memory. The 82596 has a self-test mode that enables it to run an internal self-test and place the results in system memory.

82586 SOFTWARE COMPATIBILITY

The 82596 has a software-compatible state in which all its memory structures are compatible with the 82586 memory structure. This includes all the Action Commands, the Receive Frame Area (including the RFD, Buffer Descriptors, and Data Buffers), the System Control Block, and the initialization procedures. There are two minor differences between the 82596 in the 82586-Compatible memory structure and the 82586.

- When the internal and external loopback bits in the Configure command are set to 11 the 82596 is in external loopback and the LPBK pin is activated; in the 82586 this situation would produce internal loopback.
- During a Dump command both the 82596 and 82586 dump the same number of bytes; however, the data format is different.

INITIALIZING THE 82596

A Reset command is issued to the 82596 to prepare it for normal operation. The 82596 is initialized through two data structures that are addressed by two pointers, the System Configuration Pointer (SCP) and the Intermediate System Configuration Pointer (ISCP). The initialization procedure begins when a Channel Attention signal is asserted after RESET. The 82596 uses the address of the double word that contains the SCP as a default—00FFFFFF4h. Before the CA signal is asserted this default address can be changed to any other available address by asserting the PORT pin and providing the desired address over the D31–D4 pins of the address bus. Pins D3—D0 must be 0010; i.e., any alternative address must be aligned to 16-byte boundaries. All addresses sent to the 82596 must be word aligned, which means that all pointers and memory structures must start on an even address (A0 = zero).

SYSTEM CONFIGURATION POINTER (SCP)

The SCP contains the sysbus byte and the location of the next structure of the initialization process, the ISCP. The following parameters are selected in the SYSBUS.

- The 82596 operation mode.
- The Bus Throttle timer triggering method.
- Lock enabled.
- Interrupt polarity.

Byte ordering is determined by the LE/BE pin. LE/BE = 1 selects Little Endian byte ordering and LE/BE = 0 selects Big Endian byte ordering.

NOTE:

In the following, X indicates a bit not checked 82586 mode. This bit must be set to 0 in all other modes.
The following diagram illustrates the format of the SCP.

<table>
<thead>
<tr>
<th>31</th>
<th>ODD WORD</th>
<th>16 15</th>
<th>EVEN WORD</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>X</td>
<td>X X X</td>
<td>X X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>X</td>
<td>X X X X</td>
<td>X X</td>
<td>X X X X X</td>
<td>X</td>
</tr>
<tr>
<td>A31</td>
<td>A24</td>
<td>A23</td>
<td>ISCP ADDRESS</td>
<td>A0</td>
</tr>
</tbody>
</table>

A31 .............. A24 are not checked in 82586 mode
X .................. X areas are not checked in 82586 mode; they must be 0 in all other modes.

Figure 15. The System Configuration Pointer

Writing the Sysbus

When writing the sysbus byte it is important to pay attention to the byte order.

- When a Little Endian processor is used, the sysbus byte is located at byte address 00FF F FF 6 h (or address \( n + 2 \) if an alternative SCP address \( n \) was programmed).
- When a processor using Big Endian byte ordering is used, the sysbus, alternative SCP, and ISCP addresses will be different.
  - The sysbus byte is located at 00 F F F FF 5 h.
  - If an alternative SCP address is programmed, the sysbus byte should be at byte address \( n + 1 \).
INTERMEDIATE SYSTEM CONFIGURATION POINTER (ISCP)

The ISCP indicates the location of the System Control Block. Often the SCP is in ROM and the ISCP is in RAM. The CPU loads the SCP address (or an equivalent data structure) into the ISCP and asserts CA. This Channel Attention signal causes the 82596 to begin its initialization procedure and to get the SCP address from the ISCP and SCP. In 82586 and 32-bit Segmented modes the SCP base address is also the base address of all Command Blocks, Frame Descriptors, and Buffer Descriptors (but not buffers). All these data structures must reside in one 64-KB segment; however, in Linear mode no such limitation is imposed.

The following diagram illustrates the ISCP format.

```
<table>
<thead>
<tr>
<th>ODD WORD</th>
<th>EVEN WORD</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>16 15</td>
</tr>
<tr>
<td></td>
<td>8 7</td>
</tr>
<tr>
<td>A15</td>
<td>A0 BUSY</td>
</tr>
<tr>
<td>A23</td>
<td>SCB ADDRESS</td>
</tr>
</tbody>
</table>

0 0 0 0 0 0 0 0 0 0 — in 82586 mode
A31 ................. A24 — in all other modes

BUSY — Indicates that the 82596 is being initialized. The CPU sets the ISCP to 01h before it gives the first CA to the 82596. The ISCP is cleared by the 82596 after the SCP base and offset are read. Note that the most significant byte of the first word of the ISCP is not modified when BUSY is cleared.

SCB OFFSET — This 16-bit quantity specifies the offset portion of the address of the SCB.

SCB BASE — Specifies the base portion of the address of the SCB. The base of SCB is also the base of all 82596 Command Blocks, Frame Descriptors and Buffer Descriptors. In the 82586 mode, bits A31–A24 are considered to be zero.
```

Figure 16. The Intermediate System Configuration Pointer—82586 and 32-Bit Segmented Modes

INITIALIZATION PROCESS

The CPU sets up the SCP, ISCP, and the SCB structures, and, if desired, an alternative SCP address. It also sets BUSY to 01h. The 82596 is initialized when a Channel Attention signal follows a Reset signal, causing the 82596 to access the System Configuration Pointer. The sysbus byte, the operational mode, the bus throttle timer triggering method, the interrupt polarity, and the state of LOCK are read. After reset the Bus Throttle timers are essentially disabled—the T-ON value is infinite, the T-OFF value is zero. After the SCP is read, the 82596 reads the ISCP and saves the SCB address. In 82586 and 32-bit Segmented modes this address is represented as a base address plus the offset (this base address is also the base address of all the control blocks). In Linear mode the base address is also an absolute address. The 82596 clears BUSY, sets CX and CNR to equal 1 in the SCB, clears the SCB command word, sends an interrupt to the CPU, and awaits another Channel Attention signal. RESET configures the 82596 to its default state before CA is asserted.
CONTROLLING THE 82596CA

The host CPU controls the 82596 with the commands, data structures, and methods described in this section. The CPU and the 82596 communicate through shared memory structures. The 82596 contains two independent units: the Command Unit and the Receive Unit. The Command Unit executes commands from the CPU, and the Receive Unit handles frame reception. These two units are controlled and monitored by the CPU through a shared memory structure called the System Control Block (SCB). The CPU and the 82596 use the CA and INT signals to communicate with the SCB.

82596 CPU ACCESS INTERFACE (PORT)

The 82596 has a CPU access interface that allows the host CPU to do four things.
- Write an alternative System Configuration Pointer address.
- Write an alternative Dump area pointer and perform Dump.
- Execute a software reset.
- Execute a self-test.

The following events initiate the CPU access state.
- Presence of an address on the D31–D4 data bus pins.
- The D2–D0 pins are used to select one of the four functions.
- The PORT input pin is asserted, as in a regular write cycle.

**NOTE.** The SCP Dump and Self-Test addresses must be 16-byte aligned.

The 82596 requires two 16-bit write cycles for a port command. The first write holds the internal machines and reads the first 16 bits; the second activates the PORT command and reads the second 16 bits.

The PORT Reset is useful when only the 82596 needs to be reset. The CPU must wait for 10-system and 5-serial clocks before issuing another CA to the 82596; this new CA begins a new initialization process.

The Dump function is useful for troubleshooting No Response problems. If the chip is in a No Response state, the PORT Dump operation can be executed and a PORT Reset can be used to reinitialize the 82596 without disturbing the rest of the system.

The Self-Test function can be used for board testing; the 82596 will execute a self-test and write the results to memory.

<table>
<thead>
<tr>
<th>Function</th>
<th>D31</th>
<th>D4</th>
<th>D3</th>
<th>D2</th>
<th>D1</th>
<th>D0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reset</td>
<td>Don't Care</td>
<td>A4</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Self-Test</td>
<td>Self-Test Results Address</td>
<td>A4</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>SCP</td>
<td>Alternative SCP Address</td>
<td>A4</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Dump</td>
<td>Dump Area Pointer</td>
<td>A4</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
MEMORY ADDRESSING FORMATS

The 82596 accesses memory by 32-bit addresses. There are two types of 32-bit addresses: linear and segmented. The type of address used depends on the 82596 operating mode and the type of memory structure it is addressing. The 82596 has three operating modes.

- 82586 Mode
  - A Linear address is a single 24-bit entity. Address pins A31–A24 are always zero.
  - A Segmented address uses a 24-bit base and a 16-bit offset.

- 32-bit Segmented Mode
  - A Linear address is a single 32-bit entity.
  - A Segmented address uses a 32-bit base and a 16-bit offset.

**NOTE**

In the previous two memory addressing modes, each command header (CB, TBD, RFD, RBD, and SCB) must wholly reside within one segment. If the 82596 encounters a memory structure that does not follow this restriction, the 82596 will fetch the next contiguous location in memory (beyond the segment).

- Linear Mode
  - A Linear address is a single 32-bit entity.
  - There are no Segmented addresses.

Linear addresses are primarily used to address transmit and receive data buffers. In the 82586 and 32-bit Segmented modes, segmented addresses (base plus offset) are used for all Command Blocks, Buffer Descriptors, Frame Descriptors, and System Control Blocks. When using Segmented addresses, only the offset portion of the entity being addressed is specified in the block. The base for all offsets is the same—that of the SCB. See Table 1.

LITTLE ENDIAN AND BIG ENDIAN BYTE ORDERING

The 82596 supports both Little Endian and Big Endian byte ordering for its memory structures.

The 82596 supports Big Endian byte ordering for word and byte entities. Dword entities are not supported with Big Endian byte ordering. This results in slightly different 82596 memory structures for Big Endian operation. These structures are defined in the 82596 User's Manual.

**NOTE**

All 82596 memory entities must be word or dword aligned.

An example of a dword entity is a frame descriptor command/status dword, whereas the raw data of the frame are byte entities. Both 32- and 16-bit buses are supported. When a 16-bit bus is used with Big Endian memory organization, data lines D15–D0 are used. The 82596 has an internal crossover that handles these swap operations.

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 is associated with each Action Command. The CU is modeled as a logical machine that takes, at any given time, one of the following states.

- **Idle.** The CU is not executing a command and is not associated with a CB on the list. This is the initial state.
- **Suspended.** The CU is not executing a command; however, it is associated with a CB on the list.
- **Active.** The CU is executing an Action Command and pointing to its CB.

The CPU can affect CU operation in two ways: by issuing a CU Control Command or by setting bits in the Command word of the Action Command.
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.** The RU has no memory resources and is discarding incoming frames. This is the initial state.
- **No Resources.** The RU has no memory resources and is discarding incoming frames. This state differs from Idle in that the RU accumulates statistics on the number of discarded frames.
- **Suspended.** The RU has memory available for storing frames, but is discarding them. The suspend state can only be reached if the CPU forces this through the SCB or sets the suspend bit in the RFD.
- **Ready.** The RU has memory available and is storing incoming frames.

The CPU can affect RU operation in three ways: by issuing an RU Control Command, by setting bits in the Frame Descriptor Command word of the frame being received, or by setting the EL bit of the current buffer's Buffer Descriptor.

SYSTEM CONTROL BLOCK (SCB)

The SCB is a memory block that plays a major role in communications between the CPU and the 82596. Such communications include the following.

- Commands issued by the CPU
- Status reported by the 82596

Control commands are sent to the 82596 by writing them into the SCB and then asserting CA. The 82596 examines the command, performs the required action, and then clears the SCB command word. Control commands perform the following types of tasks.

- Operation of the Command Unit (CU). The SCB controls the CU by specifying the address of the Command Block List (CBL) and by starting, suspending, resuming, or aborting execution of CBL commands.
- Operation of the Bus Throttle. The SCB controls the Bus Throttle timers by providing them with new values and sending the Load and Start timer commands. The timers can be operated in both the 32-bit Segmented and Linear modes.
- Reception of frames by the Receive Unit (RU). The SCB controls the RU by specifying the address of the Receive Frame Area and by starting, suspending, resuming, or aborting frame reception.
- Acknowledgment of events that cause interrupts.
- Resetting the chip.

The 82596 sends status reports to the CPU via the System Control Block. The SCB contains four types of status reports.

- The cause of the current interrupts. These interrupts are caused by one or more of the following 82596 events.
  - The Command Unit completes an Action Command that has its I bit set.
  - The Receive Unit receives a frame.
  - The Command Unit becomes inactive.
  - The Receive Unit becomes not ready.
  - The status of the Command Unit.
  - The status of the Receive Unit.
  - Status reports from the 82596 regarding reception of corrupted frames.
Events can be cleared only by CPU acknowledgment. If some events are not acknowledged by the ACK field the Interrupt signal (INT) will be reissued after Channel Attention (CA) is processed. Furthermore, if a new event occurs while an interrupt is set, the interrupt is temporarily cleared to trigger edge-triggered interrupt controllers.

The CPU uses the Channel Attention line to cause the 82596 to examine the SCB. This signal is trailing-edge triggered—the 82596 latches CA on the trailing edge. The latch is cleared by the 82596 before the SCB control command is read.

### Figure 18. SCB—82586 Mode

<table>
<thead>
<tr>
<th>31</th>
<th>ODD WORD</th>
<th>16 15</th>
<th>EVEN WORD</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACK</td>
<td>X</td>
<td>CUC</td>
<td>R</td>
<td>RUČ</td>
</tr>
<tr>
<td>RFA OFFSET</td>
<td></td>
<td>CBL OFFSET</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ALIGNMENT ERRORS</td>
<td></td>
<td>CRC ERRORS</td>
<td></td>
<td></td>
</tr>
<tr>
<td>OVERRUN ERRORS</td>
<td></td>
<td>RESOURCE ERRORS</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SCB + 4</td>
<td></td>
<td>SCB + 8</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Figure 19. SCB—32-Bit Segmented Mode

<table>
<thead>
<tr>
<th>31</th>
<th>ODD WORD</th>
<th>16 15</th>
<th>EVEN WORD</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACK</td>
<td>0</td>
<td>CUC</td>
<td>R</td>
<td>RUČ</td>
</tr>
<tr>
<td>RFA OFFSET</td>
<td></td>
<td>CBL OFFSET</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CRC ERRORS</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ALIGNMENT ERRORS</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RESOURCE ERRORS (*)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>OVERRUN ERRORS (*)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RCVCVT ERRORS (*)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SHORT FRAME ERRORS</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SCB + 4</td>
<td></td>
<td>SCB + 8</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T-ON TIMER</td>
<td></td>
<td>T-OFF TIMER</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SCB + 32</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

*In monitor mode these counters change function

### Figure 20. SCB—Linear Mode

<table>
<thead>
<tr>
<th>31</th>
<th>ODD WORD</th>
<th>16 15</th>
<th>EVEN WORD</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACK</td>
<td>0</td>
<td>CUC</td>
<td>R</td>
<td>RUČ</td>
</tr>
<tr>
<td>COMMAND BLOCK ADDRESS</td>
<td></td>
<td>RECEIVING FRAME AREA ADDRESS</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CRC ERRORS</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ALIGNMENT ERRORS</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RESOURCE ERRORS (*)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>OVERRUN ERRORS (*)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RCVCVT ERRORS (*)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SHORT FRAME ERRORS</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SCB + 4</td>
<td></td>
<td>SCB + 8</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T-ON TIMER</td>
<td></td>
<td>T-OFF TIMER</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SCB + 36</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

*In MONITOR mode these counters change function
### Command Word

<table>
<thead>
<tr>
<th></th>
<th>ACK</th>
<th>0</th>
<th>CUC</th>
<th>R</th>
<th>RUC</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>SCB + 2</th>
</tr>
</thead>
</table>

These bits specify the action to be performed as a result of a CA. This word is set by the CPU and cleared by the 82596. Defined bits are:

- **Bit 31 ACK-CX** — Acknowledges that the CU completed an Action Command.
- **Bit 30 ACK-FR** — Acknowledges that the RU received a frame.
- **Bit 29 ACK-CNA** — Acknowledges that the Command Unit became not active.
- **Bit 28 ACK-RNR** — Acknowledges that the Receive Unit became not ready.
- **Bits 24–26 CUC** — (3 bits) This field contains the command to the Command Unit. Valid values are:
  - 0 — NOP (does not affect current state of the unit).
  - 1 — Start execution of the first command on the CBL. If a command is executing, complete it before starting the new CBL. The beginning of the CBL is in CBL OFFSET (address).
  - 2 — Resume the operation of the Command Unit by executing the next command. This operation assumes that the Command Unit has been previously suspended.
  - 3 — Suspend execution of commands on CBL after current command is complete.
  - 4 — Abort current command immediately.
  - 5 — Loads the Bus Throttle timers so they will be initialized with their new values after the active timer (T-ON or T-OFF) reaches Terminal Count. If no timer is active new values will be loaded immediately. This command is not valid in 82586 mode.
  - 6 — Loads and immediately restarts the Bus Throttle timers with their new values. This command is not valid in 82586 mode.
  - 7 — Reserved.

- **Bit 23 RESET** — Reset chip (logically the same as hardware RESET).
- **Bits 20–22 RUC** — (3 bits) This field contains the command to the Receive Unit. Valid values are:
  - 0 — NOP (does not alter current state of unit).
  - 1 — Start reception of frames. The beginning of the RFA is contained in the RFA OFFSET (address). If a frame is being received complete reception before starting.
  - 2 — Resume frame reception (only when in suspended state).
  - 3 — Suspend frame reception. If a frame is being received complete its reception before suspending.
  - 4 — Abort receiver operation immediately.
  - 5–7 — Reserved.
Status Word

<table>
<thead>
<tr>
<th>15</th>
<th>STAT</th>
<th>0</th>
<th>CUS</th>
<th>0</th>
<th>RUS</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>SCB</th>
</tr>
</thead>
</table>

82586 mode

<table>
<thead>
<tr>
<th>15</th>
<th>STAT</th>
<th>0</th>
<th>CUS</th>
<th>RUS</th>
<th>T</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>SCB</th>
</tr>
</thead>
</table>

32-Bit Segmented and Linear mode.

Indicates the status of the 82596. This word is modified only by the 82596. Defined bits are:

- **Bit 15 CX**: The CU finished executing a command with its I (interrupt) bit set.
- **Bit 14 FR**: The RU finished receiving a frame.
- **Bit 13 CNA**: The Command Unit left the Active state.
- **Bit 12 RNR**: The Receive Unit left the Ready state.
- **Bits 8–10 CUS**: (3 bits) This field contains the status of the command unit. Valid values are:
  - 0: Idle
  - 1: Suspended
  - 2: Active
  - 3–7: Not used
- **Bits 4–7 RUS**: This field contains the status of the receive unit. Valid values are:
  - 0h (0000): Idle
  - 1h (0001): Suspended
  - 2h (0010): No Resources. This bit indicates both no resources due to lack of RFDs in the RDL and no resources due to lack of RBDs in the FBL.
  - 4h (0100): Ready
  - 8h (1000): No more RBDs (not in the 82586 mode)
  - Ah (1010): No resources due to no more RBDs (not in the 82586 mode).
  - No other combinations are allowed
- **Bit 3 T**: Bus Throttle timers loaded (not in 82586 mode).

SCB OFFSET ADDRESSES

**CBL Offset (Address)**

In 82586 and 32-bit Segmented modes this 16-bit quantity indicates the offset portion of the address for the first Command Block on the CBL. In Linear mode it is a 32-bit linear address for the first Command Block on the CBL. It is accessed only if CUC equals Start.

**RFA Offset (Address)**

In 82586 and 32-bit Segmented modes this 16-bit quantity indicates the offset portion of the address for the Receive Frame Area. In Linear mode it is a 32-bit linear address for the Receive Frame Area. It is accessed only if RUC equals Start.
SCB STATISTICAL COUNTERS

Statistical Counter Operation

- The CPU is responsible for clearing all error counters before initializing the 82596. The 82596 updates these counters by reading them, adding 1, and then writing them back to the SCB.
- The counters are wraparound counters. After reaching FFFFFFFFh the counters wrap around to zero.
- The 82596 updates the required counters for each frame. It is possible for more than one counter to be updated; multiple errors will result in all affected counters being updated.
- The 82596 executes the read-counter/increment/write-counter operation without relinquishing the bus (locked operation). This is to ensure that no logical contention exists between the 82596 and the CPU due to both attempting to write to the counters simultaneously. In the dual-port memory configuration the CPU should not execute any write operation to a counter if LOCK is asserted.
- The counters are 32-bits wide and their behavior is fully compatible with the IEEE 802.3 standard. The 82596 supports all relevant statistics (mandatory, optional, and desired) through the status of the transmit and receive header and directly through SCB statistics.

CRCERRS

This 32-bit quantity contains the number of aligned frames discarded because of a CRC error. This counter is updated, if needed, regardless of the RU state.

ALNERRS

This 32-bit quantity contains the number of frames that both are misaligned (i.e., where CRS deasserts on a nonoctet boundary) and contain a CRC error. The counter is updated, if needed, regardless of the RU state.

SHRTFRM

This 32-bit quantity contains the number of received frames shorter than the minimum frame length.

The last three counters change function in monitor mode.

RSCERRS

This 32-bit quantity contains the number of good frames discarded because there were no resources to contain them. Frames intended for a host whose RU is in the No Receive Resources state, fall into this category. This counter is updated only if the RU is in the No Resources state. When in Monitor mode this counter counts the total number of frames—good and bad.
OVRNERRS
This 32-bit quantity contains the number of frames known to be lost because the local system bus was not available. If the traffic problem lasts longer than the duration of one frame, the frames that follow the first are lost without an indicator, and they are not counted. This counter is updated, if needed, regardless of the RU state.

RCVCDT
This 32-bit quantity contains the number of collisions detected during frame reception. In Monitor mode this counter counts the total number of good frames.

ACTION COMMANDS AND OPERATING MODES
This section lists all the Action Commands of the Command Unit Command Block List (CBL). Each command contains the Command field, the Status and Control fields, the link to the next Action Command, and any command-specific parameters. There are three basic types of action commands: 82596 Configuration and Setup, Transmission, and Diagnostics. The following is a list of the actual commands.

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

The 82596 has three addressing modes. In the 82586 mode all the Action Commands look exactly like those of the 82586.

- **82586 Mode.** The 82596 software and memory structure is compatible with the 82586.
- **32-Bit Segmented Mode.** The 82596 can access the entire system memory and use the two new memory structures—Simplified and Flexible—while still using the segmented approach. This does not require any significant changes to existing software.
- **Linear Mode.** The 82596 operates in a flat, linear, 4 gigabyte memory space without segmentation. It can also use the two new memory structures.

In the 32-bit Segmented mode there are some differences between the 82596 and 82586 action commands, mainly in programming and activating new 82596 features. Those bits marked “don’t care” in the compatible mode are not checked; however, we strongly recommend that those bits all be zeroes; this will allow future enhancements and extensions.

In the Linear mode all of the address offsets become 32-bit address pointers. All new 82596 features are accessible in this mode, and all bits previously marked “don’t care” must be zeroes.

The Action Commands, and all other 82596 memory structures, must begin on even byte boundaries, i.e., they must be word aligned.
NOP

This command results in no action by the 82596 except for those performed in the normal command processing. It is used to manipulate the CBL manipulation. The format of the NOP command is shown in Figure 21.

### NOP—82586 and 32-Bit Segmented Modes

<table>
<thead>
<tr>
<th>31</th>
<th>ODD WORD</th>
<th>16</th>
<th>15</th>
<th>EVEN WORD</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>EL</td>
<td>S</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
</tbody>
</table>

### NOP—Linear Mode

<table>
<thead>
<tr>
<th>31</th>
<th>ODD WORD</th>
<th>16</th>
<th>15</th>
<th>EVEN WORD</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>EL</td>
<td>S</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>A31</td>
<td>LINK ADDRESS</td>
<td>A0</td>
<td>4</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure 21

where:

**LINK POINTER** — In the 82586 or 32-bit Segmented modes this is a 16-bit offset to the next Command Block. In the Linear mode this is the 32-bit address of the next Command Block.

**EL** — If set, this bit indicates that this command block is the last on the CBL.

**S** — If set to one, suspend the CU upon completion of this CB.

**I** — If set to one, the 82596 will generate an interrupt after execution of the command is complete. If I is not set to one, the CX bit will not be set.

**CMD (bits 16–18)** — The NOP command. Value: 0h.

**Bits 19–28** — Reserved (zero in the 32-bit Segmented and Linear modes).

**C** — This bit indicates the execution status of the command. The CPU initially resets it to zero when the Command Block is placed on the CBL. Following a command Completion, the 82596 will set it to one.

**B** — This bit indicates that the 82596 is currently executing the NOP command. It is initially reset to zero by the CPU. The 82596 sets it to one when execution begins and to zero when execution is completed. This bit is also set when the 82596 prefetches the command.

**NOTE:**

The C and B bits are modified in one operation.

**OK** — Indicates that the command was executed without error. If set to one no error occurred (command executed OK). If zero an error occurred.

### Individual Address Setup

This command is used to load the 82596 with the Individual Address. This address is used by the 82596 for inserting the Source Address during transmission and recognizing the Destination Address during reception. After RESET, and prior to Individual Address Setup Command execution, the 82596 assumes the Broadcast Address is the Individual Address in all aspects, i.e.:

- This will be the Individual Address Match reference.
- This will be the Source Address of a transmitted frame (for AL-LOC = 0 mode only).
The format of the Individual Address Setup command is shown in Figure 22.

### IA Setup—82586 and 32-Bit Segmented Modes

<table>
<thead>
<tr>
<th>31</th>
<th>ODD WORD</th>
<th>16</th>
<th>15</th>
<th>EVEN WORD</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>EL</td>
<td>S</td>
<td>I</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>INDIVIDUAL ADDRESS</td>
<td>1st byte</td>
<td>A15</td>
<td>LINK OFFSET</td>
<td>A0</td>
<td>4</td>
</tr>
<tr>
<td>6th byte</td>
<td>5th byte</td>
<td>4th byte</td>
<td>3rd byte</td>
<td>8</td>
<td></td>
</tr>
</tbody>
</table>

### IA Setup—Linear Mode

<table>
<thead>
<tr>
<th>31</th>
<th>ODD WORD</th>
<th>16</th>
<th>15</th>
<th>EVEN WORD</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>EL</td>
<td>S</td>
<td>I</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>A31</td>
<td>LINK ADDRESS</td>
<td>A0</td>
<td>4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4th byte</td>
<td>3rd byte</td>
<td>INDIVIDUAL ADDRESS</td>
<td>1st byte</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>6th byte</td>
<td>5th byte</td>
<td>C</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Figure 22**

where:

- **LINK ADDRESS**, EL, B, C, I, S — As per standard Command Block (see the NOP command for details)
- **A** — Indicates that the command was abnormally terminated due to CU Abort control command. If one, then the command was aborted, and if necessary it should be repeated. If this bit is zero, the command was not aborted.

- **Bits 19–28** — Reserved (zero in the 32-bit Segmented and Linear modes).
- **CMD (bits 16–18)** — The Address Setup command. Value: 1h.
- **INDIVIDUAL ADDRESS** — The individual address of the node, 0 to 6 bytes long.

The least significant bit of the Individual Address must be zero for Ethernet (see the Command Structure). However, no enforcement of 0 is provided by the 82596. Thus, an Individual Address with 1 as its least significant bit is a valid Individual Address in all aspects.

The default address length is 6 bytes long, as in 802.3. If a different length is used the IA Setup command should be executed after the Configure command.

### Configure

The Configure command loads the 82596 with its operating parameters. It allows changing some of the parameters by specifying a byte count less than the maximum number of configuration bytes (12 in the 82586 mode, 16 in the 32-Bit Segmented and Linear modes). The 82596 configuration depends on its mode of operation.

- In the 82586 mode the maximum number of configuration bytes is 12. Any number larger than 12 will be reduced to 12 and any number less than 4 will be increased to 4.
- The additional features of the serial side are disabled in the 82586 mode.
- In both the 32-Bit Segmented and Linear modes there are four additional configuration bytes, which hold parameters for additional 82596 features. If these parameters are not accessed, the 82596 will follow their default values.
- For more detailed information refer to the 32-Bit LAN Components User’s Manual.
The format of the Configure command is shown in Figure 23, 24 and 25.

**Figure 23. CONFIGURE—82586 Mode**

<table>
<thead>
<tr>
<th>Byte 1</th>
<th>Byte 0</th>
<th>A15</th>
<th>LINK OFFSET</th>
<th>A0</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Byte 5</td>
<td>Byte 4</td>
<td>Byte 3</td>
<td>Byte 2</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>Byte 9</td>
<td>Byte 8</td>
<td>Byte 7</td>
<td>Byte 6</td>
<td>12</td>
<td></td>
</tr>
<tr>
<td>X X X X X X X X X X X X X X X X X</td>
<td>Byte 10</td>
<td>16</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Figure 24. CONFIGURE—32-Bit Segmented Mode**

<table>
<thead>
<tr>
<th>Byte 1</th>
<th>Byte 0</th>
<th>A15</th>
<th>LINK OFFSET</th>
<th>A0</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Byte 5</td>
<td>Byte 4</td>
<td>Byte 3</td>
<td>Byte 2</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>Byte 9</td>
<td>Byte 8</td>
<td>Byte 7</td>
<td>Byte 6</td>
<td>12</td>
<td></td>
</tr>
<tr>
<td>Byte 13</td>
<td>Byte 12</td>
<td>Byte 11</td>
<td>Byte 10</td>
<td>16</td>
<td></td>
</tr>
</tbody>
</table>

**Figure 25. CONFIGURE—Linear Mode**

- **LINK ADDRESS,** — As per standard Command Block (see the NOP command for details)
- **EL, B, C, I, S**
- **A** — Indicates that the command was abnormally terminated due to a CU Abort control command. If 1, then the command was aborted and if necessary it should be repeated. If this bit is 0, the command was not aborted.
- **Bits 19–28** — Reserved (zero in the 32-Bit Segmented and Linear Modes)
- **CMD (bits 16–18)** — The CONFIGURE command. Value: 2h.

The interpretation of the fields follows:

<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>P</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>BYTE COUNT</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- **BYTE 0**
- **BYTE CNT (Bits 0–3)** — Byte Count. Number of bytes, including this one, that hold parameters to be configured.
- **PREFETCHED (Bit 7)** — Enable the 82596 to write the prefetched bit in all prefetch RBDs.
NOTE:
The P bit is valid only in the new memory structure modes. In 82586 mode this bit is disabled (i.e., no prefetched mark).

<table>
<thead>
<tr>
<th>BYTE 1</th>
<th>MONITOR</th>
<th>FIFO LIMIT</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>X</td>
<td>X</td>
</tr>
</tbody>
</table>

**FIFO Limit (Bits 0–3)**
FIFO limit.

**MONITOR# (Bits 6–7)**
Receive monitor options. If the Byte Count of the configure command is less than 12 bytes then these Monitor bits are ignored.

**DEFAULT:** C8h

<table>
<thead>
<tr>
<th>BYTE 2</th>
<th>SAV BF</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1</td>
</tr>
</tbody>
</table>

**SAV BF (Bit 7)**
0—Received bad frames are not saved in the memory.
1—Received bad frames are saved in the memory.

**DEFAULT:** 40h

<table>
<thead>
<tr>
<th>BYTE 3</th>
<th>LOOP BACK MODE</th>
<th>PREAMBLE LENGTH</th>
<th>NO SRC ADD INS</th>
<th>ADDRESS LENGTH</th>
</tr>
</thead>
</table>

**ADR LEN (Bits 0–2)**
Address length (any kind).

**NO SCR ADD INS (Bit 3)**
No Source Address Insertion.
In the 82586 this bit is called AL LOC.

**PREAM LEN (Bits 4–5)**
Preamble length.

**LP BCK MODE (Bits 6–7)**
Loopback mode.

**DEFAULT:** 26h

<table>
<thead>
<tr>
<th>BYTE 4</th>
<th>BOF METD</th>
<th>EXPONENTIAL PRIORITY</th>
<th>LINEAR PRIORITY</th>
</tr>
</thead>
</table>

**BOF METD (Bit 7)**
Exponential Backoff method.

**DEFAULT:** 00h

<table>
<thead>
<tr>
<th>BYTE 5</th>
<th>INTERFRAME SPACING</th>
</tr>
</thead>
</table>

**INTERFRAME SPACING**
Interframe spacing.

**DEFAULT:** 60h
### BYTE 6

<table>
<thead>
<tr>
<th>SLOT TIME (L)</th>
<th>Slot time, low byte.</th>
</tr>
</thead>
<tbody>
<tr>
<td>DEFAULT: 00h</td>
<td></td>
</tr>
</tbody>
</table>

### BYTE 7

<table>
<thead>
<tr>
<th>SLOT TIME (H)</th>
<th>Slot time, high part. (Bits 0–2)</th>
</tr>
</thead>
<tbody>
<tr>
<td>RETRY NUM (Bits 4–7)</td>
<td>Number of transmission retries on collision.</td>
</tr>
<tr>
<td>DEFAULT: F2h</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>PAD</th>
<th>BIT STUFF</th>
<th>CRC16/ CRC32</th>
<th>NO CRC INS</th>
<th>TONO CRS</th>
<th>MAN/ NRZ</th>
<th>BC DIS</th>
<th>PRM MODE</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### BYTE 8

| PRM (Bit 0) | Promiscuous mode. |
| BC DIS (Bit 1) | Broadcast disable. |
| MANCH/NRZ (Bit 2) | Manchester or NRZ encoding. See specific timing requirements for TXC in Manchester mode. |
| TONO CRS (Bit 3) | Transmit on no CRS. |
| NO_CRC INS (Bit 4) | No CRC insertion. |
| CRC-16/CRC-32 (Bit 5) | CRC type. |
| BIT STF (Bit 6) | Bit stuffing. |
| PAD (Bit 7) | Padding. |
| DEFAULT: 00h  |                      |

### BYTE 9

| CRSF (Bits 0–2) | Carrier Sense filter (length). |
| CRS SRC (Bit 3) | Carrier Sense source. |
| CDTF (Bits 4–6) | Collision Detect filter (length). |
| CDT SRC (Bit 7) | Collision Detect source. |
| DEFAULT: 00h  |                      |
7

| MINIMUM FRAME LENGTH |

BYTE 10

MIN FRAME LEN
Minimum frame length.
DEFAULT: 40h

| MONITOR | MC_ALL | CDBSAC | AUTOTX | CRCINM | LNGFLD | PRECRS |

BYTE 11

PRECRS (Bit 0)
Preamble until Carrier Sense
LNGFLD (Bit 1)
Length field. Enables padding at the End-of-Carrier framing (802.3).
CRCINM (Bit 2)
Rx CRC appended to the frame in memory.
AUTOTX (Bit 3)
Auto retransmit when a collision occurs during the preamble.
CDBSAC (Bit 4)
Collision Detect by source address recognition.
MC_ALL (Bit 5)
Enable to receive all MC frames.
MONITOR (Bits 6–7)
Receive monitor options.
DEFAULT: FFH

| DCR SLOT ADDRESS |

BYTE 12

DCR SLOT ADDRESS
Station index in DCR mode.
(Bits 0–5)
FDX (Bit 6)
Enables Full Duplex operation.
DCR (Bit 7)
Enables Deterministic collision resolution.
DEFAULT: 00h

| DCR NUMBER OF STATIONS |

BYTE 13

DCR NUMBER OF STATIONS
Number of stations in DCR mode.
(Bits 0–5)
MULT_IA (Bit 6)
Multiple individual address.
DIS_BOF (Bit 7)
Disable the backoff algorithm.
DEFAULT: 3Fh
A reset (hardware or software) configures the 82596 according to the following defaults.

### Table 4. Configuration Defaults

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Default Value</th>
<th>Units/ Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADDRESS LENGTH</td>
<td><strong>6</strong> Bytes</td>
<td></td>
</tr>
<tr>
<td>A/L FIELD LOCATION</td>
<td>0</td>
<td>Located in FD</td>
</tr>
<tr>
<td>* AUTO RETRANSMIT</td>
<td>1</td>
<td>Auto Retransmit Enable</td>
</tr>
<tr>
<td>BITSTUFFING/EOC</td>
<td>0 EOC</td>
<td></td>
</tr>
<tr>
<td>BROADCAST DISABLE</td>
<td>0</td>
<td>Broadcast Reception Enabled</td>
</tr>
<tr>
<td>CDBSAC</td>
<td>1</td>
<td>Disabled</td>
</tr>
<tr>
<td>CDT FILTER</td>
<td>0</td>
<td>Bit Times</td>
</tr>
<tr>
<td>CDT SRC</td>
<td>0</td>
<td>External Collision Detection</td>
</tr>
<tr>
<td>* CRC IN MEMORY</td>
<td>1</td>
<td>CRC Not Transferred to Memory</td>
</tr>
<tr>
<td>CRC-16/CRC-32</td>
<td><strong>0</strong> CRC-32</td>
<td></td>
</tr>
<tr>
<td>CRS FILTER</td>
<td>0</td>
<td>0 Bit Times</td>
</tr>
<tr>
<td>CRS SRC</td>
<td>0</td>
<td>External CRS</td>
</tr>
<tr>
<td>* DCR</td>
<td>0</td>
<td>Disable DCR Protocol</td>
</tr>
<tr>
<td>DCR Slot Number</td>
<td>0</td>
<td>DCR Disabled</td>
</tr>
<tr>
<td>DCR Number of Stations</td>
<td>63</td>
<td>Stations</td>
</tr>
<tr>
<td>* DISBOF</td>
<td>0</td>
<td>Backoff Enabled</td>
</tr>
<tr>
<td>EXT LOOPBACK</td>
<td>0</td>
<td>Disabled</td>
</tr>
<tr>
<td>EXPONENTIAL PRIORITY</td>
<td><strong>0</strong> 802.3</td>
<td>802.3 Algorithm</td>
</tr>
<tr>
<td>EXPONENTIAL BACKOFF METHOD</td>
<td><strong>0</strong> 802.3</td>
<td>802.3 Algorithm</td>
</tr>
<tr>
<td>* FULL DUPLEX (FDX)</td>
<td>0</td>
<td>CSMA/CD Protocol (No FDX)</td>
</tr>
<tr>
<td>FIFO THRESHOLD</td>
<td>8 TX: 32 Bytes, RX: 64 Bytes</td>
<td></td>
</tr>
<tr>
<td>INT LOOPBACK</td>
<td>0</td>
<td>Disabled</td>
</tr>
<tr>
<td>INTERFRAME SPACING</td>
<td><strong>96</strong> Bit Times</td>
<td></td>
</tr>
<tr>
<td>LINEAR PRIORITY</td>
<td><strong>0</strong> 802.3</td>
<td>802.3 Algorithm</td>
</tr>
<tr>
<td>* LENGTH FIELD</td>
<td>1</td>
<td>Padding Disabled</td>
</tr>
<tr>
<td>MIN FRAME LENGTH</td>
<td><strong>64</strong> Bytes</td>
<td></td>
</tr>
<tr>
<td>* MC ALL</td>
<td>1</td>
<td>Disabled</td>
</tr>
<tr>
<td>* MONITOR</td>
<td>11</td>
<td>Disabled</td>
</tr>
<tr>
<td>MANCHESTER/NRZ</td>
<td>0</td>
<td>NRZ</td>
</tr>
<tr>
<td>* MULTI 1A</td>
<td>0</td>
<td>Disabled</td>
</tr>
<tr>
<td>NUMBER OF RETRIES</td>
<td><strong>15</strong></td>
<td>Maximum Number of Retries</td>
</tr>
<tr>
<td>NO CRC INSERTION</td>
<td>0</td>
<td>CRC Appended to Frame</td>
</tr>
<tr>
<td>PREFETCH BIT IN RBD</td>
<td>0</td>
<td>Disabled (Valid Only in New Modes)</td>
</tr>
<tr>
<td>PREAMBLE LENGTH</td>
<td><strong>7</strong> Bytes</td>
<td></td>
</tr>
<tr>
<td>* Preamble Until CRS</td>
<td>1</td>
<td>Disabled</td>
</tr>
<tr>
<td>PROMISCUOUS MODE</td>
<td>0</td>
<td>Address Filter On</td>
</tr>
<tr>
<td>PADDING</td>
<td>0</td>
<td>No Padding</td>
</tr>
<tr>
<td>SLOT TIME</td>
<td><strong>512</strong> Bit Times</td>
<td></td>
</tr>
<tr>
<td>SAVE BAD FRAME</td>
<td>0</td>
<td>Discards Bad Frames</td>
</tr>
<tr>
<td>TRANSMIT ON NO CRS</td>
<td>0</td>
<td>Disabled</td>
</tr>
</tbody>
</table>

**NOTES**
1. This configuration setup is compatible with the IEEE 802.3 specification.
2. The Asterisk "**" signifies a new configuration parameter not available in the 82586.
3. The default value of the Auto retransmit configuration parameter is enabled(1).
4. Double Asterisk "****" signifies IEEE 802.3 requirements.
Multicast-Setup

This command is used to load the 82596 with the Multicast-IDs that should be accepted. As noted previously, the filtering done on the Multicast-IDs is not perfect and some unwanted frames may be accepted. This command resets the current filter and reloads it with the specified Multicast-IDs. The format of the Multicast-addresses setup command is:

```
31  ODD WORD  16 15  EVEN WORD  0
   E L  S   X   X   X   X   X   X   X   X   X   X   X   X   X   X   X   0   1   1  C   B   O   K   A
X   X  MC COUNT   A15  LINK OFFSET   A0
```

4th byte

Nth byte

**Figure 26. MC Setup—82586 and 32-Bit Segmented Modes**

```
31  ODD WORD  16 15  EVEN WORD  0
   E L  S   X   X   X   X   X   X   X   X   X   X   X   X   X   X   X   X   X   X   X   X   X   A31  LINK ADDRESS   A0

2nd byte  1st byte X   X   MC COUNT
Nth byte
```

**Figure 27. MC Setup—Linear Mode**

where:

- **LINK ADDRESS, EL, B, C, I, S** — As per standard Command Block (see the NOP command for details)
- **A** — Indicates that the command was abnormally terminated due to a CU Abort control command. If one, then the command was aborted and if necessary it should be repeated. If this bit is zero, the command was not aborted.
- **Bits 19–28** — Reserved (0 in both the 32-Bit Segmented and Linear Modes).
- **CMD (bits 16–18)** — The MC SETUP command value: 3h.
- **MC-CNT** — This 14-bit field indicates the number of bytes in the MC LIST field. The MC CNT must be a multiple of the ADDR LEN; otherwise, the 82596 reduces the MC CNT to the nearest ADDR LEN multiple. MC CNT = 0 implies resetting the Hash table which is equivalent to disabling the Multicast filtering mechanism.
- **MC LIST** — A list of Multicast Addresses to be accepted by the 82596. The least significant bit of each MC address must be 1.

**NOTE:**

The list is sequential; i.e., the most significant byte of an address is immediately followed by the least significant byte of the next address.

- When the 82596 is configured to recognize multiple Individual Address (Multi-IA), the MC-Setup command is also used to set up the Hash table for the individual address.

The least significant bit in the first byte of each IA address must be 0.
Transmit

This command is used to transmit a frame of user data onto the serial link. The format of a Transmit command is as follows.

<table>
<thead>
<tr>
<th>31</th>
<th>ODD WORD</th>
<th>16 15</th>
<th>EVEN WORD</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>EL</td>
<td>S</td>
<td>1</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>A15</td>
<td>TBD OFFSET</td>
<td>A0</td>
<td>A15</td>
<td>LINK OFFSET</td>
</tr>
</tbody>
</table>

4th byte

LENGTH FIELD

6th byte

Figure 28. TRANSMIT—82586 Mode

<table>
<thead>
<tr>
<th>31</th>
<th>ODD WORD</th>
<th>16 15</th>
<th>EVEN WORD</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>EL</td>
<td>S</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>A15</td>
<td>TBD OFFSET</td>
<td>A0</td>
<td>A15</td>
<td>LINK OFFSET</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

4th byte

LENGTH FIELD

6th byte

Figure 29. TRANSMIT—32-Bit Segmented Mode

<table>
<thead>
<tr>
<th>31</th>
<th>ODD WORD</th>
<th>16 15</th>
<th>EVEN WORD</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>EL</td>
<td>S</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>A31</td>
<td>LINK ADDRESS</td>
<td>A0</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>A31</td>
<td>TRANSMIT BUFFER DESCRIPTOR ADDRESS</td>
<td>A0</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

4th byte

LENGTH FIELD

6th byte

Figure 30. TRANSMIT—Linear Mode

0: No CRC Insertion disable; when the configure command is configured to not insert the CRC during transmission the NC bit has no effect.

1: No CRC Insertion enable; when the configure command is configured to insert the CRC during transmission the CRC will not be inserted when NC = 1.

0: Simplified Mode, all the Tx data is in the Transmit Command Block. The Transmit Buffer Descriptor Address field is all 1s.

1: Flexible Mode. Data is in the TCB and in a linked list of TBDs.
where:

EL, B, C, I, S — As per standard Command Block (see the NOP command for details).
OK (Bit 13) — Error free completion.
A (Bit 12) — Indicates that the command was abnormally terminated due to CU Abort control command. If 1, then the command was aborted, and if necessary it should be repeated. If this bit is 0, the command was not aborted.

Bits 19–28 — Reserved (0 in the 32-bit Segmented and Linear modes).
CMD (Bits 16–18) — The transmit command: 4h.
Status Bit 11 — Late collision. A late collision (a collision after the slot time is elapsed) is detected.
Status Bit 10 — No Carrier Sense signal during transmission. Carrier Sense signal is monitored from the end of Preamble transmission until the end of the Frame Check Sequence for TONOCRS=1 (Transmit On No Carrier Sense mode) it indicates that transmission has been executed despite a lack of CRS. For TONOCRS=0 (Ethernet mode), this bit also indicates unsuccessful transmission (transmission stopped when lack of Carrier Sense has been detected).
Status Bit 9 — Transmission unsuccessful (stopped) due to Loss of CTS.
Status Bit 8 — Transmission unsuccessful (stopped) due to DMA Underrun; i.e., the system did not supply data for transmission.
Status Bit 7 — Transmission Deferred, i.e., transmission was not immediate due to previous link activity.
Status Bit 6 — Heartbeat Indicator, Indicates that after a previously performed transmission, and before the most recently performed transmission, (Interframe Spacing) the CDT signal was monitored as active. This indicates that the Ethernet Transceiver Collision Detect logic is performing properly. The Heartbeat is monitored during the Interframe Spacing period.
Status Bit 5 — Transmission attempt was stopped because the number of collisions exceeded the maximum allowable number of retries.
MAX-COL (Bits 3–0) — The number of Collisions experienced during this frame. Max Col = 0 plus S5 = 1 indicates 16 collisions.
LINK OFFSET — As per standard Command Block (see the NOP Command for details)
TBD POINTER — In the 82586 and 32-bit Segmented modes this is the offset of the first Tx Buffer Descriptor containing the data to be transmitted. In the Linear mode this is the 32-bit address of the first Tx Buffer Descriptor on the list. If the TBD POINTER is all 1s it indicates that no TBD is used.
DEST ADDRESS — Contains the Destination Address of the frame. The least significant bit (MC) indicates the address type.
MC = 0: Individual Address.
MC = 1: Multicast or Broadcast Address.
If the Destination Address bits are all 1s this is a Broadcast Address.
LENGTH FIELD — The contents of this 2-byte field are user defined. In 802.3 it contains the length of the data field. It is placed in memory in the same order it is transmitted; i.e., most significant byte first, least significant byte second.
TCB COUNT — This 14-bit counter indicates the number of bytes that will be transmitted from the Transmit Command Block, starting from the third byte after the TCB COUNT field (address n+12 in the 32-bit Segmented mode, n+16 in the Linear mode). The TCB COUNT field can be any number of bytes (including an odd byte), this allows the user to transmit a frame with a header having an odd number of bytes. The TCB COUNT field is not used in the 82586 mode.
EOF Bit — Indicates that the whole frame is kept in the Transmit Command Block. In the Simplified memory model it must be always asserted.
The interpretation of what is transmitted depends on the No Source Address insertion configuration bit and the memory model being used.

NOTES
1. The Destination Address and the Length Field are sequential. The Length Field immediately follows the most significant byte of the Destination Address.
2. In case the 82596 is configured with No Source Address insertion bit equal to 0, the 82596 inserts its configured Source Address in the transmitted frame.
   - In the 82586 mode, or when the Simplified memory model is used, the Destination and Length fields of the transmitted frame are taken from the Transmit Command Block.
   - If the FLEXIBLE memory model is used, the Destination and Length fields of the transmitted frame can be found either in the TCB or TBD, depending on the TCB COUNT.
3. If the 82596 is configured with the Address/Length Field Location equal to 1, the 82596 does not insert its configured Source Address in the transmitted frame. The first \((2 \times \text{Address Length}) + 2\) bytes of the transmitted frame are interpreted as Destination Address, Source Address, and Length fields respectively. The location of the first transmitted byte depends on the operational mode of the 82596:
   - In the 82586 mode, it is always the first byte of the first Tx Buffer.
   - In both the 32-bit Segmented and Linear modes it depends on the SF bit and TCB COUNT:
     - In the Simplified memory mode the first transmitted byte is always the third byte after the TCB COUNT field.
     - In the Flexible mode, if the TCB COUNT is greater than 0 then it is the third byte after the TCB COUNT field. If TCB COUNT equals 0 then it is first byte of the first Tx Buffer.
   - Transmit frames shorter than six bytes are invalid. The transmission will be aborted (only in 82586 mode) because of a DMA Underrun.
4. Frames which are aborted during transmission are jammed. Such an interruption of transmission can be caused by any reason indicated by any of the status bits 8, 9, 10 and 12.

Jamming Rules
1. Jamming will not start before completion of preamble transmission.
2. Collisions detected during transmission of the last 11 bits will not result in jamming.

The format of a Transmit Buffer Descriptor is:

<table>
<thead>
<tr>
<th>31</th>
<th>ODD WORD</th>
<th>16 15 13</th>
<th>EVEN WORD</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>NEXT TBD OFFSET</td>
<td>EOF</td>
<td>X</td>
<td>SIZE (ACT COUNT)</td>
<td>0</td>
</tr>
<tr>
<td>X X X X X X X X</td>
<td>TRANSMIT BUFFER ADDRESS</td>
<td>4</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>31</th>
<th>ODD WORD</th>
<th>16 15 13</th>
<th>EVEN WORD</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>NEXT TBD OFFSET</td>
<td>EOF</td>
<td>0</td>
<td>SIZE (ACT COUNT)</td>
<td>0</td>
</tr>
<tr>
<td>TRANSMIT BUFFER ADDRESS</td>
<td>4</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>31</th>
<th>ODD WORD</th>
<th>16 15 13</th>
<th>EVEN WORD</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 EOF</td>
<td>0</td>
<td>SIZE (ACT COUNT)</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>NEXT TBD ADDRESS</td>
<td>4</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TRANSMIT BUFFER ADDRESS</td>
<td>8</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Figure 31**
where:

EOF — This bit indicates that this TBD is the last one associated with the frame being transmitted. It is set by the CPU before transmit.

SIZE (ACT COUNT) — This 14-bit quantity specifies the number of bytes that hold information for the current buffer. It is set by the CPU before transmission.

NEXT TBD ADDRESS — In the 82586 and 32-bit Segmented modes, it is the offset of the next TBD on the list. In the Linear mode this is the 32-bit address of the next TBD on the list. It is meaningless if EOF = 1.

BUFFER ADDRESS — The starting address of the memory area that contains the data to be sent. In the 82586 mode, this is a 24-bit address (A31–A24 are considered to be zero). In the 32-bit Segmented and Linear modes this is a 32-bit address.

TDR

This operation activates Time Domain Reflectometry, which is a mechanism to detect open or short circuits on the link and their distance from the diagnosing station. The TDR command has no parameters. The TDR transmit sequence was changed, compared to the 82586, to form a regular transmission. The TDR bit stream is as follows.

— Preamble
— Source address
— Another Source address (the TDR frame is transmitted back to the sending station, so DEST ADR = SRC ADR).
— Data field containing 7Eh patterns.
— Jam Pattern, which is the inverse CRC of the transmitted frame.

Maximum length of the TDR frame is 2048 bits. If the 82596 senses collision while transmitting the TDR frame it transmits the jam pattern and stops the transmission. The 82596 then triggers an internal timer (STC); the timer is reset at the beginning of transmission and reset if CRS is returned. The timer measures the time elapsed from the start of transmission until an echo is returned. The echo is indicated by Collision Detect going active or a drop in the Carrier Sense signal. The following table lists the possible cases that the 82596 is able to analyze.

<table>
<thead>
<tr>
<th>Condition</th>
<th>Transceiver Type</th>
<th>Ethernet</th>
<th>Non Ethernet</th>
</tr>
</thead>
<tbody>
<tr>
<td>Carrier Sense was inactive for 2048-bit-time periods</td>
<td>Short or Open on the Transceiver Cable</td>
<td>NA</td>
<td>NA</td>
</tr>
<tr>
<td>Carrier Sense signal dropped</td>
<td>Short on the Ethernet cable</td>
<td>NA</td>
<td>NA</td>
</tr>
<tr>
<td>Collision Detect went active</td>
<td>Open on the Ethernet cable</td>
<td>Open on the Serial Link</td>
<td></td>
</tr>
<tr>
<td>The Carrier Sense Signal did not drop or the Collision Detect did not go active within 2048-bit time period</td>
<td>No Problem</td>
<td>No Problem</td>
<td></td>
</tr>
</tbody>
</table>

An Ethernet transceiver is defined as one that returns transmitted data on the receive pair and activates the Carrier Sense Signal while transmitting. A Non-Ethernet Transceiver is defined as one that does not do so.
The format of the Time Domain Reflectometer command is:

<table>
<thead>
<tr>
<th>82586 and 32-Bit Segmented Modes</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
</tr>
<tr>
<td>EL</td>
</tr>
<tr>
<td>LNK</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Linear Mode</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
</tr>
<tr>
<td>EL</td>
</tr>
<tr>
<td>A31</td>
</tr>
<tr>
<td>0</td>
</tr>
</tbody>
</table>

where:

- **LINK ADDRESS, EL, B, C, I, S A** — As per standard Command Block (see the NOP command for details).

- **EL** indicates that the command was abnormally terminated due to CU Abort control command. If one, then the command was aborted, and if necessary it should be repeated. If this bit is zero, the command was not aborted.

- **Bits 19–28** — Reserved (0 in the 32-bit Segmented and Linear Modes).

- **CMD (Bits 16–18)** — The TDR command. Value: 5h.

- **TIME** — An 11-bit field that specifies the number of TxC cycles that elapsed before an echo was observed. No echo is indicated by a reception consisting of “1s” only. Because the network contains various elements such as transceiver links, transceivers, Ethernet, repeaters etc., the TIME is not exactly proportional to the problems distance.

- **LNK OK (Bit 15)** — No link problem identified. TIME = 7FFh.

- **XCVR PRB (Bit 14)** — Indicates a Transceiver problem. Carrier Sense was inactive for 2048-bit time period. LNK OK = 0. TIME = 7FFh.

- **ET OPN (Bit 13)** — The transmission line is not properly terminated. Collision Detect went active and LNK OK = 0.

- **ET SRT (Bit 12)** — There is a short circuit on the transmission line. Carrier Sense Signal dropped and LNK OK = 0.

Figure 32. TDR
DUMP

This command causes the contents of various 82596 registers to be placed in a memory area specified by the user. It is supplied as a 82596 self-diagnostic tool, and to provide registers of interest to the user. The format of the DUMP command is:

<table>
<thead>
<tr>
<th>31</th>
<th>ODD WORD</th>
<th>16 15 EVEN WORD</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>EL S</td>
<td>I X X X X X X X X X X 1 1 0 C B OK</td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
<td></td>
</tr>
<tr>
<td>A15 BUFFER OFFSET</td>
<td>A0 A15 LINK OFFSET</td>
<td>A0</td>
<td></td>
</tr>
</tbody>
</table>

**82586 and 32-Bit Segmented Modes**

<table>
<thead>
<tr>
<th>31</th>
<th>ODD WORD</th>
<th>16 15 EVEN WORD</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>EL S</td>
<td>I X X X X X X X X X X X X 1 1 0 C B OK</td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
<td></td>
</tr>
<tr>
<td>A31 LINK ADDRESS</td>
<td>A0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>A31 BUFFER ADDRESS</td>
<td>A0</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Linear Mode**

<table>
<thead>
<tr>
<th>31</th>
<th>ODD WORD</th>
<th>16 15 EVEN WORD</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>EL S</td>
<td>I X X X X X X X X X X X 1 1 0 C B OK</td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
<td></td>
</tr>
<tr>
<td>A15 BUFFER OFFSET</td>
<td>A0 A15 LINK OFFSET</td>
<td>A0</td>
<td></td>
</tr>
</tbody>
</table>

**Figure 33. Dump**

where:

- LINK ADDRESS — As per standard Command Block (see the NOP command for details).
- EL, B, C, I, S
- OK — Indicates error free completion.
- Bits 19–28 — Reserved (0 in the 32-bit Segmented and Linear Modes).
- CMD (Bits 16–18) — The Dump command. Value: 6h.
- BUFFER POINTER — In the 82586 and 32-bit Segmented modes this is the 16-bit-offset portion of the dump area address. In the Linear mode this is the 32-bit linear address of the dump area.

**Dump Area Information Format**

- The 82596 is not Dump compatible with the 82586 because of the 32-bit internal architecture. In 82586 mode the 82596 will dump the same number of bytes as the 82586. The compatible data will be marked with an asterisk.
- In 82586 mode the dump area is 170 bytes.
- The DUMP area format of the 32-bit Segmented and Linear modes is described in Figure 35.
- The size of the dump area of the 32-bit Segmented and Linear modes is 304 bytes.
- When the Dump is executed by the Port command an extra word will be appended to the Dump Area. The extra word is a copy of the Dump Area status word (containing the C, B, and OK Bits). The C and OK Bits are set when the 82596 has completed the Port Dump command.
The 82596 is not Dump compatible with the 82586 because of the 32-bit internal architecture. In 82586 mode the 82596 will dump the same number of bytes as the 82586.

**These bytes are not user defined, results may vary from Dump command to Dump command.**

![Figure 34. Dump Area Format—82586 Mode](image)
**THE 82596 IS NOT DUMP COMPATIBLE WITH THE 82586 BECAUSE OF THE 32-BIT INTERNAL ARCHITECTURE. IN 82586 MODE THE 82596 WILL DUMP THE SAME NUMBER OF BYTES AS THE 82586.**

**THESE BYTES ARE NOT USER DEFINED, RESULTS MAY VARY FROM DUMP COMMAND TO DUMP COMMAND.**

### Figure 35. Dump Area Format—Linear and 32-Bit Segmented Mode

<table>
<thead>
<tr>
<th>31</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>CONFIGURE BYTES 5, 4, 3, 2</td>
<td>00</td>
</tr>
<tr>
<td>CONFIGURE BYTES 9, 8, 7, 6</td>
<td>04</td>
</tr>
<tr>
<td>CONFIGURE BYTES 13, 12, 11, 10</td>
<td>08</td>
</tr>
<tr>
<td>I.A. BYTES 1, 0</td>
<td>XX X X X X X X X</td>
</tr>
<tr>
<td>I.A. BYTES 5, 2</td>
<td>10</td>
</tr>
<tr>
<td>TX CRC BYTES 0, 1</td>
<td>LAST T.X. STATUS</td>
</tr>
<tr>
<td>RX CRC BYTES 0, 1</td>
<td>TX CRC BYTES 3, 2</td>
</tr>
<tr>
<td>RX TEMP MEMORY 1, 0</td>
<td>RX CRC BYTES 3, 2</td>
</tr>
<tr>
<td>R.X. TEMP MEMORY 5, 2</td>
<td>20</td>
</tr>
<tr>
<td>HASH REGISTERS 1, 0</td>
<td>LAST R.X. STATUS</td>
</tr>
<tr>
<td>HASH REGISTER BYTES 5, 2</td>
<td>28</td>
</tr>
<tr>
<td>SLOT TIME COUNTER</td>
<td>HASH REGISTERS 7, 6</td>
</tr>
<tr>
<td>RECEIVE FRAME LENGTH</td>
<td>WAIT-TIME COUNTER</td>
</tr>
<tr>
<td>MICRO MACHINE**</td>
<td>34</td>
</tr>
<tr>
<td>REGISTER FILE</td>
<td></td>
</tr>
<tr>
<td>128 BYTES</td>
<td>80</td>
</tr>
<tr>
<td>MICRO MACHINE LFSR**</td>
<td>B4</td>
</tr>
<tr>
<td>MICRO MACHINE**</td>
<td>B8</td>
</tr>
<tr>
<td>FLAG ARRAY</td>
<td></td>
</tr>
<tr>
<td>28 BYTES</td>
<td>D0</td>
</tr>
<tr>
<td>M.M. INPUT PORT**</td>
<td>D4</td>
</tr>
<tr>
<td>16 BYTES</td>
<td>E0</td>
</tr>
<tr>
<td>MICRO MACHINE ALU**</td>
<td>E4</td>
</tr>
<tr>
<td>RESERVED**</td>
<td>E8</td>
</tr>
<tr>
<td>M.M. TEMP A ROTATE R.**</td>
<td>EC</td>
</tr>
<tr>
<td>M.M. TEMP A**</td>
<td>F0</td>
</tr>
<tr>
<td>T.X. DMA BYTE COUNT**</td>
<td>F4</td>
</tr>
<tr>
<td>M.M. INPUT PORT ADDRESS REGISTER**</td>
<td>F8</td>
</tr>
<tr>
<td>T.X. DMA ADDRESS**</td>
<td>FC</td>
</tr>
<tr>
<td>M.M. OUTPUT PORT REGISTER**</td>
<td>100</td>
</tr>
<tr>
<td>R.X. DMA BYTE COUNT**</td>
<td>104</td>
</tr>
<tr>
<td>M.M. OUTPUT PORT ADDRESS REGISTER**</td>
<td>108</td>
</tr>
<tr>
<td>R.X. DMA ADDRESS REGISTER**</td>
<td>10C</td>
</tr>
<tr>
<td>RESERVED**</td>
<td>110</td>
</tr>
<tr>
<td>BUS THROTTLE TIMERS</td>
<td>114</td>
</tr>
<tr>
<td>DIU CONTROL REGISTER**</td>
<td>118</td>
</tr>
<tr>
<td>RESERVED**</td>
<td>11C</td>
</tr>
<tr>
<td>DMA CONTROL REGISTER**</td>
<td>120</td>
</tr>
<tr>
<td>BIU CONTROL REGISTER**</td>
<td>124</td>
</tr>
<tr>
<td>M.M. DISPATCHER REG.**</td>
<td>128</td>
</tr>
<tr>
<td>M.M. STATUS REGISTER**</td>
<td>12C</td>
</tr>
</tbody>
</table>
Diagnose

The Diagnose Command triggers an internal self-test procedure that checks internal 82596 hardware, which includes:

- Exponential Backoff Random Number Generator (Linear Feedback Shift Register).
- Exponential Backoff Timeout Counter.
- Slot Time Period Counter.
- Collision Number Counter.
- Exponential Backoff Shift Register.
- Exponential Backoff Mask Logic.
- Timer Trigger Logic.

This procedure checks the operation of the Backoff block, which resides in the serial side and is not easily controlled. The Diagnose command is performed in two phases.

The format of the 82596 Diagnose command is:

<table>
<thead>
<tr>
<th>31</th>
<th>ODD WORD</th>
<th>16 15</th>
<th>EVEN WORD</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>EL</td>
<td>S I X X X X X X X X 1 1 1 C B OK 0 F</td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>X X X X X X X X X X X X X X X X X X</td>
<td>A15</td>
<td>LINK OFFSET</td>
<td>A0</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>31</th>
<th>ODD WORD</th>
<th>16 15</th>
<th>EVEN WORD</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>EL</td>
<td>S I 0 0 0 0 0 0 0 0 0 0 0 1 1 1 C B OK 0 F</td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>A31</td>
<td>LINK ADDRESS</td>
<td>A0</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure 36. Diagnose

where:

- LINK ADDRESS, EL, B, C, I, S — As per standard Command Block (see the NOP command for details).
- Bits 19–28 — Reserved (0 in the 32-bit Segmented and Linear Modes).
- CMD (bits 16–18) — The Diagnose command. Value: 7h.
- OK (bit 13) — Indicates error free completion.
- F (bit 11) — Indicates that the self-test procedure has failed.
RECEIVE FRAME DESCRIPTOR

Each received frame is described by one Receive Frame Descriptor (see Figure 37). Two new memory structures are available for the received frames. The structures are available only in the Linear and 32-bit Segmented modes.

Simplified Memory Structure

The first is the Simplified memory structure, the data section of the received frame is part of the RFD and is located immediately after the Length Field. Receive Buffer Descriptors are not used with the Simplified structure, it is primarily used to make programming easier. If the length of the data area described in the Size Field is smaller than the incoming frame, the following happens.

1. The received frame is truncated.
2. The No Resource error counter is updated.
3. If the 82596 is configured to Save Bad Frames the RFD is not reused; otherwise, the same RFD is used to hold the next received frame, and the only action taken regarding the truncated frame is to update the counter.
4. The 82596 continues to receive the next frame in the next RFD.

Figure 37. The Receive Frame Area
Note that this sequence is very useful for monitoring. If the 82596 is configured to Save Bad Frames, to receive in Promiscuous mode, and to use the Simplified memory structure, any programmed length of received data can be saved in memory.

The Simplified memory structure is shown in Figure 38.

![Figure 38. RFA Simplified Memory Structure](image)

**Flexible Memory Structure**

The second structure is the Flexible memory structure, the data structure of the received frame is stored in both the RFD and in a linked list of Receive Buffers—Receive Buffer Descriptors. The received frame is placed in the RFD as configured in the Size field. Any remaining data is placed in a linked list of RBDs.

The Flexible memory structure is shown in Figure 39.
Buffers on the receive side can be different lengths. The 82596 will not place more bytes into a buffer than indicated in the associated RBD. The 82596 will fetch the next RBD before it is needed. The 82596 will attempt to receive frames as long as the FBL is not exhausted. If there are no more buffers, the 82596 Receive Unit will enter the No Resources state. Before starting the RU, the CPU must place the FBL pointer in the RBD pointer field of the first RFD. All remaining RBD pointer fields for subsequent RFDs should be "1s." If the Receive Frame Descriptor and the associated Receive Buffers are not reused (e.g., the frame is properly received or the 82596 is configured to Save Bad Frames), the 82596 writes the address of the next free RBD to the RBD pointer field of the next RFD.

**Receive Buffer Descriptor (RBD)**

The RBDs are used to store received data in a flexible set of linked buffers. The portion of the frame's data field that is outside the RFD is placed in a set of buffers chained by a sequence of RBDs. The RFD points to the first RBD, and the last RBD is flagged with an EOF bit set to 1. Each buffer in the linked list of buffers related to a particular frame can be any size up to $2^{14}$ bytes but must be word aligned (begin on an even numbered byte). This ensures optimum use of the memory resources while maintaining low overhead. All buffers in a frame are filled with the received data except for the last, in which the actual count can be smaller than the allocated buffer space.

---

**Figure 39. RFA Flexible Memory Structure**
### Figure 40. Receive Frame Descriptor—82586 Mode

<table>
<thead>
<tr>
<th>31</th>
<th>ODD WORD</th>
<th>16 15</th>
<th>EVEN WORD</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>EL</td>
<td>S</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>C</td>
<td>B</td>
<td>OK</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>STATUS BITS</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>A15</td>
<td>RBD OFFSET</td>
<td>A0</td>
<td>A15</td>
<td>LINK OFFSET</td>
</tr>
<tr>
<td>4th byte</td>
<td>DESTINATION ADDRESS</td>
<td>1st byte</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>SOURCE ADDRESS</td>
<td>1st byte</td>
<td>6th byte</td>
<td>12</td>
<td></td>
</tr>
<tr>
<td>6th byte</td>
<td>4th byte</td>
<td>16</td>
<td></td>
<td></td>
</tr>
<tr>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
</tbody>
</table>

### Figure 41. Receive Frame Descriptor—32-Bit Segmented Mode

<table>
<thead>
<tr>
<th>31</th>
<th>ODD WORD</th>
<th>16 15</th>
<th>EVEN WORD</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>EL</td>
<td>S</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>C</td>
<td>B</td>
<td>OK</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>STATUS BITS</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>A15</td>
<td>RBD OFFSET</td>
<td>A0</td>
<td>A15</td>
<td>LINK OFFSET</td>
</tr>
<tr>
<td>0</td>
<td>SIZE</td>
<td>EOF</td>
<td>F</td>
<td>ACTUAL COUNT</td>
</tr>
<tr>
<td>DESTINATION ADDRESS</td>
<td>1st byte</td>
<td>12</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SOURCE ADDRESS</td>
<td>1st byte</td>
<td>6th byte</td>
<td>16</td>
<td></td>
</tr>
<tr>
<td>6th byte</td>
<td>4th byte</td>
<td>20</td>
<td></td>
<td></td>
</tr>
<tr>
<td>LENGTH FIELD</td>
<td>24</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Figure 42. Receive Frame Descriptor—Linear Mode

<table>
<thead>
<tr>
<th>31</th>
<th>ODD WORD</th>
<th>16 15</th>
<th>EVEN WORD</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>EL</td>
<td>S</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>A31</td>
<td>LINK ADDRESS</td>
<td>A0</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>A31</td>
<td>RECEIVE BUFFER DESCRIPTOR ADDRESS</td>
<td>A0</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>SIZE</td>
<td>EOF</td>
<td>F</td>
<td>ACTUAL COUNT</td>
</tr>
<tr>
<td>DESTINATION ADDRESS</td>
<td>1st byte</td>
<td>16</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SOURCE ADDRESS</td>
<td>1st byte</td>
<td>6th byte</td>
<td>20</td>
<td></td>
</tr>
<tr>
<td>6th byte</td>
<td>4th byte</td>
<td>24</td>
<td></td>
<td></td>
</tr>
<tr>
<td>LENGTH FIELD</td>
<td>28</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
where:

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>EL</td>
<td>When set, this bit indicates that this RFD is the last one on the RDL.</td>
</tr>
<tr>
<td>S</td>
<td>When set, this bit suspends the RU after receiving the frame.</td>
</tr>
<tr>
<td>SF</td>
<td>This bit selects between the Simplified or the Flexible mode.</td>
</tr>
<tr>
<td></td>
<td>0 — Simplified mode, all the RX data is in the RFD. RBD ADDRESS field is all “1s.”</td>
</tr>
<tr>
<td></td>
<td>1 — Flexible mode. Data is in the RFD and in a linked list of Receive Buffer Descriptors.</td>
</tr>
<tr>
<td>C</td>
<td>This bit indicates the completion of frame reception. It is set by the 82596.</td>
</tr>
<tr>
<td>B</td>
<td>This bit indicates that the 82596 is currently receiving this frame, or that the 82596 is ready to receive the frame. It is initially set to 0 by the CPU. The 82596 sets it to 1 when reception set up begins, and to 0 upon completion. The C and B bits are set during the same operation.</td>
</tr>
<tr>
<td>OK (bit 13)</td>
<td>Frame received successfully, without errors. RFDs with bit 13 equal to 0 are possible only if the save bad frames, configuration option is selected. Otherwise all frames with errors will be discarded, although statistics will be collected on them.</td>
</tr>
</tbody>
</table>

**STATUS** — The results of the Receive operation. Defined bits are,

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bit 12</td>
<td>Length error if configured to check length</td>
</tr>
<tr>
<td>Bit 11</td>
<td>CRC error in an aligned frame</td>
</tr>
<tr>
<td>Bit 10</td>
<td>Alignment error (CRC error in misaligned frame)</td>
</tr>
<tr>
<td>Bit 9</td>
<td>Ran out of buffer space — no resources</td>
</tr>
<tr>
<td>Bit 8</td>
<td>DMA Overrun failure to acquire the system bus</td>
</tr>
<tr>
<td>Bit 7</td>
<td>Frame too short.</td>
</tr>
<tr>
<td>Bit 6</td>
<td>No EOP flag (for Bit stuffing only)</td>
</tr>
<tr>
<td>Bit 5</td>
<td>When the SF bit equals zero, and the 82596 is configured to save bad frames, this bit signals that the receive frame was truncated. Otherwise it is zero.</td>
</tr>
<tr>
<td>Bits 2-4</td>
<td>Zeros</td>
</tr>
<tr>
<td>Bit 1</td>
<td>When it is zero, the destination address of the received frame matches the IA address. When it is a 1, the destination address of the received frame did not match the individual address. For example, a multicast address or broadcast address will set this bit to a 1.</td>
</tr>
<tr>
<td>Bit 0</td>
<td>Receive collision, a collision is detected during reception.</td>
</tr>
</tbody>
</table>

**LINK ADDRESS** — A 16-bit offset (32-bit address in the Linear mode) to the next Receive Frame Descriptor. The Link Address of the last frame can be used to form a cyclical list.

**RBD POINTER** — The offset (address in the Linear mode) of the first RBD containing the received frame data. An RBD pointer of all ones indicates no RBD.

**EOF** — These fields are for the Simplified and Flexible memory models. They are exactly the same as the respective fields in the Receive Buffer Descriptor. See the next section for detailed explanation of their functions.

**SIZE** — Multicast bit.

**ACT COUNT** — The contents of the destination address of the receive frame. The field is 0 to 6 bytes long.

**DESTINATION ADDRESS** — The contents of the Source Address field of the received frame. It is 0 to 6 bytes long.

**SOURCE ADDRESS** — The contents of this 2-byte field are user defined. In 802.3 it contains the length of the data field. It is placed in memory in the same order it is received, i.e., most significant byte first, least significant byte second.
NOTES
1. The Destination address, Source address and Length fields are packed, i.e., one field immediately follows the next.
2. The affect of Address/Length Location (No Source Address Insertion) configuration parameter while receiving is as follows:
   — 82586 Mode: The Destination address, Source address and Length field are not used, they are placed in the RX data buffers.
   — 32-Bit Segmented and Linear Modes: when the Simplified memory model is used, the Destination address, Source address and Length fields reside in their respective fields in the RFD. When the Flexible memory structure is used, the Destination address, Source address, and Length field locations depend on the SIZE field of the RFD. They can be placed in the RFD, in the RX data buffers, or partially in the RFD and the rest in the RX data buffers, depending on the SIZE field value.

<table>
<thead>
<tr>
<th>82586 Mode</th>
<th>31</th>
<th>ODD WORD</th>
<th>16 15</th>
<th>EVEN WORD</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>A15</td>
<td>NEXT RBD OFFSET</td>
<td>A0</td>
<td>EOF</td>
<td>F</td>
<td>ACTUAL COUNT</td>
</tr>
<tr>
<td>X X X X X X X A23</td>
<td>RECEIVE BUFFER ADDRESS</td>
<td>A0</td>
<td>4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>X X X X X X X X X X X X X</td>
<td>SIZE</td>
<td>8</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>32-Bit Segmented Mode</th>
<th>31</th>
<th>ODD WORD</th>
<th>16 15</th>
<th>EVEN WORD</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>A15</td>
<td>NEXT RBD OFFSET</td>
<td>A0</td>
<td>EOF</td>
<td>F</td>
<td>ACTUAL COUNT</td>
</tr>
<tr>
<td>A31</td>
<td>RECEIVE BUFFER ADDRESS</td>
<td>A0</td>
<td>4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
<td>EL</td>
<td>P</td>
<td>SIZE</td>
<td>8</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Linear Mode</th>
<th>31</th>
<th>ODD WORD</th>
<th>16 15</th>
<th>EVEN WORD</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
<td>EOF</td>
<td>F</td>
<td>ACTUAL COUNT</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>A31</td>
<td>NEXT RBD ADDRESS</td>
<td>A0</td>
<td>4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>A31</td>
<td>RECEIVE BUFFER ADDRESS</td>
<td>A0</td>
<td>8</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
<td>EL</td>
<td>P</td>
<td>SIZE</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
where:

EOF — Indicates that this is the last buffer related to the frame. It is cleared by the CPU before starting the RU, and is written by the 82596 at the end of reception of the frame.

F — Indicates that this buffer has already been used. The Actual Count has no meaning unless the F bit equals one. This bit is cleared by the CPU before starting the RU, and is set by the 82596 after the associated buffer has been. This bit has the same meaning as the Complete bit in the RFD and CB.

ACT COUNT — This 14-bit quantity indicates the number of meaningful bytes in the buffer. It is cleared by the CPU before starting the RU, and is written by the 82596 after the associated buffer has already been used. In general, after the buffer is full, the Actual Count value equals the size field of the same buffer. For the last buffer of the frame, Actual Count can be less than the buffer size.

NEXT BD ADDRESS — The offset (absolute address in the Linear mode) of the next RBD on the list. It is meaningless if EL = 1.

BUFFER ADDRESS — The starting address of the memory area that contains the received data. In the 82586 mode, this is a 24-bit address (with pins A24–A31 = 0). In the 32-bit Segmented and Linear modes this is a 32-bit address.

EL — Indicates that the buffer associated with this RBD is last in the FBL.

P — This bit indicates that the 82596 has already prefetched the RBDs and any change in the RBD data will be ignored. This bit is valid only in the new 82596 memory modes, and if this feature has been enabled during configure command. The 82596 Prefetches the RBDs in locked cycles; after prefetching the RBD the 82596 performs a write cycle where the P bit is set to one and the rest of the data remains unchanged. The CPU is responsible for resetting it in all RBDs. The 82596 will not check this bit before setting it.

SIZE — This 14-bit quantity indicates the size, in bytes, of the associated buffer. This quantity must be an even number.
ELECTRICAL AND TIMING CHARACTERISTICS

DC Characteristics

$T_C = 0^\circ C - 85^\circ C$, $V_{CC} = 5V \pm 10\%$ LE/BE have MOS levels (see $V_{MIL}$, $V_{MIH}$). All other signals have TTL levels (see $V_{IL}$, $V_{IH}$, $V_{OL}$, $V_{OH}$).

<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>$V_{IL}$</td>
<td>Input Low Voltage (TTL)</td>
<td>$-0.3$</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>$V_{CC} + 0.3$</td>
<td>V</td>
<td></td>
</tr>
<tr>
<td>$V_{MIL}$</td>
<td>Input Low Voltage (MOS)</td>
<td>$-0.3$</td>
<td>$+0.8$</td>
<td>V</td>
<td></td>
</tr>
<tr>
<td>$V_{MIH}$</td>
<td>Input High Voltage (MOS)</td>
<td>$3.7$</td>
<td>$V_{CC} + 0.3$</td>
<td>V</td>
<td></td>
</tr>
<tr>
<td>$V_{OL}$</td>
<td>Output Low Voltage (TTL)</td>
<td>$0.45$</td>
<td></td>
<td>V</td>
<td></td>
</tr>
<tr>
<td>$V_{OH}$</td>
<td>Output High Voltage (TTL)</td>
<td>$2.4$</td>
<td></td>
<td>V</td>
<td></td>
</tr>
<tr>
<td>$I_{IL}$</td>
<td>Input Leakage Current</td>
<td>$\pm 15$</td>
<td>$\mu A$</td>
<td>$0 \leq V_{IN} \leq V_{CC}$</td>
<td></td>
</tr>
<tr>
<td>$I_{IO}$</td>
<td>Output Leakage Current</td>
<td>$\pm 15$</td>
<td>$\mu A$</td>
<td>$0.45 &lt; V_{OUT} &lt; V_{CC}$</td>
<td></td>
</tr>
<tr>
<td>$C_{IN}$</td>
<td>Capacitance of Input Buffer</td>
<td>$10$</td>
<td>$pF$</td>
<td>$FC = 1 MHz$</td>
<td></td>
</tr>
<tr>
<td>$C_{OUT}$</td>
<td>Capacitance of Input/Output Buffer</td>
<td>$12$</td>
<td>$pF$</td>
<td>$FC = 1 MHz$</td>
<td></td>
</tr>
<tr>
<td>$C_{CLK}$</td>
<td>CLK Capacitance</td>
<td>$20$</td>
<td>$pF$</td>
<td>$FC = 1 MHz$</td>
<td></td>
</tr>
<tr>
<td>$I_{CC}$</td>
<td>Power Supply</td>
<td>$200$</td>
<td>mA</td>
<td>At 25 MHz</td>
<td></td>
</tr>
<tr>
<td>$I_{CC}$</td>
<td>Power Supply</td>
<td>$300$</td>
<td>mA</td>
<td>At 33 MHz</td>
<td></td>
</tr>
</tbody>
</table>

Notes:
1. $I_{OL} = 4.0 mA$
AC Characteristics

82596CA INPUT/OUTPUT SYSTEM TIMINGS

\( T_C = 0^\circ\text{C} - 85^\circ\text{C}, \ V_{CC} = 5V \pm 10\% \). These timing assume the \( C_L \) on all outputs is 50 pF unless otherwise specified. \( C_L \) can be 20 pF to 120 pF however timings must be derated. All timing requirements are given in nanoseconds.

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Parameter</th>
<th>25 MHz</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Min</td>
<td>Max</td>
<td></td>
</tr>
<tr>
<td>T1</td>
<td>CLK Period</td>
<td>40</td>
<td>80</td>
</tr>
<tr>
<td>T1a</td>
<td>CLK Period Stability</td>
<td>0.1%</td>
<td></td>
</tr>
<tr>
<td>T2</td>
<td>CLK High</td>
<td>14</td>
<td></td>
</tr>
<tr>
<td>T3</td>
<td>CLK Low</td>
<td>14</td>
<td></td>
</tr>
<tr>
<td>T4</td>
<td>CLK Rise Time</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>T5</td>
<td>CLK Fall Time</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>T6</td>
<td>BEn, LOCK, and A2-A31 Valid Delay</td>
<td>3</td>
<td>22</td>
</tr>
<tr>
<td>T6a</td>
<td>BLAST, PCHK Valid Delay</td>
<td>3</td>
<td>27</td>
</tr>
<tr>
<td>T7</td>
<td>BEn, LOCK, BLAST, A2-A31 Float Delay</td>
<td>3</td>
<td>30</td>
</tr>
<tr>
<td>T8</td>
<td>W/R and ADS Valid Delay</td>
<td>3</td>
<td>32</td>
</tr>
<tr>
<td>T9</td>
<td>W/R and ADS Float Delay</td>
<td>3</td>
<td>30</td>
</tr>
<tr>
<td>T10</td>
<td>D0-D31, DPn Write Data Valid Delay</td>
<td>3</td>
<td>22</td>
</tr>
<tr>
<td>T11</td>
<td>D0-D31, DPn Write Data Float Delay</td>
<td>3</td>
<td>30</td>
</tr>
<tr>
<td>T12</td>
<td>HOLD Valid Delay</td>
<td>3</td>
<td>32</td>
</tr>
<tr>
<td>T13</td>
<td>CA and BREQ Setup Time</td>
<td>7</td>
<td></td>
</tr>
<tr>
<td>T14</td>
<td>CA and BREQ Hold Time</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>T15</td>
<td>BS16 Setup Time</td>
<td>8</td>
<td>2</td>
</tr>
<tr>
<td>T16</td>
<td>BS16 Hold Time</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>T17</td>
<td>BRDY, RDY Setup Time</td>
<td>8</td>
<td>2</td>
</tr>
<tr>
<td>T18</td>
<td>BRDY, RDY Hold Time</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>T19</td>
<td>D0-D31, DPn READ Setup Time</td>
<td>5</td>
<td>2</td>
</tr>
<tr>
<td>T20</td>
<td>D0-D31, DPn READ Hold Time</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>T21</td>
<td>AVOID and HLDA Setup Time</td>
<td>10</td>
<td>1, 2</td>
</tr>
<tr>
<td>T22</td>
<td>AVOID Hold Time</td>
<td>3</td>
<td>1, 2</td>
</tr>
<tr>
<td>T22a</td>
<td>HLDA Hold Time</td>
<td>3</td>
<td>1, 2</td>
</tr>
<tr>
<td>T23</td>
<td>RESET Setup Time</td>
<td>10</td>
<td>1, 2</td>
</tr>
<tr>
<td>T24</td>
<td>RESET Hold Time</td>
<td>3</td>
<td>1, 2</td>
</tr>
<tr>
<td>T25</td>
<td>INT/INT Valid Delay</td>
<td>1</td>
<td>26</td>
</tr>
<tr>
<td>T26</td>
<td>CA and BREQ, PORT Pulse Width</td>
<td>2 T1</td>
<td>1, 2, 3</td>
</tr>
<tr>
<td>T27</td>
<td>D0-D31 CPU PORT Access Setup Time</td>
<td>5</td>
<td>2</td>
</tr>
<tr>
<td>T28</td>
<td>D0-D31 CPU PORT Access Hold Time</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>T29</td>
<td>PORT Setup Time</td>
<td>7</td>
<td>2</td>
</tr>
<tr>
<td>T30</td>
<td>PORT Hold Time</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>T31</td>
<td>BOFF Setup Time</td>
<td>10</td>
<td>2</td>
</tr>
<tr>
<td>T32</td>
<td>BOFF Hold Time</td>
<td>3</td>
<td>2</td>
</tr>
</tbody>
</table>
AC Characteristics (Continued)

82596CA INPUT/OUTPUT SYSTEM TIMINGS

$T_C = 0^\circ C - 85^\circ C$, $V_{CC} = 5V \pm 5\%$. These timing assume the $C_L$ on all outputs is 50 pF unless otherwise specified. $C_L$ can be 20 pF to 120 pF, however timings must be derated. All timing requirements are given in nanoseconds.

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Parameter</th>
<th>33 MHz</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Min</td>
<td>Max</td>
</tr>
<tr>
<td></td>
<td>Operating Frequency</td>
<td>12.5 MHz</td>
<td>33 MHz</td>
</tr>
<tr>
<td>T1</td>
<td>CLK Period</td>
<td>30</td>
<td>80</td>
</tr>
<tr>
<td>T1a</td>
<td>CLK Period Stability</td>
<td></td>
<td>0.1%</td>
</tr>
<tr>
<td>T2</td>
<td>CLK High</td>
<td>11</td>
<td></td>
</tr>
<tr>
<td>T3</td>
<td>CLK Low</td>
<td>11</td>
<td></td>
</tr>
<tr>
<td>T4</td>
<td>CLK Rise Time</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>T5</td>
<td>CLK Fall Time</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>T6</td>
<td>BEn, LOCK, and A2–A31 Valid Delay</td>
<td>3</td>
<td>19</td>
</tr>
<tr>
<td>T6a</td>
<td>BLAST, PCHK Valid Delay</td>
<td>3</td>
<td>22</td>
</tr>
<tr>
<td>T7</td>
<td>BEn, LOCK, BLAST, A2–A31 Float Delay</td>
<td>3</td>
<td>20</td>
</tr>
<tr>
<td>T8</td>
<td>W/R and ADS Valid Delay</td>
<td>3</td>
<td>19</td>
</tr>
<tr>
<td>T9</td>
<td>W/R and ADS Float Delay</td>
<td>3</td>
<td>20</td>
</tr>
<tr>
<td>T10</td>
<td>D0–D31, DPn Write Data Valid Delay</td>
<td>3</td>
<td>19</td>
</tr>
<tr>
<td>T11</td>
<td>D0–D31, DPn Write Data Float Delay</td>
<td>3</td>
<td>20</td>
</tr>
<tr>
<td>T12</td>
<td>HOLD Valid Delay</td>
<td>3</td>
<td>19</td>
</tr>
<tr>
<td>T13</td>
<td>CA and BREQ Setup Time</td>
<td>7</td>
<td></td>
</tr>
<tr>
<td>T14</td>
<td>CA and BREQ Hold Time</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>T15</td>
<td>BST6 Setup Time</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>T16</td>
<td>BST6 Hold Time</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>T17</td>
<td>BRDY, RDY Setup Time</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>T18</td>
<td>BRDY, RDY Hold Time</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>T19</td>
<td>D0–D31, DPn READ Setup Time</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>T20</td>
<td>D0–D31, DPn READ Hold Time</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>T21</td>
<td>AHOLD and HLDA Setup Time</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>T22</td>
<td>AHOLD Hold Time</td>
<td>3</td>
<td></td>
</tr>
</tbody>
</table>
AC Characteristics (Continued)

82596CA INPUT/OUTPUT SYSTEM TIMINGS

$C_L$ on all outputs is 50 pF unless otherwise specified.
All timing requirements are given in nanoseconds.

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Parameter</th>
<th>$33 \text{ MHz}$</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Min</td>
<td>Max</td>
</tr>
<tr>
<td>T22a</td>
<td>HLDA Hold Time</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>T23</td>
<td>RESET Setup Time</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>T24</td>
<td>RESET Hold Time</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>T25</td>
<td>INT/INT Valid Delay</td>
<td>1</td>
<td>20</td>
</tr>
<tr>
<td>T26</td>
<td>CA and BREQ, PORT Pulse Width</td>
<td>2T1</td>
<td></td>
</tr>
<tr>
<td>T27</td>
<td>D0–D31 CPU PORT Access Setup Time</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>T28</td>
<td>D0–D31 CPU PORT Access Hold Time</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>T29</td>
<td>PORT Setup Time</td>
<td>7</td>
<td></td>
</tr>
<tr>
<td>T30</td>
<td>PORT Hold Time</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>T31</td>
<td>BOFF Setup Time</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>T32</td>
<td>BOFF Hold Time</td>
<td>3</td>
<td></td>
</tr>
</tbody>
</table>

**NOTES:**
1. RESET, HLDA, and CA are internally synchronized. This timing is to guarantee recognition at next clock for RESET, HLDA and CA.
2. All set-up, hold and delay timings are at maximum frequency specification $F_{\text{max}}$, and must be derated according to the following equation for operation at lower frequencies:
   \[
   T_{\text{derated}} = \frac{F_{\text{max}}}{F_{\text{opr}}} \times T
   \]
   where:
   - $T_{\text{derated}}$ = Specifies the value to derate the specification.
   - $F_{\text{max}}$ = Maximum operating frequency.
   - $F_{\text{opr}}$ = Actual operating frequency.
   - $T$ = Specification at maximum frequency.
   This calculation only provides a rough estimate for derating the frequency. For more detailed information, contact your Intel Sales Office for the data sheet supplement.
3. CA pulse width need only be 1 T1 wide if the set up and hold times are met; BREQ must meet setup and hold times and need only be 1 T1 wide.

**TRANSMIT/RECEIVE CLOCK PARAMETERS**

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Parameter</th>
<th>$20 \text{ MHz}$</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Min</td>
<td>Max</td>
</tr>
<tr>
<td>T36</td>
<td>TxC Cycle</td>
<td>50</td>
<td></td>
</tr>
<tr>
<td>T38</td>
<td>TxC Rise Time</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>T39</td>
<td>TxC Fall Time</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>T40</td>
<td>TxC High Time</td>
<td>19</td>
<td></td>
</tr>
<tr>
<td>T41</td>
<td>TxC Low Time</td>
<td>18</td>
<td></td>
</tr>
<tr>
<td>T42</td>
<td>TxD Rise Time</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>T43</td>
<td>TxD Fall Time</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>T44</td>
<td>TxD Transition</td>
<td>20</td>
<td></td>
</tr>
<tr>
<td>T45</td>
<td>TxC Low to TxD Valid</td>
<td>25</td>
<td></td>
</tr>
<tr>
<td>T46</td>
<td>TxC Low to TxD Transition</td>
<td>25</td>
<td></td>
</tr>
<tr>
<td>T47</td>
<td>TxC High to TxD Transition</td>
<td>25</td>
<td></td>
</tr>
<tr>
<td>T48</td>
<td>TxC Low to TxD High (At End of Transition)</td>
<td>25</td>
<td></td>
</tr>
</tbody>
</table>
## TRANSMIT/RECEIVE CLOCK PARAMETERS (Continued)

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Parameter</th>
<th>20 MHz</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Min</td>
<td>Max</td>
</tr>
</tbody>
</table>

### RTS AND CTS PARAMETERS

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Parameter</th>
<th>Min</th>
<th>Max</th>
</tr>
</thead>
<tbody>
<tr>
<td>T49</td>
<td>TxC Low to RTS Low, Time to Activate RTS</td>
<td>25</td>
<td>5</td>
</tr>
<tr>
<td>T50</td>
<td>CTS Low to TxC Low, CTS Setup Time</td>
<td>20</td>
<td></td>
</tr>
<tr>
<td>T51</td>
<td>TxC Low to CTS Invalid, CTS Hold Time</td>
<td>10</td>
<td>7</td>
</tr>
<tr>
<td>T52</td>
<td>TxC Low to RTS High</td>
<td>25</td>
<td>5</td>
</tr>
</tbody>
</table>

### RECEIVE CLOCK PARAMETERS

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Parameter</th>
<th>Min</th>
<th>Max</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>T53</td>
<td>RXC Cycle</td>
<td>50</td>
<td></td>
<td>1, 3</td>
</tr>
<tr>
<td>T54</td>
<td>RXC Rise Time</td>
<td>5</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>T55</td>
<td>RXC Fall Time</td>
<td>5</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>T56</td>
<td>RXC High Time</td>
<td>19</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>T57</td>
<td>RXC Low Time</td>
<td>18</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>

### RECEIVED DATA PARAMETERS

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Parameter</th>
<th>Min</th>
<th>Max</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>T58</td>
<td>RXD Setup Time</td>
<td>20</td>
<td></td>
<td>6</td>
</tr>
<tr>
<td>T59</td>
<td>RXD Hold Time</td>
<td>10</td>
<td></td>
<td>6</td>
</tr>
<tr>
<td>T60</td>
<td>RXD Rise Time</td>
<td></td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>T61</td>
<td>RXD Fall Time</td>
<td></td>
<td>10</td>
<td></td>
</tr>
</tbody>
</table>

### CRS AND CDT PARAMETERS

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Parameter</th>
<th>Min</th>
<th>Max</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>T62</td>
<td>CDT Low to TXC HIGH</td>
<td>20</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>External Collision Detect Setup Time</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T63</td>
<td>TXC High to CDT Inactive, CDT Hold Time</td>
<td>10</td>
<td></td>
<td></td>
</tr>
<tr>
<td>T64</td>
<td>CDT Low to Jam Start</td>
<td></td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>T65</td>
<td>CRS Low to TXC High, Carrier Sense Setup Time</td>
<td>20</td>
<td></td>
<td></td>
</tr>
<tr>
<td>T66</td>
<td>TXC High to CRS Inactive, CRS Hold Time</td>
<td>10</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>(Internal Collision Detect)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T67</td>
<td>CRS High to Jamming Start,</td>
<td></td>
<td>12</td>
<td></td>
</tr>
<tr>
<td>T68</td>
<td>Jamming Period</td>
<td></td>
<td>11</td>
<td></td>
</tr>
<tr>
<td>T69</td>
<td>CRS High to RXC High, CRS Inactive Setup Time</td>
<td>30</td>
<td></td>
<td></td>
</tr>
<tr>
<td>T70</td>
<td>RXC High to CRS High, CRS Inactive Hold Time</td>
<td>10</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### TRANSMIT/RECEIVE CLOCK PARAMETERS (Continued)

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Parameter</th>
<th>20 MHz</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Min</td>
<td>Max</td>
</tr>
<tr>
<td>INTERFRAME SPACING PARAMETERS</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T71</td>
<td>Interframe Delay</td>
<td></td>
<td></td>
</tr>
<tr>
<td>EXTERNAL LOOPBACK-PIN PARAMETERS</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T72</td>
<td>TXC Low to LPBK Low</td>
<td>T36</td>
<td></td>
</tr>
<tr>
<td>T73</td>
<td>TXC Low to LPBK High</td>
<td>T36</td>
<td></td>
</tr>
</tbody>
</table>

**NOTES:**

1. Special MOS levels. \( V_{CIL} = 0.9V \) and \( V_{CHI} = -3.0V \).
2. Manchester only.
4. 1 TTL load + 50 pF.
5. 1 TTL load + 100 pF.
6. NRZ only.
7. Abnormal end of transmission—CTS expires before RTS.
8. Normal end to transmission.
9. Programmable value:
   \[ T71 = N_{IFS} \times T36 \]
   where: \( N_{IFS} = \) the IFS configuration value
   (if \( N_{IFS} \) is less than 12 then \( N_{IFS} \) is forced to 12).
10. Programmable value:
    \[ T64 = (N_{CDF} \times T36) + x \times T36 \]
    (If the collision occurs after the preamble)
    where:
    \( N_{CDF} = \) the collision detect filter configuration value,
    and
    \( x = 12, 13, 14, \) or \( 15 \)
11. \( T68 = 32 \times T36 \)
12. Programmable value:
    \[ T67 = (N_{CSF} \times T36) + x \times T36 \]
    where: \( N_{CSF} = \) the Carrier Sense Filter configuration value,
    and
    \( x = 12, 13, 14, \) or \( 15 \)
13. To guarantee recognition on the next clock.
82596CA BUS OPERATION

The following figures show the 82596CA basic bus cycle and basic burst cycle. Please refer to the 32-Bit LAN Components Manual.

Figure 44. Basic 82596CA Bus Cycle

Figure 45. Basic 82596CA Burst Cycle
SYSTEM INTERFACE A.C. TIMING CHARACTERISTICS

The measurements should be done at:

- $T_C = 0\,^\circ\mathrm{C} - 85\,^\circ\mathrm{C}, \, V_{\text{CC}} = 5\, \text{V} \pm 10\%, \, C = 50\, \text{pF}$ unless otherwise specified.
- 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 logic “1” and “0”.
- Rise and Fall time of inputs and outputs signals are measured between 0.8V and 2.0V respectively unless otherwise specified.
- All timings are relative to CLK crossing the 1.5V level.
- All A.C. parameters are valid only after 100 $\mu$s from power up.

\[
\text{Figure 46. CLK Timings}
\]

Two types of timing specifications are presented below:

1. Input Timing—minimum setup and hold times.
2. Output Timing—output delays and float times from CLK rising edge.

Figure 47 defines how the measurements should be done:

\[
\text{Figure 47. Drive Levels and Measurements Points for A.C. Specifications}
\]

- $Ts = T_{13}, T_{15}, T_{17}, T_{19}, T_{21}, T_{23}, T_{27}, T_{29}, T_{31}$
- $Th = T_{14}, T_{16}, T_{18}, T_{20}, T_{22}, T_{22a}, T_{24}, T_{28}, T_{30}, T_{32}$
- $Tn = T_{6}, T_{6a}, T_{7}, T_{8}, T_{9}, T_{10}, T_{11}, T_{12}, T_{25}$
- $Tx = T_{6}, T_{6a}, T_{7}, T_{8}, T_{9}, T_{10}, T_{11}, T_{12}, T_{25}$
INPUT WAVEFORMS

Figure 48. CA and BREQ Input Timing

Figure 49. INT/INT Output Timing

Figure 50. HOLD/HLDA Timings

Figure 51. Input Setup and Hold Time
Figure 52. Output Valid Delay Timing

Figure 53. Output Float Delay Timing

Figure 54. PORT Setup and Hold Time
SERIAL AC TIMING CHARACTERISTICS

Figure 55. RESET Input Timing

Figure 56. Serial Input Clock Timing

Figure 57. Transmit Data Waveforms
Figure 58. Transmit Data Waveforms

Figure 59. Receive Data Waveforms (NRZ)

Figure 60. Receive Data Waveforms (CRS)
OUTLINE DIAGRAMS

132 LEAD CERAMIC PIN GRID ARRAY PACKAGE INTEL TYPE A

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Millimeters</th>
<th>Inches</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>3.56</td>
<td>0.140</td>
<td>0.180</td>
</tr>
<tr>
<td>A1</td>
<td>0.76</td>
<td>0.030</td>
<td>0.050</td>
</tr>
<tr>
<td>A2</td>
<td>2.67</td>
<td>0.105</td>
<td>0.135</td>
</tr>
<tr>
<td>A3</td>
<td>1.14</td>
<td>0.045</td>
<td>0.055</td>
</tr>
<tr>
<td>B</td>
<td>0.43</td>
<td>0.017</td>
<td>0.020</td>
</tr>
<tr>
<td>D</td>
<td>36.45</td>
<td>1.435</td>
<td>1.465</td>
</tr>
<tr>
<td>D1</td>
<td>32.89</td>
<td>1.295</td>
<td>1.305</td>
</tr>
<tr>
<td>e1</td>
<td>2.29</td>
<td>0.090</td>
<td>0.110</td>
</tr>
<tr>
<td>L</td>
<td>2.54</td>
<td>0.100</td>
<td>0.130</td>
</tr>
<tr>
<td>N</td>
<td>132</td>
<td>132</td>
<td></td>
</tr>
<tr>
<td>S1</td>
<td>1.27</td>
<td>0.050</td>
<td>0.100</td>
</tr>
</tbody>
</table>

ISSUE IWS 10/12/88
## Intel Case Outline Drawings
Plastic Quad Flat Pack (PQFP) 0.025 Inch (0.635mm) Pitch

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
<th>Min</th>
<th>Max</th>
<th>Min</th>
<th>Max</th>
<th>Min</th>
<th>Max</th>
<th>Min</th>
<th>Max</th>
<th>Min</th>
<th>Max</th>
<th>Min</th>
<th>Max</th>
</tr>
</thead>
<tbody>
<tr>
<td>N</td>
<td>Leadcount</td>
<td>68</td>
<td>84</td>
<td>100</td>
<td>132</td>
<td>164</td>
<td>196</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A</td>
<td>Package Height</td>
<td>0.160</td>
<td>0.170</td>
<td>0.160</td>
<td>0.170</td>
<td>0.160</td>
<td>0.170</td>
<td>0.160</td>
<td>0.170</td>
<td>0.160</td>
<td>0.170</td>
<td>0.160</td>
<td>0.170</td>
</tr>
<tr>
<td>A1</td>
<td>Standoff</td>
<td>0.020</td>
<td>0.030</td>
<td>0.020</td>
<td>0.030</td>
<td>0.020</td>
<td>0.030</td>
<td>0.020</td>
<td>0.030</td>
<td>0.020</td>
<td>0.030</td>
<td>0.020</td>
<td>0.030</td>
</tr>
<tr>
<td>D, E</td>
<td>Terminal Dimension</td>
<td>0.675</td>
<td>0.685</td>
<td>0.775</td>
<td>0.785</td>
<td>0.875</td>
<td>0.885</td>
<td>1.075</td>
<td>1.085</td>
<td>1.275</td>
<td>1.285</td>
<td>1.475</td>
<td>1.485</td>
</tr>
<tr>
<td>D1, E1</td>
<td>Package Body</td>
<td>0.547</td>
<td>0.553</td>
<td>0.647</td>
<td>0.653</td>
<td>0.747</td>
<td>0.753</td>
<td>0.947</td>
<td>0.953</td>
<td>1.147</td>
<td>1.153</td>
<td>1.347</td>
<td>1.353</td>
</tr>
<tr>
<td>D2, E2</td>
<td>Bumper Distance</td>
<td>0.697</td>
<td>0.703</td>
<td>0.797</td>
<td>0.803</td>
<td>0.897</td>
<td>0.903</td>
<td>1.097</td>
<td>1.103</td>
<td>1.297</td>
<td>1.303</td>
<td>1.497</td>
<td>1.503</td>
</tr>
<tr>
<td>D3, E3</td>
<td>Lead Dimension</td>
<td>0.400 REF</td>
<td>0.500 REF</td>
<td>0.600 REF</td>
<td>0.800 REF</td>
<td>1.000 REF</td>
<td>1.200 REF</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D4, E4</td>
<td>Foot Radius Location</td>
<td>0.623</td>
<td>0.637</td>
<td>0.723</td>
<td>0.737</td>
<td>0.823</td>
<td>0.837</td>
<td>1.023</td>
<td>1.037</td>
<td>1.223</td>
<td>1.237</td>
<td>1.423</td>
<td>1.437</td>
</tr>
<tr>
<td>L1</td>
<td>Foot Length</td>
<td>0.020</td>
<td>0.030</td>
<td>0.020</td>
<td>0.030</td>
<td>0.020</td>
<td>0.030</td>
<td>0.020</td>
<td>0.030</td>
<td>0.020</td>
<td>0.030</td>
<td>0.020</td>
<td>0.030</td>
</tr>
</tbody>
</table>

**Issue:** IWS Preliminary 12/12/88

---

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
<th>Min</th>
<th>Max</th>
<th>Min</th>
<th>Max</th>
<th>Min</th>
<th>Max</th>
<th>Min</th>
<th>Max</th>
<th>Min</th>
<th>Max</th>
<th>Min</th>
<th>Max</th>
</tr>
</thead>
<tbody>
<tr>
<td>N</td>
<td>Leadcount</td>
<td>68</td>
<td>84</td>
<td>100</td>
<td>132</td>
<td>164</td>
<td>196</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A</td>
<td>Package Height</td>
<td>4.06</td>
<td>4.32</td>
<td>4.06</td>
<td>4.32</td>
<td>4.06</td>
<td>4.32</td>
<td>4.06</td>
<td>4.32</td>
<td>4.06</td>
<td>4.32</td>
<td>4.06</td>
<td>4.32</td>
</tr>
<tr>
<td>A1</td>
<td>Standoff</td>
<td>0.51</td>
<td>0.76</td>
<td>0.51</td>
<td>0.76</td>
<td>0.51</td>
<td>0.76</td>
<td>0.51</td>
<td>0.76</td>
<td>0.51</td>
<td>0.76</td>
<td>0.51</td>
<td>0.76</td>
</tr>
<tr>
<td>D, E</td>
<td>Terminal Dimension</td>
<td>17.15</td>
<td>17.40</td>
<td>19.69</td>
<td>19.94</td>
<td>22.23</td>
<td>22.48</td>
<td>27.31</td>
<td>27.56</td>
<td>32.39</td>
<td>32.64</td>
<td>37.47</td>
<td>37.72</td>
</tr>
<tr>
<td>D1, E1</td>
<td>Package Body</td>
<td>13.89</td>
<td>14.05</td>
<td>16.43</td>
<td>16.59</td>
<td>18.97</td>
<td>19.13</td>
<td>24.05</td>
<td>24.21</td>
<td>29.13</td>
<td>29.29</td>
<td>34.21</td>
<td>34.37</td>
</tr>
<tr>
<td>D2, E2</td>
<td>Bumper Distance</td>
<td>17.70</td>
<td>17.85</td>
<td>20.24</td>
<td>20.39</td>
<td>22.78</td>
<td>22.93</td>
<td>27.86</td>
<td>28.01</td>
<td>32.94</td>
<td>33.09</td>
<td>38.02</td>
<td>38.18</td>
</tr>
<tr>
<td>D3, E3</td>
<td>Lead Dimension</td>
<td>10.16 REF</td>
<td>12.70 REF</td>
<td>15.24 REF</td>
<td>20.32 REF</td>
<td>25.40 REF</td>
<td>30.48 REF</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>L1</td>
<td>Foot Length</td>
<td>0.51</td>
<td>0.76</td>
<td>0.51</td>
<td>0.76</td>
<td>0.51</td>
<td>0.76</td>
<td>0.51</td>
<td>0.76</td>
<td>0.51</td>
<td>0.76</td>
<td>0.51</td>
<td>0.76</td>
</tr>
</tbody>
</table>

**Issue:** IWS Preliminary 12/12/88

---

*INCH*

---

*mm*
Figure 61. Principal Dimensions and Datums

Figure 62. Molded Details
Figure 63. Terminal Details

Figure 64. Typical Lead
REVISION HISTORY

The 82596 LAN Coprocessor data sheet version -003 contains updates and improvements to previous versions.

1. Added Pin Cross Reference table.
2. Added Bus Cycle figures.
82596DX AND 82596SX
HIGH-PERFORMANCE 32-BIT LOCAL AREA NETWORK COPROCESSOR

- Performs Complete CSMA/CD Medium Access Control (MAC) Functions—Independently of CPU
  - IEEE 802.3 (EOC) Frame Delimiting
  - HDL Frame Delimiting
- Supports Industry Standard LANs
  - IEEE TYPE 10BASE5 (Ethernet*),
  - IEEE TYPE 10BASE2 (Cheapernet),
  - IEEE TYPE 10BASE5 (StarLAN),
  - HDLC Frame Delimiting
- High-Performance 32-Bit Bus Master Access Control (MAC) Functions—Independently of CPU
- IEEE 802.3 (EOC) Frame Delimiting
- 33-MHz Clock, Two Clocks Per Transfer
- Bus Throttle Timers
- Transfers Data at 100% of Serial Bandwidth
- 128-Byte Receive FIFO, 64-Byte Transmit FIFO
- Network Management and Diagnostics
  - Monitor Mode
  - 32-Bit Statistical Counters
- Self-Test Diagnostics
- Configurable Initialization Root for Data Structures
- High-Speed, 5-V, CHMOS** IV Technology
- 132-Pin Plastic Quad Flat Pack (PQFP) and PGA Package

82586 Software Compatible

Optimized CPU Interface
- Optimized Bus Interface to Intel's 32-Bit 386™DX and 16-Bit 386™SX
- Supports Big Endian and Little Endian Byte Ordering

On-Chip Memory Management
- Automatic Buffer Chaining
- Buffer Reclamation after Receipt of Bad Frames; Optional Save Bad Frames
- 32-Bit Segmented or Linear (Flat) Memory Addressing Formats

High-Performance 32-Bit Bus Master Interface
- 66-MB/s Bus Bandwidth
- 33-MHz Clock, Two Clocks Per Transfer
- Bus Throttle Timers
- Transfers Data at 100% of Serial Bandwidth
- 128-Byte Receive FIFO, 64-Byte Transmit FIFO

Network Management and Diagnostics
- Monitor Mode
- 32-Bit Statistical Counters

Self-Test Diagnostics

Configurable Initialization Root for Data Structures

High-Speed, 5-V, CHMOS** IV Technology

132-Pin Plastic Quad Flat Pack (PQFP) and PGA Package

(See Packaging Spe. Order No. 231369)

Figure 1. 82596DX/SX Block Diagram

386™ is a trademark of Intel Corporation
*Ethernet is a registered trademark of Xerox Corporation.
**CHMOS is a patented process of Intel Corporation.
82596DX AND 82596SX HIGH-PERFORMANCE
32-BIT LOCAL AREA NETWORK COPROCESSOR

CONTENTS

INTRODUCTION ............................................. 1-112
PIN DESCRIPTIONS .......................................... 1-119
82596 AND HOST CPU INTERACTION .......................... 1-123
82596 BUS INTERFACE ........................................ 1-123
82596 MEMORY ADDRESSING .................................. 1-123
82596 SYSTEM MEMORY STRUCTURE .................................. 1-125
TRANSMIT AND RECEIVE MEMORY STRUCTURES ......................... 1-126
TRANSMITTING FRAMES ........................................... 1-129
RECEIVING FRAMES ............................................. 1-130
82596 NETWORK MANAGEMENT AND DIAGNOSTICS ...................... 1-130
NETWORK PLANNING AND MAINTENANCE ............................. 1-132
STATION DIAGNOSTICS AND SELF-TEST ............................ 1-133
82586 SOFTWARE COMPATIBILITY ..................................... 1-133
INITIALIZING THE 82596 ......................................... 1-133
SYSTEM CONFIGURATION POINTER (SCP) .......................... 1-133
Writing the Sysbus ........................................... 1-134
INTERMEDIATE SYSTEM CONFIGURATION POINTER (ISCP) ............... 1-135
INITIALIZATION PROCESS ......................................... 1-135
CONTROLLING THE 82596DX/SX ..................................... 1-136
82596 CPU ACCESS INTERFACE (PORT#) .......................... 1-136
MEMORY ADDRESSING FORMATS .................................... 1-137
LITTLE ENDIAN AND BIG ENDIAN BYTE ORDERING ...................... 1-138
COMMAND UNIT (CU) ............................................ 1-138
RECEIVE UNIT (RU) .............................................. 1-138

CONTENTS

SYSTEM CONTROL BLOCK (SCB) .................................. 1-138
SCB OFFSET ADDRESS ............................................. 1-141
CBL Offset (Address) ........................................... 1-141
RFA Offset (Address) ........................................... 1-142
SCB STATISTICAL COUNTERS ...................................... 1-142
Statistical Counter Operation ..................................... 1-142
ACTION COMMANDS AND OPERATING MODES ......................... 1-143
NOP .......................................................... 1-143
Individual Address Setup ........................................ 1-144
Configure ..................................................... 1-145
Multicast-Setup ................................................. 1-151
Transmit ..................................................... 1-152
Jamming Rules .................................................. 1-154
TDR .......................................................... 1-155
Dump .......................................................... 1-157
Diagnose ....................................................... 1-160
RECEIVE FRAME DESCRIPTOR ..................................... 1-160
Simplified Memory Structure ...................................... 1-161
Flexible Memory Structure ....................................... 1-162
Receive Buffer Descriptor (RBD) .................................. 1-163
ELECTRICAL AND TIMING CHARACTERISTICS ....................... 1-168
DC Characteristics ............................................ 1-168
AC Characteristics ............................................ 1-169
82596DX Input/Output System Timings .............................. 1-169
82596SX Input/Output System Timings ................................ 1-171
Transmit/Receive Clock Parameters .................................. 1-173
82596DX/SX BUS OPERATION ....................................... 1-175
System Interface A.C. Timing Characteristics ....................... 1-176
Input Waveforms ................................................. 1-177
Serial A.C. Timing Characteristics ................................ 1-179
OUTLINE DIAGRAMS .............................................. 1-181
REVISION HISTORY .............................................. 1-185

1-111
INTRODUCTION

The 82596DX/SX is an intelligent, high-performance 32-bit Local Area Network coprocessor. The 82596DX/SX implements the CSMA/CD access method and can be configured to support all existing IEEE 802.3 standards—TYPEs 10BASE5, 10BASE2, 1BASE5, and 10BROAD36. It can also be used to implement the proposed standards TYPE 10BASE-T and 10BASE-F. The 82596DX/SX performs high-level commands, command chaining, and interprocessor communications via shared memory, thus relieving the host CPU of many tasks associated with network control. All time-critical functions are performed independently of the CPU, this increases network performance and efficiency. The 82596DX/SX bus interface is optimized for Intel's 386TM DX, 386 SX and 376TM microprocessors.

The 82596DX/SX implements all IEEE 802.3 Medium Access Control and channel interface functions, these include framing, preamble generation and stripping, source address generation, destination address checking, short-frame detection, and automatic length-field handling. Data rates up to 20 Mb/s are supported.

The 82596DX/SX provides a powerful host system interface. It manages memory structures automatically, with command chaining and bidirectional data chaining. An on-chip DMA controller manages four channels, this allows autonomous transfer of data blocks (buffers and frames) and relieves the CPU of byte transfer overhead. Buffers containing errored or collided frames can be automatically recovered without CPU intervention. The 82596DX/SX provides an upgrade path for existing 82586 software drivers by providing an 82586-software-compatible mode that supports the current 82586 memory structure. The 82596DX/SX also has a Flexible memory structure and a Simplified memory structure. The 82596DX/SX can address up to 4 gigabytes of memory. The 82596DX/SX supports Little Endian and Big Endian byte ordering.

The 82596DX/SX bus interface is optimized to Intel's 386 DX, 386 SX, and 376 microprocessors, providing a bus transfer rate of up to 66 MB/s at 33 MHz. The bus interface employs bus throttle timers to regulate 82596DX/SX bus use. Two large, independent FIFOs—128 bytes for Receive and 64 bytes for Transmit—tolerate long bus latencies and provide programmable thresholds that allow the user to optimize bus overhead for any worst-case bus latency.

The 82596DX/SX provides a wide range of diagnostics and network management functions, these include internal and external loopback, exception condition tallies, channel activity indicators, optional capture of all frames regardless of destination address (promiscuous mode), optional capture of errored or collided frames, and time domain reflectometry for locating fault points on the network cable. The statistical counters, in 32-bit segmented and linear modes, are 32-bits each and include CRC errors, alignment errors, overrun errors, resource errors, short frames, and received collisions. The 82596DX/SX also features a monitor mode for network analysis. In this mode the 82596DX/SX can capture status bytes, and update statistical counters, of frames monitored on the link without transferring the contents of the frames to memory. This can be done concurrently while transmitting and receiving frames destined for that station.

The 82596DX/SX can be used in both baseband and broadband networks. It can be configured for maximum network efficiency (minimum contention overhead) with networks of any length. Its highly flexible CSMA/CD unit supports address field lengths of zero through six bytes—configurable to either IEEE 802.3/Ethernet or HDLC frame delimitation. It also supports 16- or 32-bit cyclic redundancy checks. The CRC can be transferred directly to memory for receive, operations or dynamically inserted for transmit operations. The CSMA/CD unit can also be configured for full duplex operation for high throughput in point-to-point connections.

The 82596DX/SX is fabricated with Intel's reliable, 5-V, CHMOS IV technology. It is available in a 132-pin PQFP or PGA package.
Figure 2a. 82596DX PQFP Pin Configuration
Figure 2b. 82596SX PQFP Pin Configuration
Figure 3a. 82596DX PGA Pin View Side
<table>
<thead>
<tr>
<th>Address</th>
<th>Data</th>
<th>Control</th>
<th>Serial Interface</th>
<th>N/C</th>
<th>Vcc</th>
<th>Vss</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A2</td>
<td>N9</td>
<td>D0</td>
<td>J2</td>
<td>ADS</td>
<td>M5</td>
<td>CDT</td>
</tr>
<tr>
<td>A3</td>
<td>M9</td>
<td>D1</td>
<td>H3</td>
<td>BE0</td>
<td>M7</td>
<td>CRS</td>
</tr>
<tr>
<td>A4</td>
<td>M10</td>
<td>D2</td>
<td>G2</td>
<td>BET</td>
<td>P5</td>
<td>CTS</td>
</tr>
<tr>
<td>A5</td>
<td>P11</td>
<td>D3</td>
<td>G3</td>
<td>BE2</td>
<td>M8</td>
<td>LPB</td>
</tr>
<tr>
<td>A6</td>
<td>N11</td>
<td>D4</td>
<td>G1</td>
<td>BE3</td>
<td>P9</td>
<td>RTS</td>
</tr>
<tr>
<td>A7</td>
<td>P12</td>
<td>D5</td>
<td>D1</td>
<td>BREQ</td>
<td>P4</td>
<td>RxC</td>
</tr>
<tr>
<td>A8</td>
<td>M11</td>
<td>D6</td>
<td>C1</td>
<td>BST</td>
<td>N1</td>
<td>RxD</td>
</tr>
<tr>
<td>A9</td>
<td>N12</td>
<td>D7</td>
<td>F3</td>
<td>CA</td>
<td>P3</td>
<td>C12</td>
</tr>
<tr>
<td>A10</td>
<td>M12</td>
<td>D8</td>
<td>D2</td>
<td>CLK</td>
<td>J3</td>
<td>TD</td>
</tr>
<tr>
<td>A11</td>
<td>P13</td>
<td>D9</td>
<td>C2</td>
<td>HLDA</td>
<td>M6</td>
<td></td>
</tr>
<tr>
<td>A12</td>
<td>L12</td>
<td>D10</td>
<td>E3</td>
<td>HOLD</td>
<td>P2</td>
<td></td>
</tr>
<tr>
<td>A13</td>
<td>N13</td>
<td>D11</td>
<td>D3</td>
<td>INT</td>
<td>N3</td>
<td></td>
</tr>
<tr>
<td>A14</td>
<td>M13</td>
<td>D12</td>
<td>B2</td>
<td>LE/BE</td>
<td>M4</td>
<td></td>
</tr>
<tr>
<td>A15</td>
<td>P14</td>
<td>D13</td>
<td>B1</td>
<td>LOCK</td>
<td>M3</td>
<td></td>
</tr>
<tr>
<td>A16</td>
<td>K12</td>
<td>D14</td>
<td>C3</td>
<td>PORT</td>
<td>B13</td>
<td></td>
</tr>
<tr>
<td>A17</td>
<td>N14</td>
<td>D15</td>
<td>A1</td>
<td>READY</td>
<td>N04</td>
<td></td>
</tr>
<tr>
<td>A18</td>
<td>J12</td>
<td>D16</td>
<td>B3</td>
<td>RESET</td>
<td>M2</td>
<td></td>
</tr>
<tr>
<td>A19</td>
<td>K13</td>
<td>D17</td>
<td>C4</td>
<td>W/R</td>
<td>B14</td>
<td></td>
</tr>
<tr>
<td>A20</td>
<td>M14</td>
<td>D18</td>
<td>A2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A21</td>
<td>H12</td>
<td>D19</td>
<td>C5</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A22</td>
<td>K14</td>
<td>D20</td>
<td>A3</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A23</td>
<td>G12</td>
<td>D21</td>
<td>B4</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A24</td>
<td>F14</td>
<td>D22</td>
<td>A4</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A25</td>
<td>F12</td>
<td>D23</td>
<td>C6</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A26</td>
<td>F13</td>
<td>D24</td>
<td>B5</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A27</td>
<td>D14</td>
<td>D25</td>
<td>C7</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A28</td>
<td>D15</td>
<td>D26</td>
<td>A5</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A29</td>
<td>D16</td>
<td>D27</td>
<td>B8</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A30</td>
<td>D17</td>
<td>D28</td>
<td>C8</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A31</td>
<td>C14</td>
<td>D29</td>
<td>A9</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>D30</td>
<td>C9</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>D31</td>
<td>B9</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
**82596DX/SX**

---

**Figure 3b. 82596SX PGA Pin View Side**

---

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
<th>G</th>
<th>H</th>
<th>J</th>
<th>K</th>
<th>L</th>
<th>M</th>
<th>N</th>
<th>P</th>
</tr>
</thead>
<tbody>
<tr>
<td>01</td>
<td></td>
<td></td>
<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>02</td>
<td></td>
<td></td>
<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>03</td>
<td></td>
<td></td>
<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>04</td>
<td></td>
<td></td>
<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>05</td>
<td></td>
<td></td>
<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>06</td>
<td></td>
<td></td>
<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>07</td>
<td></td>
<td></td>
<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>08</td>
<td></td>
<td></td>
<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>09</td>
<td></td>
<td></td>
<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>10</td>
<td></td>
<td></td>
<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>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>
</tr>
<tr>
<td>12</td>
<td></td>
<td></td>
<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>13</td>
<td></td>
<td></td>
<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>14</td>
<td></td>
<td></td>
<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>

---

**METAL LID**

82596SX

*(Pin View)*
## 82596SX PGA Cross Reference by Pin Name

<table>
<thead>
<tr>
<th>Address</th>
<th>Data</th>
<th>Control</th>
<th>Serial Interface</th>
<th>N/C</th>
<th>Vcc</th>
<th>Vss</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>Signal Pin No.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A2</td>
<td>N9</td>
<td>D0</td>
<td>J2</td>
<td>ADS</td>
<td>M5</td>
<td>CDT</td>
</tr>
<tr>
<td>A3</td>
<td>M9</td>
<td>D1</td>
<td>H3</td>
<td>BLE</td>
<td>M7</td>
<td>CRS</td>
</tr>
<tr>
<td>A4</td>
<td>M10</td>
<td>D2</td>
<td>G2</td>
<td>BHE</td>
<td>P5</td>
<td>CTS</td>
</tr>
<tr>
<td>A5</td>
<td>P11</td>
<td>D3</td>
<td>G3</td>
<td>BON</td>
<td>P9</td>
<td>LPBK</td>
</tr>
<tr>
<td>A6</td>
<td>N11</td>
<td>D4</td>
<td>G1</td>
<td>BREQ</td>
<td>P4</td>
<td>RTS</td>
</tr>
<tr>
<td>A7</td>
<td>P12</td>
<td>D5</td>
<td>D1</td>
<td>CA</td>
<td>P3</td>
<td>RxC</td>
</tr>
<tr>
<td>A8</td>
<td>M11</td>
<td>D6</td>
<td>C1</td>
<td>CLK2</td>
<td>J3</td>
<td>RxD</td>
</tr>
<tr>
<td>A9</td>
<td>N12</td>
<td>D7</td>
<td>F3</td>
<td>HLDA</td>
<td>M6</td>
<td>TxC</td>
</tr>
<tr>
<td>A10</td>
<td>M12</td>
<td>D8</td>
<td>D2</td>
<td>HOLD</td>
<td>P2</td>
<td></td>
</tr>
<tr>
<td>A11</td>
<td>P13</td>
<td>D9</td>
<td>C2</td>
<td>INT/INT</td>
<td>N3</td>
<td>TxD</td>
</tr>
<tr>
<td>A12</td>
<td>L12</td>
<td>D10</td>
<td>E3</td>
<td>LE/BE</td>
<td>B14</td>
<td></td>
</tr>
<tr>
<td>A13</td>
<td>N13</td>
<td>D11</td>
<td>D3</td>
<td>LOCK</td>
<td>M4</td>
<td></td>
</tr>
<tr>
<td>A14</td>
<td>M13</td>
<td>D12</td>
<td>B2</td>
<td>PORT</td>
<td>M2</td>
<td></td>
</tr>
<tr>
<td>A15</td>
<td>P14</td>
<td>D13</td>
<td>B1</td>
<td>RDY</td>
<td>M3 (READY)</td>
<td></td>
</tr>
<tr>
<td>A16</td>
<td>K12</td>
<td>D14</td>
<td>C3</td>
<td>RESET</td>
<td>B13</td>
<td></td>
</tr>
<tr>
<td>A17</td>
<td>N14</td>
<td>D15</td>
<td>A1</td>
<td>W/R</td>
<td>N04</td>
<td></td>
</tr>
<tr>
<td>A18</td>
<td>J12</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A19</td>
<td>K13</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A20</td>
<td>M14</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A21</td>
<td>H12</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A22</td>
<td>K14</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A23</td>
<td>G12</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A24</td>
<td>F14</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A25</td>
<td>F12</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A26</td>
<td>F13</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A27</td>
<td>D14</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A28</td>
<td>E12</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A29</td>
<td>D13</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A30</td>
<td>D12</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A31</td>
<td>C14</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
## PIN DESCRIPTIONS

<table>
<thead>
<tr>
<th>Symbol</th>
<th>PQFP Pin No.</th>
<th>Type</th>
<th>Name and Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLK2</td>
<td>9</td>
<td>I</td>
<td>CLOCK. The system clock input provides the fundamental timing for the 82596. It is internally divided by two to generate the 82596 clock. All external timing parameters are specified in reference to the rising edge of CLK2. For clock levels see D.C. Characteristics.</td>
</tr>
<tr>
<td>D31–D0</td>
<td>14–53</td>
<td>I/O</td>
<td>DATA BUS. The 32 Data Bus lines are bidirectional, tri-state lines that provide the general purpose data path between the 82596 and memory. With the 82596DX the bus can be either 16 or 32 bits wide; this is determined by the BS16 signal which is static. The 82596 always drives all 32 data lines during Write operations, even with a 16-bit bus. D0–D31 are floated after a Reset or when the bus is not acquired. These lines are inputs during a CPU Port access; in this mode the CPU writes the next address to the 82596 through the Data lines. During PORT commands (Relocatable SCP, Self-Test, and Dump) the address must be aligned to a 16 byte boundary. This frees the D3–D0 lines so they can be used to distinguish the commands. The following is a summary of the decoding data.</td>
</tr>
<tr>
<td>(D15–D0)</td>
<td></td>
<td></td>
<td>These 16 Data Bus lines are bidirectional, tri-state lines that provide the entire data path for the 82596SX. In the 82596SX D16–D31 are not connected (NC).</td>
</tr>
<tr>
<td>A31–A2</td>
<td>70–108</td>
<td>O</td>
<td>ADDRESS LINES. These 30 tri-stated Address lines output the address bits required for memory operation. These lines are floated after a Reset or when the bus is not acquired.</td>
</tr>
<tr>
<td>A1</td>
<td>112</td>
<td>O</td>
<td>The 82596SX requires this additional address line to output the address bits required for memory operation.</td>
</tr>
<tr>
<td>BE3–BE0</td>
<td>109–114</td>
<td>O</td>
<td>BYTE ENABLE. (82596DX only.) These tri-stated signals are used to indicate which bytes are involved with the current memory access. The number of Byte Enable signals asserted indicates the physical size of the data being transferred (1, 2, 3, or 4 bytes).</td>
</tr>
<tr>
<td>BHE, BLE</td>
<td>109–114</td>
<td>O</td>
<td>(82596SX only.) These signals are the Byte High Enable and Byte Low Enable signals for the 82596SX.</td>
</tr>
<tr>
<td>BON</td>
<td>109</td>
<td>O</td>
<td>BUS ON. (82596SX only.) This signal is driven high when the 82596 is holding the bus. This signal is tri-stated when the bus is relinquished. BON has the same timing as the Byte Enables.</td>
</tr>
</tbody>
</table>
### PIN DESCRIPTIONS (Continued)

<table>
<thead>
<tr>
<th>Symbol</th>
<th>PQFP Pin No.</th>
<th>Type</th>
<th>Name and Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>W/R</td>
<td>120</td>
<td>O</td>
<td>WRITE/READ. This dual-function pin is used to distinguish Write and Read cycles. This line is floated after a Reset or when the bus is not acquired.</td>
</tr>
<tr>
<td>ADS</td>
<td>124</td>
<td>O</td>
<td>ADDRESS STATUS. This tri-state pin is used by the 82596 to indicate that a valid bus cycle has begun and that A31–A2, BE3–BE0, and W/R are being driven. It is asserted during t1 bus states. This line is floated after a Reset or when the bus is not acquired.</td>
</tr>
<tr>
<td>RDY</td>
<td>130</td>
<td>I</td>
<td>READY. Active low. This signal is the acknowledgment from addressed memory that the transfer cycle can be completed. When high, it causes wait states to be inserted. It is ignored at the end of the first clock of the bus cycle’s data cycle. This active-low signal does not have an internal pull-up resistor. This signal must meet the setup and hold times to operate correctly.</td>
</tr>
<tr>
<td>LOCK</td>
<td>126</td>
<td>O</td>
<td>LOCK. This tri-state pin is used to distinguish locked and unlocked bus cycles. LOCK generates a semaphore handshake to the CPU. LOCK can be active for several memory cycles, it goes active during the first locked memory cycle (t1) and goes inactive at the last locked cycle (t2). This line is floated after a Reset or when the bus is not acquired. LOCK can be disabled via the sysbus byte in software.</td>
</tr>
<tr>
<td>BS16</td>
<td>129</td>
<td>I</td>
<td>BUS SIZE. This signal allows the 82596DX to work with either 16- or 32-bit bytes. This signal is static and should be tied high for 32-bit operation or low for 16-bit operation. In Little Endian mode the D0–D15 lines are driven when BS16 is inserted, in Big Endian mode the D16–D31 lines are driven.</td>
</tr>
<tr>
<td>HOLD</td>
<td>123</td>
<td>O</td>
<td>HOLD. The HOLD signal is active high, the 82596 uses it to request local bus mastership. In normal operation HOLD goes inactive before HLDA. The 82596 can be forced off the bus by deasserting HLDA or if the bus throttle timers expire.</td>
</tr>
<tr>
<td>HLDA</td>
<td>118</td>
<td>I</td>
<td>HOLD ACKNOWLEDGE. The HLDA signal is active high, it indicates that bus mastership has been given to the 82596. HLDA is internally synchronized; after HOLD is detected low, the CPU drives HLDA low. <strong>NOTE</strong> Do not connect HLDA to VCC—it will cause a deadlock. A user wanting to give the 82596 permanent access to the bus should connect HLDA to HOLD. If HLDA goes inactive before HOLD, the 82596 will release the bus (by deasserting HOLD) within a specified number of system clocks.</td>
</tr>
<tr>
<td>BREQ</td>
<td>115</td>
<td>I</td>
<td>BUS REQUEST. This signal, when configured to an externally activated mode, is used to trigger the bus throttle timers.</td>
</tr>
</tbody>
</table>
**PIN DESCRIPTIONS (Continued)**

<table>
<thead>
<tr>
<th>Symbol</th>
<th>PQFP Pin No.</th>
<th>Type</th>
<th>Name and Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>PORT</td>
<td>3</td>
<td>I</td>
<td><strong>PORT.</strong> When this signal is received, the 82596 latches the data on the data bus into an internal 32-bit register. When the CPU is asserting this signal it can write into the 82596 (via the data bus). This pin must be activated twice during all CPU Port access commands.</td>
</tr>
<tr>
<td>RESET</td>
<td>69</td>
<td>I</td>
<td><strong>RESET.</strong> This active high, internally synchronized signal causes the 82596 to terminate current activity. The signal must be high for at least five system clock cycles. After five system clock cycles and four TxC clock cycles the 82596 will execute a Reset when it receives a high RESET signal. When RESET returns to low, the 82596 waits for the first CA signal and then begins the initialization sequence.</td>
</tr>
<tr>
<td>LE/BE</td>
<td>65</td>
<td>I</td>
<td><strong>LITTLE ENDIAN/BIG ENDIAN.</strong> This dual-function pin is used to select byte ordering. When LE/BE is high, little endian byte ordering is used; when low, big endian byte ordering is used for data in frames (bytes) and for control (SCB, RFD, CBL, etc.).</td>
</tr>
<tr>
<td>CA</td>
<td>119</td>
<td>I</td>
<td><strong>CHANNEL ATTENTION.</strong> The CPU uses this pin to force the 82596 to begin executing memory resident Command blocks. The CA signal is internally synchronized. The signal must be high for at least one system clock. It is latched internally on the high to low edge and then detected by the 82596. The first CA after a Reset forces the 82596 into the initialization sequence beginning at location 00FFFFFF6h or an SCP address written to the 82596 using CPU Port access. All subsequent CA signals cause the 82596 to begin executing new command sequences from the SCB.</td>
</tr>
<tr>
<td>INT/INT</td>
<td>125</td>
<td>O</td>
<td><strong>INTERRUPT.</strong> A high signal on this pin notifies the CPU that the 82596 is requesting an interrupt. This signal is an edge triggered interrupt signal, and can be configured to be active high or low.</td>
</tr>
<tr>
<td>VCC</td>
<td>18 Pins</td>
<td></td>
<td><strong>POWER.</strong> +5V ± 10%.</td>
</tr>
<tr>
<td>VSS</td>
<td>18 Pins</td>
<td></td>
<td><strong>GROUND.</strong> 0V.</td>
</tr>
<tr>
<td>TxD</td>
<td>54</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>TxC</td>
<td>64</td>
<td>I</td>
<td><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. Transmit clock should always be running for proper device operation.</td>
</tr>
</tbody>
</table>
## PIN DESCRIPTIONS (Continued)

<table>
<thead>
<tr>
<th>Symbol</th>
<th>PQFP Pin No.</th>
<th>Type</th>
<th>Name and Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>LPBK</td>
<td>58</td>
<td>O</td>
<td>LOOPBACK. This TTL-level control signal enables the loopback mode. In this mode serial data on the TxD input is routed through the 82C501 internal circuits and back to the RxD output without driving the transceiver cable. To enable this signal, both internal and external loopback need to be set with the Configure command.</td>
</tr>
<tr>
<td>RxD</td>
<td>60</td>
<td>I</td>
<td>RECEIVE DATA. This pin receives NRZ serial data only. It must be high when not receiving.</td>
</tr>
<tr>
<td>RxC</td>
<td>59</td>
<td>I</td>
<td>RECEIVE CLOCK. This signal provides timing information to the internal shifting logic. For NRZ data the state of the RxD pin is sampled on the high to low transition of the clock.</td>
</tr>
<tr>
<td>RTS</td>
<td>57</td>
<td>O</td>
<td>REQUEST TO SEND. When this signal is low the 82596 informs the external interface that it has data to transmit. It is forced high after a Reset or when transmission is stopped.</td>
</tr>
<tr>
<td>CTS</td>
<td>62</td>
<td>I</td>
<td>CLEAR TO SEND. An active-low signal that enables the 82596 to send data. It is normally used as an interface handshake to RTS. Asserting CTS high stops transmission. CTS is internally synchronized. If CTS goes inactive, meeting the setup time to the TxC negative edge, the transmission will stop and RTS will go inactive within, at most, two TxC cycles.</td>
</tr>
<tr>
<td>CRS</td>
<td>63</td>
<td>I</td>
<td>CARRIER SENSE. This signal is active low, it is used to notify the 82596 that traffic is on the serial link. It is only used if the 82596 is configured for external Carrier Sense. In this configuration external circuitry is required for detecting traffic on the serial link. CRS is internally synchronized. To be accepted, the signal must remain active for at least two serial clock cycles (for CRSF = 0).</td>
</tr>
<tr>
<td>CDT</td>
<td>61</td>
<td>I</td>
<td>COLLISION DETECT. This active-low signal informs the 82596 that a collision has occurred. It is only used if the 82596 is configured for external Collision Detect. External circuitry is required for collision detection. CDT is internally synchronized. To be accepted, the signal must remain active for at least two serial clock cycles (for CDTF = 0).</td>
</tr>
</tbody>
</table>
82596 AND HOST CPU INTERACTION

The 82596DX/SX and the host CPU communicate through shared memory. Because of its on-chip DMA capability, the 82596 can make data block transfers (buffers and frames) independently of the CPU; this greatly reduces the CPU byte transfer overhead.

NOTE:
The 82596DX and 82596SX differ in their address pin definitions and their data bus sizes. Information in this data sheet applies to both versions unless otherwise stated.

The 82596 is a multitasking coprocessor that comprises two independent logical units—the Command Unit (CU) and the Receive Unit (RU). The CU executes commands from shared memory. The RU handles all activities related to frame reception. The independence of the CU and RU enables the 82596 to engage in both activities simultaneously—the CU can fetch and execute commands from memory while the RU is storing received frames in memory. The CPU is only involved with this process after the CU has executed a sequence of commands or the RU has finished storing a sequence of frames.

The CPU and the 82596 use the hardware signals Interrupt (INT) and Channel Attention (CA) to initiate communication with the System Control Block (SCB), see Figure 4. The 82596 uses INT to alert the CPU of a change in the contents of the SCB, the CPU uses CA to alert the 82596.

The 82596 has a CPU Port Access state that allows the CPU to execute certain functions without accessing memory. The 82596 PORT pin and data bus pins are used to enable this feature. The CPU can directly activate four operations when the 82596 is in this state:

- Write an alternative System Configuration Pointer (SCP). This can be used when the 82596 cannot use the default SCP address space.
- Write a different Dump Command Pointer and execute Dump. This can be used for troubleshooting No Response problems.

- The CPU can reset the 82596 via software without disturbing the rest of the system.
- A self-test can be used for board testing; the 82596 will execute a self-test and write the results to memory.

82596 BUS INTERFACE

The 82596DX/SX has bus interface timings and pin definitions that are compatible with Intel's 32-bit 386 DX, 386 SX, and 376 microprocessors. This eliminates the need for additional bus interface logic. Operating at 33 MHz, the 82596's bus bandwidth can be as high as 66 MB/s. Since Ethernet only requires 1.25 MB/s, this leaves a considerable amount of bandwidth for the CPU. The 82596 also has a bus throttle to regulate its use of the bus. Two timers can be programmed through the SCB: one controls the maximum time the 82596 can remain on the bus, the other controls the time the 82596 must stay off the bus (see Figure 5). The bus throttle can be programmed to trigger internally with HLDA or externally with BREQ. These timers can restrict the 82596 HOLD activation time and improve bus utilization.

82596 MEMORY ADDRESSING

The 82596 has a 32-bit memory address range, which allows addressing up to four gigabytes of memory. The 82596 has three memory addressing modes (see Table 1).

- **82586 Mode.** The 82596 has a 24-bit memory address range. The System Control Block, Command List, Receive Descriptor List, and Buffer Descriptors must reside in one 64-kB memory segment. Transmit and Receive buffers can reside in a 24-bit address space.
- **32-Bit Segmented Mode.** The 82596 has a 32-bit memory address range. The System Control Block, Command List, Receive Descriptor List, and Buffer Descriptors must reside in one 64-kB memory segment. Transmit and Receive buffers can reside in a 32-bit address space.
- **Linear Mode.** The 82596 has a 32-bit memory address range. Any memory structure can reside anywhere within the 32-bit memory address range.
Figure 4. 82596 and Host CPU Intervention

Figure 5. Bus Throttle Timers

Table 1. 82596 Memory Addressing Formats

<table>
<thead>
<tr>
<th>Pointer or Offset</th>
<th>82586</th>
<th>32-Bit Segmented</th>
<th>Linear</th>
</tr>
</thead>
<tbody>
<tr>
<td>ISCP ADDRESS</td>
<td>24-Bit Linear</td>
<td>32-Bit Linear</td>
<td>32-Bit Linear</td>
</tr>
<tr>
<td>SCB ADDRESS</td>
<td>Base (24) + Offset (16)</td>
<td>Base (32) + Offset (16)</td>
<td>32-Bit Linear</td>
</tr>
<tr>
<td>Command Block Pointers</td>
<td>Base (24) + Offset (16)</td>
<td>Base (32) + Offset (16)</td>
<td>32-Bit Linear</td>
</tr>
<tr>
<td>Rx Frame Descriptors</td>
<td>Base (24) + Offset (16)</td>
<td>Base (32) + Offset (16)</td>
<td>32-Bit Linear</td>
</tr>
<tr>
<td>Tx Frame Descriptors</td>
<td>Base (24) + Offset (16)</td>
<td>Base (32) + Offset (16)</td>
<td>32-Bit Linear</td>
</tr>
<tr>
<td>Rx Buffer Descriptors</td>
<td>Base (24) + Offset (16)</td>
<td>Base (32) + Offset (16)</td>
<td>32-Bit Linear</td>
</tr>
<tr>
<td>Tx Buffer Descriptors</td>
<td>Base (24) + Offset (16)</td>
<td>Base (32) + Offset (16)</td>
<td>32-Bit Linear</td>
</tr>
<tr>
<td>Rx Buffers</td>
<td>24-Bit Linear</td>
<td>32-Bit Linear</td>
<td>32-Bit Linear</td>
</tr>
<tr>
<td>Tx Buffers</td>
<td>24-Bit Linear</td>
<td>32-Bit Linear</td>
<td>32-Bit Linear</td>
</tr>
</tbody>
</table>
82596 SYSTEM MEMORY STRUCTURE

The Shared Memory structure consists of four parts: the Initialization Root, the System Control Block, the Command List, and the Receive Frame Area (see Figure 6).

The Initialization Root is in an established location known to the host CPU and the 82596 (0FFFFFF6h). However, the CPU can establish the Initialization Root in another location by using the CPU Port access. This root is accessed during initialization, and points to the System Control Block.

The System Control Block serves as a bidirectional mail drop for the host CPU and the 82596 CU and RU. It is the central point through which the CPU and the 82596 exchange control and status information. The SCB has two areas. The first contains instructions from the CPU to the 82596. These include: control of the CU and RU (Start, Abort, Suspend, and Resume), a pointer to the list of CU commands, a pointer to the Receive Frame Area, a set of Interrupt Acknowledge bits, and the T-ON and T-OFF timers for the bus throttle. The second area contains status information the 82596 is sending to the CPU. Such as, the CU and RU states (Idle, Active...
In interrupt bits (Command Completed, Frame Received, CU Not Ready, and RU Not Ready), and contain the parameters and status of specific counters.

The Command List functions as a program for the CU; individual commands are placed in memory units called Command Blocks (CBs). These CBs contain the parameters and status of specific high-level commands called Action Commands; e.g., Transmit or Configure.

Transmit causes the 82596 to transmit a frame. The Transmit CB contains the destination address, the length field, and a pointer to a list of linked buffers holding the frame that is to be constructed from several buffers scattered throughout memory. The Command Unit operates without CPU intervention; the DMA for each buffer, and the prefetching of references to new buffers, is performed in parallel. The CPU is notified only after a transmission is complete.

The Receive Frame Area is a list of Free Frame Descriptors (descriptors not yet used) and a list of user-prepared buffers. Frames arrive at the 82596 unsolicited; the 82596 must always be ready to receive and store them in the Free Frame Area. The Receive Unit fills the buffers when it receives frames, and reformats the Free Buffer List into received-frame structures. The frame structure is, for all practical purposes, identical to the format of the frame to be transmitted. The first Frame descriptor is referenced by the SCB. Unless the 82596 is configured to Save Bad Frames, the frame descriptor, and the associated buffer descriptor, which is wasted when a bad frame is received, are automatically reclaimed and returned to the Free Buffer List.

Receive buffer chaining (storing incoming frames in a linked buffer list) significantly improves memory utilization. Without buffer chaining, the user must allocate consecutive blocks of memory, each capable of containing a maximum frame (for Ethernet, 1518 bytes). Since an average frame is about 200 bytes, this is very inefficient. With buffer chaining, the user can allocate small buffers and the 82596 will only use those that are needed.

Figure 7 A-D illustrates how the 82596 uses the Receive Frame Area. Figure 7A shows an unused Receive Frame Area composed of Free Frame Descriptors and Free Receive Buffers prepared by the user. The SCB points to the first Frame Descriptor of the Frame Descriptor List. Figure 7B shows the same Receive Frame Area after receiving one frame. This first frame occupies two Receive Buffers and one Frame Descriptor—a valid received frame will only occupy one Frame Descriptor. After receiving this frame the 82596 sets the next Free Frame Descriptor RBD pointer to the next Free RBD. Figure 7C shows the RFA after receiving a second frame. In this example the second frame occupies only one Receive Buffer and one RFD. The 82596 again sets the RBD pointer. This process is repeated again in Figure 7D, showing the reception of another frame using one Receive Buffer; in this example there is an extra Frame Descriptor.

TRANSMIT AND RECEIVE MEMORY STRUCTURES

There are three memory structures for reception and transmission. The 82586 memory structure, the Flexible memory structure, and the Simplified memory structure. The 82586 mode is selected by configuring the 82596 during initialization. In this mode all the 82596 memory structures are compatible with the 82586 memory structures.

When the 82596 is not configured to the 82586 mode, the other two memory structures, Simplified and Flexible, are available for transmitting and receiving. These structures can be selected on a frame-by-frame basis by setting the S/F bit in the Transmit Command and the Receive Frame Descriptor (see Figures 29, 30, 41, and 42). The Simplified memory structure offers a simple structure for ease of programming (see Figure 8). All information about a frame is contained in one structure; for example, during reception the RFD and data field are contained in one structure.

The Flexible memory structure (see Figure 9) has a control field that allows the programmer to specify the amount of receive data the RFD will contain for receive operations and the amount of transmit data the Transmit Command Block will contain for transmit operations. For example, when the control field in the RFD is set to 20 bytes during a reception, the first 20 bytes of the data field are stored in the RFD (6 Bytes of Destination Address, 6 Bytes of Source Address, 2 Bytes of Length Field, and 6 Bytes of Data), and the remainder of the data field is stored in the Receive Data Buffers. This is useful for capturing frame headers when header information is contained in the data field. The header information can then be automatically stored in the RFD partitioned from the Receive Data Buffer.

The control field can also be used for the Transmit Command when the Flexible memory structure is used. The quantity of data field bytes to be transmitted from the Transmit Command Block is specified by the variable control field.
Figure 7. Frame Reception in the RFA
Figure 8. Simplified Memory Structure

Figure 9. Flexible Memory Structure
TRANSMITTING FRAMES

The 82596 executes high-level Action Commands from the Command List in system memory. Action Commands are fetched and executed in parallel with the host CPU operation, thereby significantly improving system performance. The format of the Action Commands is shown in Figure 10. Figure 28 shows the 82586 mode, and Figures 29 and 30 shows the command formats of the Linear and 32-bit Segmented modes.

A single Transmit command contains, as part of the command-specific parameters, the destination address and length field of the transmitted frame and a pointer to buffer area in memory containing the data portion of the frame. The data field is contained in a memory data structure consisting of a buffer descriptor (BD) and a data buffer—or a linked list of buffer descriptors and buffers—as shown in Figure 11.

Multiple data buffers can be chained together using the BDs. Thus, a frame with a long data field can be transmitted using several (shorter) data buffers chained together. This chaining technique allows the system designer to develop efficient buffer management.

The 82596 automatically generates the preamble (alternating 1s and 0s) 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 specified by the Transmit command, and computes and appends the CRC to the end of the frame (see Figure 12). In the Linear and 32-bit Segmented mode the CRC can be optionally inserted on a frame-by-frame basis by setting the NC bit in the Transmit Command Block (see Figures 29 and 30).

The 82596 can be configured to generate two types of start and end frame delimiters—End of Carrier (EOC) or HDLC. In EOC mode the start frame delimiter is 10101011 and the end frame delimiter is indicated by the lack of a signal after the last bit of the frame check sequence field has been transmitted. In EOC mode the 82596 can be configured to extend short frames by adding pad bytes (7Eh) during transmission, according to the length field. In HDLC mode the 82596 will generate the 01111110 flag for the start and end frame delimiters, and do standard bit stuffing and stripping. Furthermore, the 82596 can be configured to pad frames shorter than the specified minimum frame length by appending the appropriate number of flags to the end of the frame.

When a collision occurs, the 82596 manages the jam, random wait, and retry processes, 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 larger than the maximum frame size (1518 bytes for Ethernet).

![Figure 10. Action Command Format](image1)

![Figure 11. Data Buffer Descriptor and Data Buffer Structure](image2)
To reduce CPU overhead, the 82596 is designed to receive frames without CPU supervision. The host CPU first sets aside an adequate receive buffer space and then enables the 82596 Receive Unit. Once enabled, the RU watches for arriving frames and automatically stores them in the Receive Frame Area (RFA). The RFA contains Receive Frame Descriptors, Receive Buffer Descriptors, and Data Buffers (see Figure 13). The individual Receive Frame Descriptors make up a Receive Descriptor List (RDL) used by the 82596 to store the destination and source addresses, the length field, and the status of each frame received (see Figure 14).

Once enabled, the 82596 checks each passing frame for an address match. The 82596 will recognize its own unique address, one or more multicast addresses, or the broadcast address. If a match is found the 82596 stores the destination and source addresses and the length field in the next available RFD. It then begins filling the next available Data Buffer on the FBL, which is pointed to by the current RFD, with the data portion of the incoming frame. As one Data Buffer is filled, the 82596 automatically fetches the next DB on the FBL until the entire frame is received. This buffer chaining technique is particularly memory efficient because it allows the system designer to set aside buffers to fit frames much shorter than the maximum allowable frame length. If AL-LOC = 1, or if the flexible memory structure is used, the addresses and length field can be placed in the receive buffer.

Once the entire frame is received without error, the 82596 does the following housekeeping tasks.

- The actual count field of the last Buffer Descriptor used to hold the frame just received is updated with the number of bytes stored in the associated Data Buffer.
- The next available Receive Frame Descriptor is fetched.
- The address of the next available Buffer Descriptor is written to the next available Receive Frame Descriptor.
- A frame received interrupt status bit is posted in the SCB.
- An interrupt is sent to the CPU.

If a frame error occurs, for example a CRC error, the 82596 automatically reinitializes its DMA pointers and reclaims any data buffers containing the bad frame. The 82596 will continue to receive frames without CPU help as long as Receive Frame Descriptors and Data Buffers are available.

### 82596 NETWORK MANAGEMENT AND DIAGNOSTICS

The behavior of data communication networks is normally very complex because of their distributed and asynchronous nature. It is particularly difficult to pinpoint a failure when it occurs. The 82596 has extensive diagnostic and network management functions that help improve reliability and testability. The 82596 reports on the following events after each frame is transmitted.

- Transmission successful.
- Transmission unsuccessful. Lost Carrier Sense.
- Transmission unsuccessful. Lost Clear to Send.
- Transmission unsuccessful. A DMA underrun occurred because the system bus did not keep up with the transmission.
- Transmission unsuccessful. The number of collisions exceeded the maximum allowed.
- Number of Collisions. The number of collisions experienced during the frame.
- Heartbeat Indicator. This indicates the presence of a heartbeat during the last Interframe Spacing (IFS) after transmission.

When configured to Save Bad Frames the 82596 checks each incoming frame and reports the following errors.

- CRC error. Incorrect CRC in a properly aligned frame.
- Alignment error. Incorrect CRC in a misaligned frame.
- Frame too short. The frame is shorter than the value configured for minimum frame length.
- Overrun. Part of the frame was not placed in memory because the system bus did not keep up with incoming data.
- Out of buffer. Part of the frame was discarded because of insufficient memory storage space.
- Receive collision. A collision was detected during reception.
- Length error. A frame not matching the frame length parameter was detected.
Figure 13. Receive Frame Area Diagram

Figure 14. Receive Frame Descriptor
The 82596 updates its 32-bit statistical counters after each received frame that both passes address filtering and is longer than the Minimum Frame Length configuration parameter. The 82596 reports the following statistics:

- CRC errors. The number of well-aligned frames that experienced a CRC error.
- Alignment errors. The number of misaligned frames that experienced a CRC error.
- No resources. The number of frames that were discarded because of insufficient resources for reception.
- Overrun errors. The number of frames that were not completely stored in memory because the system bus did not keep up with incoming data.
- Receive Collision counter. The number of collisions detected during receive.
- Short Frame counter. The number of frames that were discarded because they were shorter than the configured minimum frame length.

The 82596 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 when implementing a monitoring station to capture all frames for analysis.

A useful method of capturing frame headers is to use the Simplified memory mode, configure the 82596 to Save Bad Frames, and configure the 82596 to Promiscuous mode with space in the RFD allocated for specific number of receive data bytes. The 82596 will receive all frames and put them in the RFD. Frames that exceed the available space in the RFD will be truncated, the status will be updated, and the 82596 will retrieve the next RFD. This allows the user to capture the initial data bytes of each frame (for instance, the header) and discard the remainder of the frame.

The 82596 also has a monitor mode for network analysis. During normal operation the receive function enables the 82596 to receive frames which pass address filtering. These frames must have the Start of Frame Delimiter (SFD) field and must be longer than the absolute minimum frame length of 5 bytes (6 bytes in case of Multicast address filtering). Contents and status of the received frames are transferred to memory. The monitor function enables the 82596 to simply evaluate the incoming frames. The 82596 can monitor the frames that pass or do not pass the address filtering. It can also monitor frames which do not have the SFD fields. The 82596 can be configured to only keep statistical information about monitor frames. Three options are available in the Monitor mode. These modes are selectable by the two monitor mode configuration bits available in the configuration command.

When the first option is selected, the 82596 receives good frames that pass address filtering and transfers them to memory while monitoring frames that do not pass address filtering or are shorter than the minimum frame size (these frames are not transferred to memory). When this option is used the 82596 updates six counters: CRC errors, alignment errors, no resource errors, overrun errors, short frames, and total good frames received.

When the second option is selected, the receive function is completely disabled. The 82596 monitors only those frames that pass address filterings and meet the minimum frame length requirement. When this option is used the 82596 updates six counters: CRC errors, alignment errors, total frames (good and bad), short frames, collisions detected, and total good frames.

When the third option is selected, the receive function is completely disabled. The 82596 monitors all frames, including frames that do not have a Start Frame Delimiter. When this option is used the 82596 updates six counter (CRC errors, alignment errors, total frames (good and bad), short frames, collisions detected, and total good frames.
STATION DIAGNOSTICS AND SELF-TEST

The 82596 provides a large set of diagnostic and network management functions. These include internal and external loopback and time domain reflectometry for locating fault points in the network cable. The 82596 ensures software reliability by dumping the contents of the 82596 internal registers into system memory. The 82596 has a self-test mode that enables it to run an internal self-test and place the results in system memory.

82586 SOFTWARE COMPATIBILITY

The 82596 has a software-compatible state in which all its memory structures are compatible with the 82586 memory structure. This includes all the Action Commands, the Receive Frame Area (including the RFD, Buffer Descriptors, and Data Buffers), the System Control Block, and the initialization procedures. There are two minor differences between the 82596 in the 82586-Compatible memory structure and the 82586.

- When the internal and external loopback bits in the Configure command are set to 11 the 82596 is in external loopback and the LPBK pin is activated; in the 82586 this situation would produce internal loopback.
- During a Dump command both the 82596 and 82586 dump the same number of bytes; however, the data format is different.

INITIALIZING THE 82596

A Reset command is issued to the 82596 to prepare it for normal operation. The 82596 is initialized through two data structures that are addressed by two pointers, the System Configuration Pointer (SCP) and the Intermediate System Configuration Pointer (ISCP). The initialization procedure begins when a Channel Attention signal is asserted after RESET. The 82596 uses the address of the double word that contains the SCP as a default—00FFFFFF4h. Before the CA signal is asserted this default address can be changed to any other available address by asserting the PORT pin and providing the desired address over the D31–D4 pins of the address bus. Pins D3–D0 must be 0010; i.e., any alternative address must be aligned to 16 byte boundaries. All addresses sent to the 82596 must be word aligned, which means that all pointers and memory structures must start on an even address (A0 = zero).

SYSTEM CONFIGURATION POINTER (SCP)

The SCP contains the SYSBUS byte and the location of the next structure of the initialization process, the ISCP. The following parameters are selected in the SYSBUS.

- The 82596 operation mode.
- The Bus Throttle timer triggering method.
- Lock enabled.
- Interrupt polarity.

Byte ordering is determined by the LE/BE pin. LE/BE = 1 selects little endian byte ordering and LE/BE = 0 selects big endian byte ordering.

NOTE:

In the following, X indicates a bit not checked in 82586 mode. This bit must be set to 0 in all other modes.
The following diagram illustrates the format of the SCP.

<table>
<thead>
<tr>
<th>31</th>
<th>ODD WORD</th>
<th>16</th>
<th>15</th>
<th>EVEN WORD</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>X</td>
<td>..........</td>
<td>X</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>X</td>
<td>..........</td>
<td>X</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>X</td>
<td>..........</td>
<td>X</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>X</td>
<td>..........</td>
<td>X</td>
<td></td>
<td>X</td>
<td></td>
</tr>
</tbody>
</table>

SYSBUS

<table>
<thead>
<tr>
<th>A31</th>
<th>A24</th>
<th>A23</th>
<th>ISCP ADDRESS</th>
<th>A0</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

A31 ............. A24 are not checked in 82586 mode.
X ............. X areas are not checked in 82586 mode; they must be 0 in all other modes.

SYSBUS

<table>
<thead>
<tr>
<th>23</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

- INT
- LOCK
- TRG
- M1
- MO
- X

Interrupt polarity
0 = Interrupt pin is active high
1 = Interrupt pin is active low

0 : NOT CHECKED

0 0 : 82586 mode
0 1 : 32-Bit Segmented mode
1 0 : Linear mode
1 1 : Reserved

0 : Internal triggering of the Bus Throttle timers
1 : external triggering of the Bus Throttle timers

0 : LOCK function enabled
1 : LOCK function disabled

ISCP ADDRESS— The physical address of the ISCP. In the 82586 mode, bits A31—A24 are considered to be zero.

Figure 15. The System Configuration Pointer

Writing the Sysbus

When writing the Sysbus byte it is important to pay attention to the byte order.

- When a Little Endian processor is used, the Sysbus byte is located at byte address 00FFFFFF6h (or address \( n + 2 \) if an alternative SCP address \( n \) was programmed).

- When a processor using Big Endian byte ordering is used, the SYSBUS, alternative SCP, and ISCP addresses will be different.
  - The Sysbus byte is located at 00FFFFFF5h.
  - If an alternative SCP address is programmed, the SYSBUS byte should be at byte address \( n + 1 \).
INTERMEDIATE SYSTEM CONFIGURATION POINTER (ISCP)

The ISCP indicates the location of the System Control Block. Often the SCP is in ROM and the ISCP is in RAM. The CPU loads the SCB address (or an equivalent data structure) into the ISCP and asserts CA. This Channel Attention signal causes the 82596 to begin its initialization procedure and to get the SCB address from the ISCP and SCP. In 82586 and 32-bit Segmented modes the SCP base address is also the base address of all Command Blocks, Frame Descriptors, and Buffer Descriptors (but not buffers). All these data structures must reside in one 64-kB segment; however, in Linear mode no such limitation is imposed.

The following diagram illustrates the ISCP format.

<table>
<thead>
<tr>
<th>ODD WORD</th>
<th>EVEN WORD</th>
</tr>
</thead>
<tbody>
<tr>
<td>A15</td>
<td>SCB OFFSET</td>
</tr>
<tr>
<td>A23</td>
<td>SCB BASE ADDRESS</td>
</tr>
</tbody>
</table>

BUSY — Indicates that the 82596 is being initialized. The CPU sets the ISCP to 01h before it gives the first CA to the 82596. The ISCP is cleared by the 82596 after the SCB base and offset are read. Note that the most significant byte of the first word of the ISCP is not modified when BUSY is cleared.

SCB OFFSET — This 16-bit quantity specifies the offset portion of the address of the SCB.

SCB BASE — Specifies the base portion of the address of the SCB. The base of SCB is also the base of all 82596 Command Blocks, Frame Descriptors and Buffer Descriptors. In the 82586 mode, bits A31–A24 are considered to be zero.

Figure 16. The Intermediate System Configuration Pointer—82586 and 32-Bit Segmented Modes

<table>
<thead>
<tr>
<th>ODD WORD</th>
<th>EVEN WORD</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 0 0 0 0 0 0 0 0 0 0</td>
<td>BUSY</td>
</tr>
</tbody>
</table>
| BUSY — Indicates that the 82596 is being initialized. The ISCP is set to 01h by the CPU before its first CA to the 82596. It is cleared by the 82596 after the SCB address is read.

SCB ADDRESS — This 32-bit quantity specifies the physical address of the SCB.

Figure 17. The Intermediate System Configuration Pointer—Linear Mode.

INITIALIZATION PROCESS

The CPU sets up the SCP, ISCP, and the SCB structures, and, if desired, an alternative SCP address. It also sets BUSY to 01h. The 82596 is initialized when a Channel Attention signal follows a Reset signal, causing the 82596 to access the System Configuration Pointer. The sysbus byte, the operational mode, the bus throttle timer triggering method, the interrupt polarity, and the state of LOCK are read. After reset the bus throttle
timers are essentially disabled—the T-ON value is infinite, the T-OFF value is zero. After the SCP is read, the 82596 reads the ISCP and saves the SCB address. In 82586 and 32-bit Segmented modes this address is represented as a base address plus the offset (this base address is also the base address of all the control blocks). In Linear mode the base address is also an absolute address. The 82596 clears BUSY, sets CX and CNR to equal 1 in the SCB, clears the SCB command word, sends an interrupt to the CPU, and awaits another Channel Attention signal. RESET configures the 82596 to its default state before CA is asserted.

CONTROLLING THE 82596DX/SX

The host CPU controls the 82596 with the commands, data structures, and methods described in this section. The CPU and the 82596 communicate through shared memory structures. The 82596 contains two independent units: the Command Unit and the Receive Unit. The Command Unit executes commands from the CPU, and the Receive Unit handles frame reception. These two units are controlled and monitored by the CPU through a shared memory structure called the System Control Block (SCB). The CPU and the 82596 use the CA and INT signals to communicate with the SCB.

82596 CPU ACCESS INTERFACE (PORT)

The 82596 has a CPU access interface that allows the host CPU to do four things.

- Write an alternative System Configuration Pointer address.
- Write an alternative Dump area pointer and perform Dump.
- Execute a software reset.
- Execute a self-test.

The following events initiate the CPU access state.

- Presence of an address on the D31–D4 data bus pins.
- The D3–D0 pins are used to select one of the four functions.
- The PORT input pin is asserted, as in a regular write cycle.

NOTE

The SCP Dump and Self-Test addresses must be 16-byte aligned.

The 82596 requires two 16-bit write cycles for a port command. The first write holds the internal machines and reads the first 16 bits, the second activates the PORT command and reads the second 16 bits.

The PORT Reset is useful when only the 82596 needs to be reset. The CPU must wait for 10-system and 5-serial clocks before issuing another CA to the 82596; this new CA begins a new initialization process.

The Dump function is useful for troubleshooting No Response problems. If the chip is in a No Response state, the PORT Dump operation can be executed and a PORT Reset can be used to reinitialize the 82596 without disturbing the rest of the system.

The Self-Test function can be used for board testing; the 82596 will execute a self-test and write the results to memory.

<table>
<thead>
<tr>
<th>Function</th>
<th>D31</th>
<th>D4</th>
<th>D3</th>
<th>D2</th>
<th>D1</th>
<th>D0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reset</td>
<td>A31</td>
<td>Don't Care</td>
<td>A4</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Self-Test</td>
<td>A31</td>
<td>Self-Test Results Address</td>
<td>A4</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>SCP</td>
<td>A31</td>
<td>Alternative SCP Address</td>
<td>A4</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Dump</td>
<td>A31</td>
<td>Dump Area Pointer</td>
<td>A4</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

Table 2. PORT Function Selection
MEMORY ADDRESSING FORMATS

The 82596 accesses memory by 32-bit addresses. There are two types of 32-bit addresses: linear and segmented. The type of address used depends on the 82596 operating mode and the type of memory structure it is addressing. The 82596 has three operating modes.

- **82586 Mode**
  - A Linear address is a single 24-bit entity. Address pins A31–A24 are always zero.
  - A Segmented address uses a 24-bit base and a 16-bit offset.

- **32-bit Segmented Mode**
  - A Linear address is a single 32-bit entity.
  - A Segmented address uses a 32-bit base and a 16-bit offset.

**NOTE**

In the previous two memory addressing modes, each command header (CB, TBD, RFD, RBD, and SCB) must wholly reside within one segment. If the 82596 encounters a memory structure that does not follow this restriction, the 82596 will fetch the next contiguous location in memory (beyond the segment).

- **Linear Mode**
  - A Linear address is a single 32-bit entity.
  - There are no Segmented addresses.

Linear addresses are primarily used to address transmit and receive data buffers. In the 82586 and 32-bit Segmented modes, segmented addresses (base plus offset) are used for all Command Blocks, Buffer Descriptors, Frame Descriptors, and System Control Blocks. When using Segmented addresses, only the offset portion of the entity being addressed is specified in the block. The base for all offsets is the same—that of the SCB.
LITTLE ENDIAN AND BIG ENDIAN BYTE ORDERING

The 82596 supports both Little Endian and Big Endian byte ordering for its memory structures.

The 82596 supports Big Endian byte ordering for word and byte entities. Dword entities are not supported with Big Endian byte ordering. This results in slightly different 82596 memory structures for Big Endian operation. These structures are defined in the 32-Bit LAN Components User's Manual.

NOTE

All 82596 memory entities must be word or dword aligned.

An example of a double word entity is a frame descriptor command/status dword, whereas the raw data of the frame are byte entities. Both 32- and 16-bit buses are supported. When a 16-bit bus is used with Big Endian memory organization, data lines \( D_{15}-D_0 \) are used. The 82596 has an internal crossover that handles these swap operations.

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 is associated with each Action Command. The CU is modeled as a logical machine that takes, at any given time, one of the following states.

* **Idle.** The CU is not executing a command and is not associated with a CB on the list. This is the initial state.

* **Suspended.** The CU is not executing a command; however, it is associated with a CB on the list. The suspend state can only be reached if the CPU forces it through the SCB or sets the suspend bit in the RFD.

* **Active.** The CU is executing an Action Command and pointing to its CB.

The CPU can affect CU operation in two ways: by issuing a CU Control Command or by setting bits in the Command word of the Action Command.

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.** The RU has no memory resources and is discarding incoming frames. This is the initial state.

* **No Resources.** The RU has no memory resources and is discarding incoming frames. This state differs from Idle in that the RU accumulates statistics on the number of discarded frames.

* **Suspended.** The RU has memory available for storing frames, but is discarding them. The suspend state can only be reached if the CPU forces it through the SCB or sets the suspend bit in the RFD.

* **Ready.** The RU has memory available and is storing incoming frames.

The CPU can affect RU operation in three ways: by issuing an RU Control Command, by setting bits in the Frame Descriptor Command word of the frame being received, or by setting the EL bit of the current buffer's Buffer Descriptor.

SYSTEM CONTROL BLOCK (SCB)

The SCB is a memory block that plays a major role in communications between the CPU and the 82596. Such communications include the following.

* **Commands issued by the CPU**

* **Status reported by the 82596**

Control commands are sent to the 82596 by writing them into the SCB and then asserting CA. The 82596 examines the command, performs the required action, and then clears the SCB command word. Control commands perform the following types of tasks.

* **Operation of the Command Unit (CU).** The SCB controls the CU by specifying the address of the Command Block List (CBL) and by starting, suspending, resuming, or aborting execution of CBL commands.
• Operation of the Bus Throttle. The SCB controls the Bus Throttle timers by providing them with new values and sending the Load and Start timer commands. The timers can be operated in both the 32-bit Segmented and Linear modes.
• Reception of frames by the Receive Unit (RU). The SCB controls the RU by specifying the address of the Receive Frame Area and by starting, suspending, resuming, or aborting frame reception.
• Acknowledgment of events that cause interrupts.
• Resetting the chip.

The 82596 sends status reports to the CPU via the System Control Block. The SCB contains four types of status reports:
• The cause of the current interrupts. These interrupts are caused by one or more of the following 82596 events.
  • The Command Unit completes an Action Command that has its / bit set.
  • The Receive Unit receives a frame.
  • The Command Unit becomes inactive.
  • The Receive Unit becomes not ready.
• The status of the Command Unit.
• The status of the Receive Unit.
• Status reports from the 82596 regarding reception of corrupted frames.

Events can be cleared only by CPU acknowledgment. If some events are not acknowledged by the ACK field the Interrupt signal (INT) will be reissued after Channel Attention (CA) is processed. Furthermore, if a new event occurs while an interrupt is set, the interrupt is temporarily cleared to trigger edge-triggered interrupt controllers.

The CPU uses the Channel Attention line to cause the 82596 to examine the SCB. This signal is trailing-edge triggered—the 82596 latches CA on the trailing edge. The latch is cleared by the 82596 before the SCB control command is read.

![Figure 18. SCB—82586 Mode](image1)

![Figure 19. SCB—32-Bit Segmented Mode](image2)

*In MONITOR mode these counters change function*
Inter

Figure 20. SCB—Linear Mode

These bits specify the action to be performed as a result of a CA. This word is set by the CPU and cleared by the 82596. Defined bits are:

Bit 31 ACK-CX — Acknowledges that the CU completed an Action Command.
Bit 30 ACK-FR — Acknowledges that the RU received a frame.
Bit 29 ACK-CNA — Acknowledges that the Command Unit became not active.
Bit 28 ACK-RNR — Acknowledges that the Receive Unit became not ready.
Bits 24–26 CUC — (3 bits) This field contains the command to the Command Unit. Valid values are:

0 — NOP (does not affect current state of the unit).
1 — Start execution of the first command on the CBL. If a command is executing, complete it before starting the new CBL. The beginning of the CBL is in CBL OFFSET (address).
2 — Resume the operation of the Command Unit by executing the next command. This operation assumes that the Command Unit has been previously suspended.
3 — Suspend execution of commands on CBL after current command is complete.
4 — Abort current command immediately.
5 — Loads the Bus Throttle timers so they will be initialized with their new values after the active timer (T-ON or T-OFF) reaches Terminal Count. If no timer is active new values will be loaded immediately. This command is not valid in 82586 mode.
6 — Loads and immediately restarts the Bus Throttle timers with their new values. This command is not valid in 82586 mode.
7 — Reserved.
Bits 20–22 RUC — (3 bits) This field contains the command to the Receive Unit. Valid values are:
0 — NOP (does not alter current state of unit).
1 — Start reception of frames. The beginning of the RFA is contained in the RFA OFFSET (address). If a frame is being received complete reception before starting.
2 — Resume frame reception (only when in suspended state).
3 — Suspend frame reception. If a frame is being received complete its reception before suspending.
4 — Abort receiver operation immediately.
5–7 — Reserved.

Bit 23 RESET — Reset chip (logically the same as hardware RESET).

### Status Word

<table>
<thead>
<tr>
<th>Bit 15</th>
<th>Bit 14</th>
<th>Bit 13</th>
<th>Bit 12</th>
<th>Bit 11–10</th>
<th>Bit 8–7</th>
<th>Bit 6–5</th>
<th>Bit 4–3</th>
<th>Bit 2</th>
<th>Bit 1</th>
<th>Bit 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>STAT</td>
<td>CUS</td>
<td>RUS</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>SCB</td>
</tr>
</tbody>
</table>

#### 82586 Mode

<table>
<thead>
<tr>
<th>Bit 15</th>
<th>Bit 14</th>
<th>Bit 13</th>
<th>Bit 12</th>
<th>Bit 11–10</th>
<th>Bit 8–7</th>
<th>Bit 6–5</th>
<th>Bit 4–3</th>
<th>Bit 2</th>
<th>Bit 1</th>
<th>Bit 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>STAT</td>
<td>CUS</td>
<td>RUS</td>
<td>T</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>SCB</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

#### 32-Bit Segmented and Linear Modes

Indicates the status of the 82596. This word is modified only by the 82596. Defined bits are:

- **Bit 15 CX** — The CU finished executing a command with its I (interrupt) bit set.
- **Bit 14 FR** — The RU finished receiving a frame.
- **Bit 13 CNA** — The Command Unit left the Active state.
- **Bit 12 RNR** — The Receive Unit left the Ready state.
- **Bits 8–10 CUS** — (3 bits) This field contains the status of the command unit. Valid values are:
  0 — Idle
  1 — Suspended
  2 — Active
  3–7 — Not used
- **Bits 4–7 RUS** — This field contains the status of the receive unit. Valid values are:
  0h (0000) — Idle
  1h (0001) — Suspended
  2h (0010) — No resources. This bit indicates both no resources due to lack of RFDs in the RDL and no resources due to lack of RBDs in the FBL.
  4h (0100) — Ready
  8h (1000) — No more RBDs (not in the 82586 mode).
  Ah (1010) — No resources due to no more RBDs. (Not in the 82586 mode.)
  No other combinations are allowed.
- **Bit 3 T** — Bus Throttle timers loaded (not in 82586 mode).

### SCB OFFSET ADDRESSES

#### CBL Offset (Address)

In 82586 and 32-bit Segmented modes this 16-bit quantity indicates the offset portion of the address for the first Command Block on the CBL. In Linear mode it is a 32-bit linear address for the first Command Block on the CBL. It is accessed only if CUC equals Start.
RFA Offset (Address)

In 82586 and 32-bit Segmented modes this 16-bit quantity indicates the offset portion of the address for the Receive Frame Area. In Linear mode it is a 32-bit linear address for the Receive Frame Area. It is accessed only if RUC equals Start.

SCB STATISTICAL COUNTERS

Statistical Counter Operation

- The CPU is responsible for clearing all error counters before initializing the 82596. The 82596 updates these counters by reading them, adding 1, and then writing them back to the SCB.
- The counters are wraparound counters. After reaching FFFFFFFFH the counters wrap around to zero.
- The 82596 updates the required counters for each frame. It is possible for more than one counter to be updated; multiple errors will result in all affected counters being updated.
- The 82596 executes the read-counter/increment/write-counter operation without relinquishing the bus (locked operation). This is to ensure that no logical contention exists between the 82596 and the CPU due to both attempting to write to the counters simultaneously. In the dual-port memory configuration the CPU should not execute any write operation to a counter if LOCK is asserted.
- The counters are 32-bits wide and their behavior is fully compatible with the IEEE 802.3 standard. The 82596 supports all relevant statistics (mandatory, optional, and desired) through the status of the transmit and receive header and directly through SCB statistics.

CRCERRS

This 32-bit quantity contains the number of aligned frames discarded because of a CRC error. This counter is updated, if needed, regardless of the RU state.

ALNERRS

This 32-bit quantity contains the number of frames that both are misaligned (i.e., where CRS deasserts on a nonoctet boundary) and contain a CRC error. The counter is updated, if needed, regardless of the RU state.

SHRTFRM

This 32-bit quantity contains the number of received frames shorter than the minimum frame length.

The last three counters change function in monitor mode.

RSCERRS

This 32-bit quantity contains the number of good frames discarded because there were no resources to contain them. Frames intended for a host whose RU is in the No Receive Resources state, fall into this category. This counter is updated only if the RU is in the No Resources state. When in Monitor mode, this counter counts the total number of frames.

OVRNERRS

This 32-bit quantity contains the number of frames known to be lost because the local system bus was not available. If the traffic problem lasts longer than the duration of one frame, the frames that follow the first are lost without an indicator, and they are not counted. This counter is updated, if needed, regardless of the RU state.

RCVCDT

This 32-bit quantity contains the number of collisions detected during frame reception. In Monitor mode this counter counts the total number of good frames.
ACTION COMMANDS AND OPERATING MODES

This section lists all the Action Commands of the Command Unit Command Block List (CBl). Each command contains the Command field, the Status and Control fields, the link to the next Action Command, and any command-specific parameters. There are three basic types of action commands: 82596 Configuration and Setup, Transmission, and Diagnostics. The following is a list of the actual commands.

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

The 82596 has three addressing modes. In the 82586 mode all the Action Commands look exactly like those of the 82586.

- **82586 Mode.** The 82596 software and memory structure is compatible with the 82586.
- **32-Bit Segmented Mode.** The 82596 can access the entire system memory and use the two new memory structures—Simplified and Flexible—while still using the segmented approach. This does not require any significant changes to existing software.
- **Linear Mode.** The 82596 operates in a flat, linear, 4 gigabyte memory space without segmentation. It can also use the two new memory structures.

In the 32-bit Segmented mode there are some differences between the 82596 and 82586 action commands, mainly in programming and activating new 82596 features. Those bits marked “don’t care” in the compatible mode are not checked; however, we strongly recommend that those bits all be zeroes; this will allow future enhancements and extensions.

In the Linear mode all of the address offsets become 32-bit address pointers. All new 82596 features are accessible in this mode, and all bits previously marked “don’t care” must be zeroes.

The Action Commands, and all other 82596 memory structures, must begin on even byte boundaries, i.e., they must be word aligned.

**NOP**

This command results in no action by the 82596 except for those performed in the normal command processing. It is used to manipulate the CBL manipulation. The format of the NOP command is shown in Figure 21.

![NOP—82586 and 32-Bit Segmented Modes](image)

![NOP—Linear Mode](image)

**Figure 21**

1-143
where:

**LINK POINTER** — In the 82586 or 32-bit Segmented modes this is a 16-bit offset to the next Command Block. In the Linear mode this is the 32-bit address of the next Command Block.

**EL** — If set, this bit indicates that this command block is the last on the CBL.

**S** — If set to one, suspend the CU upon completion of this CB.

**I** — If set to one, the 82596 will generate an interrupt after execution of the command is complete. If I is not set to one, the CX bit will not be set.

**CMD (bits 16–18)** — The NOP command. Value: 0h.

**Bits 19–28** — Reserved (zero in the 32-bit Segmented and Linear modes).

**C** — This bit indicates the execution status of the command. The CPU initially resets it to zero when the Command Block is placed on the CBL. Following a command Completion, the 82596 will set it to one.

**B** — This bit indicates that the 82596 is currently executing the NOP command. It is initially reset to zero by the CPU. The 82596 sets it to one when execution begins and to zero when execution is completed. This bit is also set when the 82596 prefetches the command.

**NOTE:**

The C and B bits are modified in one operation.

**OK** — Indicates that the command was executed without error. If set to one no error occurred (command executed OK). If zero an error occur.

**INDIVIDUAL ADDRESS SETUP**

This command is used to load the 82596 with the Individual Address. This address is used by the 82596 for inserting the Source Address during transmission and recognizing the Destination Address during reception. After RESET, and prior to Individual Address Setup Command execution, the 82596 assumes the Broadcast Address is the Individual Address in all aspects, i.e.:

- This will be the Individual Address Match reference.
- This will be the Source Address of a transmitted frame (for AL-LOC=0 mode only).

The format of the Individual Address Setup command is shown in Figure 22.

---

**Figure 22**

```
<table>
<thead>
<tr>
<th>IA Setup—82586 and 32-Bit Segmented Modes</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
</tr>
<tr>
<td>----</td>
</tr>
<tr>
<td>EL</td>
</tr>
<tr>
<td>INDIVIDUAL ADDRESS</td>
</tr>
<tr>
<td>6th byte</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>IA Setup—Linear Mode</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
</tr>
<tr>
<td>----</td>
</tr>
<tr>
<td>EL</td>
</tr>
<tr>
<td>A31</td>
</tr>
<tr>
<td>4th byte</td>
</tr>
<tr>
<td>6th byte</td>
</tr>
</tbody>
</table>
```

where:

**LINK ADDRESS,** **EL, B, C, I, S** — As per standard Command Block (see the NOP command for details)

**A** — Indicates that the command was abnormally terminated due to CU Abort control command. If one, then the command was aborted, and if necessary it should be repeated. If this bit is zero, the command was not aborted.
Bits 19–28 — Reserved (zero in the 32-bit Segmented and Linear modes).
CMD (bits 16–18) — The Address Setup command. Value: 1h.
INDIVIDUAL ADDRESS — The individual address of the node, 0 to 6 bytes long.

The least significant bit of the Individual Address must be zero for Ethernet (see the Command Structure). However, no enforcement of 0 is provided by the 82596. Thus, an Individual Address with 1 as its least significant bit is a valid Individual Address in all aspects.

The default address length is 6 bytes long, as in 802.3. If a different length is used the IA Setup command should be executed after the Configure command.

**CONFIGURE**

The Configure command loads the 82596 with its operating parameters. It allows changing some of the parameters by specifying a byte count less than the maximum number of configuration bytes (12 in the 82586 mode, 16 in the 32-Bit Segmented and Linear modes). The 82596 configuration depends on its mode of operation.

- In the 82586 mode the maximum number of configuration bytes is 12. Any number larger than 12 will be reduced to 12 and any number less than 4 will be increased to 4.
- The additional features of the serial side are disabled in the 82586 mode.
- In both the 32-Bit Segmented and Linear modes there are four additional configuration bytes, which hold parameters for additional 82596 features. If these parameters are not accessed, the 82596 will follow their default values.
- For more detailed information refer to the 82596 User's Manual.

The format of the Configure command is shown in Figures 23, 24, and 25.

<table>
<thead>
<tr>
<th>31</th>
<th>ODD WORD</th>
<th>16 15</th>
<th>EVEN WORD</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>EL</td>
<td>S I X X X X X X X X X X</td>
<td>0 1 0</td>
<td>C B OK A 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
<td></td>
</tr>
<tr>
<td>Byte 1</td>
<td>Byte 0</td>
<td>A15</td>
<td>LINK OFFSET</td>
<td>A0</td>
</tr>
<tr>
<td>Byte 5</td>
<td>Byte 4</td>
<td>Byte 3</td>
<td>Byte 2</td>
<td>8</td>
</tr>
<tr>
<td>Byte 9</td>
<td>Byte 8</td>
<td>Byte 7</td>
<td>Byte 6</td>
<td>1</td>
</tr>
<tr>
<td>X X X X X X X X X X X X X X X X X X X X X X</td>
<td>Byte 10</td>
<td>1</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

*Figure 23. CONFIGURE—82586 Mode*

<table>
<thead>
<tr>
<th>31</th>
<th>ODD WORD</th>
<th>16 15</th>
<th>EVEN WORD</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>EL</td>
<td>S I 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Byte 1</td>
<td>Byte 0</td>
<td>A15</td>
<td>LINK OFFSET</td>
<td>A0</td>
</tr>
<tr>
<td>Byte 5</td>
<td>Byte 4</td>
<td>Byte 3</td>
<td>Byte 2</td>
<td>8</td>
</tr>
<tr>
<td>Byte 9</td>
<td>Byte 8</td>
<td>Byte 7</td>
<td>Byte 6</td>
<td>12</td>
</tr>
<tr>
<td>Byte 13</td>
<td>Byte 12</td>
<td>Byte 11</td>
<td>Byte 10</td>
<td>16</td>
</tr>
</tbody>
</table>

*Figure 24. CONFIGURE—32-Bit Segmented Mode*
The interpretation of the fields follows:

<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>P</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**BYTE 0**

- **BYTE CNT (Bits 0-3)**: Byte Count. Number of bytes, including this one, that hold parameters to be configured.
- **PREFETCHED (Bit 7)**: Enable the 82596 to write the prefetched bit in all prefetch RBDs.

**NOTE:**
The P bit is valid only in the new memory structure modes. In 82586 mode this bit is disabled (i.e., no prefetched mark).

<table>
<thead>
<tr>
<th>7</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>MONITOR</td>
<td>X</td>
</tr>
</tbody>
</table>

**BYTE 1**

- **FIFO Limit (Bits 0-3)**: FIFO limit.
- **MONITOR # (Bits 6-7)**: Receive monitor options. If the Byte Count of the configure command is less than 12 bytes then these Monitor bits are ignored.

**DEFAULT: C8h**

<table>
<thead>
<tr>
<th>7</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>SAV BF</td>
<td>1</td>
</tr>
</tbody>
</table>

**BYTE 2**

- **SAV BF (Bit 7)**: 0—Received bad frames are not saved in the memory.
  1—Received bad frames are saved in the memory.

**DEFAULT: 40h**
### BYTE 3

<table>
<thead>
<tr>
<th>LOOP BACK MODE</th>
<th>PREAMBLE LENGTH</th>
<th>NO SRC ADD INS</th>
<th>ADDRESS LENGTH</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

ADR LEN (Bits 0–2) Address length (any kind).

NO SCR ADD INS (Bit 3) No Source Address Insertion.

In the 82586 this bit is called AL LOC.

PREAM LEN (Bits 4–5) Preamble length.

LP BCK MODE (Bits 6–7) Loopback mode.

DEFAULT: 26h

### BYTE 4

<table>
<thead>
<tr>
<th>BOF METD</th>
<th>EXPONENTIAL PRIORITY</th>
<th>0</th>
<th>LINEAR PRIORITY</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

LIN PRIO (Bits 0–2) Linear Priority.

EXP PRIO (Bits 4–6) Exponential Priority.

BOF METD (Bit 7) Exponential Backoff method.

DEFAULT: 00h

### BYTE 5

<table>
<thead>
<tr>
<th>INTERFRAME SPACING</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
</tr>
</tbody>
</table>

INTERFRAME SPACING Interframe spacing.

DEFAULT: 60h

### BYTE 6

<table>
<thead>
<tr>
<th>SLOT TIME - LOW</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
</tr>
</tbody>
</table>

SLOT TIME (L) Slot time, low byte.

DEFAULT: 00h

### BYTE 7

<table>
<thead>
<tr>
<th>MAXIMUM RETRY NUMBER</th>
<th>0</th>
<th>SLOT TIME - HIGH</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

SLOT TIME (H) Slot time, high part.

RETRY NUM (Bits 4–7) Number of transmission retries on collision.

DEFAULT: F2h
<table>
<thead>
<tr>
<th>BYTE 8</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>PAD</td>
<td>BIT STUFF</td>
<td>CRC16/ CRC32</td>
<td>NO CRC</td>
<td>Tx ON</td>
<td>MAN/ NRZ</td>
<td>BC DIS</td>
</tr>
<tr>
<td>PRM (Bit 0)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BC DIS (Bit 1)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MANCH/NRZ (Bit 2)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TONO CRS (Bit 3)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NOCRC INS (Bit 4)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CRC-16/CRC-32 (Bit 5)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BIT STF (Bit 6)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PAD (Bit 7)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DEFAULT: 00h</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>BYTE 9</th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>CDT SRC</td>
<td>COLLISION DETECT FILTER</td>
<td>CRS SRC</td>
<td>CARRIER SENSE FILTER</td>
</tr>
<tr>
<td>CRSF (Bits 0–2)</td>
<td></td>
<td></td>
<td></td>
<td>Carrier Sense filter (length).</td>
</tr>
<tr>
<td>CRS SRC (Bit 3)</td>
<td></td>
<td></td>
<td></td>
<td>Carrier Sense source.</td>
</tr>
<tr>
<td>CDTF (Bits 4–6)</td>
<td></td>
<td></td>
<td></td>
<td>Collision Detect filter (length).</td>
</tr>
<tr>
<td>CDT SRC (Bit 7)</td>
<td></td>
<td></td>
<td></td>
<td>Collision Detect source.</td>
</tr>
<tr>
<td>DEFAULT: 00h</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>BYTE 10</th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>MINIMUM FRAME LENGTH</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MIN FRAME LEN</td>
<td></td>
<td></td>
<td>Minimum frame length.</td>
<td></td>
</tr>
<tr>
<td>DEFAULT: 40h</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>BYTE 11</th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>MONITOR</td>
<td>MC_ALL</td>
<td>CDBSAC</td>
<td>AUTOTX</td>
</tr>
<tr>
<td>PRECRS (Bit 0)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LNGFLD (Bit 1)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CRCINM (Bit 2)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>AUTOTX (Bit 3)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CDBSAC (Bit 4)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MC_ALL (Bit 5)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MONITOR (Bits 6–7)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DEFAULT: FFH</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>DCR</td>
<td>FDX</td>
<td>DCR SLOT ADDRESS</td>
<td></td>
</tr>
<tr>
<td>---</td>
<td>-----</td>
<td>-----</td>
<td>------------------</td>
<td></td>
</tr>
</tbody>
</table>

**BYTE 12**

<table>
<thead>
<tr>
<th></th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DCR SLOT ADDRESS</td>
<td>Station index in DCR mode.</td>
</tr>
<tr>
<td>(Bits 0–5)</td>
<td></td>
</tr>
<tr>
<td>FDX (Bit 6)</td>
<td>Enables Full Duplex operation.</td>
</tr>
<tr>
<td>DCR (Bit 7)</td>
<td>Enables Deterministic collision resolution.</td>
</tr>
<tr>
<td>DEFAULT: 00h</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>7</th>
<th>DIS__BOF</th>
<th>MULT__IA</th>
<th>DCR NUMBER OF STATIONS</th>
</tr>
</thead>
</table>

**BYTE 13**

<table>
<thead>
<tr>
<th></th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DCR NUMBER OF STATIONS</td>
<td>Number of stations in DCR mode.</td>
</tr>
<tr>
<td>(Bits 0–5)</td>
<td></td>
</tr>
<tr>
<td>MULT__IA (Bit 6)</td>
<td>Multiple individual address.</td>
</tr>
<tr>
<td>DIS__BOF (Bit 7)</td>
<td>Disable the backoff algorithm.</td>
</tr>
<tr>
<td>DEFAULT: 3Fh</td>
<td></td>
</tr>
</tbody>
</table>
A reset (hardware or software) configures the 82596 according to the following defaults.

**Table 4. Configuration Defaults**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Default Value</th>
<th>Units/meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADDRESS LENGTH</td>
<td><strong>6</strong> Bytes</td>
<td></td>
</tr>
<tr>
<td>A/L FIELD LOCATION</td>
<td>0</td>
<td>Located in FD</td>
</tr>
<tr>
<td>* AUTO RETRANSMIT</td>
<td>1</td>
<td>Auto Retransmit Enable</td>
</tr>
<tr>
<td>BITSTUFFING/EOC</td>
<td>0</td>
<td>EOC</td>
</tr>
<tr>
<td>BROADCAST DISABLE</td>
<td>0</td>
<td>Broadcast Reception Enabled</td>
</tr>
<tr>
<td>* CDBSAC</td>
<td>1</td>
<td>Disabled</td>
</tr>
<tr>
<td>CDT FILTER</td>
<td>0</td>
<td>Bit Times</td>
</tr>
<tr>
<td>CDT SRC</td>
<td>0</td>
<td>External Collision Detection</td>
</tr>
<tr>
<td>* CRC IN MEMORY</td>
<td>1</td>
<td>CRC Not Transferred to Memory</td>
</tr>
<tr>
<td>CRC-16/CRC-32</td>
<td><strong>0</strong> CRC-32</td>
<td></td>
</tr>
<tr>
<td>CRS FILTER</td>
<td>0</td>
<td>0 Bit Times</td>
</tr>
<tr>
<td>CRS SRC</td>
<td>0</td>
<td>External CRS</td>
</tr>
<tr>
<td>* DCR</td>
<td>0</td>
<td>Disable DCR Protocol</td>
</tr>
<tr>
<td>DCR Slot Number</td>
<td>0</td>
<td>DCR Disabled</td>
</tr>
<tr>
<td>* DCR Number of Stations</td>
<td>63</td>
<td>Stations</td>
</tr>
<tr>
<td>* DISBOF</td>
<td>0</td>
<td>Backoff Enabled</td>
</tr>
<tr>
<td>EXT LOOPBACK</td>
<td>0</td>
<td>Disabled</td>
</tr>
<tr>
<td>EXPONENTIAL PRIORITY</td>
<td><strong>0</strong> 802.3 Algorithm</td>
<td></td>
</tr>
<tr>
<td>EXPONENTIAL BACKOFF METHOD</td>
<td><strong>0</strong> 802.3 Algorithm</td>
<td></td>
</tr>
<tr>
<td>* FULL DUPLEX (FDX)</td>
<td>0</td>
<td>CSMA/CD Protocol (No FDX)</td>
</tr>
<tr>
<td>FIFO THRESHOLD</td>
<td>8</td>
<td>TX: 32 Bytes, RX: 64 Bytes</td>
</tr>
<tr>
<td>INT LOOPBACK</td>
<td>0</td>
<td>Disabled</td>
</tr>
<tr>
<td>INTERFRAME SPACING</td>
<td><strong>96</strong> Bit Times</td>
<td></td>
</tr>
<tr>
<td>LINEAR PRIORITY</td>
<td><strong>0</strong> 802.3 Algorithm</td>
<td></td>
</tr>
<tr>
<td>* LENGTH FIELD</td>
<td>1</td>
<td>Padding Disabled</td>
</tr>
<tr>
<td>MIN FRAME LENGTH</td>
<td><strong>64</strong> Bytes</td>
<td></td>
</tr>
<tr>
<td>* MC ALL</td>
<td>1</td>
<td>Disabled</td>
</tr>
<tr>
<td>* MONITOR</td>
<td>11</td>
<td>Disabled</td>
</tr>
<tr>
<td>MANCHESTER/NRZ</td>
<td>0</td>
<td>NRZ</td>
</tr>
<tr>
<td>* MULTI IA</td>
<td>0</td>
<td>Disabled</td>
</tr>
<tr>
<td>NUMBER OF RETRIES</td>
<td><strong>15</strong> Maximum Number of Retries</td>
<td></td>
</tr>
<tr>
<td>NO CRC INSERTION</td>
<td>0</td>
<td>CRC Appended to Frame</td>
</tr>
<tr>
<td>PREFETCH BIT IN RBD</td>
<td>0</td>
<td>Disabled (Valid Only in New Modes)</td>
</tr>
<tr>
<td>PREAMBLE LENGTH</td>
<td><strong>7</strong> Bytes</td>
<td></td>
</tr>
<tr>
<td>* Preamble Until CRS</td>
<td>1</td>
<td>Disabled</td>
</tr>
<tr>
<td>PROMISCUOUS MODE</td>
<td>0</td>
<td>Address Filter On</td>
</tr>
<tr>
<td>PADDING</td>
<td>0</td>
<td>No Padding</td>
</tr>
<tr>
<td>SLOT TIME</td>
<td><strong>512</strong> Bit Times</td>
<td></td>
</tr>
<tr>
<td>SAVE BAD FRAME</td>
<td>0</td>
<td>Discards Bad Frames</td>
</tr>
<tr>
<td>TRANSMIT ON NO CRS</td>
<td>0</td>
<td>Disabled</td>
</tr>
</tbody>
</table>

**NOTES:**

1. This configuration setup is compatible with the IEEE 802.3 specification.
2. The Asterisk "**" signifies a new configuration parameter not available in the 82586.
3. The default value of the Auto retransmit configuration parameter is enabled (1).
4. Double Asterisk "***" signifies IEEE 802.3 requirements.
MULTICAST-SETUP

This command is used to load the 82596 with the Multicast-IDs that should be accepted. As noted previously, the filtering done on the Multicast-IDs is not perfect and some unwanted frames may be accepted. This command resets the current filter and reloads it with the specified Multicast-IDs. The format of the Multicast-addresses setup command is:

```
31      ODD WORD     16 15     EVEN WORD    0
EL S I X X X X X X X X 0 1 1 C B OK A 0 0 0 0 0 0 0 0 0 0 0 0
X X      MC COUNT    A15      LINK OFFSET A0
4th byte
Nth byte
```

Figure 26. MC Setup—82586 and 32-Bit Segmented Modes

```
31      ODD WORD     16 15     EVEN WORD    0
EL S I 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
A31      LINK ADDRESS A0
2nd byte
1st byte X X      MC COUNT
Nth byte
```

Figure 27. MC Setup—Linear Mode

where:

- **LINK ADDRESS**, **EL, B, C, I, S**
  - As per standard Command Block (see the NOP command for details)
- **A**
  - Indicates that the command was abnormally terminated due to a CU Abort control command. If one, then the command was aborted and if necessary it should be repeated. If this bit is zero, the command was not aborted.
- **Bits 19–28**
  - Reserved (0 in both the 32-Bit Segmented and Linear Modes).
- **CMD (bits 16–18)**
  - The MC SETUP command value: 3h.
- **MC-CNT**
  - This 14-bit field indicates the number of bytes in the MC LIST field. The MC CNT must be a multiple of the ADDR LEN; otherwise, the 82596 reduces the MC CNT to the nearest ADDR LEN multiple. MC CNT = 0 implies resetting the Hash table which is equivalent to disabling the Multicast filtering mechanism.
- **MC LIST**
  - A list of Multicast Addresses to be accepted by the 82596. The least significant bit of each MC address must be 1.

**NOTE:**
The list is sequential; i.e., the most significant byte of an address is immediately followed by the least significant byte of the next address.

- When the 82596 is configured to recognize multiple Individual Address (Multi-IA), the MC-Setup command is also used to set up the Hash table for the individual address.

The least significant bit in the first byte of each IA address must be 0.
TRANSMIT

This command is used to transmit a frame of user data onto the serial link. The format of a Transmit command is as follows.

<table>
<thead>
<tr>
<th>31</th>
<th>ODD WORD</th>
<th>16 15</th>
<th>EVEN WORD</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>EL</td>
<td>S</td>
<td>I</td>
<td>0 0 0 0</td>
<td>NC</td>
</tr>
<tr>
<td>SF</td>
<td>1 0 0 0</td>
<td>C</td>
<td>B</td>
<td>STATUS BITS</td>
</tr>
<tr>
<td>A15</td>
<td>TBD OFFSET</td>
<td>A0</td>
<td>A15</td>
<td>LINK OFFSET</td>
</tr>
<tr>
<td>4th byte</td>
<td>DESTINATION ADDRESS</td>
<td>1st byte</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>LENGTH FIELD</td>
<td>6th byte</td>
<td>12</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure 28. TRANSMIT—82586 Mode

<table>
<thead>
<tr>
<th>31</th>
<th>ODD WORD</th>
<th>16 15</th>
<th>EVEN WORD</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>EL</td>
<td>S</td>
<td>I</td>
<td>0 0 0 0</td>
<td>NC</td>
</tr>
<tr>
<td>SF</td>
<td>1 0 0 0</td>
<td>C</td>
<td>B</td>
<td>STATUS BITS</td>
</tr>
<tr>
<td>A15</td>
<td>TBD OFFSET</td>
<td>A0</td>
<td>A15</td>
<td>LINK OFFSET</td>
</tr>
<tr>
<td>4th byte</td>
<td>DESTINATION ADDRESS</td>
<td>1st byte</td>
<td>12</td>
<td></td>
</tr>
<tr>
<td>LENGTH FIELD</td>
<td>6th byte</td>
<td>16</td>
<td></td>
<td></td>
</tr>
<tr>
<td>OPTIONAL DATA</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure 29. TRANSMIT—32-Bit Segmented Mode

<table>
<thead>
<tr>
<th>31</th>
<th>ODD WORD</th>
<th>16 15</th>
<th>EVEN WORD</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>EL</td>
<td>S</td>
<td>I</td>
<td>0 0 0 0</td>
<td>NC</td>
</tr>
<tr>
<td>SF</td>
<td>1 0 0 0</td>
<td>C</td>
<td>B</td>
<td>STATUS BITS</td>
</tr>
<tr>
<td>A31</td>
<td>LINK ADDRESS</td>
<td>A0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>A31</td>
<td>TRANSMIT BUFFER DESCRIPTOR ADDRESS</td>
<td>A0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4th byte</td>
<td>DESTINATION ADDRESS</td>
<td>1st byte</td>
<td>16</td>
<td></td>
</tr>
<tr>
<td>LENGTH FIELD</td>
<td>6th byte</td>
<td>20</td>
<td></td>
<td></td>
</tr>
<tr>
<td>OPTIONAL DATA</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure 30. TRANSMIT—Linear Mode

0: No CRC Insertion disable; when the configure command is configured to not insert the CRC during transmission the NC has no effect.
1: No CRC Insertion enable; when the configure command is configured to insert the CRC during transmission the CRC will not be inserted when NC = 1.

0: Simplified Mode, all the Tx data is in the Transmit Command Block. The Transmit Buffer Descriptor Address field is all 1s.
1: Flexible Mode. Data is in the TCB and in a linked list of TBDs.
where:

EL, B, C, I, S — As per standard Command Block (see the NOP command for details).
OK (Bit 13) — Error free completion.
A (Bit 12) — Indicates that the command was abnormally terminated due to CU Abort control command. If 1, then the command was aborted, and if necessary it should be repeated. If this bit is 0, the command was not aborted.

Bits 19-28 — Reserved (0 in the 32-bit Segmented and Linear modes).
CMD (Bits 16-18) — The transmit command: 4h.

Status Bit 11 — Late collision. A late collision (a collision after the slot time is elapsed) is detected.
Status Bit 10 — No Carrier Sense signal during transmission. Carrier Sense signal is monitored from the end of Preamble transmission until the end of the Frame Check Sequence for TONOCRS = 1 (Transmit On No Carrier Sense mode) it indicates that transmission has been executed despite a lack of CRS. For TONOCRS = 0 (Ethernet mode), this bit also indicates unsuccessful transmission (transmission stopped when lack of Carrier Sense has been detected).

Status Bit 9 — Transmission unsuccessful (stopped) due to Loss of CTS.
Status Bit 8 — Transmission unsuccessful (stopped) due to DMA Underrun; i.e., the system did not supply data for transmission.
Status Bit 7 — Transmission Deferred, i.e., transmission was not immediate due to previous link activity.
Status Bit 6 — Heartbeat Indicator, Indicates that after a previously performed transmission, and before the most recently performed transmission, (Interframe Spacing) the CDT signal was monitored as active. This indicates that the Ethernet Transceiver Collision Detect logic is performing properly. The Heartbeat is monitored during the Interframe Spacing period.

Status Bit 5 — Transmission attempt was stopped because the number of collisions exceeded the maximum allowable number of retries.

MAX-COL (Bits 3–0) — The number of Collisions experienced during this frame. Max Col = 0 plus S5 = 1 indicates 16 collisions.

LINK OFFSET — As per standard Command Block (see the NOP for details).

TBD POINTER — In the 82586 and 32-bit Segmented modes this is the offset of the first Tx Buffer Descriptor containing the data to be transmitted. In the Linear mode this is the 32-bit address of the first Tx Buffer Descriptor on the list. If the TBD POINTER is all 1s it indicates that no TBD is used.

DEST ADDRESS — Contains the Destination Address of the frame. The least significant bit (MC) indicates the address type.
MC = 0: Individual Address.
MC = 1: Multicast or Broadcast Address.

If the Destination Address bits are all 1s this is a Broadcast Address.

LENGTH FIELD — The contents of this 2-byte field are user defined. In 802.3 it contains the length of the data field. It is placed in memory in the same order it is transmitted; i.e., most significant byte first, least significant byte second.

TCB COUNT — This 14-bit counter indicates the number of bytes that will be transmitted from the Transmit Command Block, starting from the third byte after the TCB COUNT field (address n + 12 in the 32-bit Segmented mode, N + 16 in the Linear mode). The TCB COUNT field can be any number of bytes (including an odd byte), this allows the user to transmit a frame with a header having an odd number of bytes. The TCB COUNT field is not used in the 82586 mode.

EOF Bit — Indicates that the whole frame is kept in the Transmit Command Block. In the Simplified memory model it must be always asserted.
The interpretation of what is transmitted depends on the No Source Address insertion configuration bit and the memory model being used.

NOTES
1. The Destination Address and the Length Field are sequential of the Length Field immediately follows the most significant byte of the Destination Address.
2. In case the 82596 is configured with No Source Address insertion bit equal to 0, the 82596 inserts its configured Source Address in the transmitted frame.
   - In the 82586 mode, or when the Simplified memory model is used, the Destination and Length fields of the transmitted frame are taken from the Transmit Command Block.
   - If the FLEXIBLE memory model is used, the Destination and Length fields of the transmitted frame can be found either in the TCB or TBD, depending on the TCB COUNT.
3. If the 82596 is configured with the Address/Length Field Location equal to 1, the 82596 does not insert its configured Source Address in the transmitted frame. The first \((2 \times \text{Address Length}) + 2\) bytes of the transmitted frame are interpreted as Destination Address, Source Address, and Length fields respectively.
   - In the 82586 mode, it is always the first byte of the first Tx Buffer.
   - In both the 32-bit Segmented and Linear modes it depends on the SF bit and TCB COUNT:
     - In the Simplified memory mode the first transmitted byte is always the third byte after the TCB COUNT field.
     - In the Flexible mode, if the TCB COUNT is greater than 0 then it is the third byte after the TCB COUNT field. If TCB COUNT equals 0 then it is first byte of the first Tx Buffer.
   - Transmit frames shorter than six bytes are invalid. The transmission will be aborted (only in 82586 mode) because of a DMA Underrun.
4. Frames which are aborted during transmission are jammed. Such an interruption of transmission can be caused by any reason indicated by any of the status bits 8, 9, 10 and 12.

JAMMING RULES
1. Jamming will not start before completion of preamble transmission.
2. Collisions detected during transmission of the last 11 bits will not result in jamming.

The format of a Transmit Buffer Descriptor is:

<table>
<thead>
<tr>
<th>82586 Mode</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
</tr>
<tr>
<td>NEXT TBD OFFSET</td>
</tr>
<tr>
<td>X X X X X X X X</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>32-Bit Segmented Mode</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
</tr>
<tr>
<td>NEXT TBD OFFSET</td>
</tr>
<tr>
<td>TRANSmit BUFFER ADDRESS</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Linear Mode</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
</tr>
<tr>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 EOF</td>
</tr>
<tr>
<td>NEXT TBD ADDRESS</td>
</tr>
<tr>
<td>TRANSmit BUFFER ADDRESS</td>
</tr>
</tbody>
</table>

Figure 31
where:

**EOF** — This bit indicates that this TBD is the last one associated with the frame being transmitted. It is set by the CPU before transmit.

**SIZE (ACT COUNT)** — This 14-bit quantity specifies the number of bytes that hold information for the current buffer. It is set by the CPU before transmission.

**NEXT TBD ADDRESS** — In the 82586 and 32-bit Segmented modes, it is the offset of the next TBD on the list. In the Linear mode this is the 32-bit address of the next TBD on the list. It is meaningless if EOF = 1.

**BUFFER ADDRESS** — The starting address of the memory area that contains the data to be sent. In the 82586 mode, this is a 24-bit address (A31–A24 are considered to be zero). In the 32-bit Segmented and Linear modes this is a 32-bit address.

**TDR**

This operation activates Time Domain Reflectometry, which is a mechanism to detect open or short circuits on the link and their distance from the diagnosing station. The TDR command has no parameters. The TDR transmit sequence was changed, compared to the 82586, to form a regular transmission. The TDR bit stream is as follows.

- Preamble
- Source address
- Another Source address (the TDR frame is transmitted back to the sending station, so DEST ADR = SRC ADR).
- Data field containing 7Eh patterns.
- Jam Pattern, which is the inverse CRC of the transmitted frame.

Maximum length of the TDR frame is 2048 bits. If the 82596 senses collision while transmitting the TDR frame it transmits the jam pattern and stops the transmission. The 82596 then triggers an internal timer (STC); the timer is reset at the beginning of transmission and reset if CRS is returned. The timer measures the time elapsed from the start of transmission until an echo is returned. The echo is indicated by Collision Detect going active or a drop in the Carrier Sense signal. The following table lists the possible cases that the 82596 is able to analyze.

<table>
<thead>
<tr>
<th>Conditions of TDR as Interpreted by the 82596</th>
</tr>
</thead>
<tbody>
<tr>
<td>Condition</td>
</tr>
<tr>
<td>Carrier Sense was inactive for 2048-bit-time</td>
</tr>
<tr>
<td>periods</td>
</tr>
<tr>
<td>Carrier Sense signal dropped</td>
</tr>
<tr>
<td>Collision Detect went active</td>
</tr>
<tr>
<td>The Carrier Sense Signal did not drop or the</td>
</tr>
<tr>
<td>Collision Detect did not go active within 2048-bit time period</td>
</tr>
</tbody>
</table>

An Ethernet transceiver is defined as one that returns transmitted data on the receive pair and activates the Carrier Sense Signal while transmitting. A Non-Ethernet Transceiver is defined as one that does not do so.
The format of the Time Domain Reflectometer command is:

<table>
<thead>
<tr>
<th>31</th>
<th>ODD WORD</th>
<th>16 15</th>
<th>EVEN WORD</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>EL</td>
<td>S</td>
<td>I</td>
<td>X X X X X X X X X 1 0 1</td>
<td>C</td>
</tr>
<tr>
<td>LNK</td>
<td>XVR</td>
<td>ET</td>
<td>ET</td>
<td>X</td>
</tr>
<tr>
<td>OK</td>
<td>PRB</td>
<td>OPN</td>
<td>SRT</td>
<td></td>
</tr>
</tbody>
</table>

**Linear Mode**

<table>
<thead>
<tr>
<th>31</th>
<th>ODD WORD</th>
<th>16 15</th>
<th>EVEN WORD</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>EL</td>
<td>S</td>
<td>I 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
<td>1 0 1</td>
<td>C</td>
</tr>
<tr>
<td>A31</td>
<td></td>
<td></td>
<td>LINK ADDRESS</td>
<td>A0</td>
</tr>
<tr>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
<td>LNK</td>
<td>XVR</td>
<td>ET</td>
<td>ET</td>
</tr>
</tbody>
</table>

where:

- **LINK ADDRESS**, **EL, B, C, I, S**
  - As per standard Command Block (see the NOP command for details).
- **A**
  - Indicates that the command was abnormally terminated due to CU Abort control command. If one, then the command was aborted, and if necessary it should be repeated. If this bit is zero, the command was not aborted.
- **Bits 19–28**
  - Reserved (0 in the 32-bit Segmented and Linear Modes).
- **CMD (Bits 16–18)**
  - The TDR command. Value: 5h.
- **TIME**
  - An 11-bit field that specifies the number of TxC cycles that elapsed before an echo was observed. No echo is indicated by a reception consisting of “1s” only. Because the network contains various elements such as transceiver links, transceivers, Ethernet, repeaters etc., the TIME is not exactly proportional to the problems distance.
- **LNK OK (Bit 15)**
  - No link problem identified. TIME = 7FFh.
- **XCVR PRB (Bit 14)**
  - Indicates a Transceiver problem. Carrier Sense was inactive for 2048-bit time period. LNK OK = 0. TIME = 7FFh.
- **ET OPN (Bit 13)**
  - The transmission line is not properly terminated. Collision Detect went active and LNK OK = 0.
- **ET SRT (Bit 12)**
  - There is a short circuit on the transmission line. Carrier Sense Signal dropped and LNK OK = 0.
DUMP

This command causes the contents of various 82596 registers to be placed in a memory area specified by the user. It is supplied as a 82596 self-diagnostic tool, and to provide registers of interest to the user. The format of the DUMP command is:

<table>
<thead>
<tr>
<th>ODD WORD</th>
<th>EVEN WORD</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>EL S I X X X X X X X X</td>
<td>11 0 C B OK</td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
</tr>
<tr>
<td>A15 BUFFER OFFSET</td>
<td>A0 A15 LINK OFFSET</td>
<td>A0</td>
</tr>
</tbody>
</table>

82586 and 32-Bit Segmented Modes

<table>
<thead>
<tr>
<th>ODD WORD</th>
<th>EVEN WORD</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>EL S I X X X X X X X X</td>
<td>11 0 C B OK</td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
</tr>
<tr>
<td>A31 LINK ADDRESS</td>
<td>A0</td>
<td></td>
</tr>
<tr>
<td>A31 BUFFER ADDRESS</td>
<td>A0</td>
<td></td>
</tr>
</tbody>
</table>

Linear Mode

where:

- LINK ADDRESS, EL, B, C, I, S — As per standard Command Block (see the NOP command for details).
- OK — Indicates error free completion.
- Bits 19–28 — Reserved (0 in the 32-bit Segmented and Linear Modes).
- CMD (Bits 16–18) — The Dump command. Value: 6h.
- BUFFER POINTER — In the 82586 and 32-bit Segmented modes this is the 16-bit-offset portion of the dump area address. In the Linear mode this is the 32-bit linear address of the dump area.

Dump Area Information Format

- The 82596 is not Dump compatible with the 82586 because of the 32-bit internal architecture. In 82586 mode the 82596 will dump the same number of bytes as the 82586. The compatible data will be marked with an asterisk.
- In 82586 mode the dump area is 170 bytes.
- The dump area format of the 32-bit Segmented and Linear modes is described in Figure 35.
- The size of the dump area of the 32-bit Segmented and Linear modes is 304 bytes.
- When the dump is executed by the Port command an extra word will be appended to the Dump Area. The extra word is a copy of the Dump Area status word (containing the C, B, and OK bits). The C and OK bits are set when the 82596 has completed the Port Dump command.
<table>
<thead>
<tr>
<th>Byte 15</th>
<th>Byte 14</th>
<th>Byte 13</th>
<th>Byte 12</th>
<th>Byte 11</th>
<th>Byte 10</th>
<th>Byte 9</th>
<th>Byte 8</th>
<th>Byte 7</th>
<th>Byte 6</th>
<th>Byte 5</th>
<th>Byte 4</th>
<th>Byte 3</th>
<th>Byte 2</th>
<th>Byte 1</th>
<th>Byte 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>DMA CONTROL REGISTER*</td>
<td>00</td>
<td>CONFIGURE BYTES 3, 2</td>
<td>02</td>
<td>CONFIGURE BYTES 5, 4</td>
<td>04</td>
<td>CONFIGURE BYTES 7, 6</td>
<td>06</td>
<td>CONFIGURE BYTES 9, 8</td>
<td>08</td>
<td>CONFIGURE BYTES 10</td>
<td>0A</td>
<td>I.A. BYTES 1, 0*</td>
<td>0C</td>
<td>I.A. BYTES 3, 2*</td>
<td>0E</td>
</tr>
</tbody>
</table>

**NOTE:**

*The 82596 is not Dump compatible with the 82586 because of the 32-bit internal architecture. In 82586 mode the 82596 will dump the same number of bytes as the 82586. The compatible data will be marked with an asterisk.

**These bytes are not user defined, results may vary from Dump command to Dump command.

Figure 34. Dump Area Format—82586 Mode
<table>
<thead>
<tr>
<th>31</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>CONFIGURE BYTES 5, 4, 3, 2</td>
<td>00</td>
</tr>
<tr>
<td>CONFIGURE BYTES 9, 8, 7, 6</td>
<td>04</td>
</tr>
<tr>
<td>CONFIGURE BYTES 13, 12, 11, 10</td>
<td>08</td>
</tr>
<tr>
<td>I.A. BYTES 1, 0</td>
<td>X X X X X X X X</td>
</tr>
<tr>
<td>I.A. BYTES 5, 2</td>
<td>10</td>
</tr>
<tr>
<td>TX CRC BYTES 0, 1</td>
<td>LAST T.X. STATUS</td>
</tr>
<tr>
<td>RX CRC BYTES 0, 1</td>
<td>TX CRC BYTES 3, 2</td>
</tr>
<tr>
<td>RX TEMP MEMORY 1, 0</td>
<td>RX CRC BYTES 3, 2</td>
</tr>
<tr>
<td>R.X. TEMP MEMORY 5, 2</td>
<td>20</td>
</tr>
<tr>
<td>HASH REGISTERS 1, 0</td>
<td>LAST R.X. STATUS</td>
</tr>
<tr>
<td>HASH REGISTER BYTES 5, 2</td>
<td>24</td>
</tr>
<tr>
<td>SLOT TIME COUNTER</td>
<td>HASH REGISTERS 7, 6</td>
</tr>
<tr>
<td>RECEIVE FRAME LENGTH</td>
<td>WAIT-TIME COUNTER</td>
</tr>
<tr>
<td>MICRO MACHINE**</td>
<td>34</td>
</tr>
<tr>
<td>REGISTER FILE</td>
<td></td>
</tr>
<tr>
<td>128 BYTES</td>
<td>B0</td>
</tr>
<tr>
<td>MICRO MACHINE LFSR**</td>
<td>B4</td>
</tr>
<tr>
<td>MICRO MACHINE**</td>
<td>B8</td>
</tr>
<tr>
<td>FLAG ARRAY</td>
<td></td>
</tr>
<tr>
<td>28 BYTES</td>
<td>D0</td>
</tr>
<tr>
<td>M.M. INPUT PORT**</td>
<td>D4</td>
</tr>
<tr>
<td>16 BYTES</td>
<td>E0</td>
</tr>
<tr>
<td>MICRO MACHINE ALU**</td>
<td>E4</td>
</tr>
<tr>
<td>RESERVED**</td>
<td>E8</td>
</tr>
<tr>
<td>M.M. TEMP A ROTATE R.**</td>
<td>EC</td>
</tr>
<tr>
<td>M.M. TEMP A**</td>
<td>F0</td>
</tr>
<tr>
<td>T.X. DMA BYTE COUNT**</td>
<td>F4</td>
</tr>
<tr>
<td>M.M. INPUT PORT ADDRESS REGISTER**</td>
<td>F8</td>
</tr>
<tr>
<td>T.X. DMA ADDRESS**</td>
<td>FC</td>
</tr>
<tr>
<td>M.M. OUTPUT PORT REGISTER**</td>
<td>100</td>
</tr>
<tr>
<td>R.X. DMA BYTE COUNT**</td>
<td>104</td>
</tr>
<tr>
<td>M.M. OUTPUT PORT ADDRESS REGISTER**</td>
<td>108</td>
</tr>
<tr>
<td>R.X. DMA ADDRESS REGISTER**</td>
<td>10C</td>
</tr>
<tr>
<td>RESERVED**</td>
<td>110</td>
</tr>
<tr>
<td>BUS THROTTLE TIMERS</td>
<td>114</td>
</tr>
<tr>
<td>DIU CONTROL REGISTER**</td>
<td>118</td>
</tr>
<tr>
<td>RESERVED**</td>
<td>11C</td>
</tr>
<tr>
<td>DMA CONTROL REGISTER**</td>
<td>120</td>
</tr>
<tr>
<td>BIU CONTROL REGISTER**</td>
<td>124</td>
</tr>
<tr>
<td>M.M. DISPATCHER REG.**</td>
<td>128</td>
</tr>
<tr>
<td>M.M. STATUS REGISTER**</td>
<td>12C</td>
</tr>
</tbody>
</table>

NOTE:
*The 82596 is not Dump compatible with the 82586 because of the 32-bit internal architecture. In 82586 mode the 82596 will dump the same number of bytes as the 82586. The compatible data will be marked with an asterisk.
**These bytes are not user defined, results may vary from Dump command to Dump command.

Figure 35. Dump Area Format—Linear and 32-Bit Segmented Mode
DIAGNOSE

The Diagnose Command triggers an internal self-test procedure that checks internal 82596 hardware, which includes:

- Exponential Backoff Random Number Generator (Linear Feedback Shift Register).
- Exponential Backoff Timeout Counter.
- Slot Time Period Counter.
- Collision Number Counter.
- Exponential Backoff Shift Register.
- Exponential Backoff Mask Logic.
- Timer Trigger Logic.

This procedure checks the operation of the Backoff block, which resides in the serial side and is not easily controlled. The Diagnose command is performed in two phases.

The format of the 82596 Diagnose command is:

```
82586 and 32-Bit Segmented Modes

<table>
<thead>
<tr>
<th>Bit</th>
<th>ODD WORD</th>
<th>EVEN WORD</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>16 15</td>
<td>0</td>
</tr>
<tr>
<td>EL</td>
<td>S I</td>
<td></td>
</tr>
<tr>
<td>X X X X X X X X X X X X X X X</td>
<td>C B OK F 0 0 0 0 0 0 0 0 0 0 0 0</td>
<td></td>
</tr>
<tr>
<td>X X X X X X X X X X X X X X X</td>
<td>A15 LINK OFFSET A0</td>
<td></td>
</tr>
</tbody>
</table>

Linear Mode

<table>
<thead>
<tr>
<th>Bit</th>
<th>ODD WORD</th>
<th>EVEN WORD</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>16 15</td>
<td>0</td>
</tr>
<tr>
<td>EL</td>
<td>S I</td>
<td></td>
</tr>
<tr>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
<td>C B OK F 0 0 0 0 0 0 0 0 0 0 0 0</td>
<td></td>
</tr>
<tr>
<td>A31</td>
<td>LINK ADDRESS A0</td>
<td></td>
</tr>
</tbody>
</table>
```

where:

- LINK ADDRESS; — As per standard Command Block (see the NOP command for details).
- EL, B, C, I, S — Reserved (0 in the 32-bit Segmented and Linear Modes).
- Bits 19–28 — The Diagnose command. Value: 7h.
- OK (bit 13) — Indicates error free completion.
- F (bit 11) — Indicates that the self-test procedure has failed.

RECEIVE FRAME DESCRIPTOR

Each received frame is described by one Receive Frame Descriptor (see Figure 37). Two new memory structures are available for the received frames. The structures are available only in the Linear and 32-bit Segmented modes.
Simplified Memory Structure

The first is the Simplified memory structure, the data section of the received frame is part of the RFD and is located immediately after the Length Field. Receive Buffer Descriptors are not used with the Simplified structure, it is primarily used to make programming easier. If the length of the data area described in the Size Field is smaller than the incoming frame, the following happens.

1. The received frame is truncated.
2. The No Resource error counter is updated.
3. If the 82596 is configured to Save Bad Frames the RFD is not reused; otherwise, the same RFD is used to hold the next received frame, and the only action taken regarding the truncated frame is to update the counter.
4. The 82596 continues to receive the next frame in the next RFD.
Note that this sequence is very useful for monitoring. If the 82596 is configured to Save Bad Frames, to receive in Promiscuous mode, and to use the Simplified memory structure, any programmed length of received data can be saved in memory.

The Simplified memory structure is shown in Figure 38.

**Flexible Memory Structure**

The second structure is the Flexible memory structure, the data structure of the received frame is stored in both the RFD and in a linked list of Receive Buffers—Receive Buffer Descriptors. The received frame is placed in the RFD as configured in the Size field. Any remaining data is placed in a linked list of RBDs.

The Flexible memory structure is shown in Figure 39.
Buffers on the receive side can be different lengths. The 82596 will not place more bytes into a buffer than indicated in the associated RBD. The 82596 will fetch the next RBD before it is needed. The 82596 will attempt to receive frames as long as the FBL is not exhausted. If there are no more buffers, the 82596 Receive Unit will enter the No Resources state. Before starting the RU, the CPU must place the FBL pointer in the RBD pointer field of the first RFD. All remaining RBD pointer fields for subsequent RFDs should be "1s." If the Receive Frame Descriptor and the associated Receive Buffers are not reused (e.g., the frame is properly received or the 82596 is configured to Save Bad Frames), the 82596 writes the address of the next free RBD to the RBD pointer field of the next RFD.

**RECEIVE BUFFER DESCRIPTOR (RBD)**

The RBDs are used to store received data in a flexible set of linked buffers. The portion of the frame's data field that is outside the RFD is placed in a set of buffers chained by a sequence of RBDs. The RFD points to the first RBD, and the last RBD is flagged with an EOF bit set to 1. Each buffer in the linked list of buffers related to a particular frame can be any size up to $2^{14}$ bytes but must be word aligned (begin on an even numbered byte). This ensures optimum use of the memory resources while maintaining low overhead. All buffers in a frame are filled with the received data except for the last, in which the actual count can be smaller than the allocated buffer space.

---

![Figure 39. RFA Flexible Memory Structure](image-url)
### Figure 40. Receive Frame Descriptor—82586 Mode

<table>
<thead>
<tr>
<th>31</th>
<th>ODD WORD</th>
<th>16 15</th>
<th>EVEN WORD</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>EL</td>
<td>S</td>
<td>C</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>X X X X X X</td>
<td>B</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>X X X X X X</td>
<td>OK</td>
<td>STATUS BITS</td>
<td>0 0 0 0 0 0 0 0</td>
</tr>
<tr>
<td>A15</td>
<td>RBD OFFSET</td>
<td>A0</td>
<td>LINK OFFSET</td>
<td>A0 4</td>
</tr>
<tr>
<td>4th byte</td>
<td>DESTINATION ADDRESS</td>
<td>1st byte</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>SOURCE ADDRESS</td>
<td>1st byte</td>
<td>6th byte</td>
<td>12</td>
<td></td>
</tr>
<tr>
<td>6th byte</td>
<td>4th byte</td>
<td>16</td>
<td></td>
<td></td>
</tr>
<tr>
<td>X X X X X X X X X X X X X X X X X</td>
<td>LENGTH FIELD</td>
<td>20</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Figure 41. Receive Frame Descriptor—32-Bit Segmented Mode

<table>
<thead>
<tr>
<th>31</th>
<th>ODD WORD</th>
<th>16 15</th>
<th>EVEN WORD</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>EL</td>
<td>S</td>
<td>C</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0 0 0 0 0</td>
<td>SF</td>
<td>STATUS BITS</td>
<td>0</td>
</tr>
<tr>
<td>A15</td>
<td>RBD OFFSET</td>
<td>A0</td>
<td>LINK OFFSET</td>
<td>A0 4</td>
</tr>
<tr>
<td>0</td>
<td>SIZE</td>
<td>E0F</td>
<td>ACTUAL COUNT</td>
<td>8</td>
</tr>
<tr>
<td>4th byte</td>
<td>DESTINATION ADDRESS</td>
<td>1st byte</td>
<td>12</td>
<td></td>
</tr>
<tr>
<td>SOURCE ADDRESS</td>
<td>1st byte</td>
<td>6th byte</td>
<td>16</td>
<td></td>
</tr>
<tr>
<td>6th byte</td>
<td>4th byte</td>
<td>20</td>
<td></td>
<td></td>
</tr>
<tr>
<td>X X X X X X X X X X X X X X X X X</td>
<td>LENGTH FIELD</td>
<td>24</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Figure 42. Receive Frame Descriptor—Linear Mode

<table>
<thead>
<tr>
<th>31</th>
<th>ODD WORD</th>
<th>16 15</th>
<th>EVEN WORD</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>EL</td>
<td>S</td>
<td>C</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0 0 0 0 0</td>
<td>SF</td>
<td>STATUS BITS</td>
<td>0</td>
</tr>
<tr>
<td>A31</td>
<td>LINK ADDRESS</td>
<td>A0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>A31</td>
<td>RECEIVE BUFFER DESCRIPTOR ADDRESS</td>
<td>A0 8</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>SIZE</td>
<td>E0F</td>
<td>ACTUAL COUNT</td>
<td>12</td>
</tr>
<tr>
<td>4th byte</td>
<td>DESTINATION ADDRESS</td>
<td>1st byte</td>
<td>16</td>
<td></td>
</tr>
<tr>
<td>SOURCE ADDRESS</td>
<td>1st byte</td>
<td>6th byte</td>
<td>20</td>
<td></td>
</tr>
<tr>
<td>6th byte</td>
<td>4th byte</td>
<td>24</td>
<td></td>
<td></td>
</tr>
<tr>
<td>X X X X X X X X X X X X X X X X X</td>
<td>LENGTH FIELD</td>
<td>28</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

1-164
where:

EL — When set, this bit indicates that this RFD is the last one on the RDL.
S — When set, this bit suspends the RU after receiving the frame.
SF — This bit selects between the Simplified or the Flexible mode.

0 — Simplified mode, all the RX data is in the RFD. RBD ADDRESS field is all “1s.”
1 — Flexible mode. Data is in the RFD and in a linked list of Receive Buffer Descriptors.

C — This bit indicates the completion of frame reception. It is set by the 82596.
B — This bit indicates that the 82596 is currently receiving this frame, or that the 82596 is ready to receive the frame. It is initially set to 0 by the CPU. The 82596 sets it to 1 when reception set up begins, and to 0 upon completion. The C and B bits are set during the same operation.

OK (bit 13) — Frame received successfully, without errors. RFDs with bit 13 equal to 0 are possible only if the save bad frames configuration option is selected. Otherwise all frames with errors will be discarded, although statistics will be collected on them.

STATUS — The results of the Receive operation. Defined bits are,

Bit 12: Length error if configured to check length
Bit 11: CRC error in an aligned frame
Bit 10: Alignment error (CRC error in misaligned frame)
Bit 9: Ran out of buffer space—no resources
Bit 8: DMA Overrun failure to acquire the system bus.
Bit 7: Frame too short.
Bit 6: No EOP flag (for Bit stuffing only)
Bit 5: When the SF bit equals zero, and the 82596 is configured to save bad frames, this bit signals that the received frame was truncated. Otherwise it is zero.

Bits 2-4: Zeros

Bit 1: When it is zero, the destination address of the received frame matches the IA address. When it is 1, the destination address of the received frame does not match the individual address. For example, a multicast address or broadcast address will set this bit to a 1.

Bit 0: Receive collision, a collision is detected during reception.

LINK ADDRESS — A 16-bit offset (32-bit address in the Linear mode) to the next Receive Frame Descriptor. The Link Address of the last frame can be used to form a cyclical list.

RBD POINTER — The offset (address in the Linear mode) of the first RBD containing the received frame data. An RBD pointer of all ones indicates no RBD.

EOF F — These fields are for the Simplified and Flexible memory models. They are exactly the same as the respective fields in the Receive Buffer Descriptor. See the next section for detailed explanation of their functions.

SIZE ACT COUNT MC — Multicast bit.

DESTINATION ADDRESS — The contents of the destination address field of the received frame. The field is 0 to 6 bytes long.

SOURCE ADDRESS — The contents of the Source Address field of the received frame. It is 0 to 6 bytes long.
LENGTH FIELD — The contents of this 2-byte field are user defined. In 802.3 it contains the length of the data field. It is placed in memory in the same order it is received, i.e., most significant byte first, least significant byte second.

NOTES
1. The Destination address, Source address and Length fields are packed, i.e., one field immediately follows the next.
2. The affect of Address/Length Location (No Source Address insertion) configuration parameter while receiving is as follows:
   — 82586 Mode: The Destination address, Source address and Length field are not used, they are placed in the RX data buffers.
   — 32-Bit Segmented and Linear Modes: when the Simplified memory model is used, the Destination address, Source address and Length fields reside in their respective fields in the RFD. When the Flexible memory structure is used the Destination address, Source address, and Length field locations depend on the SIZE field of the RFD. They can be placed in the RFD, in the RX data buffers, or partially in the RFD and the rest in the RX data buffers, depending on the SIZE field value.

<table>
<thead>
<tr>
<th>82586 Mode</th>
<th>31</th>
<th>ODD WORD</th>
<th>16 15</th>
<th>EVEN WORD</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>A15</td>
<td>NEXT RBD OFFSET</td>
<td>A0 EOF F</td>
<td>ACTUAL COUNT</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>X X X X X X X X</td>
<td>A23</td>
<td>RECEIVE BUFFER ADDRESS</td>
<td>A0 4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>X X X X X X X X X X X X X X X X</td>
<td>EL X</td>
<td>SIZE</td>
<td>8</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>32-Bit Segmented Mode</th>
<th>31</th>
<th>ODD WORD</th>
<th>16 15</th>
<th>EVEN WORD</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>A15</td>
<td>NEXT RBD OFFSET</td>
<td>A0 EOF F</td>
<td>ACTUAL COUNT</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>A31</td>
<td>RECEIVE BUFFER ADDRESS</td>
<td>A0 4</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
<td>EL P</td>
<td>SIZE</td>
<td>8</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Linear Mode</th>
<th>31</th>
<th>ODD WORD</th>
<th>16 15</th>
<th>EVEN WORD</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
<td>EOF F</td>
<td>ACTUAL COUNT</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>A31</td>
<td>NEXT RBD ADDRESS</td>
<td>A0 4</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A31</td>
<td>RECEIVE BUFFER ADDRESS</td>
<td>A0 8</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
<td>EL P</td>
<td>SIZE</td>
<td>8</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure 43. Receive Buffer Descriptor
where:

EOF — Indicates that this is the last buffer related to the frame. It is cleared by the CPU before starting the RU, and is written by the 82596 at the end of reception of the frame.

F — Indicates that this buffer has already been used. The Actual Count has no meaning unless the F bit equals one. This bit is cleared by the CPU before starting the RU, and is set by the 82596 after the associated buffer has been. This bit has the same meaning as the Complete bit in the RFD and CB.

ACT COUNT — This 14-bit quantity indicates the number of meaningful bytes in the buffer. It is cleared by the CPU before starting the RU, and is written by the 82596 after the associated buffer has already been used. In general, after the buffer is full, the Actual Count value equals the size field of the same buffer. For the last buffer of the frame, Actual Count can be less than the buffer size.

NEXT BD ADDRESS — The offset (absolute address in the Linear mode) of the next RBD on the list. It is meaningless if EL = 1.

BUFFER ADDRESS — The starting address of the memory area that contains the received data. In the 82586 mode, this is a 24-bit address (with pins A24–A31 = 0). In the 32-bit Segmented and Linear modes this is a 32-bit address.

EL — Indicates that the buffer associated with this RBD is last in the FBL.

P — This bit indicates that the 82596 has already prefetched the RBDs and any change in the RBD data will be ignored. This bit is valid only in the new 82596 memory modes, and if this feature has been enabled during configure command. The 82596 Prefetches the RBDs in locked cycles; after prefetching the RBD the 82596 performs a write cycle where the P bit is set to one and the rest of the data remains unchanged. The CPU is responsible for resetting it in all RBDs. The 82596 will not check this bit before setting it.

SIZE — This 14-bit quantity indicates the size, in bytes, of the associated buffer. This quantity must be an even number.
**D.C. CHARACTERISTICS**

\( T_C = 0^\circ C - 85^\circ C, V_{CC} = 5V \pm 10\% \) CLK2 and LE/BE have MOS levels (see \( V_{MIL}, V_{MIH} \)).
All other signals have TTL levels (see \( V_{IL}, V_{IH}, V_{OL}, V_{OH} \)).

<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>( V_{IL} )</td>
<td>Input Low Voltage (TTL)</td>
<td>-0.3</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} + 0.3 )</td>
<td>V</td>
<td></td>
</tr>
<tr>
<td>( V_{MIL} )</td>
<td>Input Low Voltage (MOS)</td>
<td>-0.3</td>
<td>+0.8</td>
<td>V</td>
<td></td>
</tr>
<tr>
<td>( V_{MIH} )</td>
<td>Input High Voltage (MOS)</td>
<td>3.7</td>
<td>( V_{CC} + 0.3 )</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></td>
<td>( I_{OL} = 4.0 , mA^1 )</td>
</tr>
<tr>
<td>( V_{CIL} )</td>
<td>( RxC, TxC ) Input Low Voltage</td>
<td>-0.5</td>
<td>0.6</td>
<td>V</td>
<td></td>
</tr>
<tr>
<td>( V_{CIH} )</td>
<td>( RxC, TxC ) Input High Voltage</td>
<td>3.3</td>
<td>( V_{CC} + 0.5 )</td>
<td>V</td>
<td></td>
</tr>
<tr>
<td>( V_{OH} )</td>
<td>Output High Voltage (TTL)</td>
<td>2.4</td>
<td>V</td>
<td></td>
<td>( I_{OH} = 0.9mA - 1mA^1 )</td>
</tr>
<tr>
<td>( I_{LI} )</td>
<td>Input Leakage Current</td>
<td>( \pm 15 ) ( \mu A )</td>
<td>( 0 \leq V_{IN} \leq V_{CC} )</td>
<td></td>
<td></td>
</tr>
<tr>
<td>( I_{LO} )</td>
<td>Output Leakage Current</td>
<td>( \pm 15 ) ( \mu A )</td>
<td>( 0.45 &lt; V_{OUT} &lt; V_{CC} )</td>
<td></td>
<td></td>
</tr>
<tr>
<td>( C_{IN} )</td>
<td>Capacitance of Input Buffer</td>
<td>10</td>
<td>( pF )</td>
<td>( FC = 1 , MHz )</td>
<td></td>
</tr>
<tr>
<td>( C_{OUT} )</td>
<td>Capacitance of Input/Output Buffer</td>
<td>12</td>
<td>( pF )</td>
<td>( FC = 1 , MHz )</td>
<td></td>
</tr>
<tr>
<td>( C_{CLK} )</td>
<td>CLK Capacitance</td>
<td>20</td>
<td>( pF )</td>
<td>( FC = 1 , MHz )</td>
<td></td>
</tr>
<tr>
<td>( I_{CC} )</td>
<td>Power Supply</td>
<td>150</td>
<td>( mA )</td>
<td>At 16 MHz for the 82596SX</td>
<td></td>
</tr>
<tr>
<td>( I_{CC} )</td>
<td>Power Supply</td>
<td>200</td>
<td>( mA )</td>
<td>At 25 MHz</td>
<td></td>
</tr>
<tr>
<td>( I_{CC} )</td>
<td>Power Supply</td>
<td>300</td>
<td>( mA )</td>
<td>At 33 MHz</td>
<td></td>
</tr>
</tbody>
</table>

---

1-168
A.C. CHARACTERISTICS

82596DX INPUT/OUTPUT SYSTEM TIMINGS \( T_C = 0^\circ \text{C} \) to \( +85^\circ \text{C} \), \( V_{CC} = 5V \pm 10\% \)

These timings assume the \( C_L \) on all outputs is 50 pF unless otherwise specified. \( C_L \) can be 20 pF to 120 pF, however, timings must be derated.
All timing requirements are given in nanoseconds.

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Parameter</th>
<th>25 MHz</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Min</td>
<td>Max</td>
</tr>
<tr>
<td>T1</td>
<td>Operating Frequency</td>
<td>12 MHz</td>
<td>25 MHz</td>
</tr>
<tr>
<td>T2</td>
<td>CLK2 Period</td>
<td>20</td>
<td>40</td>
</tr>
<tr>
<td>T3</td>
<td>CLK2 High</td>
<td>4</td>
<td>3.7V</td>
</tr>
<tr>
<td>T4</td>
<td>CLK2 Low</td>
<td>5</td>
<td>0.8V</td>
</tr>
<tr>
<td>T5</td>
<td>CLK2 Rise Time</td>
<td>—</td>
<td>7</td>
</tr>
<tr>
<td>T13</td>
<td>CLK2 Fall Time</td>
<td>—</td>
<td>7</td>
</tr>
<tr>
<td>T14</td>
<td>CA and BREQ Setup Time</td>
<td>7</td>
<td>1, 2, 3</td>
</tr>
<tr>
<td>T26</td>
<td>CA and BREQ Hold Time</td>
<td>3</td>
<td>1, 2, 3</td>
</tr>
<tr>
<td>T25</td>
<td>CA and BREQ, PORT Pulse Width</td>
<td>4 T1</td>
<td>3</td>
</tr>
<tr>
<td>T6</td>
<td>IN Valid Delay</td>
<td>1</td>
<td>26</td>
</tr>
<tr>
<td>T7</td>
<td>BEx, LOCK, and A2–A31 Valid Delay</td>
<td>4</td>
<td>21</td>
</tr>
<tr>
<td>T8</td>
<td>W/R and ADS Valid Delay</td>
<td>4</td>
<td>21</td>
</tr>
<tr>
<td>T9</td>
<td>W/R and ADS Float Delay</td>
<td>4</td>
<td>30</td>
</tr>
<tr>
<td>T10</td>
<td>D0–D31 Write Data Valid Delay</td>
<td>3</td>
<td>22</td>
</tr>
<tr>
<td>T11</td>
<td>D0–D31 Write Data Float Delay</td>
<td>4</td>
<td>22</td>
</tr>
<tr>
<td>T27</td>
<td>D0–D31 CPU PORT Access Setup Time</td>
<td>7</td>
<td>2</td>
</tr>
<tr>
<td>T28</td>
<td>D0–D31 CPU PORT Access Hold Time</td>
<td>5</td>
<td>2</td>
</tr>
<tr>
<td>T29</td>
<td>PORT Setup Time</td>
<td>7</td>
<td>2</td>
</tr>
<tr>
<td>T30</td>
<td>PORT Hold Time</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>T17</td>
<td>RDY Setup Time</td>
<td>9</td>
<td>2</td>
</tr>
<tr>
<td>T18</td>
<td>RDY Hold Time</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>T19</td>
<td>D0–D31 READ Setup Time</td>
<td>7</td>
<td>2</td>
</tr>
<tr>
<td>T20</td>
<td>D0–D31 READ Hold Time</td>
<td>5</td>
<td>2</td>
</tr>
<tr>
<td>T12</td>
<td>HOLD Valid Delay</td>
<td>4</td>
<td>22</td>
</tr>
<tr>
<td>T21</td>
<td>HLDA Setup Time</td>
<td>10</td>
<td>1, 2</td>
</tr>
<tr>
<td>T22a</td>
<td>HLDA Hold Time</td>
<td>3</td>
<td>1, 2</td>
</tr>
<tr>
<td>T23</td>
<td>RESET Setup Time</td>
<td>10</td>
<td>2</td>
</tr>
<tr>
<td>T24</td>
<td>RESET Hold Time</td>
<td>3</td>
<td>2</td>
</tr>
</tbody>
</table>
A.C. CHARACTERISTICS (Continued)

82596DX INPUT/OUTPUT SYSTEM TIMINGS \( T_C = 0^\circ C \text{ to } +85^\circ C, V_{CC} = 5V \pm 5\% \)

These timings assume the \( C_L \) on all outputs is 50 pF unless otherwise specified. \( C_L \) can be 20 pF to 120 pF, however, timings must be derated.

All timing requirements are given in nanoseconds.

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Parameter</th>
<th>33 MHz</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Min</td>
<td>Max</td>
</tr>
<tr>
<td>T1</td>
<td>Operating Frequency</td>
<td>12.5 MHz</td>
<td>33 MHz</td>
</tr>
<tr>
<td>T2</td>
<td>CLK2 Period</td>
<td>15</td>
<td>40</td>
</tr>
<tr>
<td>T3</td>
<td>CLK2 High</td>
<td>4.5</td>
<td></td>
</tr>
<tr>
<td>T4</td>
<td>CLK2 Low</td>
<td>4.5</td>
<td></td>
</tr>
<tr>
<td>T5</td>
<td>CLK2 Rise Time</td>
<td>—</td>
<td>4</td>
</tr>
<tr>
<td>T6</td>
<td>CLK2 Fall Time</td>
<td>—</td>
<td>4</td>
</tr>
<tr>
<td>T7</td>
<td>CA and BREQ Setup Time</td>
<td>7</td>
<td>1, 2, 3</td>
</tr>
<tr>
<td>T8</td>
<td>CA and BREQ Hold Time</td>
<td>3</td>
<td>1, 2, 3</td>
</tr>
<tr>
<td>T9</td>
<td>CA and BREQ, PORT Pulse Width</td>
<td>4 T1</td>
<td>3</td>
</tr>
<tr>
<td>T10</td>
<td>BEX, LOCK, and A2–A31 Valid Delay</td>
<td>4</td>
<td>15</td>
</tr>
<tr>
<td>T11</td>
<td>BEX, LOCK, and A2–A31 Float Delay</td>
<td>4</td>
<td>20</td>
</tr>
<tr>
<td>T12</td>
<td>W/R and ADS Valid Delay</td>
<td>4</td>
<td>15</td>
</tr>
<tr>
<td>T13</td>
<td>W/R and ADS Float Delay</td>
<td>4</td>
<td>20</td>
</tr>
<tr>
<td>T14</td>
<td>D0–D31 Write Data Valid Delay</td>
<td>3</td>
<td>19</td>
</tr>
<tr>
<td>T15</td>
<td>D0–D31 Write Data Float Delay</td>
<td>4</td>
<td>17</td>
</tr>
<tr>
<td>T16</td>
<td>D0–D31 CPU PORT Access Setup Time</td>
<td>5</td>
<td>2</td>
</tr>
<tr>
<td>T17</td>
<td>D0–D31 CPU PORT Access Hold Time</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>T18</td>
<td>PORT Setup Time</td>
<td>7</td>
<td>2</td>
</tr>
<tr>
<td>T19</td>
<td>PORT Hold Time</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>T20</td>
<td>RDY Setup Time</td>
<td>7</td>
<td>2</td>
</tr>
<tr>
<td>T21</td>
<td>RDY Hold Time</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>T22</td>
<td>D0–D31 READ Setup Time</td>
<td>5</td>
<td>2</td>
</tr>
<tr>
<td>T23</td>
<td>D0–D31 READ Hold Time</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>T24</td>
<td>HOLD Valid Delay</td>
<td>3</td>
<td>19</td>
</tr>
<tr>
<td>T25</td>
<td>INT Valid Delay</td>
<td>1</td>
<td>20</td>
</tr>
<tr>
<td>T26</td>
<td>HLDQ Setup Time</td>
<td>8</td>
<td>1, 2</td>
</tr>
<tr>
<td>T27</td>
<td>HLDQ Hold Time</td>
<td>3</td>
<td>1, 2</td>
</tr>
<tr>
<td>T28</td>
<td>RESET Setup Time</td>
<td>8</td>
<td>2</td>
</tr>
<tr>
<td>T29</td>
<td>RESET Hold Time</td>
<td>3</td>
<td>2</td>
</tr>
</tbody>
</table>
A.C. CHARACTERISTICS (Continued)

82596SX INPUT/OUTPUT SYSTEM TIMINGS \( T_C = 0^\circ\text{C to } +85^\circ\text{C}, V_{CC} = 5V \pm 10\% \)

These timings assume the \( C_L \) on all outputs is 50 pF unless otherwise specified. \( C_L \) can be 20 pF to 120 pF, however, timings must be derated.

All timing requirements are given in nanoseconds.

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Parameter</th>
<th>( 20 \text{ MHz} )</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Min</td>
<td>Max</td>
</tr>
<tr>
<td>Operating Frequency</td>
<td>12.5 MHz</td>
<td>20 MHz</td>
<td>CLK2/2</td>
</tr>
<tr>
<td>T1</td>
<td>CLK2 Period</td>
<td>25</td>
<td>40</td>
</tr>
<tr>
<td>T2</td>
<td>CLK2 High</td>
<td>8</td>
<td>at 2.0V</td>
</tr>
<tr>
<td>T3</td>
<td>CLK2 Low</td>
<td>8</td>
<td>at 2.0V</td>
</tr>
<tr>
<td>T4</td>
<td>CLK2 Rise Time</td>
<td>—</td>
<td>8</td>
</tr>
<tr>
<td>T5</td>
<td>CLK2 Fall Time</td>
<td>—</td>
<td>8</td>
</tr>
<tr>
<td>T13</td>
<td>CA and BREQ Setup Time</td>
<td>10</td>
<td>1, 2, 3</td>
</tr>
<tr>
<td>T14</td>
<td>CA and BREQ Hold Time</td>
<td>8</td>
<td>1, 2, 3</td>
</tr>
<tr>
<td>T26</td>
<td>CA and BREQ, PORT Pulse Width</td>
<td>4 T1</td>
<td>3</td>
</tr>
<tr>
<td>T25</td>
<td>INT Valid Delay</td>
<td>1</td>
<td>35</td>
</tr>
<tr>
<td>T6</td>
<td>BHE, BLE, LOCK, BON, and A1–A31 Valid Delay</td>
<td>4</td>
<td>30</td>
</tr>
<tr>
<td>T7</td>
<td>BHE, BLE, LOCK, BON, and A1–A31 Float Delay</td>
<td>4</td>
<td>30</td>
</tr>
<tr>
<td>T8</td>
<td>W/R and ADS Valid Delay</td>
<td>4</td>
<td>26</td>
</tr>
<tr>
<td>T9</td>
<td>W/R and ADS Float Delay</td>
<td>4</td>
<td>30</td>
</tr>
<tr>
<td>T10</td>
<td>D0–D15 Write Data Valid Delay</td>
<td>4</td>
<td>38</td>
</tr>
<tr>
<td>T11</td>
<td>D0–D15 Write Data Float Delay</td>
<td>4</td>
<td>27</td>
</tr>
<tr>
<td>T27</td>
<td>D0–D15 CPU PORT Access Setup Time</td>
<td>9</td>
<td>2</td>
</tr>
<tr>
<td>T28</td>
<td>D0–D15 CPU PORT Access Hold Time</td>
<td>6</td>
<td>2</td>
</tr>
<tr>
<td>T29</td>
<td>PORT Setup Time</td>
<td>10</td>
<td>2</td>
</tr>
<tr>
<td>T30</td>
<td>PORT Hold Time</td>
<td>8</td>
<td>2</td>
</tr>
<tr>
<td>T17</td>
<td>RDY Setup Time</td>
<td>12</td>
<td>2</td>
</tr>
<tr>
<td>T18</td>
<td>RDY Hold Time</td>
<td>5.5</td>
<td>2</td>
</tr>
<tr>
<td>T19</td>
<td>D0–D15 READ Setup Time</td>
<td>9</td>
<td>2</td>
</tr>
<tr>
<td>T20</td>
<td>D0–D15 READ Hold Time</td>
<td>6</td>
<td>2</td>
</tr>
<tr>
<td>T12</td>
<td>HOLD Valid Delay</td>
<td>4</td>
<td>28</td>
</tr>
<tr>
<td>T21</td>
<td>HLDA Setup Time</td>
<td>15</td>
<td>1, 2</td>
</tr>
<tr>
<td>T22a</td>
<td>HLDA Hold Time</td>
<td>8</td>
<td>1, 2</td>
</tr>
<tr>
<td>T23</td>
<td>RESET Setup Time</td>
<td>12</td>
<td>1, 2</td>
</tr>
<tr>
<td>T24</td>
<td>RESET Hold Time</td>
<td>4</td>
<td>1, 2</td>
</tr>
</tbody>
</table>
A.C. CHARACTERISTICS (Continued)

82596SX INPUT/OUTPUT SYSTEM TIMINGS $T_C = 0^\circ C$ to $+85^\circ C$, $V_{CC} = 5V \pm 10\%$

These timings assume the $C_L$ on all outputs is 50 pF unless otherwise specified. $C_L$ can be 20 pF to 120 pF, however, timings must be derated.

All timing requirements are given in nanoseconds.

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Parameter</th>
<th>16 MHz</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Operating Frequency</td>
<td></td>
<td></td>
</tr>
<tr>
<td>T1</td>
<td>CLK2 Period</td>
<td>31</td>
<td>40</td>
</tr>
<tr>
<td>T2</td>
<td>CLK2 High</td>
<td>9</td>
<td>2.0V</td>
</tr>
<tr>
<td>T3</td>
<td>CLK2 Low</td>
<td>9</td>
<td>2.0V</td>
</tr>
<tr>
<td>T4</td>
<td>CLK2 Rise Time</td>
<td></td>
<td>0.8V to 3.7V</td>
</tr>
<tr>
<td>T5</td>
<td>CLK2 Fall Time</td>
<td></td>
<td>3.7V to 0.8V</td>
</tr>
<tr>
<td>T13</td>
<td>CA and BREQ Setup Time</td>
<td>11</td>
<td>1, 2, 3</td>
</tr>
<tr>
<td>T14</td>
<td>CA and BREQ Hold Time</td>
<td>8</td>
<td>1, 2, 3</td>
</tr>
<tr>
<td>T26</td>
<td>CA and BREQ, PORT Pulse Width</td>
<td>4 T1</td>
<td>3</td>
</tr>
<tr>
<td>T25</td>
<td>INT Valid Delay</td>
<td>1</td>
<td>40</td>
</tr>
<tr>
<td>T6</td>
<td>BHE, BLE, LOCK, BON, and A1–A31 Valid Delay</td>
<td>4</td>
<td>36</td>
</tr>
<tr>
<td>T7</td>
<td>BHE, BLE, LOCK, BON, and A1–A31 Float Delay</td>
<td>4</td>
<td>40</td>
</tr>
<tr>
<td>T8</td>
<td>W/R and ADS Valid Delay</td>
<td>4</td>
<td>33</td>
</tr>
<tr>
<td>T9</td>
<td>W/R and ADS Float Delay</td>
<td>4</td>
<td>35</td>
</tr>
<tr>
<td>T10</td>
<td>D0–D15 Write Data Valid Delay</td>
<td>4</td>
<td>40</td>
</tr>
<tr>
<td>T11</td>
<td>D0–D15 Write Data Float Delay</td>
<td>4</td>
<td>35</td>
</tr>
<tr>
<td>T27</td>
<td>D0–D15 CPU PORT Access Setup Time</td>
<td>9</td>
<td>2</td>
</tr>
<tr>
<td>T28</td>
<td>D0–D15 CPU PORT Access Hold Time</td>
<td>6</td>
<td>2</td>
</tr>
<tr>
<td>T29</td>
<td>PORT Setup Time</td>
<td>11</td>
<td>2</td>
</tr>
<tr>
<td>T30</td>
<td>PORT Hold Time</td>
<td>8</td>
<td>2</td>
</tr>
<tr>
<td>T17</td>
<td>RDY Setup Time</td>
<td>19</td>
<td>2</td>
</tr>
<tr>
<td>T18</td>
<td>RDY Hold Time</td>
<td>6</td>
<td>2</td>
</tr>
<tr>
<td>T19</td>
<td>D0–D15 READ Setup Time</td>
<td>9</td>
<td>2</td>
</tr>
<tr>
<td>T20</td>
<td>D0–D15 READ Hold Time</td>
<td>6</td>
<td>2</td>
</tr>
<tr>
<td>T12</td>
<td>HOLD Valid Delay</td>
<td>4</td>
<td>33</td>
</tr>
<tr>
<td>T21</td>
<td>HLDA Setup Time</td>
<td>15</td>
<td>1, 2</td>
</tr>
<tr>
<td>T22a</td>
<td>HLDA Hold Time</td>
<td>8</td>
<td>1, 2</td>
</tr>
<tr>
<td>T23</td>
<td>RESET Setup Time</td>
<td>13</td>
<td>1, 2</td>
</tr>
<tr>
<td>T24</td>
<td>RESET Hold Time</td>
<td>4</td>
<td>1, 2</td>
</tr>
</tbody>
</table>

NOTES:
1. RESET, HLDA, and CA are internally synchronized. This timing is to guarantee recognition at next clock for RESET, HLDA, and CA.
2. All set-up, hold, and delay timings are at the maximum frequency specification $F_{max}$, and must be derated according to the following equation for operation at lower frequencies:

$$T_{derated} = \frac{(F_{max}/F_{opr}) \times T}{F_{opr}}$$

where:
- $T_{derated}$ = Specifies the value to derate the specification.
- $F_{max}$ = Maximum operating frequency.
- $F_{opr}$ = Actual operating frequency.
- $T$ = Specification at maximum frequency.

This calculation only provides a rough estimate for derating the frequency. For more detailed information contact your Intel sales office for the data sheet supplement.
3. CA is internally synchronized; if the setup and hold times are met then CA needs to be only 2 T1. BREQ and PORT are not internally synchronized. BREQ must meet setup and hold times and need only be 2 T1 wide.
## TRANSMIT/RECEIVE CLOCK PARAMETERS

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Parameter</th>
<th>20 MHz</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Min</td>
<td>Max</td>
</tr>
<tr>
<td>T36</td>
<td>TxC Cycle</td>
<td>50</td>
<td></td>
</tr>
<tr>
<td>T38</td>
<td>TxC Rise Time</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>T39</td>
<td>TxC Fall Time</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>T40</td>
<td>TxC High Time</td>
<td>19</td>
<td></td>
</tr>
<tr>
<td>T41</td>
<td>TxC Low Time</td>
<td>18</td>
<td></td>
</tr>
<tr>
<td>T42</td>
<td>TxD Rise Time</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>T43</td>
<td>TxD Fall Time</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>T44</td>
<td>TxD Transition</td>
<td>20</td>
<td></td>
</tr>
<tr>
<td>T45</td>
<td>TxC Low to TxD Valid</td>
<td>25</td>
<td></td>
</tr>
<tr>
<td>T46</td>
<td>TxC Low to TxD Transition</td>
<td>25</td>
<td></td>
</tr>
<tr>
<td>T47</td>
<td>TxC High to TxD Transition</td>
<td>25</td>
<td></td>
</tr>
<tr>
<td>T48</td>
<td>TxC Low to TxD High (At End of Transition)</td>
<td>25</td>
<td></td>
</tr>
</tbody>
</table>

### RTS AND CTS PARAMETERS

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Parameter</th>
<th>20 MHz</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Min</td>
<td>Max</td>
</tr>
<tr>
<td>T49</td>
<td>TxC Low to RTS Low, Time to Activate RTS</td>
<td>25</td>
<td></td>
</tr>
<tr>
<td>T50</td>
<td>CTS Low to TxC Low, CTS Setup Time</td>
<td>20</td>
<td></td>
</tr>
<tr>
<td>T51</td>
<td>TxC Low to CTS Invalid, CTS Hold Time</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>T52</td>
<td>TxC Low to RTS High</td>
<td>25</td>
<td></td>
</tr>
</tbody>
</table>

### RECEIVE CLOCK PARAMETERS

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Parameter</th>
<th>20 MHz</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Min</td>
<td>Max</td>
</tr>
<tr>
<td>T53</td>
<td>RxC Cycle</td>
<td>50</td>
<td></td>
</tr>
<tr>
<td>T54</td>
<td>RxC Rise Time</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>T55</td>
<td>RxC Fall Time</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>T56</td>
<td>RxC High Time</td>
<td>19</td>
<td></td>
</tr>
<tr>
<td>T57</td>
<td>RxC Low Time</td>
<td>18</td>
<td></td>
</tr>
</tbody>
</table>

### RECEIVED DATA PARAMETERS

| Symbol | Parameter          | 20 MHz |  |
|--------|--------------------|--------| |
| T58    | RxD Setup Time     | 20     | 6 |
| T59    | RxD Hold Time      | 10     | 6 |
### TRANSMIT/RECEIVE CLOCK PARAMETERS (Continued)

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Parameter</th>
<th>20 MHz</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>T60</td>
<td>RxD Rise Time</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>T61</td>
<td>RxD Fall Time</td>
<td>10</td>
<td></td>
</tr>
</tbody>
</table>

### RECEIVED DATA PARAMETERS (Continued)

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Parameter</th>
<th>20 MHz</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>T62</td>
<td>CDT Low to TxC HIGH External Collision Detect Setup Time</td>
<td>20</td>
<td></td>
</tr>
<tr>
<td>T63</td>
<td>TxC High to CDT Inactive, CDT Hold Time</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>T64</td>
<td>CDT Low to Jam Start</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>T65</td>
<td>CRS Low to TxC High, Carrier Sense Setup Time</td>
<td>20</td>
<td></td>
</tr>
<tr>
<td>T66</td>
<td>TxC High to CRS Inactive, CRS Hold Time</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>T67</td>
<td>CRS High to Jamming Start, (Internal Collision Detect)</td>
<td>12</td>
<td></td>
</tr>
<tr>
<td>T68</td>
<td>Jamming Period</td>
<td>11</td>
<td></td>
</tr>
<tr>
<td>T69</td>
<td>CRS High to RxC High, CRS Inactive Setup Time</td>
<td>30</td>
<td></td>
</tr>
<tr>
<td>T70</td>
<td>RxC High to CRS High, CRS Inactive Hold Time</td>
<td>10</td>
<td></td>
</tr>
</tbody>
</table>

### INTERFRAME SPACING PARAMETERS

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Parameter</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>T71</td>
<td>Interframe Delay</td>
<td>9</td>
</tr>
</tbody>
</table>

### EXTERNAL LOOPBACK-PIN PARAMETERS

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Parameter</th>
<th>T36</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>T72</td>
<td>TxC Low to LPBK Low</td>
<td></td>
<td>4</td>
</tr>
<tr>
<td>T73</td>
<td>TxC Low to LPBK High</td>
<td>T36</td>
<td>4</td>
</tr>
</tbody>
</table>

### NOTES:

1. Special MOS levels, $V_{CIL} = 0.9V$ and $V_{CIH} = 3.0V$.
2. Manchester only.
4. 1 TTL load + 50 pF.
5. 1 TTL load + 100 pF.
6. NRZ only.
7. Abnormal end of transmission—CTS expires before RTS.
8. Normal end to transmission.
9. Programmable value:

   $$ T71 = N_{IFS} \cdot T36 $$

   where: $N_{IFS}$ = the IFS configuration value

   (if $N_{IFS}$ is less than 12 then $N_{IFS}$ is forced to 12).

10. Programmable value:

    $$ T64 = (N_{CDF} \cdot T36) + x \cdot T36 $$

    (If the collision occurs after the preamble)

    where:

    $N_{CDF}$ = the collision detect filter configuration value, and

    $x = 12, 13, 14, or 15$

11. $T68 = 32 \cdot T36$

12. Programmable value:

    $$ T67 = (N_{CSF} \cdot T36) + x \cdot T36 $$

    where: $N_{CSF}$ = the Carrier Sense Filter configuration value, and

    $x = 12, 13, 14, or 15$

13. To guarantee recognition on the next clock.
82596DX/SX BUS OPERATION

The following figures show the basic bus cycles for the 82596DX and 82596SX.

For more details refer to the 32-Bit LAN Components Manual.

Figure 44. Basic 82596DX Bus Cycles

Figure 45. Basic 82596SX Bus Cycles
SYSTEM INTERFACE A.C. TIMING CHARACTERISTICS

The measurements should be done at:

- $T_C = 0^\circ\text{C} - 85^\circ\text{C}$, $V_{CC} = 5\text{V} \pm 10\%$, $C = 50\ \mu\text{F}$ unless otherwise specified.
- 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 logic "1" and "0".
- Rise and Fall time of inputs and outputs signals are measured between 0.8V and 2.0V respectively unless otherwise specified.
- All timings are relative to CLK2 crossing the 1.5V level.
- All A.C. parameters are valid only after 100 $\mu$s from power up.

Two types of timing specifications are presented below:
1. Input Timing—minimum setup and hold times.
2. Output Timings—output delays and float times from CLK2 rising edge.

Figure 46. CLK2 Timings

Figure 45 defines how the measurements should be done:

LEGEND:
$Ts = \text{Input Setup Time}$
$Th = \text{Input Hold Time}$
$Tn = \text{Minimum output delay or Minimum float delay}$
$Tx = \text{Maximum output delay or Maximum float delay}$

Figure 47. Drive Levels and Measurements Points for A.C. Specifications
INPUT WAVEFORMS

Ts = T13, T15, T17, T19, T21, T23, T27, T29, T31
Th = T14, T16, T18, T20, T22, T22a, T24, T28, T30, T32

Figure 48. CA and BREQ Input Timing

Figure 49. INT/INT Output Timing

Figure 50. HOLD/HLDA Timings

Figure 51. Input Setup and Hold Time
Figure 52. Output Valid Delay Timing

Figure 53. Output Float Delay Timing

Figure 54. PORT Setup and Hold Time
SERIAL A.C. TIMING CHARACTERISTICS

Figure 55. RESET Input Timing

Figure 56. Serial Input Clock Timing

Figure 57. Transmit Data Waveforms
Figure 58. Transmit Data Waveforms

Figure 59. Receive Data Waveforms (NRZ)

Figure 60. Receive Data Waveforms (CRS)
OUTLINE DIAGRAMS

132 LEAD CERAMIC PIN GRID ARRAY PACKAGE INTEL TYPE A

Family: Ceramic Pin Grid Array Package

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Millimeters</th>
<th>Inches</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Min</td>
<td>Max</td>
<td></td>
</tr>
<tr>
<td>A</td>
<td>3.56</td>
<td>4.57</td>
<td></td>
</tr>
<tr>
<td>A₁</td>
<td>0.76</td>
<td>1.27</td>
<td></td>
</tr>
<tr>
<td>A₂</td>
<td>2.67</td>
<td>3.43</td>
<td></td>
</tr>
<tr>
<td>A₃</td>
<td>1.14</td>
<td>1.40</td>
<td></td>
</tr>
<tr>
<td>B</td>
<td>0.43</td>
<td>0.51</td>
<td></td>
</tr>
<tr>
<td>D</td>
<td>36.45</td>
<td>37.21</td>
<td></td>
</tr>
<tr>
<td>D₁</td>
<td>32.89</td>
<td>33.15</td>
<td></td>
</tr>
<tr>
<td>e₁</td>
<td>2.29</td>
<td>2.79</td>
<td></td>
</tr>
<tr>
<td>L</td>
<td>2.54</td>
<td>3.30</td>
<td></td>
</tr>
<tr>
<td>N</td>
<td>132</td>
<td></td>
<td></td>
</tr>
<tr>
<td>S₁</td>
<td>1.27</td>
<td>2.54</td>
<td></td>
</tr>
</tbody>
</table>

ISSUE IWS 10/12/88
### Intel Case Outline Drawings

Plastic Quad Flat Pack (PQFP)
0.025 Inch (0.635mm) Pitch

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
<th>Min</th>
<th>Max</th>
<th>Min</th>
<th>Max</th>
<th>Min</th>
<th>Max</th>
<th>Min</th>
<th>Max</th>
<th>Min</th>
<th>Max</th>
<th>Min</th>
<th>Max</th>
</tr>
</thead>
<tbody>
<tr>
<td>N</td>
<td>Leadcount</td>
<td>68</td>
<td>84</td>
<td>100</td>
<td>132</td>
<td>164</td>
<td>196</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A</td>
<td>Package Height</td>
<td>0.160</td>
<td>0.170</td>
<td>0.160</td>
<td>0.170</td>
<td>0.160</td>
<td>0.170</td>
<td>0.160</td>
<td>0.170</td>
<td>0.160</td>
<td>0.170</td>
<td></td>
<td></td>
</tr>
<tr>
<td>A1</td>
<td>Standoff</td>
<td>0.020</td>
<td>0.030</td>
<td>0.020</td>
<td>0.030</td>
<td>0.020</td>
<td>0.030</td>
<td>0.020</td>
<td>0.030</td>
<td>0.020</td>
<td>0.030</td>
<td></td>
<td></td>
</tr>
<tr>
<td>D, E</td>
<td>Terminal Dimension</td>
<td>0.675</td>
<td>0.685</td>
<td>0.775</td>
<td>0.785</td>
<td>0.875</td>
<td>0.885</td>
<td>1.075</td>
<td>1.085</td>
<td>1.275</td>
<td>1.285</td>
<td>1.475</td>
<td>1.485</td>
</tr>
<tr>
<td>D1, E1</td>
<td>Package Body</td>
<td>0.547</td>
<td>0.553</td>
<td>0.647</td>
<td>0.653</td>
<td>0.747</td>
<td>0.753</td>
<td>0.947</td>
<td>0.953</td>
<td>1.147</td>
<td>1.153</td>
<td>1.347</td>
<td>1.353</td>
</tr>
<tr>
<td>D2, E2</td>
<td>Bumper Distance</td>
<td>0.697</td>
<td>0.703</td>
<td>0.797</td>
<td>0.803</td>
<td>0.897</td>
<td>0.903</td>
<td>1.097</td>
<td>1.103</td>
<td>1.297</td>
<td>1.303</td>
<td>1.497</td>
<td>1.503</td>
</tr>
<tr>
<td>D3, E3</td>
<td>Lead Dimension</td>
<td>0.400 REF</td>
<td>0.500 REF</td>
<td>0.600 REF</td>
<td>0.800 REF</td>
<td>1.000 REF</td>
<td>1.200 REF</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D4, E4</td>
<td>Foot Radius Location</td>
<td>0.623</td>
<td>0.637</td>
<td>0.723</td>
<td>0.737</td>
<td>0.823</td>
<td>0.837</td>
<td>1.023</td>
<td>1.037</td>
<td>1.223</td>
<td>1.237</td>
<td>1.423</td>
<td>1.437</td>
</tr>
<tr>
<td>L1</td>
<td>Foot Length</td>
<td>0.020</td>
<td>0.030</td>
<td>0.020</td>
<td>0.030</td>
<td>0.020</td>
<td>0.030</td>
<td>0.020</td>
<td>0.030</td>
<td>0.020</td>
<td>0.030</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Issue**: IWS Preliminary 12/12/88

---

### Symbol Description

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
<th>Min</th>
<th>Max</th>
<th>Min</th>
<th>Max</th>
<th>Min</th>
<th>Max</th>
<th>Min</th>
<th>Max</th>
<th>Min</th>
<th>Max</th>
<th>Min</th>
<th>Max</th>
</tr>
</thead>
<tbody>
<tr>
<td>N</td>
<td>Leadcount</td>
<td>68</td>
<td>84</td>
<td>100</td>
<td>132</td>
<td>164</td>
<td>196</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A</td>
<td>Package Height</td>
<td>4.06</td>
<td>4.32</td>
<td>4.06</td>
<td>4.32</td>
<td>4.06</td>
<td>4.32</td>
<td>4.06</td>
<td>4.32</td>
<td>4.06</td>
<td>4.32</td>
<td></td>
<td></td>
</tr>
<tr>
<td>A1</td>
<td>Standoff</td>
<td>0.51</td>
<td>0.76</td>
<td>0.51</td>
<td>0.76</td>
<td>0.51</td>
<td>0.76</td>
<td>0.51</td>
<td>0.76</td>
<td>0.51</td>
<td>0.76</td>
<td></td>
<td></td>
</tr>
<tr>
<td>D, E</td>
<td>Terminal Dimension</td>
<td>17.15</td>
<td>17.40</td>
<td>19.69</td>
<td>19.94</td>
<td>22.23</td>
<td>22.48</td>
<td>27.31</td>
<td>27.56</td>
<td>32.39</td>
<td>32.64</td>
<td>37.47</td>
<td>37.72</td>
</tr>
<tr>
<td>D1, E1</td>
<td>Package Body</td>
<td>13.89</td>
<td>14.05</td>
<td>16.43</td>
<td>16.59</td>
<td>18.97</td>
<td>19.13</td>
<td>24.05</td>
<td>24.21</td>
<td>29.13</td>
<td>29.29</td>
<td>34.21</td>
<td>34.37</td>
</tr>
<tr>
<td>D2, E2</td>
<td>Bumper Distance</td>
<td>17.70</td>
<td>17.85</td>
<td>20.24</td>
<td>20.39</td>
<td>22.78</td>
<td>22.93</td>
<td>27.86</td>
<td>28.01</td>
<td>32.94</td>
<td>33.09</td>
<td>38.02</td>
<td>38.18</td>
</tr>
<tr>
<td>D3, E3</td>
<td>Lead Dimension</td>
<td>10.16 REF</td>
<td>12.70 REF</td>
<td>15.24 REF</td>
<td>15.32 REF</td>
<td>20.32 REF</td>
<td>25.40 REF</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>L1</td>
<td>Foot Length</td>
<td>0.51</td>
<td>0.76</td>
<td>0.51</td>
<td>0.76</td>
<td>0.51</td>
<td>0.76</td>
<td>0.51</td>
<td>0.76</td>
<td>0.51</td>
<td>0.76</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Issue**: IWS Preliminary 12/12/88

---

**Note**: The table above provides dimensions for the Intel Case Outline Drawings for Plastic Quad Flat Pack (PQFP) with a 0.025 inch (0.635mm) pitch. The dimensions are given in both INCH and mm.
Figure 61. Principal Dimensions and Datums

Figure 62. Molded Details
Figure 63. Terminal Details

Figure 64. Typical Lead
REVISION HISTORY

The 82596 LAN Coprocessor data sheet version -003 contains updates and improvements to previous versions.

1. Updated A.C. Timings
2. Added Pin Cross Reference tables.
3. Added Bus Cycle figures.
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>D6</td>
<td>7</td>
<td>11</td>
<td>I/O</td>
</tr>
<tr>
<td>D5</td>
<td>8</td>
<td>12</td>
<td>I/O</td>
</tr>
<tr>
<td>D4</td>
<td>9</td>
<td>13</td>
<td>I/O</td>
</tr>
<tr>
<td>D3</td>
<td>10</td>
<td>14</td>
<td>I/O</td>
</tr>
<tr>
<td>D2</td>
<td>11</td>
<td>18</td>
<td>I/O</td>
</tr>
<tr>
<td>D1</td>
<td>12</td>
<td>19</td>
<td>I/O</td>
</tr>
<tr>
<td>D0</td>
<td>13</td>
<td>20</td>
<td>I/O</td>
</tr>
<tr>
<td>RD</td>
<td>5</td>
<td>9</td>
<td>I</td>
</tr>
<tr>
<td>WR</td>
<td>3</td>
<td>4</td>
<td>I</td>
</tr>
<tr>
<td>CS</td>
<td>2</td>
<td>3</td>
<td>I</td>
</tr>
<tr>
<td>CLK</td>
<td>4</td>
<td>5</td>
<td>I</td>
</tr>
<tr>
<td>RESET</td>
<td>25</td>
<td>40</td>
<td>I</td>
</tr>
<tr>
<td>INT</td>
<td>26</td>
<td>41</td>
<td>O</td>
</tr>
<tr>
<td>DRQ0</td>
<td>17</td>
<td>26</td>
<td>O</td>
</tr>
<tr>
<td>DRQ1</td>
<td>18</td>
<td>27</td>
<td>O</td>
</tr>
<tr>
<td>DACK0</td>
<td>1</td>
<td>2</td>
<td>I</td>
</tr>
<tr>
<td>DACK1</td>
<td>27</td>
<td>42</td>
<td>I</td>
</tr>
</tbody>
</table>

**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.

**READ:** Together with CS, DACK0 or DACK1, Read controls data or status transfers out of the 82588 registers.

**WRITE:** Together with CS, DACK0 or DACK1, Write controls data or command transfers into the 82588 registers.

**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.

**CLOCK:** System clock. TTL compatible signal.

**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.

**INTERRUPT:** Active HIGH signal indicates to the CPU that the 82588 is requesting an interrupt.

**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.

**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.

**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.

**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.
### 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>X1/X2</td>
<td>15/16</td>
<td>24/25</td>
<td>High Integration Mode</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>OSCILLATOR INPUTS: These inputs may be used to connect a quartz crystal that controls the internal clock generator for the serial unit.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>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>High Speed Mode</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>TRANSMIT CLOCK: 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>RECEIVE CLOCK: 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.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>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.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>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>COLLISION DETECT: 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>RECEIVE DATA: This pin receives serial data.</td>
</tr>
<tr>
<td>TxD</td>
<td>20</td>
<td>32</td>
<td>TRANSMIT DATA: 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>REQUEST TO SEND: 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>CLEAR TO SEND: 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>POWER: +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>DIP</td>
<td>PLCC</td>
<td></td>
<td></td>
</tr>
<tr>
<td>NC</td>
<td>6</td>
<td></td>
<td>NO CONNECT: These pins are reserved for future use.</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>
</tbody>
</table>

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.
<table>
<thead>
<tr>
<th>Pin Name</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>CS*</td>
<td>RD</td>
</tr>
<tr>
<td>1</td>
<td>x</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>DACK0[DACK1]*</th>
<th>RD</th>
<th>WR</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</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

Figure 5. Command Format and Operation Values

COMMAND REGISTER

<table>
<thead>
<tr>
<th>COMMANDS</th>
<th>VALUE</th>
<th>COMMANDS</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>
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 Retransmit 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](image)

![Figure 8. Single Buffer Reception](image)
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.

![Diagram of 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: This is a production data sheet. The specifications are subject to change without notice.

*WARNING: Stressing the device beyond the "Absolute Maximum Ratings" may cause permanent damage. These are stress ratings only. Operation beyond the "Operating Conditions" is not recommended and extended exposure beyond the "Operating Conditions" may affect device reliability.

D.C. Characteristics

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

\(TxC, RxC\) have MOS levels (See VMIL, VMIH). All other signals have TTL levels (See VIL, VIH, VOL, VOH).

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Parameter Description</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</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</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

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

System Clock Parameters

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Parameter Description</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>1000</td>
<td>ns</td>
<td>*5</td>
</tr>
<tr>
<td>T2</td>
<td>CLK Low Time</td>
<td>53</td>
<td>1000</td>
<td>ns</td>
<td>*6</td>
</tr>
<tr>
<td>T3</td>
<td>CLK High Time</td>
<td>53</td>
<td>ns</td>
<td></td>
<td>*1</td>
</tr>
<tr>
<td>T4</td>
<td>CLK Rise Time</td>
<td>15</td>
<td>ns</td>
<td></td>
<td>*2</td>
</tr>
<tr>
<td>T5</td>
<td>CLK Fall Time</td>
<td>15</td>
<td>ns</td>
<td></td>
<td></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><strong>Reset Parameters</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<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></td>
<td>T1</td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td><strong>Interrupt Timing Parameters</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<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>
<tr>
<td><strong>Write Parameters</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<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>
<tr>
<td><strong>Read Parameters</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<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>
<tr>
<td><strong>DMA Parameters</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<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
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 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.

Clocks MOS Input Voltage Levels for Timing Measurements

Interrupt Timing (Going Active)
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:

<table>
<thead>
<tr>
<th>For Oscillator Frequency = 1 to 16 MHz (High)</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>External (Fast) Clock Parameters</th>
<th>Min</th>
<th>Max</th>
<th>Units</th>
<th>Test Conditions</th>
</tr>
</thead>
<tbody>
<tr>
<td>T24 Fast Clock Cycle</td>
<td>62.5</td>
<td>ns</td>
<td>*1</td>
<td></td>
</tr>
<tr>
<td>T25 TFC High Time</td>
<td>18.5</td>
<td>1000</td>
<td>ns</td>
<td>*1, *14</td>
</tr>
<tr>
<td>T26 TFC Low Time</td>
<td>23.5</td>
<td>ns</td>
<td>*1</td>
<td></td>
</tr>
<tr>
<td>T27 TFC Rise Time</td>
<td>5</td>
<td>ns</td>
<td>*1</td>
<td></td>
</tr>
<tr>
<td>T28 TFC Fall Time</td>
<td>5</td>
<td>ns</td>
<td>*1</td>
<td></td>
</tr>
</tbody>
</table>

Transmit Clock Parameters

<table>
<thead>
<tr>
<th>Transmit Clock Parameters</th>
<th>Min</th>
<th>Max</th>
<th>Units</th>
<th>Test Conditions</th>
</tr>
</thead>
<tbody>
<tr>
<td>T29 Transmit Clock Cycle</td>
<td>500</td>
<td>ns</td>
<td>*3, *12</td>
<td></td>
</tr>
<tr>
<td>T30 TCLK High Time</td>
<td>*8</td>
<td>1070</td>
<td>ns</td>
<td>*3</td>
</tr>
<tr>
<td>T31 TCLK Low Time</td>
<td>*9</td>
<td></td>
<td>*3</td>
<td></td>
</tr>
<tr>
<td>T32 TCLK Rise Time</td>
<td>15</td>
<td>ns</td>
<td>*3</td>
<td></td>
</tr>
<tr>
<td>T33 TCLK Fall Time</td>
<td>15</td>
<td>ns</td>
<td>*3</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><strong>Transmit Data Parameters (Manchester, Differential Manchester)</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T34</td>
<td>TxD Transition-Transition</td>
<td>4T24-10</td>
<td></td>
<td>ns</td>
<td>*12</td>
</tr>
<tr>
<td>T35</td>
<td>TCLK Low to TxD Transition Half Bit Cell</td>
<td></td>
<td>*10</td>
<td></td>
<td>*2, *12</td>
</tr>
<tr>
<td>T36</td>
<td>TCLK Low to TxD Transition Full Bit Cell</td>
<td></td>
<td>*11</td>
<td></td>
<td>*2, *12</td>
</tr>
<tr>
<td>T37</td>
<td>TxD Rise Time</td>
<td>15</td>
<td></td>
<td>ns</td>
<td>*2</td>
</tr>
<tr>
<td>T38</td>
<td>TxD Fall Time</td>
<td>15</td>
<td></td>
<td>ns</td>
<td>*2</td>
</tr>
<tr>
<td><strong>Transmit Data Parameters (NRZI)</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T39</td>
<td>TxD Transition-Transition</td>
<td>8T24-10</td>
<td></td>
<td>ns</td>
<td>*12</td>
</tr>
<tr>
<td>T40</td>
<td>TCLK Low to TxD Transition</td>
<td></td>
<td>*10</td>
<td></td>
<td>*2, *12</td>
</tr>
<tr>
<td>T41</td>
<td>TxD Rise Time</td>
<td>15</td>
<td></td>
<td>ns</td>
<td>*2</td>
</tr>
<tr>
<td>T42</td>
<td>TxD Fall Time</td>
<td>15</td>
<td></td>
<td>ns</td>
<td>*2</td>
</tr>
<tr>
<td><strong>RTS, CTS, Parameters</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T43</td>
<td>TCLK Low To RTS Low</td>
<td></td>
<td>*10</td>
<td></td>
<td>*3, *12</td>
</tr>
<tr>
<td>T44</td>
<td>CTS Low to TCLK Low CTS Setup Time</td>
<td>65</td>
<td></td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>T45</td>
<td>TCLK low to RTS High</td>
<td></td>
<td>*10</td>
<td></td>
<td>*3, *12</td>
</tr>
<tr>
<td>T46</td>
<td>TCLK Low to CTS Invalid, CTS Hold Time</td>
<td>20</td>
<td></td>
<td>ns</td>
<td>*4, *13</td>
</tr>
<tr>
<td>T47</td>
<td>CTS High to TCLK Low, CTS Setup Time to Stop Transmission</td>
<td>65</td>
<td></td>
<td>ns</td>
<td>*4</td>
</tr>
<tr>
<td><strong>IFS Parameters</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T48</td>
<td>Interframe Delay</td>
<td></td>
<td>*5</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Collision Detect Parameter</strong></td>
<td></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></td>
<td>ns</td>
<td>*13</td>
</tr>
<tr>
<td>T50</td>
<td>CDT High to TCLK Low</td>
<td>50</td>
<td></td>
<td>ns</td>
<td>*13</td>
</tr>
<tr>
<td>T51</td>
<td>TCLK High to CDT Inactive, CDT Hold Time</td>
<td>20</td>
<td></td>
<td>ns</td>
<td>*13</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</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>(Continued)</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>
</tbody>
</table>

|        | **Received Data Parameters**       |      |      |       |                 |
|        | (Manchester)                       |      |      |       |                 |
| T54    | RxD Transition-Transition          | 4T24 | ns   | *12   |                 |

|        | **Received Data Parameters**       |      |      |       |                 |
|        | (Manchester)                       |      |      |       |                 |
| T55    | RxD Rise Time                      | 10   | ns   | *1    |                 |
| T56    | RxD Fall Time                      | 10   | ns   | *1    |                 |

|        | **Received Data Parameters**       |      |      |       |                 |
|        | (NRZI)                             |      |      |       |                 |
| T57    | RxD Transition-Transition          | 8T24 | ns   | *12   |                 |
| T58    | RxD Rise Time                      | 10   | ns   | *1    |                 |
| T59    | RxD Fall Time                      | 10   | ns   | *1    |                 |

**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: \( T_{48} = \text{NIFS} \times T_{29} \) (ns) NIFS—the IFS configuration value.
  If NIFS is less than 12, then it is enforced to 12.
*6—Programmable value:
  \( T_{52} = \text{NCDF} \times T_{29} + (12 \text{ to } 15) \times T_{29} \) (if collision occurs after preamble).
*7—\( T_{53} = 32 \times T_{29} \)
*8—Depends on T24 frequency range:
  High Range: T24 = 10
  Low Range: T25 = 10
*9—\( T_{31} = T_{29} - T_{30} - T_{32} - T_{33} \)
*10—2T24 + 40 ns
*11—6T24 + 40 ns
*12—For \( x \times 16 \) 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](image-url)
Read Timing

DMA Request (Going Active)

DMA Request (Going Inactive)
Transmit Timings: Clocks RTS and CTS

Transmit Timings—Manchester Data Encoding

Transmit Timings—Lost CTS
Transmit Timings—NRZI Data Encoding

Transmit Timings—Lost CTS

Receive Data Timings (Manchester)
Receive Data Timings (NRZI)

Transmit Timings—Interframe Spacing

Transmit Timings—Collision Detect and Jamming
### 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>Transmit/Receive Clock Parameters</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T60</td>
<td>RxCl TxCl Cycle</td>
<td>200</td>
<td>*13</td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>T61</td>
<td>TxCl Rise Time</td>
<td>10</td>
<td></td>
<td>ns</td>
<td>*1</td>
</tr>
<tr>
<td>T62</td>
<td>TxCl Fall Time</td>
<td>10</td>
<td></td>
<td>ns</td>
<td>*1</td>
</tr>
<tr>
<td>T63</td>
<td>TxCl High</td>
<td>80</td>
<td>1000</td>
<td>ns</td>
<td>*1, *3</td>
</tr>
<tr>
<td>T64</td>
<td>TxCl Low</td>
<td>80</td>
<td></td>
<td>ns</td>
<td>*1, *3</td>
</tr>
<tr>
<td>Transmit Data Parameters</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>TxCl Low to TxD Valid</td>
<td>60</td>
<td></td>
<td>ns</td>
<td>*4, *6</td>
</tr>
<tr>
<td>T68</td>
<td>TxCl Low to TxD Transition</td>
<td>60</td>
<td></td>
<td>ns</td>
<td>*2, *4</td>
</tr>
<tr>
<td>T69</td>
<td>TxCl 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>TxCl Low to TxD High (At the Transmission End)</td>
<td>60</td>
<td></td>
<td>ns</td>
<td>*4</td>
</tr>
<tr>
<td>RTS, CTS Parameters</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T72</td>
<td>TxCl, 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 TxCl Low CTS Setup Time</td>
<td>65</td>
<td></td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>T74</td>
<td>TxCl Low to RTS High</td>
<td>60</td>
<td></td>
<td>ns</td>
<td>*5</td>
</tr>
<tr>
<td>T75</td>
<td>TxCl Low to CTS Invalid</td>
<td>20</td>
<td></td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>T75A</td>
<td>CTS High to TxCl Low CTS Set-up Time to Stop Transmission</td>
<td>65</td>
<td></td>
<td>ns</td>
<td>*7</td>
</tr>
<tr>
<td>Interframe Spacing Parameters</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>CRS, CDT, Parameters</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T77</td>
<td>CDT Low to TxCl High External Collision Detect Setup Time</td>
<td>45</td>
<td></td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>T78</td>
<td>TxCl High to CDT Inactive CDT Hold Time</td>
<td>20</td>
<td></td>
<td>ns</td>
<td>*14</td>
</tr>
<tr>
<td>T79</td>
<td>CDT Low to Jamming Start</td>
<td></td>
<td></td>
<td>*10</td>
<td></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 TxCl High Carrier Sense Setup Time</td>
<td>45</td>
<td></td>
<td>ns</td>
<td>*14</td>
</tr>
<tr>
<td>T82</td>
<td>TxCl High to CRS Inactive CRS Hold Time</td>
<td>20</td>
<td></td>
<td>ns</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 Jamming (Internal Collision Detect)</td>
<td></td>
<td>*12</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></td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>T85</td>
<td>RxC High to CRS High. End of Receive Packet</td>
<td>20</td>
<td></td>
<td>ns</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></td>
<td>ns</td>
<td>*1</td>
</tr>
<tr>
<td>T87</td>
<td>RxC Fall Time</td>
<td>10</td>
<td></td>
<td>ns</td>
<td>*1</td>
</tr>
<tr>
<td>T88</td>
<td>RxC High Time</td>
<td>80</td>
<td></td>
<td>ns</td>
<td>*1</td>
</tr>
<tr>
<td>T89</td>
<td>RxC Low Time</td>
<td>80</td>
<td></td>
<td>ns</td>
<td>*1</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></td>
<td>ns</td>
<td>*1</td>
</tr>
<tr>
<td>T91</td>
<td>RxD Hold Time</td>
<td>45</td>
<td></td>
<td>ns</td>
<td>*1</td>
</tr>
<tr>
<td>T92</td>
<td>RxD Rise Time</td>
<td>20</td>
<td></td>
<td>ns</td>
<td>*1</td>
</tr>
<tr>
<td>T93</td>
<td>RxD Fall Time</td>
<td>20</td>
<td></td>
<td>ns</td>
<td>*1</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
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
- 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, 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, CST, 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>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>D2</td>
<td>11</td>
<td>18</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>D1</td>
<td>12</td>
<td>19</td>
<td></td>
<td>DMA REQUEST (CHANNEL 0)—This pin is used by the 82590 to request DMA transfers. 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>D0</td>
<td>13</td>
<td>20</td>
<td></td>
<td>DMA REQUEST (CHANNEL 0)—This pin is used by the 82590 to request DMA transfers. 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>RD</td>
<td>5</td>
<td>9</td>
<td>I</td>
<td>READ—Together with CS0, CS1, DACK0, or DACK1, Read controls data or status transfers out of the 82590.</td>
</tr>
<tr>
<td>WR</td>
<td>3</td>
<td>4</td>
<td>I</td>
<td>WRITE—Together with CS0, CST, DACK0, or DACK1, Write controls data or command transfers into the 82590.</td>
</tr>
<tr>
<td>CS0</td>
<td>2</td>
<td>3</td>
<td>I</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>RESET</td>
<td>25</td>
<td>40</td>
<td>I</td>
<td>DMA REQUEST (CHANNEL 0)—This pin is used by the 82590 to request DMA transfers. 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>INT</td>
<td>26</td>
<td>41</td>
<td>O</td>
<td>INTERRUPT—A HIGH signal on this pin notifies the CPU that the 82590 is requesting an interrupt.</td>
</tr>
<tr>
<td>DRQ0</td>
<td>17</td>
<td>26</td>
<td>O</td>
<td>DMA REQUEST (CHANNEL 0)—This pin is used by the 82590 to request DMA transfers. 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>DRQ1</td>
<td>18</td>
<td>27</td>
<td>O</td>
<td>DMA REQUEST (CHANNEL 0)—This pin is used by the 82590 to request DMA transfers. 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>DACK0</td>
<td>1</td>
<td>2</td>
<td>I</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. DMA ACKNOWLEDGE (CHANNELS 0 AND 1)—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>
</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</td>
<td>27</td>
<td>42</td>
<td>I/O</td>
<td>This is a multifunction, bidirectional pin which can be programmed to DACK1 or CS1/EOP during configuration. When it is configured for EOP, it provides an open-drain output.</td>
</tr>
<tr>
<td>CS1/EOP</td>
<td></td>
<td></td>
<td>I/O</td>
<td>DMA ACKNOWLEDGE (CHANNEL 1)—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></td>
<td>CHIP SELECT (PORT 1)—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>END OF PROCESS—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/O</td>
<td>CLOCK—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></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 FREQUENCY. 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></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>CLKSRC</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Type</td>
</tr>
<tr>
<td></td>
<td>0 (LOW)</td>
<td>Don’t Care</td>
<td>I</td>
<td>Clock</td>
</tr>
<tr>
<td></td>
<td>1 (HIGH)</td>
<td>1</td>
<td>O</td>
<td>Internal Parallel Subsystem Clock Divided by Two</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>0</td>
<td>O</td>
<td>Internal Parallel Subsystem Clock</td>
</tr>
<tr>
<td>CLKSRC</td>
<td>NA</td>
<td>6</td>
<td>I</td>
<td>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.</td>
</tr>
<tr>
<td>FREQUENCY</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 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.</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>X1/X2</td>
<td>15/16</td>
<td>24/25</td>
<td>I</td>
<td>High Integration Mode</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>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>15</td>
<td>24</td>
<td>I</td>
<td>High Speed Mode</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>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>16</td>
<td>25</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>24</td>
<td>36</td>
<td>I</td>
<td>CARRIER SENSE—in High Speed Mode this pin is Carrier Sense, CRS, and is used to notify the 82590 that the serial link is active.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>O</td>
<td>TRANSMIT CLOCK—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>COLLISION DETECT—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>RECEIVE DATA—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>TRANSMIT DATA—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>REQUEST TO SEND—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>CLEAR TO SEND—An active LOW signal which enables the 82590 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>28</td>
<td>1, 43, 44</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 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×, 10×, 16×, or 18× 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])
Figure 5. Port 0 Commands

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

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

**RESULT 1**

**RESULT 2**

**RECEIVE BYTE COUNT (LOW)/FRAME COUNTER**

**RECEIVE BYTE COUNT (HIGH)**

<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>RCV CHNL</td>
<td>RCV STATE</td>
<td>BUF. CHAIN'G NO. OF BUF.</td>
<td>EXEC CHNL</td>
<td>EXEC STATE</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Status Registers—4 Bytes**

(82588 Compatible Modes)

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

**RESULT 1**

**RESULT 2**

<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>RCV CHNL</td>
<td>RCV STATE</td>
<td>BUF. CHAIN'G NO. OF BUF.</td>
<td>EXEC CHNL</td>
<td>EXEC STATE</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Events**

<table>
<thead>
<tr>
<th>Event</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>
</tbody>
</table>

**Value (Status 0)**

- Diagnose-Passed: 7
- End-Of-Frame: 8
- Request-Next-Buffer: 9
- Reception-Aborted: 10
- Retransmit-Done: 12
- Execution-Aborted: 13
- Diagnose-Failed: 15

*Available only after Hardware or Software Reset*

**Figure 7. Port 0 Status Registers**

**Timer/Counter Events**

<table>
<thead>
<tr>
<th>Value* (Status 0)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(T/C = 1)</td>
</tr>
<tr>
<td>Timer Expired</td>
</tr>
<tr>
<td>Counter 1 Expired</td>
</tr>
<tr>
<td>Counter 2 Expired</td>
</tr>
<tr>
<td>Counter 3 Expired</td>
</tr>
</tbody>
</table>

**General Purpose Event**

<table>
<thead>
<tr>
<th>Value* (Status 0)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(GP = 1)</td>
</tr>
<tr>
<td>REM-PWR-UP</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**
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.
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 jabber inhibit function.

Figure 13. Demultiplexing DACK/CS1/EOP Pin

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>
 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.

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
Figure 15. 82590/82560 High-Integration Adapter
ABSOLUTE MAXIMUM RATINGS*

<table>
<thead>
<tr>
<th>Case Temperature (Tc) Under Bias</th>
</tr>
</thead>
<tbody>
<tr>
<td>Plastic ..........................</td>
</tr>
<tr>
<td>PLCC .............................</td>
</tr>
<tr>
<td>Storage Temperature ...............</td>
</tr>
<tr>
<td>Voltage on any Pin with Respect to Ground ..........</td>
</tr>
<tr>
<td>Power Dissipation ................</td>
</tr>
</tbody>
</table>

NOTICE: This is a production data sheet. The specifications are subject to change without notice.

WARNING: Stressing the device beyond the "Absolute Maximum Ratings" may cause permanent damage. These are stress ratings only. Operation beyond the "Operating Conditions" is not recommended and extended exposure beyond the "Operating Conditions" may affect device reliability.

D.C. CHARACTERISTICS

(Tc [Plastic] = 0°C to +85°C, Tc [PLCC] = 0°C to +85°C, Vcc = +5V ±10%)

TxC and RxC have 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>
</tr>
</thead>
<tbody>
<tr>
<td>VIL</td>
<td>Input Low Voltage (TTL)</td>
<td>-0.5</td>
<td>0.8</td>
<td>V</td>
</tr>
<tr>
<td>VIH</td>
<td>Input High Voltage (TTL)</td>
<td>2.0</td>
<td>Vcc + 0.5</td>
<td>V</td>
</tr>
<tr>
<td>VOL</td>
<td>Output Low Voltage (TTL)</td>
<td>0.45</td>
<td>V</td>
<td></td>
</tr>
<tr>
<td>VOH</td>
<td>Output High Voltage (TTL)</td>
<td>2.4</td>
<td>V</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>
</tr>
<tr>
<td>VMIH</td>
<td>Input High Voltage (MOS)</td>
<td>3.9</td>
<td>Vcc + 0.5</td>
<td>V</td>
</tr>
<tr>
<td>IIL</td>
<td>Input Leakage Current</td>
<td>±10</td>
<td>μA</td>
<td></td>
</tr>
<tr>
<td>ILIO</td>
<td>I/O Leakage Current</td>
<td>±10</td>
<td>μA</td>
<td></td>
</tr>
<tr>
<td>CIN</td>
<td>Capacitance of Input Buffer</td>
<td>10</td>
<td>pF</td>
<td></td>
</tr>
<tr>
<td>CO</td>
<td>Capacitance of I/O Buffer</td>
<td>20</td>
<td>pF</td>
<td></td>
</tr>
<tr>
<td>ICC</td>
<td>Power Supply Current</td>
<td>80</td>
<td>mA</td>
<td></td>
</tr>
</tbody>
</table>

NOTES:
1. System side (CLK) at 16 MHz, serial side (TxC) at 20 MHz in High-Speed Mode.
2. Remote power-up mode.
3. Power-down mode—all inputs connected to Vcc level.
4. General formulas for current are: (a) f(CLK) × 2.9 + f(TxC) × 1.8 for High Speed Mode. (b) f(CLK) × 2.9 + f(TFC) × 0.2 + f(TFC) × 1.8/SR, where SR is the sampling rate in High-Integration Mode, and f = frequency in MHz.
### A.C. CHARACTERISTICS

(C\textsubscript{L} on all outputs is 20 pF–125 pF unless otherwise specified.)

<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\textsubscript{1}</td>
<td>CLK Cycle Time</td>
<td>62.5</td>
<td></td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>t\textsubscript{2}</td>
<td>CLK Low Time</td>
<td>27</td>
<td></td>
<td>ns</td>
<td>(Note 5)</td>
</tr>
<tr>
<td>t\textsubscript{3}</td>
<td>CLK High Time</td>
<td>27</td>
<td></td>
<td>ns</td>
<td>(Note 5)</td>
</tr>
<tr>
<td>t\textsubscript{4}</td>
<td>CLK Rise Time</td>
<td>5</td>
<td></td>
<td>ns</td>
<td>(Note 1)</td>
</tr>
<tr>
<td>t\textsubscript{5}</td>
<td>CLK Fall Time</td>
<td>5</td>
<td></td>
<td>ns</td>
<td>(Note 2)</td>
</tr>
</tbody>
</table>

**SYSTEM CLOCK INPUT 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>t\textsubscript{98}</td>
<td>CLK Cycle Time</td>
<td>120</td>
<td></td>
<td>ns</td>
<td>(Notes 4, 7)</td>
</tr>
<tr>
<td>t\textsubscript{99}</td>
<td>CLK Low Time</td>
<td>50</td>
<td></td>
<td>ns</td>
<td>(Note 4)</td>
</tr>
<tr>
<td>t\textsubscript{100}</td>
<td>CLK High Time</td>
<td>50</td>
<td></td>
<td>ns</td>
<td>(Note 4)</td>
</tr>
<tr>
<td>t\textsubscript{101}</td>
<td>CLK Rise Time</td>
<td>15</td>
<td></td>
<td>ns</td>
<td>(Notes 1, 4)</td>
</tr>
<tr>
<td>t\textsubscript{102}</td>
<td>CLK Fall Time</td>
<td>15</td>
<td></td>
<td>ns</td>
<td>(Notes 2, 4)</td>
</tr>
</tbody>
</table>

**SYSTEM CLOCK OUTPUT 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>t\textsubscript{6}</td>
<td>Reset Active to Clock Low</td>
<td>20</td>
<td></td>
<td>ns</td>
<td>(Note 3)</td>
</tr>
<tr>
<td>t\textsubscript{8}</td>
<td>Reset Pulse Width</td>
<td>4t\textsubscript{1}</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>t\textsubscript{9}</td>
<td>Control Inactive after Reset</td>
<td>2t\textsubscript{1}</td>
<td></td>
<td>ns</td>
<td></td>
</tr>
</tbody>
</table>

**RESET 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>t\textsubscript{10}</td>
<td>CLK High to Interrupt Active</td>
<td>55</td>
<td></td>
<td>ns</td>
<td>(Note 4)</td>
</tr>
<tr>
<td>t\textsubscript{11}</td>
<td>WR Inactive to Interrupt</td>
<td>55</td>
<td></td>
<td>ns</td>
<td>(Note 4)</td>
</tr>
<tr>
<td>t\textsubscript{103}</td>
<td>Int Low to Int High Gap</td>
<td>2t\textsubscript{1}</td>
<td></td>
<td>ns</td>
<td>(Note 4)</td>
</tr>
</tbody>
</table>

**INTERRUPT 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>t\textsubscript{12}</td>
<td>CS\textsubscript{0}, CS\textsubscript{1}, DACK\textsubscript{0}, or DACK\textsubscript{1} Setup to WR Low</td>
<td>0</td>
<td></td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>t\textsubscript{13}</td>
<td>WR Pulse Width</td>
<td>55</td>
<td></td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>t\textsubscript{14}</td>
<td>CS\textsubscript{0}, CS\textsubscript{1}, DACK\textsubscript{0}, or DACK\textsubscript{1} Hold after WR High</td>
<td>0</td>
<td></td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>t\textsubscript{15}</td>
<td>Data Setup to WR High</td>
<td>30</td>
<td></td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>t\textsubscript{16}</td>
<td>Data Hold after WR High</td>
<td>3</td>
<td></td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>t\textsubscript{94}</td>
<td>Write Cycle Time</td>
<td>2t\textsubscript{1}</td>
<td></td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>t\textsubscript{96}</td>
<td>WR Inactive Time</td>
<td>55</td>
<td></td>
<td>ns</td>
<td></td>
</tr>
</tbody>
</table>

**WRITE PARAMETERS**

**NOTE:**
To achieve socket compatibility with the Intel 82588, the rise and fall time specifications of the Intel 82588 can be applied for the 82590 clock inputs (CLK, TXC, RXC, TFC). This is valid only for standard 82588 operating frequencies. Refer to the Intel 82588 data sheet for these specifications.
### A.C. CHARACTERISTICS

(C\textsubscript{L} on all outputs is 20 pF–125 pF unless otherwise specified.) (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>( t_{17} )</td>
<td>CS0, CST, DACK0, or DACK1 Setup to RD Low</td>
<td>0 ns</td>
<td>55 ns</td>
<td></td>
<td></td>
</tr>
<tr>
<td>( t_{18} )</td>
<td>RD Pulse Width</td>
<td>0 ns</td>
<td>45 ns</td>
<td></td>
<td></td>
</tr>
<tr>
<td>( t_{19} )</td>
<td>CS0, CST DACK0, or DACK1 Hold after RD High</td>
<td>0 ns</td>
<td>45 ns</td>
<td></td>
<td></td>
</tr>
<tr>
<td>( t_{20} )</td>
<td>RD Low to Data Valid</td>
<td>0 ns</td>
<td>45 ns</td>
<td></td>
<td></td>
</tr>
<tr>
<td>( t_{21} )</td>
<td>Data Float after RD High</td>
<td>0 ns</td>
<td>45 ns</td>
<td></td>
<td></td>
</tr>
<tr>
<td>( t_{95} )</td>
<td>Read Cycle Time</td>
<td>2( t_{1} ) ns</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( t_{97} )</td>
<td>RD Inactive Time</td>
<td>55 ns</td>
<td></td>
<td></td>
<td></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>( t_{22} )</td>
<td>CLK Low to DRQ0 or DRQ1 Active</td>
<td>55 ns</td>
<td></td>
<td></td>
<td>(Note 4)</td>
</tr>
<tr>
<td>( t_{23} )</td>
<td>WR or RD Low to DRQ0 or DRQ1 Inactive</td>
<td>45 ns</td>
<td></td>
<td></td>
<td>(Note 4)</td>
</tr>
<tr>
<td>( t_{104} )</td>
<td>WR or RD High to DRQ0 or DRQ1 Inactive—82560 Retransmit or Receive Buffer Reclaim</td>
<td>2.5 ns</td>
<td>65 ns</td>
<td></td>
<td>(Note 4)</td>
</tr>
<tr>
<td>( t_{105} )</td>
<td>WR or RD Low to EOP Active</td>
<td>45 ns</td>
<td></td>
<td></td>
<td>(Note 6)</td>
</tr>
<tr>
<td>( t_{106} )</td>
<td>EOP Float after DACK0 or DACK1 Going Inactive</td>
<td>40 ns</td>
<td></td>
<td></td>
<td>(Note 6)</td>
</tr>
</tbody>
</table>

#### NOTES:

1. 0.8V to 2.0V.
2. 2.0V to 0.8V.
3. To guarantee recognition on the next clock.
4. C\textsubscript{L} = 50 pF.
5. Measured at 1.5V.
6. Open drain I/O pin.
7. None of the A.C. Parameters are related to the CLK output pin.

#### A.C. TESTING INPUT/OUTPUT WAVEFORMS

**TTL Input/Output Voltage Levels for Timing Measurements**

<table>
<thead>
<tr>
<th>Voltage Level</th>
<th>Test Points</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.4V</td>
<td>TEST POINTS</td>
</tr>
<tr>
<td>1.5V</td>
<td>1.5V</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”. Rise and fall times are measured between 0.8V and 2.0V.

**System Clock I/O Voltage Levels for Timing Measurements**

- \( T_{4}, T_{101} \)
- \( T_{5}, T_{102} \)
- \( T_{4}, T_{101} \)
- \( T_{5}, T_{102} \)
- \( T_{1}, T_{98} \)
- \( T_{2}, T_{99} \)
- \( T_{3}, T_{100} \)

290147–18
Reset Timing

Interrupt Timing (Going Active)

Interrupt Timing (Going Inactive)
WRITE TIMING

NOTE:
DACK0 or DACK1 may remain active continuously for consecutive DMA cycles.

READ TIMING

NOTE:
DACK0 or DACK1 may remain active continuously for consecutive DMA cycles.
DMA Request (Going Active)

CLK

DRQ0 DRQ1

T22

DMA Request (Going Inactive)

DRQ0 DRQ1

DACK0 DACK1

WR RD

T23

Tightly Coupled Interface

DRQ0, DRQ1

DACK0 DACK1

WR, RD

EOP

T23

T104

T105

T106
**SERIAL INTERFACE A.C. TIMING CHARACTERISTICS**

TFC is the Crystal or Serial Clock Input at X1.

<table>
<thead>
<tr>
<th></th>
<th>X8 Sampling</th>
<th>X10 Sampling</th>
<th>X16 Sampling</th>
<th>X18 Sampling</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>High Integration Mode</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>For TFC Frequency = 1 MHz to 32 MHz (High)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TCLK Frequency</td>
<td>0.125 MHz–4 MHz</td>
<td>100 kHz–3.2 MHz</td>
<td>62.5 kHz–2 MHz</td>
<td>55.6 kHz–1.78 MHz</td>
</tr>
<tr>
<td>(t_{29} = TCLK)</td>
<td>(8 \times t_{24})</td>
<td>(10 \times t_{24})</td>
<td>(16 \times t_{24})</td>
<td>(18 \times t_{24})</td>
</tr>
<tr>
<td>Cycle Time</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(t_{30} = TCLK)</td>
<td>(t_{24}) (Typically)</td>
<td>(t_{24}) (Typically)</td>
<td>(t_{24}) (Typically)</td>
<td>(t_{24}) (Typically)</td>
</tr>
<tr>
<td>High Time</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(t_{31} = TCLK)</td>
<td>(7 \times t_{24}) (Typically)</td>
<td>(9 \times t_{24}) (Typically)</td>
<td>(15 \times t_{24}) (Typically)</td>
<td>(17 \times t_{24}) (Typically)</td>
</tr>
<tr>
<td>Low Time</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

For TFC Frequency = 0 MHz to 1 MHz (Low)

<table>
<thead>
<tr>
<th>TCLK Frequency</th>
<th>0–0.125 MHz</th>
<th>0–100 kHz</th>
<th>0–62.5 kHz</th>
<th>0–55.6 kHz</th>
</tr>
</thead>
<tbody>
<tr>
<td>(t_{29} = TCLK)</td>
<td>(8 \times t_{24})</td>
<td>(10 \times t_{24})</td>
<td>(16 \times t_{24})</td>
<td>(18 \times t_{24})</td>
</tr>
<tr>
<td>Cycle Time</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(t_{30} = TCLK)</td>
<td>(t_{25}) (Typically)</td>
<td>(t_{25}) (Typically) &amp; (t_{25}) (Typically)</td>
<td>(t_{25}) (Typically)</td>
<td></td>
</tr>
<tr>
<td>High Time</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(t_{31} = TCLK)</td>
<td>(7 \times t_{24} + t_{26}) (Typically)</td>
<td>(9 \times t_{24} + t_{26}) (Typically)</td>
<td>(15 \times t_{24} + t_{26}) (Typically)</td>
<td>(17 \times t_{24} + t_{26}) (Typically)</td>
</tr>
<tr>
<td>Low Time</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**NOTES:**

X10 and X18 are available only for Manchester or Differential Manchester encoding/decoding.

\(t_{24}\) = Serial Clock Cycle Time

\(t_{25}\) = Serial Clock High Time

\(t_{26}\) = Serial Clock Low Time

**HIGH SPEED MODE**

- Applies for TxC, RxC
- \(f_{\text{max}} = 20\text{ MHz} \pm 100\text{ ppm}\)
- For Manchester, symmetry is required: \(t_{64}, t_{64} = \frac{1}{2f} \pm 5\%\)
### 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></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

#### EXTERNAL (FAST) CLOCK PARAMETERS

<p>| | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>t24</td>
<td>Fast Clock (TFC) Cycle Time</td>
<td>31.25</td>
<td>ns</td>
<td>(Notes 1, 16)</td>
<td></td>
</tr>
<tr>
<td>t25</td>
<td>TFC High Time</td>
<td>(Note 13)</td>
<td>ns</td>
<td>(Notes 1, 7)</td>
<td></td>
</tr>
<tr>
<td>t26</td>
<td>TFC Low Time</td>
<td>12</td>
<td>ns</td>
<td>(Notes 1, 17)</td>
<td></td>
</tr>
<tr>
<td>t27</td>
<td>TFC Rise Time</td>
<td>3</td>
<td>ns</td>
<td>(Note 1)</td>
<td></td>
</tr>
<tr>
<td>t28</td>
<td>TFC Fall Time</td>
<td>3</td>
<td>ns</td>
<td>(Note 1)</td>
<td></td>
</tr>
</tbody>
</table>

#### TRANSMIT CLOCK PARAMETERS

<p>| | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>t29</td>
<td>Transmit Clock (TCLK) Cycle Time</td>
<td>(Note 13)</td>
<td>ns</td>
<td>(Note 2)</td>
<td></td>
</tr>
<tr>
<td>t30</td>
<td>TCLK High Time</td>
<td>(Note 7)</td>
<td>ns</td>
<td>(Note 2)</td>
<td></td>
</tr>
<tr>
<td>t31</td>
<td>TCLK Low Time</td>
<td>(Note 8)</td>
<td>ns</td>
<td>(Note 2)</td>
<td></td>
</tr>
<tr>
<td>t32</td>
<td>TCLK Rise Time</td>
<td>10</td>
<td>ns</td>
<td>(Note 2)</td>
<td></td>
</tr>
<tr>
<td>t33</td>
<td>TCLK Fall Time</td>
<td>10</td>
<td>ns</td>
<td>(Note 2)</td>
<td></td>
</tr>
</tbody>
</table>

#### TRANSMIT DATA PARAMETERS (MANCHESTER, DIFFERENTIAL MANCHESTER)

<p>| | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>t34</td>
<td>TxD Transition-Transition</td>
<td>(Note 14)</td>
<td>ns</td>
<td></td>
<td></td>
</tr>
<tr>
<td>t35</td>
<td>TCLK Low to TxD Mid Bit Cell Transition</td>
<td>(Note 10)</td>
<td>ns</td>
<td>(Note 2)</td>
<td></td>
</tr>
<tr>
<td>t36</td>
<td>TCLK Low to TxD Bit Cell Boundary Transition</td>
<td>(Note 9)</td>
<td>ns</td>
<td>(Note 2)</td>
<td></td>
</tr>
<tr>
<td>t37</td>
<td>TxD Rise Time</td>
<td>10</td>
<td>ns</td>
<td>(Note 2)</td>
<td></td>
</tr>
<tr>
<td>t38</td>
<td>TxD Fall Time</td>
<td>10</td>
<td>ns</td>
<td>(Note 2)</td>
<td></td>
</tr>
</tbody>
</table>

#### TRANSMIT DATA PARAMETERS (NRZI)

<p>| | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>t39</td>
<td>TxD Transition-Transition</td>
<td>(Note 15)</td>
<td>ns</td>
<td></td>
<td></td>
</tr>
<tr>
<td>t40</td>
<td>TCLK Low to TxD Transition</td>
<td>(Note 9)</td>
<td>ns</td>
<td>(Note 2)</td>
<td></td>
</tr>
<tr>
<td>t41</td>
<td>TxD Rise Time</td>
<td>10</td>
<td>ns</td>
<td>(Note 2)</td>
<td></td>
</tr>
<tr>
<td>t42</td>
<td>TxD Fall Time</td>
<td>10</td>
<td>ns</td>
<td>(Note 2)</td>
<td></td>
</tr>
</tbody>
</table>

#### RTS, CTS PARAMETERS

<p>| | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>t43</td>
<td>TCLK Low to RTS Low</td>
<td>(Note 9)</td>
<td>ns</td>
<td>(Note 2)</td>
<td></td>
</tr>
<tr>
<td>t44</td>
<td>CTS Low to TCLK Low</td>
<td>35</td>
<td>ns</td>
<td></td>
<td></td>
</tr>
<tr>
<td>t45</td>
<td>TCLK Low to RTS High</td>
<td>(Note 9)</td>
<td>ns</td>
<td>(Note 2)</td>
<td></td>
</tr>
<tr>
<td>t46</td>
<td>TCLK Low to CTS Invalid CTS Hold Time</td>
<td>10</td>
<td>ns</td>
<td>(Notes 3, 12)</td>
<td></td>
</tr>
<tr>
<td>t47</td>
<td>CTS High to TCLK Low; CTS Setup Time to Stop Transmission</td>
<td>35</td>
<td>ns</td>
<td>(Note 3)</td>
<td></td>
</tr>
</tbody>
</table>

#### IFS PARAMETERS

<p>| | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>t48</td>
<td>Interframe Delay</td>
<td>(Note 4)</td>
<td>ns</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

1-236
## 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>(t_{49})</td>
<td>CDT Low to TCLK High External Collision Detect Setup Time</td>
<td>35 ns</td>
<td></td>
<td>ns</td>
<td>(Note 12)</td>
</tr>
<tr>
<td>(t_{50})</td>
<td>CDT High to TCLK High</td>
<td>35 ns</td>
<td></td>
<td>ns</td>
<td>(Note 12)</td>
</tr>
<tr>
<td>(t_{51})</td>
<td>TCLK High to CDT Inactive CDT Hold Time</td>
<td>10 ns</td>
<td></td>
<td>ns</td>
<td>(Note 12)</td>
</tr>
<tr>
<td>(t_{52})</td>
<td>CDT Low to Jamming Start</td>
<td></td>
<td>(Note 5)</td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>(t_{53})</td>
<td>Jamming Period</td>
<td></td>
<td></td>
<td>ns</td>
<td></td>
</tr>
</tbody>
</table>

### RECEIVED DATA PARAMETERS (MANCHESTER, DIFFERENTIAL MANCHESTER)

<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_{54})</td>
<td>RxD Transition-Transition</td>
<td>115 ns</td>
<td></td>
<td>ns</td>
<td>(Note 14)</td>
</tr>
<tr>
<td>(t_{55})</td>
<td>RxD Rise Time</td>
<td>10 ns</td>
<td></td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>(t_{56})</td>
<td>RxD Fall Time</td>
<td>10 ns</td>
<td></td>
<td>ns</td>
<td></td>
</tr>
</tbody>
</table>

### RECEIVED DATA PARAMETERS (NRZI)

<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_{57})</td>
<td>RxD Transition-Transition</td>
<td>240 ns</td>
<td></td>
<td>ns</td>
<td>(Note 11)</td>
</tr>
<tr>
<td>(t_{58})</td>
<td>RxD Rise Time</td>
<td>10 ns</td>
<td></td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>(t_{59})</td>
<td>RxD Fall Time</td>
<td>10 ns</td>
<td></td>
<td>ns</td>
<td></td>
</tr>
</tbody>
</table>

### EXTERNAL LOOPBACK 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>(t_{107})</td>
<td>TCLK Low to LPBK High</td>
<td>50 ns</td>
<td></td>
<td>ns</td>
<td>(Note 2)</td>
</tr>
<tr>
<td>(t_{108})</td>
<td>TCLK Low to LPBK Float</td>
<td>50 ns</td>
<td></td>
<td>ns</td>
<td>(Note 2)</td>
</tr>
</tbody>
</table>

### NOTES:

1. MOS Levels
2. 1 TTL Load + 50 pF.
3. Abnormal End of Transmission: CTS expires before RTS.
4. Programmable value: \(t_{28} = N_{IFS} \times t_{29}\) (ns)
   - If \(N_{IFS}\) is less than 12 then it is enforced to 12.
5. \(t_{52} = N_{CDF} \times t_{29} + (12 \text{ to } 15) \times t_{29}\) (if collision occurs after preamble).
6. \(t_{53} = 32 \times t_{29}\)
7. Depends on frequency range:
   - High Range: \(t_{34} = 10\) ns
   - Low Range: \(t_{35} = 10\) ns
8. \(t_{51} = t_{30} - t_{32} - t_{33}\)
9. \(2 \times t_{24} + 40\) ns for 8X or 10X
   - \(4 \times t_{24} + 40\) ns for 16X
10. \(6 \times t_{24} + 40\) ns for 8X
   - \(12 \times t_{24} + 40\) ns for 16X
   - \(7 \times t_{24} + 40\) ns for 10X
   - \(13 \times t_{24} + 40\) ns for 18X
11. \(8 \times t_{24} - 10\) ns for 8X
   - \(10 \times t_{24} - 10\) ns for 10X
   - \(16 \times t_{24} - 10\) ns for 16X
   - \(18 \times t_{24} - 10\) ns for 18X
12. To Guarantee recognition on the next clock.
13. \(t_{29} = 10\) ns for High Range
   - \(30\) ns for Low Range
14. \(4 \times t_{24} - 10\) ns for 8X
   - \(5 \times t_{24} - 10\) ns for 10X
   - \(8 \times t_{24} - 10\) ns for 16X
   - \(9 \times t_{24} - 10\) ns for 18X
15. \(10 \times t_{24} - 10\) ns for 16X
16. \(10 \times t_{24} - 10\) ns for 18X
17. Maximum capacitance load on the X2 pin when an external MOS clock is connected to X1:
   - 15 pF for DC to 16 MHz
   - 5 pF for 16 MHz to 32 MHz

---

1-237
NOTES:
1. High-quality, parallel resonant, fundamental-mode crystals are recommended for maximum accuracy.
2. C1, C2, and stray capacitance of the board should be adjusted so the total capacitance load on the crystal is approx. 15 pF.
3. For IEEE 802.3 applications, the crystal must be accurate to ±35 PPM over a range of 0°C to 70°C.
RECEIVE DATA TIMINGS (NRZI)

TRANSMIT TIMINGS—INTERFRAME SPACING

TRANSMIT TIMINGS—COLLISION DETECT AND JAMMING

LOOPBACK OUTPUT SIGNAL TIMINGS
## 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></td>
<td><strong>TRANSMIT/RECEIVE CLOCK PARAMETERS</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$t_{60}$</td>
<td>RxC, TxC Cycle Time</td>
<td>50</td>
<td></td>
<td>ns</td>
<td>(Notes 1, 3)</td>
</tr>
<tr>
<td>$t_{61}$</td>
<td>TxC Rise Time</td>
<td></td>
<td>5</td>
<td>ns</td>
<td>(Note 1)</td>
</tr>
<tr>
<td>$t_{62}$</td>
<td>TxC Fall Time</td>
<td></td>
<td>5</td>
<td>ns</td>
<td>(Note 1)</td>
</tr>
<tr>
<td>$t_{63}$</td>
<td>TxC High Time</td>
<td></td>
<td>18</td>
<td>ns</td>
<td>(Notes 1, 3)</td>
</tr>
<tr>
<td>$t_{64}$</td>
<td>TxC Low Time</td>
<td></td>
<td>19</td>
<td>ns</td>
<td>(Notes 1, 3)</td>
</tr>
<tr>
<td></td>
<td><strong>TRANSMIT DATA PARAMETERS</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$t_{65}$</td>
<td>TxD Rise Time</td>
<td></td>
<td>10</td>
<td>ns</td>
<td>(Note 4)</td>
</tr>
<tr>
<td>$t_{66}$</td>
<td>TxD Fall Time</td>
<td></td>
<td>10</td>
<td>ns</td>
<td>(Note 4)</td>
</tr>
<tr>
<td>$t_{67}$</td>
<td>TxC Low to TxD Valid</td>
<td></td>
<td>30</td>
<td>ns</td>
<td>(Notes 4, 5)</td>
</tr>
<tr>
<td>$t_{68}$</td>
<td>TxC Low to TxD Transition</td>
<td></td>
<td>30</td>
<td>ns</td>
<td>(Notes 2, 4)</td>
</tr>
<tr>
<td>$t_{69}$</td>
<td>TxC High to TxD Transition</td>
<td></td>
<td>30</td>
<td>ns</td>
<td>(Notes 2, 4)</td>
</tr>
<tr>
<td>$t_{70}$</td>
<td>TxD Transition-Transition</td>
<td></td>
<td>20</td>
<td>ns</td>
<td>(Notes 2, 4)</td>
</tr>
<tr>
<td>$t_{71}$</td>
<td>TxC Low to TxD High (At the Transmission End)</td>
<td></td>
<td>30</td>
<td>ns</td>
<td>(Note 4)</td>
</tr>
<tr>
<td></td>
<td><strong>RTS, CTS PARAMETERS</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$t_{72}$</td>
<td>TxC Low to RTS Low Time to Activate RTS</td>
<td></td>
<td>30</td>
<td>ns</td>
<td>(Note 4)</td>
</tr>
<tr>
<td>$t_{73}$</td>
<td>CTS Low to TxC Low CTS Setup Time</td>
<td></td>
<td>20</td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>$t_{74}$</td>
<td>TxC Low to RTS High</td>
<td></td>
<td>30</td>
<td>ns</td>
<td>(Note 4)</td>
</tr>
<tr>
<td>$t_{75}$</td>
<td>TxC Low to CTS Invalid. CTS Hold Time</td>
<td></td>
<td>10</td>
<td>ns</td>
<td>(Note 6)</td>
</tr>
<tr>
<td>$t_{75a}$</td>
<td>CTS High to TxC Low. CTS Setup Time to Stop Transmission</td>
<td></td>
<td>20</td>
<td>ns</td>
<td>(Note 6)</td>
</tr>
<tr>
<td></td>
<td><strong>INTERFRAME SPACING PARAMETER</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$t_{76}$</td>
<td>Inter Frame Delay</td>
<td></td>
<td>(Note 8)</td>
<td>ns</td>
<td></td>
</tr>
</tbody>
</table>

---

Notes:
1. Parameter values depend on the output driver type.
2. Parameter values depend on the input driver type.
3. Parameter values may vary with the output driver type and input driver type.
4. Parameter values depend on the output driver type and input driver type.
5. Parameter values depend on the input driver type and output driver type.
6. Parameter values depend on the input driver type and output driver type.
7. Parameter values may vary with the input driver type and output driver type.
8. Parameter values may vary with the input driver type and output driver type.
## 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>t77</td>
<td>CDT Low to TxC High; External Collision Detect Setup Time</td>
<td>20</td>
<td></td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>t78</td>
<td>TxC High to CDT Inactive; CDT Hold Time</td>
<td>10</td>
<td></td>
<td>ns</td>
<td>(Note 12)</td>
</tr>
<tr>
<td>t79</td>
<td>CDT Low to Jam Start</td>
<td>(Note 9)</td>
<td></td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>t80</td>
<td>Jamming Period</td>
<td>(Note 10)</td>
<td></td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>t81</td>
<td>CRS Low to TxC High; Carrier Sense Setup Time</td>
<td>25</td>
<td></td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>t82</td>
<td>TxC High to CRS Inactive; CRS Hold Time</td>
<td>10</td>
<td></td>
<td>ns</td>
<td>(Note 12)</td>
</tr>
<tr>
<td>t83</td>
<td>CRS High to Jamming Start (Internal Collision Detect)</td>
<td>(Note 11)</td>
<td></td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>t84</td>
<td>CRS High to RxC High; CRS Inactive Setup Time</td>
<td>30</td>
<td></td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>t85</td>
<td>RxC High to CRS High; CRS Inactive Hold Time</td>
<td>10</td>
<td></td>
<td>ns</td>
<td></td>
</tr>
</tbody>
</table>

### RECEIVE CLOCK PARAMETERS

| t86    | RxC Rise Time | 5  |  | ns   | (Note 1)       |
| t87    | RxC Fall Time | 5  |  | ns   | (Note 1)       |
| t88    | RxC High Time | 18 |  | ns   | (Note 1)       |
| t89    | RxC Low Time | 19 |  | ns   | (Note 1)       |

### RECEIVED DATA PARAMETERS

| t90    | RxD Setup Time | 15 |  | ns   | (Note 5)       |
| t91    | RxD Hold Time | 15 |  | ns   | (Note 5)       |
| t92    | RxD Rise Time | 10 |  | ns   |                |
| t93    | RxD Fall Time | 10 |  | ns   |                |

### EXTERNAL LOOPBACK PARAMETERS

| t109, t110 | TxC Low to LPBK High | t60 |  | ns   | (Note 4)       |

### NOTES:

1. MOS Levels.
2. Manchester Only.
4. 1 TTL Load + 50 pF.
5. NRZ only.
6. Abnormal End of Transmission: CTS expires before RTS.
8. Programmable value:
   \[ t_{T76} = N_{IFS} \times t_{60} \]
   \[ N_{IFS} \]: the IFS configuration value.
   If \( N_{IFS} \) is less than 12 then \( N_{IFS} \) is enforced to 12.
9. Programmable Value:
   \[ t_{T79} = N_{CDF} \times t_{60} + (12 \text{ to } 15) \times t_{60} \] (if collision occurs after preamble).
   \[ N_{CDF} \]: The collision detect filter configuration value.
10. \( t_{T80} = 32 \times t_{60} \)
11. Programmable Value:
   \[ t_{T83} = N_{CDF} \times t_{60} + (12 \text{ to } 15) \times t_{60} \]
12. To guarantee recognition on the next clock.
TRANSMIT DATA WAVEFORMS

1-243
RECEIVE DATA WAVEFORMS (NRZ)

RECEIVE DATA WAVEFORMS

LOOPBACK OUTPUT SIGNAL TIMINGS
The 82C501AD Ethernet Serial Interface (ESI) chip is designed to work directly with Intel LAN Controllers (82586, 82590, and 82596) in IEEE 802.3 (10BASE5 and 10BASE2), 10-Mb/s, 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.
## Table 1. Pin Description

<table>
<thead>
<tr>
<th>Symbol</th>
<th>DIP Pin No.</th>
<th>PLCC Pin No.</th>
<th>Type</th>
<th>Name and Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>ENETV1</td>
<td>1</td>
<td>2</td>
<td>I</td>
<td><strong>ETHERNET VERSION 1.0:</strong> An active low, MOS-level input intended for use as a strapping option. When ENETV1 is asserted, the TRMT/TRMT 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/TRMT differential voltage gradually approaches 0V at the end of transmission.</td>
</tr>
<tr>
<td>NOOR</td>
<td>2</td>
<td>3</td>
<td>I</td>
<td><strong>CRS 'OR':</strong> An active low, MOS-level input intended for use as a strapping option. 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 inactive high, either the presence of a valid signal on CLSN/CLSN or on RCV/RCV will force CRS active.</td>
</tr>
<tr>
<td>LPBK/WDT</td>
<td>3</td>
<td>5</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/TRMT output pair to the transceiver cable. During loopback CDT is asserted at the end of each transmission to simulate the SQE test. The LPBK signal should be driven high once Vcc is stabilized. <strong>WATCHDOG TIMER DISABLE:</strong> An input voltage of 10 to 16 V through a 1 kΩ resistor will disable the on-chip watchdog timer.</td>
</tr>
<tr>
<td>RCV</td>
<td>4</td>
<td>6</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 is negative-going to indicate the beginning of a frame. The last transition is 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>7</td>
<td>I</td>
<td></td>
</tr>
<tr>
<td>CRS</td>
<td>6</td>
<td>8</td>
<td>O</td>
<td><strong>CARRIER SENSE:</strong> An active low, MOS-level output which notifies 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, when NOOR = 1, CRS is inhibited for a period of 5 μs minimum to 7 μs maximum, regardless of any activity on the collision-presence signal (CLSN/CLSN) and RCV/RCV inputs. When NOOR = 0, CRS is not inhibited.</td>
</tr>
<tr>
<td>CDT</td>
<td>7</td>
<td>10</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>11</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>13</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>DIP Pin No.</th>
<th>PLCC Pin No.</th>
<th>Type</th>
<th>Name and Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>GND</td>
<td>10</td>
<td>14</td>
<td></td>
<td>GROUND</td>
</tr>
<tr>
<td></td>
<td></td>
<td>15</td>
<td>I</td>
<td></td>
</tr>
<tr>
<td>CLSN</td>
<td>12</td>
<td>17</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.</td>
</tr>
<tr>
<td></td>
<td>11</td>
<td>16</td>
<td>I</td>
<td></td>
</tr>
<tr>
<td>X1</td>
<td>14</td>
<td>19</td>
<td>I</td>
<td>CLOCK CRYSTAL: 20-MHz crystal inputs. When X2 is floated, X1 can be driven by an external MOS level input clock.</td>
</tr>
<tr>
<td>X2</td>
<td>13</td>
<td>18</td>
<td>I</td>
<td></td>
</tr>
<tr>
<td>TEN</td>
<td>15</td>
<td>20</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 signal from the Intel LAN Controller.</td>
</tr>
<tr>
<td>TXC</td>
<td>16</td>
<td>22</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>24</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>28</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 ENETVy is asserted this voltage stepping is disabled.</td>
</tr>
<tr>
<td></td>
<td>18</td>
<td>27</td>
<td>O</td>
<td></td>
</tr>
<tr>
<td>VCC</td>
<td>20</td>
<td>1</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% at 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

The accuracy of the Crystal Oscillator frequency depends on the PC board characteristics, therefore it is advised to keep the X1 and X2 traces as short as possible. The optimum value of C1 and C2 should be determined experimentally under nominal operating conditions. The typical value of C1 and C2 is between 22 and 35 pF.

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 Intel LAN Controller 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 can 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/ITRMT 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, after the last transmit data transition, then it is gradually reduced. The TRMT/ITRMT differential voltage will become less than 40 mV within t18 after the last positive 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.

If an active signal is present at the ENETV1 input at the end of transmission, the TRMT/ITRMT pair output will remain at a high differential voltage. As a result there is 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 (when NOOR = 1). 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/ITRMT 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 to +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.

**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 the Manchester decode circuitry.

**NOTE:**

C1 = C2 = 22 pF to 35 pF.

For best operation, a decoupling capacitor should be used between VCC and GND.

**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 RXC 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 ±25%/−15% square-wave generated by the transceiver whenever two or more data frames are superimposed on the coaxial cable. The pulse width of the CLSN/CLSN signal can be no less than 35 ns and no greater than 70 ns measured at the 0-V crossing.

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 and NOOR = 1. 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 SGE (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.
• A pulse is rejected if:
  a. Its peak voltage is more positive than −150 mV, with no restriction on width, or:
  b. Its peak voltage is more positive than −600 mV and its width is less than 5 ns (measured at a reference level of −285 mV).
• The filter is turned off by the first valid negative pulse on the RCV or CLSN pair. A pulse whose peak voltage is more negative than −300 mV and whose width is greater than 30 ns (measured at −285 mV) is considered valid.

• 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 SGE test (heartbeat) signal within 1 μs after the end of the frame. Thus, the collision circuits are also tested in loopback mode. During loopback, as in any normal reception, 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.

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

![Figure 6. Watchdog Timer Disable](image)

<table>
<thead>
<tr>
<th>LPBK</th>
<th>WDTD</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>X</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>

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 Intel LAN Controller/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 Intel LAN Controller in IEEE 802.3 10 Mb/s, as well as other 10 Mb/s LAN applications. The control and data signals connect directly between the two devices without the need for additional external logic. The complete Intel LAN Controller/82C501AD Ethernet Transceiver interface is shown in Figure 4. The 82C501AD provides the driver and receivers needed to directly connect to the transceiver cable or requiring only terminating resistors on each input signal pair and 240Ω pull-down resistors.

It is recommended that a decoupling capacitor be used between \( V_{CC} \) and GND.

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 (Intel LAN Controller/82C501AD) and the transceiver. In an Ethernet/Cheapernet design, a single transformer can be used for both connections at minimal additional cost.

The pulse transformer should have the following characteristics:

1. A minimum inductance of 75 \( \mu \)H.
3. 2000V isolation between primaries of separate transformers.

Since Ethernet Version 1.0 transceivers can 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:** This is a production data sheet. The specifications are subject to change without notice.

*WARNING: Stressing the device beyond the "Absolute Maximum Ratings" may cause permanent damage. These are stress ratings only. Operation beyond the "Operating Conditions" is not recommended and extended exposure beyond the "Operating Conditions" may affect device reliability.*

### D.C. CHARACTERISTICS  \( T_C = 0°C \) to +85°C, \( V_{CC} = 5V \pm 10\% \)

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Parameter</th>
<th>Min</th>
<th>Max</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>TTL</td>
<td>MOS</td>
</tr>
<tr>
<td>( V_{IL} )</td>
<td>Input Low Voltage</td>
<td>-0.5V</td>
<td>0.8V</td>
</tr>
<tr>
<td>( V_{IH} )</td>
<td>Input High Voltage</td>
<td>2.0V</td>
<td>( V_{CC} + 0.5V )</td>
</tr>
<tr>
<td>( V_{ACCEPT} )</td>
<td>Differential Input Accept Voltage</td>
<td>±285 mV</td>
<td>±150 mV</td>
</tr>
<tr>
<td>( V_{REJECT} )</td>
<td>Differential Input Reject Voltage</td>
<td></td>
<td>±12V</td>
</tr>
<tr>
<td>( V_{CM} )</td>
<td>Input Common Mode Voltage</td>
<td>0V</td>
<td>( V_{CC} )</td>
</tr>
<tr>
<td>( V_{OCM} )</td>
<td>Common Mode Output Voltage(2)</td>
<td>0.5V</td>
<td>5.0V</td>
</tr>
<tr>
<td>( V_{OL} )</td>
<td>Output Low Voltage @ ( I_{OL} = 4 \text{ mA} )</td>
<td>0.45V</td>
<td></td>
</tr>
<tr>
<td>( V_{OH} )</td>
<td>Output High Voltage (MOS) @ ( I_{OH} = -500 \mu \text{A} )</td>
<td>3.9V</td>
<td></td>
</tr>
<tr>
<td>( V_{ODF} )</td>
<td>Differential Output Voltage(2)</td>
<td>±0.45V</td>
<td>±1.2V</td>
</tr>
<tr>
<td>( V_U )</td>
<td>TRMT Pair Differential Return to Zero Undershoot(2)</td>
<td></td>
<td>-100 mV</td>
</tr>
<tr>
<td>( V_{DI} )</td>
<td>TRMT Pair Differential Idle Voltage(2)</td>
<td></td>
<td>±40 mV(5)</td>
</tr>
<tr>
<td>( I_{LI} )</td>
<td>Input Leakage Current @ ( V_{IN} = 0V ) to ( V_{CC}(3) )</td>
<td>±10 \mu A</td>
<td></td>
</tr>
<tr>
<td>( I_{CC} )</td>
<td>Power Supply Current @ ( T_C = 85°C(4) )</td>
<td>135 mA</td>
<td></td>
</tr>
<tr>
<td>( I_{SP} )</td>
<td>Short Protection Activation Current</td>
<td>60 mA</td>
<td>150 mA</td>
</tr>
<tr>
<td>( I_L )</td>
<td>Input Load Current(6)</td>
<td></td>
<td>±1 mA</td>
</tr>
<tr>
<td>( C_{IN} )</td>
<td>Input Capacitance @ ( f_c = 1 \text{ MHz}(7) )</td>
<td>10 pF</td>
<td></td>
</tr>
</tbody>
</table>

**NOTES:**
1. The voltage levels for CLSN/CLSN, RCV/RCV inputs are −0.75V to +10V.
2. The testing load is a 38Ω ±1% resistor in parallel with a 27 μH ±1% inductor and two 240Ω ±5% pulldown resistors.
3. Applies to TXD and TE pins.
4. Part of the power is dissipated through the pulldown resistors connected to the TRMT/TRMT outputs.
5. Measured after \( t_{18} \) has expired.
6. Applies to RCV/RCV, \( X_1 \), CLSN/CLSN, LPBK/WDTO, NOCR, and ENETVT inputs for input voltages from 0V to \( V_{CC} \).
7. Characterized, not tested.
A.C. MEASUREMENT CONDITIONS

1. \( T_C = 0^\circ C \) to \(+85^\circ C\), \( V_{CC} = 5V \pm 10\%\).
2. The AC MOS, TTL and differential signals 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, two 240\( \Omega \) \( \pm 5\% \) pull down resistors, and a load resistor of 78\( \Omega \) \( \pm 1\% \) in parallel with a 27 \( \mu \)H \( \pm 1\% \) inductor between terminals.
4. All AC Parameters become valid 100 \( \mu s \) after the supply voltage has stabilized.

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>( t_1 )</td>
<td>( X_1 ) Cycle Time</td>
<td>49.995</td>
<td>50.005</td>
<td>ns</td>
</tr>
<tr>
<td>( t_2 )</td>
<td>( X_1 ) Fall Time(^{(2)})</td>
<td>5</td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>( t_3 )</td>
<td>( X_1 ) Rise Time(^{(2)})</td>
<td>5</td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>( t_4 )</td>
<td>( X_1 ) Low Time(^{(2)})</td>
<td>15</td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>( t_5 )</td>
<td>( X_1 ) High Time(^{(2)})</td>
<td>15</td>
<td>ns</td>
<td></td>
</tr>
</tbody>
</table>

NOTES:
1. Refer to Figure 9.
2. 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>( t_6 )</td>
<td>( TXC ) Cycle Time(^{(2)})</td>
<td>99.99</td>
<td>100.01</td>
<td>ns</td>
</tr>
<tr>
<td>( t_7 )</td>
<td>( TXC ) Rise/Fall Time</td>
<td>5</td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>( t_8 )</td>
<td>( TXD ) Rise/Fall Time</td>
<td>10</td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>( t_9 )</td>
<td>( TXC ) Low Time</td>
<td>40</td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>( t_{10} )</td>
<td>( TXC ) High Time</td>
<td>40</td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>( t_{11} )</td>
<td>Transmit Enable/Disable to ( TXC ) Low</td>
<td>45</td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>( t_{12} )</td>
<td>( TXD ) Stable to ( TXC ) Low</td>
<td>45</td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>( t_{13} )</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>( t_{14} )</td>
<td>( TEN ) Rise/Fall Time</td>
<td>10</td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>( t_{15} )</td>
<td>Transmit Differential Signal Rise/Fall Time</td>
<td>5.0</td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>( t_{16} )</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>( t_{17} )</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>( t_{18} )</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 exactly twice \( t_1 \).
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 at the Start of a Packet</td>
<td>1400</td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>t20</td>
<td>Receive Pair Signal Rise/Fall Time(5)</td>
<td>10</td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>t21</td>
<td>Receive Pair Bit Cell Center Jitter in Preamble(2)</td>
<td>±12</td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>t22</td>
<td>Receive Pair Bit Cell Center Jitter in Data(2)</td>
<td>±18</td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>t23</td>
<td>Receive Idle Time after Transmission in a Transmitting Station</td>
<td>8</td>
<td>µs</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>ns</td>
<td></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>ns</td>
<td></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>ns</td>
<td></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>ns</td>
<td></td>
</tr>
<tr>
<td>t29</td>
<td>RXC Low Time</td>
<td>40</td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>t30</td>
<td>RXC High Time</td>
<td>36</td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>t31</td>
<td>Receive Data Stable Before the Negative Edge of RXC</td>
<td>30</td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>t32</td>
<td>Receive Data Held Valid Past the Negative Edge of RXC</td>
<td>30</td>
<td>ns</td>
<td></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(5)</td>
<td>10</td>
<td>ns</td>
<td></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. Applies when NOOR = 1. For NOOR = 0 there is no inhibit of CRS.
5. Characterized, not tested.
### COLLISION 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>t36</td>
<td>CLSN/CLSN Cycle Time</td>
<td>80</td>
<td>118</td>
<td>ns</td>
</tr>
<tr>
<td>t37</td>
<td>CLSN/CLSN Rise/Fall Time(2)</td>
<td>10</td>
<td></td>
<td>ns</td>
</tr>
<tr>
<td>t38</td>
<td>CLSN/CLSN High/Low 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>450</td>
<td></td>
<td>ns</td>
</tr>
</tbody>
</table>

**NOTE:**
1. Refer to Figure 14.
2. Characterized, not tested.

### LOOPBACK 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>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(1)

<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>25</td>
<td></td>
<td>ns</td>
</tr>
</tbody>
</table>

**NOTE:**
1. Refer to Figure 16.
A.C. TIMING CHARACTERISTICS

Figure 7. TTL Input Voltage Levels for Timing Measurements (TEN, TXD, LPBK/WDTD).

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

Figure 9. X1 Input Voltage Levels for Timing Measurements.

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

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

RECEIVE TIMING: START OF FRAME

Figure 11

Figure 12
RECEIVE TIMING: END OF FRAME

![Diagram of Receive Timing](image)

Figure 13

COLLISION TIMING

![Diagram of Collision Timing](image)

Figure 14
LOOPBACK TIMING

Figure 15

NOISE FILTER TIMING

Accepted Pulse

Rejected Pulse

Figure 16. Noise Filter Characteristics
## 20 LEAD CERDIP DUAL IN-LINE PACKAGE INTEL TYPE D

![Diagram of 20 lead CERDIP dual-in-line package Intel Type D](image)

### Family: CerDIP Dual-In-Line Package

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Millimeters</th>
<th>Inches</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Min</td>
<td>Max</td>
</tr>
<tr>
<td>α</td>
<td>0°</td>
<td>10°</td>
</tr>
<tr>
<td>A</td>
<td></td>
<td>5.08</td>
</tr>
<tr>
<td>A₁</td>
<td>0.38</td>
<td></td>
</tr>
<tr>
<td>A₂</td>
<td>3.56</td>
<td>4.24</td>
</tr>
<tr>
<td>A₃</td>
<td>3.56</td>
<td>4.24</td>
</tr>
<tr>
<td>B</td>
<td>0.41</td>
<td>0.51</td>
</tr>
<tr>
<td>B₁</td>
<td>1.52</td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>0.23</td>
<td>0.30</td>
</tr>
<tr>
<td>D</td>
<td>24.38</td>
<td>25.27</td>
</tr>
<tr>
<td>D₂</td>
<td>22.86</td>
<td></td>
</tr>
<tr>
<td>E</td>
<td>7.62</td>
<td>8.13</td>
</tr>
<tr>
<td>E₁</td>
<td>7.11</td>
<td>7.90</td>
</tr>
<tr>
<td>e₁</td>
<td>2.29</td>
<td>2.79</td>
</tr>
<tr>
<td>eₐ</td>
<td>7.87</td>
<td></td>
</tr>
<tr>
<td>eₐB</td>
<td>8.13</td>
<td>10.16</td>
</tr>
<tr>
<td>L</td>
<td>3.18</td>
<td>3.81</td>
</tr>
<tr>
<td>N</td>
<td>20</td>
<td></td>
</tr>
<tr>
<td>S</td>
<td>0.38</td>
<td>1.78</td>
</tr>
<tr>
<td>S₁</td>
<td>0.13</td>
<td></td>
</tr>
</tbody>
</table>

**IWS** IWS 1/15/87
### Plastic Leaded Chip Carrier Rectangular

<table>
<thead>
<tr>
<th>Dimension</th>
<th>28 Lead (Inch)</th>
<th>28 Lead (mm)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Overall Height (A)</td>
<td>0.126 - 0.140</td>
<td>3.20 - 3.56</td>
</tr>
<tr>
<td>Shoulder to Board Height (A₁)</td>
<td>0.076 - 0.090</td>
<td>1.93 - 2.29</td>
</tr>
<tr>
<td>Outside Dimension (D)</td>
<td>0.385 - 0.396</td>
<td>9.78 - 10.0</td>
</tr>
<tr>
<td>Plastic Body Dimension (D₁)</td>
<td>0.347 - 0.353</td>
<td>8.81 - 8.97</td>
</tr>
<tr>
<td>Foot Print (D₂)</td>
<td>0.290 - 0.330</td>
<td>7.37 - 8.38</td>
</tr>
<tr>
<td>Foot Print (D₃)</td>
<td>0.200 Ref.</td>
<td>5.08 Ref.</td>
</tr>
<tr>
<td>Outside Dimension (E)</td>
<td>0.585 - 0.595</td>
<td>14.9 - 15.1</td>
</tr>
<tr>
<td>Plastic Body Dimension (E₁)</td>
<td>0.547 - 0.553</td>
<td>13.9 - 14.0</td>
</tr>
<tr>
<td>Foot Print (E₂)</td>
<td>0.490 - 0.530</td>
<td>12.4 - 13.5</td>
</tr>
<tr>
<td>Foot Print (E₃)</td>
<td>0.400 Ref.</td>
<td>10.2 Ref.</td>
</tr>
<tr>
<td># of Leads (N)</td>
<td>28</td>
<td>28</td>
</tr>
<tr>
<td># of Leads on Short Side (Nₐ)</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td># of Leads on Long Side (Nₑ)</td>
<td>9</td>
<td>9</td>
</tr>
<tr>
<td>Seating Plane Coplanarity (CP)</td>
<td>0.000 - 0.004</td>
<td>0.00 - 0.10</td>
</tr>
<tr>
<td>Tweezing Coplanarity (TCP)</td>
<td>0.000 - 0.004</td>
<td>0.000 - 0.10</td>
</tr>
<tr>
<td>Lead Thickness (LT)</td>
<td>0.009 - 0.015</td>
<td>0.23 - 0.38</td>
</tr>
<tr>
<td>Issue</td>
<td>IWS 1/15/87</td>
<td></td>
</tr>
</tbody>
</table>

231926-21
The Twisted Pair Ethernet Serial Supercomponent (SSC) provides the complete serial and analog Twisted Pair Ethernet interface required to connect the Ethernet LAN controller directly to a 10BASE-T connector. It is designed for node applications in 10-Mb/s, CSMA/CD networks as defined by the IEEE 802.3-1985 standard; for example, PCs, workstations, and file servers. The SSC includes the serial interface, transceiver, Manchester encoder, Manchester decoder, 10BASE-T functionality, line drivers, line receivers, analog filters, protection circuitry and isolation transformers in a single package. It provides all the required circuitry to give the LAN designer immediate access to the twisted pair Ethernet environment (10BASE-T).

Existing Ethernet/Cheapernet designs can be easily modified to take advantage of cost-effective twisted pair wire. The SSC can be soldered or socketed onto a host adapter card or motherboard without any adjustments or configuration. It is compatible with the pending IEEE 802.3 draft standard 10BASE-T, and can be used with non-Intel LAN controllers. The SSC is designed to satisfy FCC class A test requirements and to meet all standard host-system size and power requirements.

Figure 1. 82521TB Block Diagram

*Ethernet is a registered trademark of Xerox Corporation.
**Figure 2. 82521TB Pin Configuration**

<table>
<thead>
<tr>
<th>Pin</th>
<th>Symbol</th>
<th>Type</th>
<th>Name and Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>CRS</td>
<td>O</td>
<td><strong>CARRIER SENSE</strong>: Active low output that alerts the Ethernet controller or coprocessor (82586, 82590, 82592, or 82596) that activity is present on the twisted pair link. This pin is directly connected to the CRS input of the controller.</td>
</tr>
<tr>
<td>2</td>
<td>RxD</td>
<td>O</td>
<td><strong>RECEIVE DATA</strong>: NRZ data passed to the Ethernet controller. This pin is directly connected to the RxD pin on the controller.</td>
</tr>
<tr>
<td>3</td>
<td>CTS</td>
<td>O</td>
<td><strong>CLEAR TO SEND</strong>: An active low output that alerts the Ethernet controller that the device is ready to accept data. This function is optional. This pin can be tied directly to the controller’s CTS pin or left floating.</td>
</tr>
<tr>
<td>4</td>
<td>Vss</td>
<td></td>
<td><strong>GROUND</strong>.</td>
</tr>
<tr>
<td>5</td>
<td>TxD</td>
<td>I</td>
<td><strong>TRANSMIT DATA</strong>: NRZ or Manchester encoded serial data is clocked in on TxD from the Ethernet controller. This pin is directly connected to the TxD pin of the controller. The state of the MENC pin determines NRZ or Manchester encoded input.</td>
</tr>
</tbody>
</table>

*Pins 14, 15, and 17 have been enhanced since the 82521TA. This does not affect compatibility with the 82521TA.*

†Pins 26 and 31 have been removed for improved electrical isolation. This does not affect compatibility with the 82521TA.

‡The 12-V power requirement of the 82521 has been removed on the 82521TB. To maintain socket compatibility these pins are not connected internally; they can be connected to a power supply on the host card or left floating.
<table>
<thead>
<tr>
<th>Pin</th>
<th>Symbol</th>
<th>Type</th>
<th>Name and Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>VSS</td>
<td></td>
<td>GROUND.</td>
</tr>
<tr>
<td>7</td>
<td>TxC</td>
<td>O</td>
<td>TRANSMIT CLOCK: A 10-MHz clock output tied directly to the TxC pin of the Ethernet controller.</td>
</tr>
<tr>
<td>8</td>
<td>VSS</td>
<td></td>
<td>GROUND.</td>
</tr>
<tr>
<td>9</td>
<td>VEE</td>
<td></td>
<td>VEE: A – 5 V ± 10% power supply.</td>
</tr>
<tr>
<td>10</td>
<td>RxC</td>
<td>O</td>
<td>RECEIVE CLOCK: A 10-MHz clock connected directly to the RxC input of the Ethernet controller. This clock is the recovered clock from the incoming data on the twisted pair.</td>
</tr>
<tr>
<td>11</td>
<td>VSS</td>
<td></td>
<td>GROUND.</td>
</tr>
<tr>
<td>12</td>
<td>ECLK</td>
<td>O</td>
<td>EXTERNAL CLOCK: A 20-MHz ± 0.01%, TTL-level input clock with a 40/60 duty cycle. ECLK attaches to ICLK for normal asynchronous operation. Synchronous operation of the device can be obtained by connecting an external clock to ECLK.</td>
</tr>
<tr>
<td>13</td>
<td>ICLK</td>
<td>I</td>
<td>INTERNAL CLOCK: A 20-MHz ± 0.01%, TTL-level output clock with a 40/60 duty cycle. For normal asynchronous operation this pin is directly connected to ECLK. ICLK can be used for synchronous operation of interface circuitry.</td>
</tr>
<tr>
<td>14</td>
<td>LID/WDT</td>
<td>I</td>
<td>LINK INTEGRITY DISABLE, WATCHDOG TIMER DISABLE: When connected to VCC1, this pin disables the link integrity processor, linkbeat generator, and watchdog timer. This ensures compatibility with the 82521TA. When connected to VSS, or not connected, these functions are enabled and the device is compatible with 10BASE-T.</td>
</tr>
<tr>
<td>15</td>
<td>MENC</td>
<td>I</td>
<td>MANCHESTER ENCODING: When tied to VSS, or not connected, this pin enables internal Manchester encoding of the NRZ data on TxD. When tied to VCC1, Manchester encoded data is expected on TxD.</td>
</tr>
<tr>
<td>16</td>
<td>LPBK2</td>
<td>I</td>
<td>LOOPBACK 2: An active high input signal that causes the 82521TB to enter diagnostic loopback mode. The twisted pair medium will be removed from the circuit, thus isolating the node from the network. When not connected this pin assumes the inactive (normal) state. Diagnostic loopback mode does not affect the operation of the link integrity processor or linkbeat generator. The watchdog timer will not operate in diagnostic loopback mode.</td>
</tr>
<tr>
<td>17</td>
<td>WDT</td>
<td>O</td>
<td>WATCHDOG TIMER: An active high output that indicates expiration of the watchdog timer (jabber protection) in the super-component. The output deasserts when the jabber function is reset.</td>
</tr>
<tr>
<td>18</td>
<td>VSS</td>
<td></td>
<td>GROUND.</td>
</tr>
<tr>
<td>19</td>
<td>RTS</td>
<td>I</td>
<td>REQUEST TO SEND: An active low input signal synchronous to TxC; it enables data transmission on the twisted pair link segment.</td>
</tr>
<tr>
<td>20</td>
<td>LPBK1</td>
<td>I</td>
<td>LOOPBACK 1: An active low input signal that causes the 82521TB to enter diagnostic loopback mode. The twisted pair medium will be removed from the circuit, thus isolating the node from the network. When not connected this pin assumes the inactive (normal) state. Diagnostic loopback mode does not affect the operation of the link integrity processor, or linkbeat generator. The watchdog timer will not operate in diagnostic loopback mode.</td>
</tr>
<tr>
<td>Pin</td>
<td>Symbol</td>
<td>Type</td>
<td>Name and Function</td>
</tr>
<tr>
<td>-----</td>
<td>--------</td>
<td>------</td>
<td>-------------------</td>
</tr>
<tr>
<td>21</td>
<td>N/C</td>
<td></td>
<td>This pin is not connected.</td>
</tr>
<tr>
<td>22</td>
<td>Reserved</td>
<td>I</td>
<td>This pin is used for testing purposes. It should be left floating or connected to VSS.</td>
</tr>
<tr>
<td>23</td>
<td>VSS</td>
<td></td>
<td>GROUND.</td>
</tr>
<tr>
<td>24</td>
<td>Reset</td>
<td>I</td>
<td>RESET: An active high input that brings the device into a known state. It must be asserted for 1 ms while the clock is running.</td>
</tr>
<tr>
<td>25</td>
<td>LI</td>
<td>O</td>
<td>LINK INTEGRITY: An active high output used to indicate the presence of link integrity faults.</td>
</tr>
<tr>
<td>27</td>
<td>TD+</td>
<td>O</td>
<td>TWISTED PAIR TRANSMIT DATA: This pin transmits outgoing Manchester data to the twisted pair link segment. It is connected directly to the Medium Dependent Interface Connector (RJ-45) pin 1. This pin, and the trace leading to it, must withstand 2250 V dc to ground without damage.</td>
</tr>
<tr>
<td>28</td>
<td>TD−</td>
<td>O</td>
<td>TWISTED PAIR TRANSMIT DATA COMPLEMENT: This pin transmits outgoing inverted Manchester data to the twisted pair link segment. It is connected directly to the Medium Dependent Interface Connector (RJ-45) pin 2. This pin, and the trace leading to it, must withstand 2250 V dc to ground without damage.</td>
</tr>
<tr>
<td>29</td>
<td>RD+</td>
<td>I</td>
<td>TWISTED PAIR RECEIVE DATA: This pin receives incoming Manchester data from the twisted pair link segment. It is connected directly to the Medium Dependent Interface Connector (RJ-45) pin 3. This pin, and the trace leading to it, must withstand 2250 V dc to ground without damage.</td>
</tr>
<tr>
<td>30</td>
<td>RD−</td>
<td>I</td>
<td>TWISTED PAIR RECEIVE DATA COMPLEMENT: This pin receives incoming inverted Manchester data from the twisted pair link segment. It is connected directly to the Medium Dependent Interface Connector (RJ-45) pin 6. This pin, and the trace leading to it, must withstand 2250 V dc to ground without damage.</td>
</tr>
<tr>
<td>32</td>
<td>N/C</td>
<td></td>
<td>This pin is not connected.</td>
</tr>
<tr>
<td>33</td>
<td>N/C</td>
<td></td>
<td>This pin is not connected.</td>
</tr>
<tr>
<td>34</td>
<td>ĈDT</td>
<td>O</td>
<td>COLLISION DETECT: An active low signal that indicates the presence of a collision to the controller.</td>
</tr>
<tr>
<td>35</td>
<td>VCC1</td>
<td></td>
<td>VCC1: A 5 V ±5% power supply.</td>
</tr>
<tr>
<td>36</td>
<td>VCC2</td>
<td></td>
<td>VCC2: A 5 V ±10% power supply. This pin can be connected to pin 35 (VCC1) if a single power supply can reliably supply the combined requirements of ICC1 and ICC2 (see DC Characteristics).</td>
</tr>
</tbody>
</table>
FUNCTIONAL DESCRIPTION

Overview

The 82521TB provides the functions required for operating Data Terminal Equipment on a 10-Mb/s, CSMA/CD, Local Area Network (LAN) using standard telephone building wiring. The 82521TB design is based on the Twisted Pair Ethernet draft standard supplement to IEEE Std. 802.3 (type 10BASE-T, P802.3I/D8).

Clock Generation

The 82521TB supports internal and external sources for the precision clock (20 MHz ± 0.01%) required in an Ethernet environment. The clock is used to retime the transmitted Manchester data, to generate the 10-MHz TxC signal, and as a precision reference for Manchester decoding and clock recovery of received data.

If ICLK (Pin 13) is strapped to ECLK (Pin 12) an onboard clock oscillator generates the precision clock. This clock can be used for synchronous operation of circuits on the host board; however, care must be taken to minimize the load on the clock. The supercomponent can be operated synchronous to the host by providing ECLK with a 20-MHz ± 0.01%, TTL-level clock with a duty cycle of 40/60 or better.

Transmit Section

The transmit section of the 82521TB is controlled by the RTS signal generated by the Ethernet LAN controller (Intel's 82586, 82592, etc). When RTS asserts, the 82521TB begins clocking in data from the controller on the TxD input. The TxD level is sampled on the falling edge of the 10-MHz TxC signal (or every edge if Manchester encoding is not enabled). The data is then encoded using the precision 20-MHz clock, and then sent to the twisted pair line drivers.

The line drivers begin transmitting the serial Manchester bit stream two bit times after the assertion of RTS. A predistortion algorithm is used by the line drivers to improve jitter performance on the twisted pair. The line drivers reduce their drive level during the second half of “fat” (100 ns) Manchester pulses, and maintain full drive level during all “thin” (50 ns) pulses. During the “fat” pulses, this reduces line overcharge, which is a major source of jitter. Figure 3 shows the difference between the familiar coax cable waveform and the predistorted waveform generated by the 82521TB. The line drivers maintain a characteristic impedance of 96 Ω typical throughout data packet transmission and the idle state.

The predistorted output of the line drivers is then passed through the transmit EMI filter to reduce the high frequency harmonics of the transmitted signal. This reduces noise, Near End Cross Talk (NEXT) in bundled twisted pair cables, and unwanted Radio Frequency (RF) interference. The filter maintains the 96 Ω (typical) characteristic impedance. The filtered signal then passes through an isolation transformer and common mode choke. These provide high voltage protection, dc isolation, and common mode noise rejection. The output of the common mode choke is directly connected to the TD+ and TD- pins. These can be connected to pins 1 and 2 of an ISO 8877 (RJ-45) connector.

After a successful transmission, the signal_quality_error test (heartbeat) function is executed in accordance with the 802.3 10BASE-T draft specification.

Figure 3. Effects of Predistortion
Receive Section

The receive section of the 82521TB processes incoming Manchester data from the twisted pair link segment, converts it to NRZ data, and recovers the embedded clock. It contains a squelch circuit that distinguishes noise from incoming data. Valid data passes through the input protection and common mode rejection of the 82521TB and the receive EMI filter, and trips the squelch circuit. The twisted pair line receiver is then enabled, and converts the signal to digital voltage levels.

The signal then passes to the Manchester decoder and clock recovery circuit. CRS asserts within nine bit times of the arrival of the data packet to indicate the presence of activity on the network. Fourteen bit times later RXC and RxD activate, passing the remaining preamble and data to the controller in NRZ format.

The 82521TB detects the Start-Of-Idle (SOI) signal at the end of a packet. CRS will be synchronously deasserted with RXC within four bit times from the beginning of the SOI. RXC and RxD then return to their idle state.

Collision Detect

Collision detection in the twisted pair environment is indicated by simultaneous transmission and reception on the twisted pair link segment. The CDT signal is asserted for the duration of both RTS and the presence of received data; CRS is asserted for the duration of either RTS or the presence of received data.

DO to DI Loopback

When the 82521TB is transmitting on the TD circuit, and not receiving on the RD circuit, it also routes the transmitted data to the receive circuitry. It returns to the controller via the CRS, RXC, and RxD signals.

Link Integrity

The 82521TB supports the link integrity function as defined in the 10BASE-T draft. During long periods of idle on the transmitter, link test pulses will be transmitted on to the twisted pair medium as an indication to the receiving MAU that the link is good. These pulses will be transmitted between 8 and 24 ms after the end of the last transmission or link test pulse.

The link integrity function continuously monitors activity on the receive circuit. If neither valid data or link test pulses are received for a period of time, the link integrity processor declares the link bad, and disables transmission and reception on the medium. Transmission of link test pulses and monitoring of receive activity are not affected. The idle time required for the link integrity processor to determine if the link is bad is between 50 and 150 ms.

Once a frame, or a sequence of 2 to 10 consecutive link test pulses, are detected, the link integrity processor declares the link good and reconnects the transmitter and receiver.

Jabber Function

The 82521TB has an onboard watchdog timer to implement the jabber function. If a transmission continues beyond the limits specified by the 10BASE-T draft standard (between 20 and 150 ms), the jabber function inhibits further transmission and asserts the collision indicator CDT. The transmission inhibit period extends until the 82521TB detects sufficient idle time (between 250 and 750 ms) on the RTS signal. Link test pulses continue to be sent during the period when the transmitter is disabled.

Diagnostic Loopback Mode

The 82521TB supports a diagnostic loopback mode in addition to the normal DO to DI loopback mode. When either LPBKT or LPBK2 are asserted, data transmission and reception on the twisted pair link is disabled, thus removing the DTE from the network. Any transmissions made in this mode are fed back into the receive circuits and subsequently passed back to the controller. Diagnostic loopback mode does not affect the link integrity function. Link test pulses are still transmitted and the twisted pair link is monitored for frames and link test pulse reception.

State Diagrams

The 82521TB operation is described in the following four state diagrams: Transmit-Receive, SQE Test, Jabber, and Link Integrity. They are based on the state diagrams of the 10BASE-T draft. These state diagrams differ from those of the draft standard because the 10BASE-T specification addresses an external MAU with AUI cable, whereas the 82521TB eliminates the AUI cable. Therefore, the state diagrams for the 82521TB reference its own interface signals, not the AUI signals. Operation of the device at the MDI connector (RJ-45) is identical.
FUNCTIONAL DESCRIPTION (Continued)

![Transmit Receive State Diagram](image)

Figure 4. Transmit Receive State Diagram
FUNCTIONAL DESCRIPTION (Continued)

**Figure 5. SQE Test State Diagram**

- **Power On**
  - **OUTPUT IDLE**
  - **RTS = active**
    - **OUTPUT DETECTED**
    - **RTS = idle**
      - **SQE TEST WAIT**
        - **Start_SQE_test_wait_timer**
          - \( xmit = disable \)
          - \( *isolate = idle \)
        - **SQE TEST**
          - **Start_SQE_test_timer**
          - **SQE**
          - **SQE_H_test_timer_done**

\[ \text{SQE_test_wait_timer_done} \] (\( xmit = enable + isolate = active \))
FUNCTIONAL DESCRIPTION (Continued)

Figure 6. Jabber Function State Diagram
FUNCTIONAL DESCRIPTION (Continued)

![Link Integrity State Diagram]

Figure 7. Link Integrity State Diagram
State Diagram Variables

The variable definitions contain a description of the function they control as well as the possible values for that variable. Many of the variables have a default value; when none of the state machines explicitly assigns a particular value a value, then that variable implicitly takes on the default value. Several constants are used and defined by either the 802.3 standard or the 10BASE-T draft, these constants are printed italic.

**RTS**
Indicates the presence of a data stream being received from the DTE on the TxD signal.
- **RTS = active.** Data is present on the TxD circuit.
- **RTS = idle.** Data is not present on the TxD circuit.

**RxD**
Controls the source of data transmitted to the DTE on the RxD circuit.
- **RxD = RD.** Data on the RxD circuit is sourced by the RD circuit input from the twisted pair. The condition "rcv = disable" overrides and disables this activity.
- **RxD = DO.** Data on the RxD circuit is sourced by the TxD circuit from the DTE. This is the loopback function. The condition "loopback = disable" overrides and disables this activity.
- **RxD = idle.** Default. There is no data being transmitted on the RxD circuit, it contains the input_idle message.

**CDT**
Controls the message the 82521TB sends to the DTE on the CDT signal.
- **CDT = SQE.** The MAU is sending the signal_quality_error message to the DTE on the CDT signal.
- **CDT = idle.** Default. The MAU is sending the MAU_available message to the DTE on the CDT signal.

**TD**
Controls the source of data transmitted to the network on the TD circuit.
- **TD = TxD.** The TD circuit is sourced by the TxD signal. The conditions "xmit = disable" or "isolate = true" overrides and disables this activity.

**RD**
Indicates the presence of an input data stream from the network on the RD circuit.
- **RD = active.** Data is present on the RD circuit.
- **RD = idle.** Data is not present on the RD circuit.

**link_test_rcvd**
Indicates the presence of a link test pulse on the RD circuit.
- **link_test_rcvd = true.** A link test pulse is present on the RD circuit.
- **link_test_rcvd = false.** Silence or data is present on the RD circuit.

**link_count**
Indicates the number of consecutive link test pulses received while in a link-fail-state.

**lc_max**
The maximum number of consecutive link test pulses required before reconnection is allowed.

**xmit**
Communication path between state machines. This variable relates the status of certain fault conditions that require the transmit and SQE functions to be disabled.
- **xmit = disable.** A condition exists that dictates the transmit and SQE test functions to be disabled.
- **xmit = enable.** Default. The transmit and SQE test functions operate normally.

**rcv**
Communication path between state machines. This variable relates the status of certain fault conditions that require the receive function to be disabled.
- **rcv = disable.** A condition exists that dictates that the receive function be disabled.
- **rcv = enable.** Default. The receive function should operate normally.

**lpbk**
Communication path between state machines. This variable relates the status of certain fault conditions that require the DO to DI loopback function to be disabled.
- **lpbk = disable.** A condition exists that dictates that the loopback function be disabled.
lpbk  (continued)  lpbk = enable. Default. The DO to DI loopback function should operate normally.

isolate  Indicates the status of the diagnostic loopback mode of the 82521TB.

isolate = false. Device is not in diagnostic loopback mode. Input LPBK1 = 1 and LPBK2 = 0.

isolate = true. Device is in diagnostic loopback mode. Input LPBK1 = 0 or LPBK2 = 1.

WDTD  Indicates the status of the watchdog timer disable mode of the 82521TB.

WDTD = true. Watchdog timer (Jabber function) is disabled. Input LID/WDTD = 1.

WDTD = false. Watchdog timer is enabled. Input LID/WDTD = 0.

LID  Indicates the status of the link integrity disable mode of the 82521TB.

LID = true. Link integrity process is disabled. Input LID/WDTD = 1.

LID = false. Link integrity process is enabled. Input LID/WDTD = 0.

State Diagram Timers

link_loss_timer. Time to wait to declare a bad receive link.

link_test_min_timer. Minimum time allowed between consecutive link test pulses.

link_test_max_timer. Maximum time allowed between consecutive link pulses.

SQE_test_wait_timer. Time to wait before executing the SQE_test function.

SQE_test_timer. Time to wait for completing the SQE_test function.

xmit_max_timer. Time to wait to interrupt jabbering transmission.

unjab_timer. Time to wait before resetting jabber function.
Dear [Recipient],

I am writing to inform you about the 82521TB ABSOLUTE MAXIMUM RATINGS and D.C. CHARACTERISTICS of the Intel IC. The data sheet contains preliminary information on new products in production. The specifications are subject to change without notice. Stressing the device beyond the “Absolute Maximum Ratings” may cause permanent damage. These are stress ratings only. Operation beyond the “Operating Conditions” is not recommended and extended exposure beyond the “Operating Conditions” may affect device reliability.

### ABSOLUTE MAXIMUM RATINGS

- Ambient Temperature Under Bias: 0° to +70°C
- Storage Temperature: -65° to +140°C
- All Output and Supply Voltages: -0.5 to +7V
- All Input Voltages: -0.5 to VCC1 + 0.5V

### D.C. CHARACTERISTICS

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Parameter</th>
<th>Min</th>
<th>Max</th>
<th>Units</th>
</tr>
</thead>
<tbody>
<tr>
<td>VIL</td>
<td>Input Low Voltage (3)</td>
<td>-0.3</td>
<td>0.8</td>
<td>V</td>
</tr>
<tr>
<td>VIH</td>
<td>Input High Voltage (3)</td>
<td>2.0</td>
<td>VCC</td>
<td>V</td>
</tr>
<tr>
<td>ILI</td>
<td>Input Leakage Current</td>
<td>±400</td>
<td>μA</td>
<td>-0.3 V ≤ VIL ≤ VCC1</td>
</tr>
<tr>
<td>VOL</td>
<td>Output Low Voltage (4)</td>
<td>0.45</td>
<td>V</td>
<td>IOL = 4 mA</td>
</tr>
<tr>
<td>VOH</td>
<td>Output High Voltage (4)</td>
<td>3.9</td>
<td>V</td>
<td>IOH = -500 μA</td>
</tr>
<tr>
<td>ICC1</td>
<td>Power Supply Current</td>
<td>410</td>
<td>mA</td>
<td>VCC = 5.25 V</td>
</tr>
<tr>
<td>ICC2</td>
<td>Power Supply Current</td>
<td>105</td>
<td>mA</td>
<td>VCC = 5.5 V</td>
</tr>
<tr>
<td>IEE</td>
<td>Power Supply Current</td>
<td>-40</td>
<td>mA</td>
<td>VEE = -5.5 V</td>
</tr>
<tr>
<td>PD</td>
<td>Power Dissipation</td>
<td>3</td>
<td>W</td>
<td>VCC = 5.0 V</td>
</tr>
</tbody>
</table>

### ANALOG CHARACTERISTICS

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Parameter</th>
<th>Min</th>
<th>Max</th>
<th>Units</th>
</tr>
</thead>
<tbody>
<tr>
<td>ZO</td>
<td>Characteristic Impedance(1, 2)</td>
<td>77</td>
<td>115</td>
<td>Ω</td>
</tr>
<tr>
<td>Return Loss(1, 2) (5 MHz to 10 MHz)</td>
<td>15</td>
<td>dB</td>
<td>85Ω ≤ RLOAD ≤ 111Ω</td>
<td></td>
</tr>
<tr>
<td>Squelch Reject Level(2)</td>
<td>300</td>
<td>mV</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Squelch Accept Level(2)</td>
<td>450</td>
<td>mV</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CMR</td>
<td>Common Mode Rejection(1, 2)</td>
<td>29</td>
<td>dB</td>
<td>At 10 MHz</td>
</tr>
<tr>
<td>VIDS</td>
<td>Input Peak Differential</td>
<td>0.500</td>
<td>V</td>
<td></td>
</tr>
<tr>
<td>VODF1</td>
<td>Output Peak Differential</td>
<td>2.2</td>
<td>2.8</td>
<td>V</td>
</tr>
</tbody>
</table>

NOTES:
1. The voltage levels for TD+ and TD− are ±2250 V with respect to ground.
2. The voltage levels for RD+ and RD− are ±2250 V with respect to ground.
3. Digital Inputs, TxD, ECLK, MENC, LID/WDT, LPBK2, RTS, LPBK1, TEST, and RESET.
4. Digital Outputs, CRS, RxD, CTS, TxC, RxC, ICLK, WDT, LI, and CDI.

Thank you for your attention.

Best regards,

[Your Name]
ANALOG CHARACTERISTICS (Continued)

$T_A = 0^\circ$ to 70°C, $V_{CC1} = 5$ V $\pm 5\%$, $V_{CC2} = 5$ V $\pm 10\%$

<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>$V_{ODF2}$</td>
<td>Output Peak Differential</td>
<td>0.585</td>
<td></td>
<td>V</td>
<td>With Cable Model(3)</td>
</tr>
<tr>
<td></td>
<td>Maximum Overshoot</td>
<td>50</td>
<td></td>
<td>mV</td>
<td>Start of Idle</td>
</tr>
<tr>
<td></td>
<td>Transmitter Impedance Balance</td>
<td>29</td>
<td></td>
<td>dB</td>
<td>At 10 MHz</td>
</tr>
<tr>
<td>$V_{OCM}$</td>
<td>Output Common Mode</td>
<td>50</td>
<td></td>
<td>mV</td>
<td>$&gt;30$ kHz</td>
</tr>
<tr>
<td></td>
<td>Harmonic Content</td>
<td>$-27$</td>
<td></td>
<td>dB</td>
<td>$\geq 30$ MHz</td>
</tr>
</tbody>
</table>

NOTES:
1. Pin 27 and 28 (TD+ and TD−).
2. Pins 29 and 30 (RD+ and RD−).
3. Cable Mode defined in 10BASE-T draft.

A.C. Timing Conditions
1. $T_A = 0^\circ$ to 70°C, $V_{CC1} = 5$ V $\pm 5\%$, $V_{CC2} = 5$ V $\pm 10\%$.
2. Digital outputs timing measurement points are 0.9 V and 3.0 V unless otherwise noted.
3. Digital input timing measurements points are 0.8 V and 2.0 V unless otherwise noted.
4. TD Pair and RD Pair timing measurements are 0 V unless otherwise noted.
5. Digital ac loads: 20 pF to ground.
6. TD+ and TD− load: 96 Ω differential load.

CLOCK TIMING CHARACTERISTICS

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Parameter</th>
<th>Min</th>
<th>Max</th>
<th>Units</th>
</tr>
</thead>
<tbody>
<tr>
<td>$t_1$</td>
<td>ECLK/ICLK Cycle Time</td>
<td>49.995</td>
<td>50.005</td>
<td>ns</td>
</tr>
<tr>
<td>$t_2$</td>
<td>ECLK/ICLK Rise and Fall Time</td>
<td>8</td>
<td></td>
<td>ns</td>
</tr>
<tr>
<td>$t_3$</td>
<td>ECLK/ICLK High and Low Time</td>
<td>15</td>
<td></td>
<td>ns</td>
</tr>
</tbody>
</table>
# Decoder Timing Characteristics

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Parameter</th>
<th>Min</th>
<th>Max</th>
<th>Units</th>
</tr>
</thead>
<tbody>
<tr>
<td>t4</td>
<td>RXC Period</td>
<td>96</td>
<td>104</td>
<td>ns</td>
</tr>
<tr>
<td>t5</td>
<td>RXC High Time</td>
<td>36</td>
<td></td>
<td>ns</td>
</tr>
<tr>
<td>t6</td>
<td>RXC Low Time</td>
<td>40</td>
<td></td>
<td>ns</td>
</tr>
<tr>
<td>t7</td>
<td>RXC Rise and Fall Time(2)</td>
<td>5</td>
<td></td>
<td>ns</td>
</tr>
<tr>
<td>t8</td>
<td>RXD Setup Time to RXC Falling Edge</td>
<td>30</td>
<td></td>
<td>ns</td>
</tr>
<tr>
<td>t9</td>
<td>RXD Hold Time from RXC Falling Edge</td>
<td>30</td>
<td></td>
<td>ns</td>
</tr>
<tr>
<td>t10</td>
<td>RXD Rise and Fall Time(2)</td>
<td>5</td>
<td></td>
<td>ns</td>
</tr>
<tr>
<td>t11</td>
<td>Receiver Steady State Delay</td>
<td>300</td>
<td></td>
<td>ns</td>
</tr>
<tr>
<td>t12</td>
<td>CRS Assertion Delay from First Valid RD Pair Edge(1)</td>
<td>600</td>
<td></td>
<td>ns</td>
</tr>
<tr>
<td>t13</td>
<td>Duration RXC is Held Low at Start of Packet</td>
<td>1900</td>
<td></td>
<td>ns</td>
</tr>
<tr>
<td>t14</td>
<td>CRS Deassertion Delay from Last Valid RD Pair Edge(1)</td>
<td>350</td>
<td></td>
<td>ns</td>
</tr>
<tr>
<td>t15</td>
<td>CRS Deassertion Hold Time from RXC High(1)</td>
<td>10</td>
<td>40</td>
<td>ns</td>
</tr>
<tr>
<td>t16</td>
<td>RD Pair Bit Cell Center Jitter in Preamble</td>
<td>±12</td>
<td></td>
<td>ns</td>
</tr>
<tr>
<td>t17</td>
<td>RD Pair Bit Cell Center Jitter in Data</td>
<td>±18</td>
<td></td>
<td>ns</td>
</tr>
<tr>
<td>t18</td>
<td>RD Pair Return to Zero from Last Valid Positive Transition</td>
<td>235</td>
<td></td>
<td>ns</td>
</tr>
<tr>
<td>t19</td>
<td>RD Idle Time After Transmission</td>
<td>8</td>
<td></td>
<td>μs</td>
</tr>
</tbody>
</table>

**NOTES:**
1. RTS inactive.
2. Characterized, not tested.

![Figure 8. Clock Timing Measurement Points](image_url)
Figure 9. Receive Timing: Start of Frame

Figure 10. Receive Timing: End of Frame
## Encoder Timing Characteristics

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Parameter</th>
<th>Min</th>
<th>Max</th>
<th>Units</th>
</tr>
</thead>
<tbody>
<tr>
<td>t20</td>
<td>TxC Period</td>
<td>99.99</td>
<td>100.01</td>
<td>ns</td>
</tr>
<tr>
<td>t21</td>
<td>TxC Rise and Fall Time(^2)</td>
<td></td>
<td>8</td>
<td>ns</td>
</tr>
<tr>
<td>t22</td>
<td>TxC High Time</td>
<td>40</td>
<td></td>
<td>ns</td>
</tr>
<tr>
<td>t23</td>
<td>TxC Low Time</td>
<td>40</td>
<td></td>
<td>ns</td>
</tr>
<tr>
<td>t24</td>
<td>RTS Assertion to TD Pair Active</td>
<td>250</td>
<td>470</td>
<td>ns</td>
</tr>
<tr>
<td>t25</td>
<td>Transmitter Steady State Delay</td>
<td>100</td>
<td>270</td>
<td>ns</td>
</tr>
<tr>
<td>t26</td>
<td>TxD Setup Time to TxC Low</td>
<td>5</td>
<td></td>
<td>ns</td>
</tr>
<tr>
<td>t27</td>
<td>TxD Hold Time from TxC Low</td>
<td>10</td>
<td></td>
<td>ns</td>
</tr>
<tr>
<td>t28</td>
<td>TxD Setup Time from TxC High(^1)</td>
<td>5</td>
<td></td>
<td>ns</td>
</tr>
<tr>
<td>t29</td>
<td>TxD Hold Time from TxC High(^1)</td>
<td>25</td>
<td></td>
<td>ns</td>
</tr>
<tr>
<td>t30</td>
<td>TxD Rise and Fall Time(^2)</td>
<td>10</td>
<td></td>
<td>ns</td>
</tr>
<tr>
<td>t31</td>
<td>RTS Setup Time to TxC Low</td>
<td>5</td>
<td></td>
<td>ns</td>
</tr>
<tr>
<td>t32</td>
<td>RTS Hold Time from TxC Low</td>
<td>10</td>
<td></td>
<td>ns</td>
</tr>
<tr>
<td>t33</td>
<td>RTS Rise and Fall Time(^2)</td>
<td>10</td>
<td></td>
<td>ns</td>
</tr>
<tr>
<td>t34</td>
<td>TD Pair Output Jitter</td>
<td>±4</td>
<td></td>
<td>ns</td>
</tr>
<tr>
<td>t35</td>
<td>RTS Deasserted to TD Pair Inactive</td>
<td>350</td>
<td>520</td>
<td>ns</td>
</tr>
<tr>
<td>t36</td>
<td>TD Pair Held at Positive Differential at SOI</td>
<td>250</td>
<td>400</td>
<td>ns</td>
</tr>
<tr>
<td>t37</td>
<td>TD Pair Return to ±50 mV after Transmission</td>
<td>4500</td>
<td></td>
<td>ns</td>
</tr>
<tr>
<td>t38</td>
<td>RTS Deasserted to CDT Asserted (Heartbeat)</td>
<td>600</td>
<td>1600</td>
<td>ns</td>
</tr>
<tr>
<td>t39</td>
<td>CDT Assertion Pulse Width (Heartbeat)</td>
<td>500</td>
<td>1500</td>
<td>ns</td>
</tr>
</tbody>
</table>

**NOTES:**
1. Manchester Encoder Disabled.
2. Characterized, not tested.

---

**Figure 11. Transmit Timing: Manchester Encoding**
### COLLISION TIMING CHARACTERISTICS

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Parameter</th>
<th>Min</th>
<th>Max</th>
<th>Units</th>
</tr>
</thead>
<tbody>
<tr>
<td>(t_{41})</td>
<td>CDT Assertion from Onset of Collision</td>
<td>20</td>
<td>900</td>
<td>ns</td>
</tr>
<tr>
<td>(t_{42})</td>
<td>CDT Deassertion from End of Collision</td>
<td>20</td>
<td>900</td>
<td>ns</td>
</tr>
<tr>
<td>(t_{43})</td>
<td>CRS Deassertion from End of Collision(^{(1)})</td>
<td>20</td>
<td>900</td>
<td>ns</td>
</tr>
</tbody>
</table>

**NOTE:**
1. Both RTS and RD Pair Idle.

### MODAL TIMING CHARACTERISTICS

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Parameter</th>
<th>Min</th>
<th>Max</th>
<th>Units</th>
</tr>
</thead>
<tbody>
<tr>
<td>(t_{44})</td>
<td>RESET Pulse Width after (V_{CC}) Stable</td>
<td>1</td>
<td></td>
<td>ms</td>
</tr>
<tr>
<td>(t_{45})</td>
<td>LID/WDTD, MENC, LPBK2, LPBK1 Setup to RTS Assert</td>
<td>10</td>
<td></td>
<td>(\mu)s</td>
</tr>
<tr>
<td>(t_{46})</td>
<td>LID/WDTD, MENC, LPBK2, LPBK1 Hold from RTS Deassert</td>
<td>10</td>
<td></td>
<td>(\mu)s</td>
</tr>
<tr>
<td>(t_{47})</td>
<td>LID/WDTD, MENC, LPBK2, LPBK1 Setup to RD Active(^{(1)})</td>
<td>10</td>
<td></td>
<td>(\mu)s</td>
</tr>
<tr>
<td>(t_{48})</td>
<td>LID/WDTD, MENC, LPBK2, LPBK1 Hold from Rd Inactive(^{(1)})</td>
<td>10</td>
<td></td>
<td>(\mu)s</td>
</tr>
</tbody>
</table>

**NOTE:**
1. Violation of this specification can result in corrupted data presented to the LAN controller. No other adverse affects will occur.

### JABBER TIMING CHARACTERISTICS

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Parameter</th>
<th>Min</th>
<th>Max</th>
<th>Units</th>
</tr>
</thead>
<tbody>
<tr>
<td>(t_{49})</td>
<td>RTS Assert to WDT Assert (TD Disabled)</td>
<td>20</td>
<td>150</td>
<td>ms</td>
</tr>
<tr>
<td>(t_{50})</td>
<td>Last RTS Deassert to WDT Deassert (TD Enabled)</td>
<td>250</td>
<td>750</td>
<td>ms</td>
</tr>
</tbody>
</table>

---

Figure 12. Transmit Timing: Non-Manchester Encoding

---

\(t_{28}\) \(t_{29}\) \(t_{26}\) \(t_{27}\)
Figure 13. Collision Timing

Figure 14. Modal Timing

LINK INTEGRITY TIMING CHARACTERISTICS

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Parameter</th>
<th>Min</th>
<th>Max</th>
<th>Units</th>
</tr>
</thead>
<tbody>
<tr>
<td>t_{51}</td>
<td>Last TD Edge to Link Test Pulse Transmission</td>
<td>8</td>
<td>24</td>
<td>ms</td>
</tr>
<tr>
<td>t_{52}</td>
<td>Link Test Pulse Width</td>
<td>80</td>
<td>120</td>
<td>ns</td>
</tr>
<tr>
<td>t_{53}</td>
<td>Last RD Edge to Link Fail (LI Assert)</td>
<td>50</td>
<td>150</td>
<td>ms</td>
</tr>
<tr>
<td>t_{54}</td>
<td>Minimum Idle Time between Consecutive Leakbeat Reception</td>
<td>6</td>
<td>8</td>
<td>ms</td>
</tr>
<tr>
<td>t_{55}</td>
<td>Maximum Idle Time between Consecutive Leakbeat Reception</td>
<td>24</td>
<td>150</td>
<td>ms</td>
</tr>
</tbody>
</table>

Figure 15. Jabber Timing
Figure 16. Link Integrity Timing

Figure 17. 82521TB Form Factor
The Intel 82504TA Transceiver Serial Interface component (TSI) is intended for Twisted Pair Ethernet LAN applications using 10 Mb/s, Manchester coded data; for example, client stations, file servers, and repeaters. The 82504TA reduces design time by providing the serial interface functions required to connect the twisted pair interface circuitry to any of Intel's Ethernet LAN controllers, including the 82586, 82590, and 82592. It offers LAN system designers an easy way to upgrade existing Ethernet/Cheapernet products to take advantage of low-cost twisted pair wire. The TSI chip performs clock recovery and Manchester decoding of 10 Mb/s data, and produces NRZ data and clock signals for the LAN controller. The TSI also supports a predistortion method to prevent line overcharge, improving jitter performance. The 82504TA is pin compatible with the AT&T T7210. It is fabricated using low-power CMOS processing technology, and is available in 24-lead plastic DIP and 28-lead SOJ packages.
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>VCCA</td>
<td>1</td>
<td></td>
<td>Analog VCC. +5V power supply.</td>
</tr>
<tr>
<td>RSV1</td>
<td>2</td>
<td>I</td>
<td>Reserved. This pin is reserved and must be connected to VSSD for proper operation.</td>
</tr>
<tr>
<td>TPS</td>
<td>3</td>
<td>I</td>
<td>Twisted Pair Sense. Active high. This pin is asserted when data is valid on TRxD (Twisted Pair Receive Data).</td>
</tr>
<tr>
<td>RESET</td>
<td>4</td>
<td>I</td>
<td>Reset. Active High. This pin is asserted to bring the TSI into a known state. It must be asserted for 1 ms while the clock is running.</td>
</tr>
<tr>
<td>TRxD</td>
<td>5</td>
<td>I</td>
<td>Twisted Pair Receive Data. Asynchronous Manchester data from the twisted pair line receiver.</td>
</tr>
<tr>
<td>RSV2</td>
<td>6</td>
<td>I</td>
<td>Reserved. This pin is reserved and must be connected to VSSD for proper operation.</td>
</tr>
</tbody>
</table>

NOTES:
I = Input
O = Output
<table>
<thead>
<tr>
<th>Symbol</th>
<th>Pin No.</th>
<th>Type</th>
<th>Name and Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>TxD</td>
<td>7</td>
<td>I</td>
<td>Transmit Data. Manchester encoded data from the 82586 (or other Ethernet controller). This pin is directly connected to the TxD controller output.</td>
</tr>
<tr>
<td>RTS</td>
<td>8</td>
<td>I</td>
<td>Request to Send. Active low. This signal is synchronous to TxC, and enables data transmission on the twisted pair link segment.</td>
</tr>
<tr>
<td>RSV3</td>
<td>9</td>
<td>I</td>
<td>Reserved. This pin is reserved and must be connected to VCCD for proper operation.</td>
</tr>
<tr>
<td>DTE</td>
<td>10</td>
<td>I</td>
<td>Data Terminal Equipment. This pin should be connected to VCCD if the TSI is used in a DTE, or to VSSD if used in a repeater.</td>
</tr>
<tr>
<td>CLK</td>
<td>11</td>
<td>I</td>
<td>Clock. A 20 MHz ± 0.01% input clock used for precision timing and encoded data transmission.</td>
</tr>
<tr>
<td>VCCD</td>
<td>12</td>
<td></td>
<td>Digital VCC. +5V Power Supply.</td>
</tr>
<tr>
<td>VSSD</td>
<td>13</td>
<td></td>
<td>Digital Ground.</td>
</tr>
<tr>
<td>TxC</td>
<td>14</td>
<td>O</td>
<td>Transmit Clock. A 10 MHz clock output tied directly to the TxC pin of the Intel Ethernet LAN Controller.</td>
</tr>
<tr>
<td>PDC</td>
<td>15</td>
<td>O</td>
<td>Predistortion Control. This signal is used to reduce jitter in a twisted pair environment by preventing line overcharge. This pin is asserted for the first 50 ns of any pulse on the TRMT pair. This allows the TP line drivers to reduce their output voltage during the last 50 ns of 100 ns Manchester pulses. PDC will not produce glitches during consecutive 50 ns pulses.</td>
</tr>
<tr>
<td>TRMT</td>
<td>16</td>
<td>O</td>
<td>Twisted Pair Transmit Pair. Serial Manchester encoded data generated for the twisted pair line drivers.</td>
</tr>
<tr>
<td>TRMT</td>
<td>17</td>
<td>O</td>
<td>Twisted Pair Transmit Pair. Serial Manchester encoded data generated for the twisted pair line drivers.</td>
</tr>
<tr>
<td>TPEN</td>
<td>18</td>
<td>O</td>
<td>Twisted Pair Enable. Active low. This pin enables the line drivers.</td>
</tr>
<tr>
<td>RxC</td>
<td>19</td>
<td>O</td>
<td>Receive Clock. A 10 MHz clock connected directly to the RxC input of the Intel Ethernet LAN Controller. This clock is the recovered clock from TRxD.</td>
</tr>
<tr>
<td>RxD</td>
<td>20</td>
<td>O</td>
<td>Receive Data. NRZ data passed to the Intel Ethernet LAN Controller. This pin is directly connected to the RxD pin on the controller.</td>
</tr>
<tr>
<td>CRS</td>
<td>21</td>
<td>O</td>
<td>Carrier Sense. Active low. A signal that alerts the Intel Ethernet LAN Controller that the twisted pair link is active. This pin is directly connected to the CRS input of the controller.</td>
</tr>
<tr>
<td>MCV</td>
<td>22</td>
<td>O</td>
<td>Manchester Code Violation. Active low. This signal indicates the presence of Manchester code violations.</td>
</tr>
<tr>
<td>NC</td>
<td>23</td>
<td></td>
<td>Not Connected.</td>
</tr>
<tr>
<td>VSSA</td>
<td>24</td>
<td></td>
<td>Analog Ground.</td>
</tr>
</tbody>
</table>
FUNCTIONAL DESCRIPTION

DTE MODE

Clock Generator

To clock the 82504TA TSI chip and provide the precision timings required in an IEEE 802.3 environment, a 20 MHz ± 0.01% clock is required. An internal divide-by-two counter generates the 10 MHz TXC signal. Several commercially available quartz crystal based clock oscillators meet these requirements. The following are two possible vendors.

- Fox Electronics
  5842 Corporation Circle
  Fort Myers, FL 33905
- M-Tron Industries, Inc.
  Yankton, SD 57078

An external TTL-compatible 20 MHz ± 0.01% clock with a duty cycle of 40/60 or better can also be used.

Transmit Section

MANCHESTER RETIMING AND PREDISTORTION

The transmit section of the 82504TA is controlled by the RTS signal generated by the Ethernet LAN controller (Intel's 82586, 82592, etc). When RTS is asserted, the 82504TA begins clocking in Manchester data from the controller on the TxD input. The TxD signal is sampled on every transition of the 10 MHz TXC signal. The serial data is then retimed by the 20 MHz input clock, and sent to the line drivers via the TRMT and TRMT pins. The enable signal for the line drivers, TPEN, asserts two bit times after the assertion of RTS to allow the input Manchester data to settle. At the end of the packet, TPEN remains asserted for three bit times to make allowance for device latency, and to append the end of packet symbol (IDL) to the data packet.

Another signal, Predistortion Control (PDC), is also generated by the transmit section. Predistortion is a technique for reducing jitter by preventing line overcharging during "fat" (100 ns) Manchester pulses. PDC is asserted during the first 50 ns of any pulse on the TRMT outputs; i.e., it is asserted throughout "thin" (50 ns) pulses and during the first half of "fat" pulses. This permits the twisted pair line driver to reduce its output drive during the second half of "fat" pulses. Internal circuitry prevents glitches on PDC.

APPLICATION EXAMPLE

The twisted-pair line driver (74ACT244) shown in Figure 1 is a rail-to-rail CMOS line driver. A resistive, voltage summing network is used to combine the individual line driver outputs into a differential signal having the required degree of predistortion. This signal is then fed through a protection circuit and an electromagnetic interference (EMI) filter. This reduces interference from the system and the TP wire, and to reduce crosstalk in bundled cables. Finally, isolation transformers and a common-mode choke are included for DC isolation and noise reduction.

Receive Section

MANCHESTER DECODING AND CLOCK RECOVERY

The Receive section of the 82504TA is enabled when incoming data from the twisted pair asserts the Twisted Pair Sense (TPS) signal. Manchester data decoding and clock recovery begin on the serial data from the Twisted Pair Receive Data (TRxD) input. RXC changes from its free running state to its locked state during the first two bit-times. CRS goes active after two bit times to guarantee reception of valid data after RXC clock stabilization. The decoded NRZ data is sent to the LAN controller on the RXD line along with the recovered clock signal, RXC.

The end of packet is detected by the presence of the IDL symbol or by the deassertion of TPS. After three bit times CRS will be deasserted synchronously with RXC, then RXC returns to its free running state.

To interface with a LAN controller that expects CRS to be asserted in response to its own transmission—Intel controllers are software configurable either way—the CRS signal from the TSI should be AND'd with the RTS signal from the controller (as shown in Figure 2); this way, CRS to the controller will assert during both transmission or reception. This is the normal mode of operation for coaxial Ethernet environments.

APPLICATION EXAMPLE

A typical DTE receiver design is shown in Figure 2. The incoming signal from the twisted pair wire passes through a common-mode choke and an isolation transformer for noise reduction. This signal runs through another filter. The filter output runs directly to a line receiver to establish a data channel, and through a DC offset to another line receiver for a squelch channel. The squelch channel is used for noise rejection, and detecting valid incoming...
data. The line receivers on both the data channel and squelch channel convert the differential signal to TTL-compatible signals.

When the incoming signal level is above the comparator's preset threshold, the comparator output triggers a Retriggerable Monostable Multivibrator. The multivibrator then asserts the TPS signal for the 82504TA. The TPS signal remains asserted for two bit times past the last input transition.

The TPS signal is used to gate the data-channel line driver from the TRxD signal to ensure proper operation. Further, the VIH level of TRxD should be held between 1.8V and 2.4V. In the example shown in Figure 2 this is accomplished by using a 100Ω pull-down resistor on the output of the AND gate (74F08), which is used to gate the data channel with TPS.

**Collision Detect**

Collision detection in the twisted pair environment occurs from simultaneous transmission and reception on the twisted pair wires. This is indicated by the assertion of both TPS and RTS. The simple logic circuit shown in Figure 2 can detect such collisions.

**Interface Example**

Figure 3 shows a typical DTE implementation circuit. When designing this type of circuit, considerable attention must be paid to power supply noise reduction, capacitive decoupling, and the layout of the line driver/receiver to the interface connector (RJ-45).

**REPEATER MODE**

**Operation in Repeater Mode**

The 82504TA can be used when the DTE pin is not asserted. There are two principal differences in this mode of operation. First, the deassertion of CRS is not synchronized to RxC—this, on the average, allows CRS to deassert one bit time earlier. Second, TPEN assertion occurs two bit times earlier than in DTE mode.

**Application Example**

A repeater design using the 82505TA Multiport Repeater controller (MPR) requires the services of an 82504 TSI. The TSI provides the Manchester decoder and clock recovery for the MPR. Figure 4 shows the appropriate interface circuitry.
Figure 1. Transmit Section

Figure 2. Receive Section
Figure 3. DTE Interface Application Diagram

Figure 4. Repeater Interface Application Diagram
**ABSOLUTE MAXIMUM RATINGS**

Ambient Operating Temperature (TA) ..................... 0°C to +70°C
Storage Temperature ..................... −40°C to +125°C
Power Dissipation ..................... 400 mW
Voltage on any Pin with Respect to Ground . . . . −0.5V to VCC + 0.5V

**NOTICE:** This data sheet contains preliminary information on new products in production. The specifications are subject to change without notice.

**WARNING:** Stressing the device beyond the “Absolute Maximum Ratings” may cause permanent damage. These are stress ratings only. Operation beyond the “Operating Conditions” is not recommended and extended exposure beyond the “Operating Conditions” may affect device reliability.

**CHARACTERISTICS**

**DC Characteristics**

\( T_A = 0°C \) to \(+70°C\), \( V_{CC} = 5V \pm 10\%\), \( V_{SS} = 0.0V \)

<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>( V_{IL} )</td>
<td>Input Low Voltage</td>
<td>−0.5</td>
<td>0.6</td>
<td>V</td>
<td></td>
</tr>
<tr>
<td>( V_{IH} )</td>
<td>Input High Voltage</td>
<td>2.0</td>
<td>( V_{DD} + 0.5 )</td>
<td>V</td>
<td>( I_{OL} = 25 ) mA</td>
</tr>
<tr>
<td>( V_{OL} )</td>
<td>Output Low Voltage</td>
<td>0.5</td>
<td>V</td>
<td>IOL = 25 mA</td>
<td></td>
</tr>
<tr>
<td>( V_{OH} )</td>
<td>Output High Voltage</td>
<td>2.4</td>
<td>V</td>
<td>IOH = −25 mA</td>
<td></td>
</tr>
<tr>
<td>( I_{CC} )</td>
<td>Power Supply Current (No Load)*</td>
<td>40</td>
<td>mA</td>
<td>( V_{CC} = 5.0) V</td>
<td></td>
</tr>
<tr>
<td>( I_{CC} )</td>
<td>Power Supply Current (Load)</td>
<td>80</td>
<td>mA</td>
<td>( V_{CC} = 5.0) V</td>
<td></td>
</tr>
<tr>
<td>( I_{LI} )</td>
<td>Input Leakage Current</td>
<td>10</td>
<td>( \mu A )</td>
<td>( V_{IH} = 5.5) V</td>
<td></td>
</tr>
<tr>
<td>PD</td>
<td>Power Dissipation (No Load)*</td>
<td>0.20</td>
<td>W</td>
<td>( V_{CC} = 5.0) V</td>
<td></td>
</tr>
<tr>
<td>PD</td>
<td>Power Dissipation (Load)*</td>
<td>0.4</td>
<td>W</td>
<td>( V_{CC} = 5.0) V</td>
<td></td>
</tr>
<tr>
<td>( t_{r} )</td>
<td>Output Rise and Fall Time</td>
<td>5</td>
<td>ns</td>
<td>( C_{LOAD} = 20 ) pF</td>
<td></td>
</tr>
<tr>
<td>( \tau_{f} )</td>
<td>Output Rise and Fall Time</td>
<td>5</td>
<td>ns</td>
<td>( C_{LOAD} = 20 ) pF</td>
<td></td>
</tr>
</tbody>
</table>

*Not including excessive output buffer loads.*
## Decoder Timing Characteristics
(Measurements are from 50% points, unless otherwise noted.)

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Parameter</th>
<th>Min</th>
<th>Max</th>
<th>Units</th>
</tr>
</thead>
<tbody>
<tr>
<td>t1</td>
<td>RXC High to CRS Asserted</td>
<td>3</td>
<td>19</td>
<td>ns</td>
</tr>
<tr>
<td>t2</td>
<td>CRS Asserted to RXC Low</td>
<td>17</td>
<td>40</td>
<td>ns</td>
</tr>
<tr>
<td>t3</td>
<td>DTE: RXC High to CRS Deasserted</td>
<td>3</td>
<td>19</td>
<td>ns</td>
</tr>
<tr>
<td>t4</td>
<td>DTE: CRS Deasserted to RXC Low</td>
<td>20</td>
<td>65</td>
<td>ns</td>
</tr>
<tr>
<td>t5</td>
<td>DTE: CRS Deasserted to RXC High</td>
<td>195</td>
<td>345</td>
<td>ns</td>
</tr>
<tr>
<td>t6</td>
<td>RXC High Pulse Width as Captured Data Clock</td>
<td>36</td>
<td>45</td>
<td>ns</td>
</tr>
<tr>
<td>t7</td>
<td>RXC Low Pulse Width as Captured Data Clock</td>
<td>38</td>
<td>80</td>
<td>ns</td>
</tr>
<tr>
<td>t8</td>
<td>RXC Period as Captured Data Clock</td>
<td>78</td>
<td>124</td>
<td>ns</td>
</tr>
<tr>
<td>t9</td>
<td>RXC High Pulse Width as Free Oscillating Clock</td>
<td>43</td>
<td>73</td>
<td>ns</td>
</tr>
<tr>
<td>t10</td>
<td>RXC Low Pulse Width as Free Oscillating Clock</td>
<td>172</td>
<td>276</td>
<td>ns</td>
</tr>
<tr>
<td>t11</td>
<td>RXC Period as Free Oscillating Clock</td>
<td>215</td>
<td>349</td>
<td>ns</td>
</tr>
<tr>
<td>t12</td>
<td>RXD Transition to RXC High</td>
<td>-5</td>
<td>5</td>
<td>ns</td>
</tr>
<tr>
<td>t13</td>
<td>RXC Low to RXD Transition</td>
<td>30</td>
<td>85</td>
<td>ns</td>
</tr>
<tr>
<td>t14</td>
<td>RXD Transition to RXC Low</td>
<td>30</td>
<td>50</td>
<td>ns</td>
</tr>
<tr>
<td>t15</td>
<td>TRxD Midbit Transition to RXC Low</td>
<td>86</td>
<td>130</td>
<td>ns</td>
</tr>
<tr>
<td>t16</td>
<td>TPS Asserted to TRxD Sampled</td>
<td>-5</td>
<td>20</td>
<td>ns</td>
</tr>
<tr>
<td>t17</td>
<td>TRxD Preamble Transition to CRS Asserted</td>
<td>53</td>
<td>95</td>
<td>ns</td>
</tr>
<tr>
<td>t18</td>
<td>DTE: Beginning of IDL to CRS Deasserted (Last Bit = 0)</td>
<td>230</td>
<td>320</td>
<td>ns</td>
</tr>
<tr>
<td></td>
<td>DTE: Beginning of IDL to CRS Deasserted (Last Bit = 1)</td>
<td>280</td>
<td>390</td>
<td>ns</td>
</tr>
<tr>
<td>t19</td>
<td>Repeater: Beginning of IDL to CRS Deasserted (Last Bit = 0)</td>
<td>180</td>
<td>220</td>
<td>ns</td>
</tr>
<tr>
<td></td>
<td>Repeater: Beginning of IDL to CRS Deasserted (Last Bit = 1)</td>
<td>170</td>
<td>220</td>
<td>ns</td>
</tr>
<tr>
<td>t20</td>
<td>Midbit to Midbit Transition on TRxD</td>
<td>80</td>
<td>120</td>
<td>ns</td>
</tr>
<tr>
<td>t21</td>
<td>Boundary to Midbit Transition on TRxD</td>
<td>30</td>
<td>70</td>
<td>ns</td>
</tr>
</tbody>
</table>

**Negative-Transition CRS Timing Relative to RXC**

![Negative-Transition CRS Timing Relative to RXC](image)
Latency-Definition Timing Measurements

Start-of-Packet Timing Measurements

DTE End-of-Normal-Packet Timing Measurements

Repeater CRS Timing Relative to IDL
**Encoder Timing Characteristics**

(Measurements are from 50% points, unless otherwise noted. The input duty cycle requirement for CLK is 60%/40%.)

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Parameter</th>
<th>Min</th>
<th>Max</th>
<th>Units</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>DTE TRANSMISSION</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>t22</td>
<td>Data Clocked to TxD to Data at Output TRMT (Latency)</td>
<td>100</td>
<td>190</td>
<td>ns</td>
</tr>
<tr>
<td>t23</td>
<td>RTS Assertion Clocked to TPEN Assertion*</td>
<td>340</td>
<td>440</td>
<td>ns</td>
</tr>
<tr>
<td>t24</td>
<td>TxD Setup Time with Respect to TxC Transition</td>
<td>10</td>
<td></td>
<td>ns</td>
</tr>
<tr>
<td>t25</td>
<td>RTS Setup Time with Respect to TxC Transition</td>
<td>10</td>
<td></td>
<td>ns</td>
</tr>
<tr>
<td>t26</td>
<td>TxD Hold Time with Respect to TxC Transition</td>
<td>0</td>
<td></td>
<td>ns</td>
</tr>
<tr>
<td>t27</td>
<td>RTS Hold Time with Respect to TxC Transition</td>
<td>0</td>
<td></td>
<td>ns</td>
</tr>
<tr>
<td>t28</td>
<td>RTS Deassertion Clocked to TPEN Deassertion†</td>
<td>340</td>
<td>440</td>
<td>ns</td>
</tr>
<tr>
<td><strong>REPEATER</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>t29</td>
<td>Data Clocked to TxD to Data at Output TRMT (Latency)</td>
<td>100</td>
<td>210</td>
<td>ns</td>
</tr>
<tr>
<td>t30</td>
<td>RTS Assertion Clocked to TPEN Assertion‡</td>
<td>140</td>
<td>260</td>
<td>ns</td>
</tr>
<tr>
<td>t31</td>
<td>TxD Setup Time with Respect to CLK High</td>
<td>20</td>
<td></td>
<td>ns</td>
</tr>
<tr>
<td>t32</td>
<td>RTS Setup Time with Respect to CLK High</td>
<td>20</td>
<td></td>
<td>ns</td>
</tr>
<tr>
<td>t33</td>
<td>TxD Hold Time with Respect to CLK High</td>
<td>0</td>
<td></td>
<td>ns</td>
</tr>
<tr>
<td>t34</td>
<td>RTS Hold Time with Respect to CLK High</td>
<td>0</td>
<td></td>
<td>ns</td>
</tr>
<tr>
<td>t35</td>
<td>RTS Deassertion Clocked to TPEN Deassertion†</td>
<td>340</td>
<td>460</td>
<td>ns</td>
</tr>
<tr>
<td><strong>OUTPUT CHARACTERISTICS</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>t36</td>
<td>Maximum Deviation from the Ideal 50 ns Strobe Point for TPEN, TRMT/TRMT, and PDC</td>
<td></td>
<td></td>
<td>1.5</td>
</tr>
<tr>
<td>t37-t38</td>
<td>TRMT/TRMT Worst Case Duty Cycle Mismatch, 10 pF Load</td>
<td></td>
<td></td>
<td>−3</td>
</tr>
<tr>
<td>t39</td>
<td>TxC High Time</td>
<td></td>
<td></td>
<td>40</td>
</tr>
<tr>
<td>t40</td>
<td>TxC Low Time</td>
<td></td>
<td></td>
<td>40</td>
</tr>
</tbody>
</table>

**NOTES:**

* DTE start-of-packet delay: 2.5 bit times of data are masked after RTS is asserted by delaying TEN assertion.
† End of Packet: 2.5 bit times of data are transmitted beyond RTS deassertion by allowing TEN to remain asserted.
‡ Repeater start-of-packet delay: 0.5 bit times of data are masked after RTS is asserted, by delaying TEN assertion.
DTE Start-of-Packet

- TXC
- RTS
- RTS DE-ASSERTION Clocked
- RTS DE-ASSERTED
- DATA IS CLOCKED
- TXD
- DATA IN
- TPEN
- t23
- TRMT
- DATA OUT
- TRMT
- PDC

DTE TxD and RTS Setup Time

- TXC
- TXD
- t26
- t24
- t27
- RTS
- t25

1-295
DTE End-of-Packet

RTS DE-ASSERTION CLOCKED

RTS DE-ASSERTED

TXD

TPEN

TRMT

TRMT

PDC

Repeater Start-of-Packet

RTS ASSERTION CLOCKED

RTS ASSERTED

DATA IS CLOCKED

DATA IN

DATA OUT

TXD

TPEN

TRMT

TRMT

PDC
Repeater TxD and RTS Setup Time

CLK

TXD

RTS

Repeater End-of-Packet

CLK

RTS

TXD

TPEN

TRMT

TRMT

PDC
OUTLINE DIAGRAM

24-Pin Plastic DIP
Dimensions are in inches and (millimeters)

INDEX MARK
INDEX AREA

SEATING PLANE

0.125 (3.17) MIN
0.065 (1.65) MAX
0.032 (0.81) MAX
0.100 (2.54) REF

24

1.270 (32.26) MAX
0.555 (14.10) MAX

28-Pin Plastic SOJ
Dimensions are in inches and (millimeters)

CODE MARK ON TOP SURFACE
CODING IS RIGHT SIDE UP WHEN
LEAD 1 IS IN LOWER LEFT SIDE

5° to 7° TYP

0.094 (2.39) MAX
0.125 (3.18) MAX

0.50 (1.27)

0.024 (0.61) MAX
0.020 (0.81) MAX
0.024 (0.61) MAX
0.040 (1.02) MAX

0.347 (8.81) MAX
0.300 (7.62) MAX
0.041 (1.04) MAX

0.296 ± 0.002 (7.66)

0.600 (15.24) BSC

0.700 (17.78) MAX

0.010 (0.254)

0.100 (2.54)

0.165 (4.19) MAX
0.015 (0.38) MIN
0.085 (2.16) MAX

0.100 (2.54)

0.50 (1.27)

SEE ENLARGED DETAIL

SEE ENLARGED DETAIL

1-299
82505TA
MULTIPORT REPEATER CONTROLLER (MPR)

- Complies with IEEE 802.3 CSMA/CD Standard for Repeaters (Std ANSI/IEEE 802.3c-1988)
- 10-Mb/s Operation
- Allows Up to Eleven Twisted Pair Ports and One AUI Port
- Supports Up to Four Cascaded Repeaters
- Automatic Preamble Regeneration
- Auto-Partitioning for System Fault Isolation
- Minimum Frame-Length Enforcement (96 bits)
- Performs Manchester Encoding of Transmitted Data
- Pin-Selectable FIFO Fill Level
- Jam Signal Generation
- Eight-Bit Blinding Timer at End of Transmission
- LED Output Control of Critical Network Parameters
  - Traffic Status
  - Jam Status
  - Per-Port Jabber Status
  - FIFO Error Status
- Single 5-V Supply, and Low-Power CMOS Processing
- Pin Compatible with the AT&T T7200

The 82505TA Multiport Repeater controller (MPR) is a VLSI device designed for use in 10-Mb/s CSMA/CD Twisted Pair Ethernet repeaters. The 82505TA combines with a single 82504TA Transceiver Serial Interface (TSI) chip to handle all necessary multiport repeater functions. The MPR controller provides automatic preamble regeneration to minimize bit loss, Manchester encoding of transmitted data, jam signal generation, and minimum frame length enforcement (96 bits). The MPR supports fault isolation by providing lockup control and auto-partitioning jabber timing. The MPR/TSI combination supports up to eleven twisted pair ports for direct connection to twisted pair client stations, file servers, repeaters, bridges, and gateways. In addition, the set supports one AUI port for interfacing twisted pair networks to existing Ethernet (IEEE 802.3 TYPE 10BASE5) or Cheapernet (IEEE 802.3 TYPE 10BASE2) networks. The MPR offers pin selectable FIFO fill levels and LED output control of traffic status, jam status, per-port jabber status, and FIFO error status for simplified network management and diagnostics. The 82505TA Multiport Repeater controller is fabricated using low-power CMOS technology, and is available in a 68-lead plastic leaded chip carrier package (PLCC).
### PIN DESCRIPTIONS

<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>Not Connected.</td>
</tr>
<tr>
<td>FILL</td>
<td>2</td>
<td>I</td>
<td>FIFO Fill Level: This pin controls the number of bits loaded into the internal or external FIFO before the bits are unloaded. When low, the fill level is seven. When high, the fill level is eight. This pin is connected to an internal pull-up device.</td>
</tr>
<tr>
<td>NC</td>
<td>3-7</td>
<td>—</td>
<td>Not Connected.</td>
</tr>
<tr>
<td>TESTO0</td>
<td>8</td>
<td>O</td>
<td>Device Test Outputs: These pins are used in conjunction with TEST0 and TEST1 to facilitate device testing. During normal operation these pins are not connected.</td>
</tr>
<tr>
<td>TESTO1</td>
<td>9</td>
<td>O</td>
<td></td>
</tr>
<tr>
<td>VSS</td>
<td>10</td>
<td>—</td>
<td>Ground: 0.0V. All ground pins must be connected together.</td>
</tr>
<tr>
<td>PD0–PD3</td>
<td>11–14</td>
<td>O</td>
<td>Port to Disable: Address of port to disable when traffic is received.</td>
</tr>
<tr>
<td>PDCTL</td>
<td>15</td>
<td>O</td>
<td>Port Disable Control: When low it indicates that the port selected by PD0–PD3 is to be disabled. When high it indicates that all ports should be enabled. The Port Disable address is invalid when PDCTL is high. This pin remains low as long as the Port Disable address pins are valid.</td>
</tr>
<tr>
<td>L0</td>
<td>16</td>
<td>O</td>
<td>LED Status Indicator: Part of address bus for LED status indicators.</td>
</tr>
<tr>
<td>VCC</td>
<td>17</td>
<td>—</td>
<td>5-V Supply: All VCC pins must be connected together.</td>
</tr>
<tr>
<td>L1–L3</td>
<td>18–20</td>
<td>O</td>
<td>LED Status Indicators: Part of Address bus for LED Status Indicators.</td>
</tr>
<tr>
<td>LEDCTL</td>
<td>21</td>
<td>O</td>
<td>LED Control: When low it indicates that the LED selected by L0–L3 is turned on. When high it indicates that the LED is turned off.</td>
</tr>
<tr>
<td>LEDSTRB</td>
<td>22</td>
<td>O</td>
<td>LED Strobe: This pin pulses low when the LED address and control pins are valid and an LED status is updated.</td>
</tr>
<tr>
<td>CLKOUT</td>
<td>23</td>
<td>O</td>
<td>20-MHz TTL Clock Output: This pin is a buffered version of CLK.</td>
</tr>
<tr>
<td>VSS</td>
<td>24</td>
<td>—</td>
<td>Ground: 0.0 V. All ground pins must be connected together.</td>
</tr>
<tr>
<td>TPS</td>
<td>25</td>
<td>O</td>
<td>Twisted Pair Sense: This pin indicates presence of carrier to the TSI. It is high while valid Manchester data is being received. If the repeater is sending jam, or is blinding inputs, this pin is driven low.</td>
</tr>
<tr>
<td>TRxD</td>
<td>26</td>
<td>O</td>
<td>TSI Received Data: Manchester data from the repeater front-end to the Manchester decoder.</td>
</tr>
<tr>
<td>TRMT</td>
<td>27</td>
<td>O</td>
<td>Transmit Output: Retimed Manchester complement to all ports (including the AUI port).</td>
</tr>
<tr>
<td>TRMT</td>
<td>28</td>
<td>O</td>
<td>Transmit Output: Retimed Manchester data to all ports (including the AUI port).</td>
</tr>
<tr>
<td>PDC</td>
<td>29</td>
<td>O</td>
<td>Predistortion Control: Active low. This signal is used to reduce jitter in a twisted pair environment by preventing overcharge. This pin is asserted for the first 50 ns of any pulse on the TRMT pair. This allows the T-P line drivers to reduce their output voltage during the last 50 ns of 100-ns Manchester pulses. PDC will not produce glitches during consecutive 50-ns pulses.</td>
</tr>
</tbody>
</table>
### PIN DESCRIPTIONS (Continued)

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Pin No.</th>
<th>Type</th>
<th>Name and Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>TPEN</td>
<td>30</td>
<td>O</td>
<td><strong>Transmit Port Enable:</strong> TPEN is low when TRMT and TRMT contain valid data, jam, or IDL.</td>
</tr>
<tr>
<td>VSS</td>
<td>31</td>
<td>—</td>
<td><strong>Ground:</strong> 0.0 V. All ground pins must be connected together.</td>
</tr>
<tr>
<td>CLK</td>
<td>32</td>
<td>I</td>
<td><strong>System Clock:</strong> 20-MHz ±0.01%, 50% nominal, 40/60% worst-case duty cycle.</td>
</tr>
<tr>
<td>TEST0</td>
<td>33</td>
<td>I</td>
<td><strong>Device Test Control:</strong> If either pin is low, internal test circuitry is enabled to facilitate device testing. If both pins are high the repeater operates normally. These pins are connected to internal pull-up devices.</td>
</tr>
<tr>
<td>TEST1</td>
<td>34</td>
<td>I</td>
<td></td>
</tr>
<tr>
<td>RESET</td>
<td>35</td>
<td>I</td>
<td><strong>Device Reset (Schmitt Input):</strong> A high on the pin causes the device to reset. RESET must be low for normal operation.</td>
</tr>
<tr>
<td>AUICRS</td>
<td>36</td>
<td>I</td>
<td><strong>AUI Carrier Sense (Schmitt input):</strong> A high on this pin indicates the presence of a carrier on the AUI port. AUICRS must be active high for at least 2.5 ± 0.5 successive 2X clock samples (1 to 1.5 bits) for the repeater to recognize valid AUI carrier.</td>
</tr>
<tr>
<td>TCS0-TCS10</td>
<td>37–47</td>
<td>I</td>
<td><strong>Twisted Pair Carrier Sense (Schmitt input):</strong> A high on any of these pins indicates the presence of carrier on that port. TCSn must be active high for at least 2.5 ± 0.5 successive 2X clock samples (1 to 1.5 bits) for the repeater to recognize valid T-P carrier.</td>
</tr>
<tr>
<td>AUICDT</td>
<td>48</td>
<td>I</td>
<td><strong>AUI Collision Detected:</strong> A low on this pin indicates the presence of a collision at the AUI port. AUICDT must be active low for at least 1.5 ± 0.5 successive 2X clock samples (0.5 to 1.0 bits) for the repeater to recognize valid AUI collision.</td>
</tr>
<tr>
<td>AUIRxD</td>
<td>49</td>
<td>I</td>
<td><strong>AUI Receive Data:</strong> Received Manchester data from the AUI line receiver.</td>
</tr>
<tr>
<td>TRD9–TRD10</td>
<td>50–51</td>
<td>I</td>
<td><strong>Twisted Pair Receive Data:</strong> Received Manchester data from the twisted pair line receivers.</td>
</tr>
<tr>
<td>VDD</td>
<td>52</td>
<td>—</td>
<td><strong>5-V Supply:</strong> Pin 17 must be connected to this pin.</td>
</tr>
<tr>
<td>TRD0–TRD8</td>
<td>53–61</td>
<td>I</td>
<td><strong>Twisted Pair Receive Data:</strong> Received Manchester data from the twisted pair line receivers.</td>
</tr>
<tr>
<td>VSS</td>
<td>62</td>
<td>—</td>
<td><strong>Ground:</strong> 0.0 V. All ground pins must be connected together.</td>
</tr>
<tr>
<td>CRS</td>
<td>63</td>
<td>I</td>
<td><strong>Carrier Sense:</strong> A low on this pin indicates that the Manchester decoder (TSI) is receiving a valid packet.</td>
</tr>
<tr>
<td>RxC</td>
<td>64</td>
<td>I</td>
<td><strong>Receive Clock:</strong> Recovered clock from the TSI decoder.</td>
</tr>
<tr>
<td>RxD</td>
<td>65</td>
<td>I</td>
<td><strong>Receive Data:</strong> Recovered NRZ data from the TSI.</td>
</tr>
<tr>
<td>MCV</td>
<td>66</td>
<td>I</td>
<td><strong>Manchester Code Violation:</strong> A low on this pin indicates that a Manchester violation was detected by the TSI. CDT must be active low for at least 1.5 ± 0.5 successive 2X clock samples (0.5 to 1.0 bits) for the repeater to recognize collision. The repeater enters the transmit collision global state when a violation is detected.</td>
</tr>
<tr>
<td>NC</td>
<td>67–68</td>
<td>—</td>
<td><strong>Not Connected.</strong></td>
</tr>
</tbody>
</table>
FUNCTIONAL DESCRIPTION

The Use of State Diagrams

State diagrams are used throughout the Functional Description section of this data sheet to facilitate concise descriptions. The state diagrams are modelled after—and use the same terminology—those used in the CSMA/CD repeater standard (ANSI/IEEE Std 802.3c-1988). Each state diagram is assumed to represent an independent process; each process communicates by using interprocess flags. Furthermore, the state diagrams are intended to convey the external operation of the MPR, they do not necessarily describe the internal circuitry. For example, the state diagrams imply independent Transmit timers for each port, while in fact, only one timer is used.

The 82505TA state diagrams adhere to the IEEE standard as closely as possible; however, several departures from the standard have been made to account for the inclusion of some of the Twisted Pair MAU’s internal functions. Should the state diagrams conflict with the text in this section, the state diagrams should be given preference.

TIMERS

Several timers and counters are implemented in the 82505TA MPR, they are described in the following two sections.

Timer Tw1. Tw1 is the wait timer for the end-of-transmit recovery time (blinding timer), its duration is eight bit times. When the repeater finishes transmitting a packet, Tw1 prevents the repeater from receiving that transmission as a new activity.

Timer Tw2. Tw2 is the wait timer for the end-of-carrier recovery time, its duration is three bit times. When AUICDT is detected making a positive transition, Tw2 prevents the repeater from prematurely detecting the real end-of-collision signal.

Timer Tw3. Tw3 is the wait timer for length of continuous output, its duration is 65,536 bit times. It is started when transmission of a packet begins. If Tw3 expires before transmission of the packet is completed the repeater enters the MAU jabber lockup-protection condition, and interrupts the transmission for a period equal to Tw4.

Timer Tw4. Tw4 is the wait timer for time to disable output for jabber lockup protection, its duration is 96 bit times. When Tw4 is active, transmission to all ports is suspended until the timer expires. The MAU lockup LED is turned on at the next LED counter interval, this indicates that transmission is suspended.

Timer Tw5. Tw5 is the wait timer for length of packet without collision, its duration is 512 bit times. It is started when a port becomes active. If a collision is detected before Tw5 expires, the collision count for that port is augmented, and the port Tw6 is begun. A separate Tw5 is implemented for each port, including the AUI port.
Timer Tw6. Tw6 is the wait timer for excessive length of collision, its duration is 1024 bit times. It is begun if a collision is detected before Tw5 expires. If the collision condition still exists when Tw6 expires, the port on which the violation occurred is partitioned (the receiver is disabled). A separate Tw6 is implemented for each port, including the AUI port.

COUNTERS

Counter CC. CC, the collision counter, maintains a record of the number of consecutive collisions for a particular port. If the collision limit is reached on a port, that port is partitioned (the receiver is disabled). A separate CC, with a limit of 31, is implemented on each port, including the AUI port.

Counter TT. TT, the transmit timer counter, maintains a record of the number of bits transmitted to any given port; its duration is 96 bit times. If the total number of bits transmitted to a port is less than 96 (due to the reception of a fragmented packet), the repeater will enter the receive collision global state and will transmit jam until TT expires, thus extending the frame to ≥ 96 bits.

The TT counter is cleared when the repeater enters the transmit collision global state. This ensures that at least 96 bits of jam are transmitted to all ports before the repeater enters the one remaining port, or blind states from the transmit collision state.

Global State Machine

A single global state machine is implemented for the MPR. The state diagram assumes multiple twisted pair ports and one AUI port.

Auto-Partition and Reconnection

The optional auto-partition and reconnection algorithm described in SC. 9.6.6.2 of the ANSI/IEEE Std 802.3c-1988 is implemented in the 82505TA MPR chip. Each port, including the AUI port, is provided with an individual partition state machine. The state machine for the AUI port corresponds to that described in the standard. The machines for the T-P ports have been modified to reflect the inclusion of several MAU functions; the operation of the machine at the MDI interface remains unchanged.

RESET

The repeater unit is reset when RESET (pin 35) is asserted high. When the 82505TA is reset, the repeater unit disconnects all ports and performs a lamp test by cycling through each LED address with LEDCTL low. All LEDs will remain on (lamp test state) for as long as RESET is held high. Minimum RESET high is 2 µs (40 CLK cycles), to ensure a device reset. The 2 µs also allow the repeater unit to cycle through each LED address at least once.

At the end of a reset (RESET goes low), all jabber, collision, and FIFO error indicators are turned off, but the traffic status indicator is left on (it blinks when packets arrive).

The repeater unit is fully operational when it exits reset.

Automatic Preamble Generation

Automatic Preamble Generation (APG) prevents the preamble from shrinking as a frame is passed from repeater to repeater. This shrinking, or loss of bits, is due to the bit cost of determining the presence of a carrier and synchronizing the Manchester data for NRZ data and clock recovery.

The APG circuit compensates for the bit loss by beginning transmission of new preamble bits before the FIFO limit is reached. When bits from the incoming frame reach the FIFO limit they are synchronously switched into the awaiting pretransmitted preamble. If the logic polarity of the first bit out of the FIFO is not the value expected, an extra preamble bit is transmitted. This prevents corruption of the preamble pattern when the transmitted bit stream is switched from the APG generator to the FIFO data.

The delay between carrier detection and start of APG depends on the FIFO fill level selected, and the type of active port.

The delay for a T-P port is such that the number of preamble bits added by the repeater is equal to the number of bits (± 1 bit) lost while detecting the frame. That is, the latency of bits through the repeater equals the delay of preamble start introduced by the repeater. If the FIFO fill level (FILL) is changed, the delay for start of APG is automatically changed to compensate for the new latency of the FIFO.

For the AUI port, the bit loss in detecting the frame can range from one to eight bits. Therefore, the beginning of APG caused by the AUI port is dependent only on a carrier detection, and is not delayed. This allows the repeater unit to recover a maximum of three bits lost by the attached MAU in frame detection.

The leading edge of the first preamble bit transmitted by the repeater (as seen on the line) indicates the beginning of a 100-ns positive voltage (TTL logic 1).
LED STATUS INDICATOR ADDRESSES

<table>
<thead>
<tr>
<th>L3</th>
<th>L2</th>
<th>L1</th>
<th>L0</th>
<th>Address</th>
<th>Description</th>
<th>Asserted LED State</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>TPP 0 Partition Jabber</td>
<td>On</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>TPP 1 Partition Jabber</td>
<td>On</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>2</td>
<td>TPP 2 Partition Jabber</td>
<td>On</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>3</td>
<td>TPP 3 Partition Jabber</td>
<td>On</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>4</td>
<td>TPP 4 Partition Jabber</td>
<td>On</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>5</td>
<td>TPP 5 Partition Jabber</td>
<td>On</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>6</td>
<td>TPP 6 Partition Jabber</td>
<td>On</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>7</td>
<td>TPP 7 Partition Jabber</td>
<td>On</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>8</td>
<td>TPP 8 Partition Jabber</td>
<td>On</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>9</td>
<td>TPP 9 Partition Jabber</td>
<td>On</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>A</td>
<td>TPP 10 Partition Jabber</td>
<td>On</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>B</td>
<td>AUI Partition Jabber</td>
<td>On</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>C</td>
<td>FIFO Error</td>
<td>Off</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>D</td>
<td>Traffic</td>
<td>Off</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>E</td>
<td>Jam*</td>
<td>On</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>F</td>
<td>MAU Lockup Protection</td>
<td>On</td>
</tr>
</tbody>
</table>

* This indicates the transmission of jam for collision, packet fragments, and FIFO errors.

LED Controller

The LED controller contains a 21-bit counter (105 ms at 20 MHz). At each 105-ms interval, the controller loads the status of each event into a shift register and shifts the status out as LEDCTL. If the port indicates a change in status from the previous interval, the appropriate LED is toggled. Thus, for each event, the minimum time an LED is on or off is 105 ms, and the LED will not change state until an additional 105 ms have elapsed.

The status indicators, with the exception of the jabber indicators (address 0 to 11), have a 50% duty cycle when they are asserted (105 ms on, 105 ms off). The jabber indicators will remain on for as long as the affected ports are partitioned (Receive disabled).

The following table shows the addressing used for the LED status indicators. An LED is turned on by a negative pulse on LEDSTRB when LEDCTL is low. An LED is turned off by a negative pulse on LEDSTRB when LEDCTL is high.

A status LED interface using two 74LS259 addressable latches is shown in the following figure.

The upper LED address bit (L3) is gated with LEDSTRB to provide the strobe signal to each 74LS259 device. The CLEAR input is tied high since the repeater will initialize the latches during the LED lamp test.

Port Disable Controller

The Port Disable Controller determines which port is receiving valid data, and outputs an address associated with that port. External circuitry uses this address to disable outgoing traffic on that port.

The controller is designed to be used with a 4-to-16 line decoder, with the address pins connected to the address inputs, and PDCTL and TPEN connected to the gating inputs.

If PDCTL is high the address is not valid and all ports should be enabled. PDCTL goes active low after the repeater unit outputs a valid port address, and remains low for as long as that port address is valid.

If the repeater is sending jam to all ports but one, and then must send jam to all the ports, the positive-going edge of PDCTL will coincide with the beginning of a 100-ns positive voltage (TTL logic 1) at the TRMT output pin. The following table shows the address associated with each port.
## PORT DISABLE ADDRESSES

<table>
<thead>
<tr>
<th>PD3</th>
<th>PD2</th>
<th>PD1</th>
<th>PD0</th>
<th>Address</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>Disable T-P Port 0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>Disable T-P Port 1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>2</td>
<td>Disable T-P Port 2</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>3</td>
<td>Disable T-P Port 3</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>4</td>
<td>Disable T-P Port 4</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>5</td>
<td>Disable T-P Port 5</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>6</td>
<td>Disable T-P Port 6</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>7</td>
<td>Disable T-P Port 7</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>8</td>
<td>Disable T-P Port 8</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>9</td>
<td>Disable T-P Port 9</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>A</td>
<td>Disable T-P Port 10</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>B</td>
<td>Disable AUI Port</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>C</td>
<td>Not Assigned</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>D</td>
<td>Not Assigned</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>E</td>
<td>Not Assigned</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>F</td>
<td>Not Assigned</td>
</tr>
</tbody>
</table>

---

**LED Interface Using Two 'LS259 Addressable Latches**
Global State Diagram for Multiple TP Ports and One AUI Port
Partitioning State Diagram for AUI Port

- **RESET**
  - COUNT CLEAR = 0
  - CC(AUI) = 0
  - Datain(AUI) = DIPresent(AUI)
  - Collin(AUI) = CIPresent(AUI)
  - DIPresent(AUI) = Idle
  - CIPresent(AUI) = SQE

- **PARTITION WAIT**
  - Datain(AUI) = Idle
  - Collin(AUI) = SQE
  - DIPresent(AUI) = Idle
  - CIPresent(AUI) = SQE

- **PARTITION HOLD**
  - Datain(AUI) = Idle
  - Collin(AUI) = SQE
  - DIPresent(AUI) = Idle
  - CIPresent(AUI) = SQE

- **PARTITION COLLISION WATCH**
  - Datain(AUI) = Idle
  - Collin(AUI) = SQE
  - EnableTw5
  - CIPresent(AUI) = SQE

- **COLLISION COUNT IDLE**
  - Datain(AUI) = DIPresent(AUI)
  - Collin(AUI) = CIPresent(AUI)
  - DIPresent(AUI) = Idle
  - CIPresent(AUI) = SQE

- **WATCH FOR COLLISION**
  - Datain(AUI) = DIPresent(AUI)
  - Collin(AUI) = CIPresent(AUI)
  - EnableTw5
  - DIPresent(AUI) = Idle
  - CIPresent(AUI) = SQE

- **COLLISION COUNT INCREMENT**
  - CC(AUI) = CC(AUI) + 1
  - Datain(AUI) = DIPresent(AUI)
  - Collin(AUI) = CIPresent(AUI)
  - EnableTw6

- **PARTITION COLLISION**
  - CC(AUI) ≥ 31
  - Datain(AUI) = DIPresent(AUI)
  - Collin(AUI) = CIPresent(AUI)
  - EnableTw6
  - CC(AUI) < 31
  - Datain(AUI) = DIPresent(AUI)

- **WAIT TO RESTORE PORT**
  - Datain(AUI) = Idle
  - Collin(AUI) = SQE
  - CC(AUI) = 0
  - DIPresent(AUI) = Idle
  - CIPresent(AUI) = SQE
Partitioning State Diagram for T-P Port x

RESET
- COUNT CLEAR = 0
  - CC(X) = 0
  - Datain(X) = DIPresent(X)
    - DIPresent(X) = idle

- PARTITION WAIT - 4
  - Datain(X) = Idle
    - DIPresent(X) = idle

- PARTITION HOLD - 5
  - Datain(X) = Idle
    - DIPresent(X) = idle

- PARTITION COLLISION WATCH - 6
  - Datain(X) = Idle
    - EnableTw5
      - TPEN(X) = Active
        - DIPresent(X) = idle

- WAIT TO RESTORE - 7
  - Datain(X) = Idle
    - CC(X) = 0
    - DIPresent(X) = idle

- COLLISION COUNT - 1
  - Datain(X) = DIPresent(X)
    - DIPresent(X) = idle

- WATCH FOR COLLISION - 2
  - Datain(X) = DIPresent(X)
    - EnableTw5
      - DIPresent(X) = idle

- COLLISION COUNT INCREMENT - 3
  - CC(X) = CC(X) + 1
    - Datain(X) = DIPresent(X)
      - EnableTw6
        - CC(X) < 31

- PARTITION COLLISION WATCH - 6
  - CC(X) ≥ 31
    - Datain(X) = DIPresent(X)
      - EnableTw6
        - CC(X) < 31

- PARTITION HOLD - 5
  - Datain(X) = Idle
    - DIPresent(X) = idle

- PARTITION WAIT - 4
  - Datain(X) = Idle
    - DIPresent(X) = idle

- COUNT CLEAR - 0
  - CC(X) = 0
  - Datain(X) = DIPresent(X)
    - DIPresent(X) = idle
Tw2 State Diagram

MAU Jabber Lockup Protection State Diagram

Transmit Timer State Diagram
CHARACTERISTICS

DC Characteristics

<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>-0.5</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>VDD + 0.5</td>
<td>V</td>
<td></td>
</tr>
<tr>
<td>VOL</td>
<td>Output Low Voltage</td>
<td></td>
<td>0.5</td>
<td>V</td>
<td>IOL = 1.6 mA</td>
</tr>
<tr>
<td>VOH</td>
<td>Output High Voltage</td>
<td>2.4</td>
<td></td>
<td>V</td>
<td>IOH = -0.4 mA</td>
</tr>
<tr>
<td>ICC</td>
<td>Power Supply Current</td>
<td></td>
<td>75</td>
<td>mA</td>
<td>VCC = 5.5 V</td>
</tr>
<tr>
<td>IIL</td>
<td>Input Leakage Current (TTL)</td>
<td></td>
<td>10</td>
<td>µA</td>
<td>VIH = 5.5 V</td>
</tr>
<tr>
<td>ILI</td>
<td>Schmitt Inputs</td>
<td></td>
<td>10</td>
<td>µA</td>
<td>VIH = 5.5 V</td>
</tr>
<tr>
<td>IIL</td>
<td>Inputs with Pull-Up</td>
<td></td>
<td>500</td>
<td>µA</td>
<td>VIH = 5.5 V</td>
</tr>
<tr>
<td>PD</td>
<td>Power Dissipation (25°C)</td>
<td></td>
<td>0.33</td>
<td>W</td>
<td>VCC = 5.0 V</td>
</tr>
<tr>
<td>PD</td>
<td>Power Dissipation (0°C)</td>
<td></td>
<td>0.4</td>
<td>W</td>
<td>VCC = 5.0 V</td>
</tr>
</tbody>
</table>

ABSOLUTE MAXIMUM RATING*

Ambient Operating Temperature (Ta) .......... 0°C, 70°C
Storage Temperature ...................... -40°C, + 125°C
Power Dissipation .......................... 400 mW
Voltage On Any Pin with Respect to Ground ....... -0.5V, VCC + 0.5V

NOTICE: This data sheet contains preliminary information on new products in production. The specifications are subject to change without notice.

*WARNING: Stressing the device beyond the “Absolute Maximum Ratings” may cause permanent damage. These are stress ratings only. Operation beyond the “Operating Conditions” is not recommended and extended exposure beyond the “Operating Conditions” may affect device reliability.

CLOCK AND RESET TIMINGS

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
<th>Min</th>
<th>Max</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>t1</td>
<td>Clock period</td>
<td>49.995</td>
<td>50.005</td>
<td>ns</td>
</tr>
<tr>
<td>t2</td>
<td>Clock high</td>
<td>20</td>
<td>30</td>
<td>ns</td>
</tr>
<tr>
<td>t3</td>
<td>Clock low</td>
<td>20</td>
<td>30</td>
<td>ns</td>
</tr>
<tr>
<td>t4</td>
<td>Clock rise time</td>
<td></td>
<td>10</td>
<td>ns</td>
</tr>
<tr>
<td>t5</td>
<td>Clock fall time</td>
<td></td>
<td>10</td>
<td>ns</td>
</tr>
<tr>
<td>t6</td>
<td>CLKOUT propagation delay</td>
<td>10</td>
<td>16</td>
<td>ns</td>
</tr>
<tr>
<td>t7</td>
<td>RESET pulse width</td>
<td>40</td>
<td>—</td>
<td>CLK</td>
</tr>
</tbody>
</table>

Clock and Reset Timing
TRANSMIT TIMINGS

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
<th>Min</th>
<th>Max</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>t8</td>
<td>Delay from CRS low to TPEN low.</td>
<td>5</td>
<td>6</td>
<td>bits</td>
</tr>
<tr>
<td></td>
<td>T-P Port, FILL = 0 (FIFO fill = 7 bits)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>T-P Port FILL = 1, (FIFO fill = 8 bits)</td>
<td>6</td>
<td>7</td>
<td>bits</td>
</tr>
<tr>
<td></td>
<td>AUI Port, FILL = x (FIFO fill = don't care)</td>
<td>4.5</td>
<td>5</td>
<td>bits</td>
</tr>
<tr>
<td>t9</td>
<td>AUICDT low to TPEN low</td>
<td>4</td>
<td>5</td>
<td>bits</td>
</tr>
<tr>
<td>t10</td>
<td>Multiple Carrier* to TPEN low</td>
<td>5</td>
<td>6</td>
<td>bits</td>
</tr>
</tbody>
</table>

* Carrier is any of TCSX or AUICRS

GLOBAL MACHINE STATES:
0. Idle
1. Send Data

Start of Transmission Timing (Normal Packet)

Receive Collision (RC)

Transmit Collision (TC)

GLOBAL MACHINE STATES:
0. Idle
1. Receive Collision
2. Transmit Collision

Start of Transmission Timing (Collision Conditions)
Manchester Encoder Timing

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
<th>Min</th>
<th>Max</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>t11</td>
<td>Clock high to TPEN low</td>
<td>4</td>
<td>22</td>
<td>ns</td>
</tr>
<tr>
<td>t12</td>
<td>Clock high to TPEN high</td>
<td>4</td>
<td>22</td>
<td>ns</td>
</tr>
<tr>
<td>t13</td>
<td>Clock high to PDC low</td>
<td>4</td>
<td>22</td>
<td>ns</td>
</tr>
<tr>
<td>t14</td>
<td>Clock high to PDC high</td>
<td>4</td>
<td>22</td>
<td>ns</td>
</tr>
<tr>
<td>t15</td>
<td>Clock high to TRMT high</td>
<td>4</td>
<td>22</td>
<td>ns</td>
</tr>
<tr>
<td>t16</td>
<td>Clock high to TRMT low</td>
<td>4</td>
<td>22</td>
<td>ns</td>
</tr>
<tr>
<td>t17</td>
<td>TRMT rise time</td>
<td>1</td>
<td>7</td>
<td>ns</td>
</tr>
<tr>
<td>t18</td>
<td>TRMT fall time</td>
<td>1</td>
<td>7</td>
<td>ns</td>
</tr>
<tr>
<td>t19</td>
<td>Clock high to TRMT low</td>
<td>4</td>
<td>22</td>
<td>ns</td>
</tr>
<tr>
<td>t20</td>
<td>Clock high to TRMT high</td>
<td>4</td>
<td>22</td>
<td>ns</td>
</tr>
<tr>
<td>t21</td>
<td>TRMT fall time</td>
<td>1</td>
<td>7</td>
<td>ns</td>
</tr>
<tr>
<td>t22</td>
<td>TRMT rise time</td>
<td>1</td>
<td>7</td>
<td>ns</td>
</tr>
</tbody>
</table>

CLK  
TPEN  
PDC  
TRMT  
TRMT  

Manchester Encoder Timing
Manchester Encoder Timing Relationships

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
<th>Min</th>
<th>Max</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>t23</td>
<td>TPEN low to TRMT low</td>
<td>1</td>
<td></td>
<td>bits</td>
</tr>
<tr>
<td>t24</td>
<td>TRMT high to TPEN high</td>
<td>2.5</td>
<td>3.0</td>
<td>bits</td>
</tr>
<tr>
<td>t25</td>
<td>TRMT high to TRMT low</td>
<td>0</td>
<td>5</td>
<td>ns</td>
</tr>
<tr>
<td>t26</td>
<td>TRMT low to TRMT high</td>
<td>0</td>
<td>5</td>
<td>ns</td>
</tr>
</tbody>
</table>

RECEIVER AND TSI INTERFACE TIMINGS

Carrier and Energy Timings (No Errors)

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
<th>Min</th>
<th>Max</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>t27</td>
<td>Carrier hold time</td>
<td>14</td>
<td></td>
<td>ns</td>
</tr>
<tr>
<td>t28</td>
<td>Carrier setup time</td>
<td>10</td>
<td></td>
<td>ns</td>
</tr>
<tr>
<td>t29</td>
<td>Clock high to TPS high</td>
<td>20</td>
<td>30</td>
<td>ns</td>
</tr>
<tr>
<td>t30</td>
<td>Clock high to TPS low</td>
<td>20</td>
<td>30</td>
<td>ns</td>
</tr>
<tr>
<td>t31</td>
<td>Carrier high to TPS high</td>
<td>4</td>
<td>5</td>
<td>CLK</td>
</tr>
<tr>
<td>t32</td>
<td>Carrier low to TPS low</td>
<td>2</td>
<td>3</td>
<td>CLK</td>
</tr>
</tbody>
</table>
Data and TRMT Timings

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
<th>Min</th>
<th>Max</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>t33</td>
<td>TPS high to TRXD valid</td>
<td>5</td>
<td>12</td>
<td>ns</td>
</tr>
<tr>
<td>t34</td>
<td>TPS low to TRXD high</td>
<td>5</td>
<td>12</td>
<td>ns</td>
</tr>
<tr>
<td>t35</td>
<td>Data† to TRSD delay</td>
<td>5</td>
<td>35</td>
<td>ns</td>
</tr>
<tr>
<td>t36</td>
<td>TRXD rise time</td>
<td>1</td>
<td>7</td>
<td>ns</td>
</tr>
<tr>
<td>t37</td>
<td>TRXD fall time</td>
<td>1</td>
<td>7</td>
<td>ns</td>
</tr>
</tbody>
</table>

†Data refers to the data signal (either TRDx or AUIRx) from the active receiving port.

![Diagram](290213-22)

*Data refers to the data signal (either TRDx or AUIRx) from the active port.

Decoded NRZ Data Timings

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
<th>Min</th>
<th>Max</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>t38</td>
<td>RxC high to CRS low</td>
<td>5</td>
<td>19</td>
<td>ns</td>
</tr>
<tr>
<td>t39</td>
<td>RxC high to CRS high</td>
<td>5</td>
<td>19</td>
<td>ns</td>
</tr>
<tr>
<td>t40</td>
<td>Time between CRS low</td>
<td>16</td>
<td>—</td>
<td>bits</td>
</tr>
<tr>
<td>t41</td>
<td>RxC period</td>
<td>78</td>
<td>—</td>
<td>ns</td>
</tr>
<tr>
<td>t42</td>
<td>RxC low</td>
<td>30</td>
<td>—</td>
<td>ns</td>
</tr>
<tr>
<td>t43</td>
<td>RxC high</td>
<td>36</td>
<td>—</td>
<td>ns</td>
</tr>
<tr>
<td>t44</td>
<td>RXD setup time</td>
<td>40</td>
<td>—</td>
<td>ns</td>
</tr>
<tr>
<td>t45</td>
<td>RXD hold time</td>
<td>30</td>
<td>—</td>
<td>ns</td>
</tr>
</tbody>
</table>

![Diagram](290213-23)

Decoded NRZ Data Timing
AUI COLLISION INTERFACE

CDT and AUICDT Timing Relationship for AUI Port

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
<th>Min</th>
<th>Max</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>t46</td>
<td>Delay for MCV low</td>
<td>10.5</td>
<td>11</td>
<td>bits</td>
</tr>
<tr>
<td>t47</td>
<td>MCV low to AUICDT low for T-COLLSN</td>
<td>9.5</td>
<td>—</td>
<td>bits</td>
</tr>
<tr>
<td>t48</td>
<td>MCV low to AUICDT low for R-COLLSN</td>
<td>—</td>
<td>9</td>
<td>bits</td>
</tr>
</tbody>
</table>

GLOBAL MACHINE STATES:
1. Send Data
2. Receive Collision
3. Transmit Collision

MCV, AUICDT Timing Relationship for AUI Port

Port Disable Control Timings

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
<th>Min</th>
<th>Max</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>t49</td>
<td>Clock high to port address valid</td>
<td>20</td>
<td>30</td>
<td>ns</td>
</tr>
<tr>
<td>t50</td>
<td>Clock high to port address invalid</td>
<td>20</td>
<td>30</td>
<td>ns</td>
</tr>
<tr>
<td>t51</td>
<td>Clock high to PDCTL low</td>
<td>20</td>
<td>30</td>
<td>ns</td>
</tr>
<tr>
<td>t52</td>
<td>Clock high to PDCTL high</td>
<td>20</td>
<td>30</td>
<td>ns</td>
</tr>
<tr>
<td>t53</td>
<td>Port address valid to PDCTL low</td>
<td>2</td>
<td>—</td>
<td>CLK</td>
</tr>
<tr>
<td>t54</td>
<td>PDCTL high to port address invalid</td>
<td>1</td>
<td>—</td>
<td>CLK</td>
</tr>
<tr>
<td>t55</td>
<td>PDCTL low to TPEN low</td>
<td>4</td>
<td>—</td>
<td>CLK</td>
</tr>
<tr>
<td>t56</td>
<td>TPEN high to PDCTI high</td>
<td>19</td>
<td>—</td>
<td>CLK</td>
</tr>
</tbody>
</table>
Port Disable Timing (One Port Left)

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
<th>Min</th>
<th>Max</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>t57</td>
<td>PDCTL high to port address invalid</td>
<td>1</td>
<td>—</td>
<td>CLK</td>
</tr>
<tr>
<td>t58</td>
<td>Port address valid to PDCTL low</td>
<td>2</td>
<td>—</td>
<td>CLK</td>
</tr>
<tr>
<td>t59</td>
<td>PDCTL high to TRMT low</td>
<td>1</td>
<td>—</td>
<td>bits</td>
</tr>
</tbody>
</table>

GLOBAL MACHINE STATES:
0. Idle
1. Send Data
3. Transmit Collision
4. One Port Left
5. Blind

LED Control Timing

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
<th>Min</th>
<th>Max</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>t60</td>
<td>Clock high to address valid</td>
<td>20</td>
<td>30</td>
<td>ns</td>
</tr>
<tr>
<td>t61</td>
<td>Clock high to address invalid</td>
<td>20</td>
<td>30</td>
<td>ns</td>
</tr>
<tr>
<td>t62</td>
<td>Clock high to LEDCTL valid</td>
<td>20</td>
<td>30</td>
<td>ns</td>
</tr>
<tr>
<td>t63</td>
<td>Clock high to LEDCTL invalid</td>
<td>20</td>
<td>30</td>
<td>ns</td>
</tr>
<tr>
<td>t64</td>
<td>Clock low to LEDSTRB low</td>
<td>20</td>
<td>30</td>
<td>ns</td>
</tr>
<tr>
<td>t65</td>
<td>Clock low to LEDSTRB high</td>
<td>20</td>
<td>30</td>
<td>ns</td>
</tr>
<tr>
<td>t66</td>
<td>LEDSTRB low</td>
<td>1</td>
<td>—</td>
<td>CLK</td>
</tr>
<tr>
<td>t67</td>
<td>Time between strobes</td>
<td>1</td>
<td>—</td>
<td>CLK</td>
</tr>
</tbody>
</table>
LED Timing Relationship

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
<th>Min</th>
<th>Max</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>t68</td>
<td>LED update duration</td>
<td>1.6</td>
<td></td>
<td>µs</td>
</tr>
<tr>
<td>t69</td>
<td>Interoperation time</td>
<td>105</td>
<td></td>
<td>ms</td>
</tr>
</tbody>
</table>

OUTLINE DIAGRAM

Dimensions are in Inches

---

1-320
82506TB
TWISTED PAIR MEDIUM ATTACHMENT UNIT (TP MAU)

- Complies with IEEE 802.3 10BASE-T Draft 11 for Twisted Pair Interface
- Conforms to IEEE 802.3 Standard for Attachment Unit Interface (AUI)
- Direct Interface to AUI and Twisted Pair Isolation Transformers
- On-Chip Line Drivers and Receivers
- LED Drivers for Transmit, Receive, Collision, and Jabber Status
- Generates Internal Predistortion Signal

The 82506TB Twisted Pair Medium Attachment Unit (TP MAU) is intended for local area network (LAN) designs that interface the IEEE 802.3-1988 AUI cable to the twisted pair wire (10BASE-T). It offers LAN designers a cost-effective, integrated solution to the problem of upgrading existing standard Ethernet networks to twisted pair. The 82506TB complies with IEEE 802.3 AUI specifications and IEEE 802.3 10BASE-T Draft 11 specifications. The device incorporates the interface circuitry and both the AUI and twisted pair line drivers and receivers in a low-power CMOS package. The 82506TB TP MAU internally generates predistortion signals to eliminate line overcharge and improve jitter performance. It provides selectable 10BASE-T features for simplified network management, including selectable signal quality error (SQE) test, link integrity test, and jabber protection. In addition, the 82506TB TP MAU supports LED status indicators for transmit, receive, jabber, and collision. It is fabricated using CMOS-process technology and is available in 28-lead plastic DIP and 28-lead SOJ packages.

Manufactured and tested for Intel by AT&T in accordance with AT&T internal standards.

*Ethernet® is a registered trademark of Xerox Corporation.
Figure 2. 82506TB Pinout

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Pin No.</th>
<th>Type</th>
<th>Name and Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>VSS</td>
<td>1</td>
<td>-</td>
<td>Analog Ground.</td>
</tr>
<tr>
<td>TRMT, TRMT</td>
<td>2, 3</td>
<td>I</td>
<td>Transmit Data. A differentially driven input tied to the D0 pair of the transceiver cable. The transmit pair of the transceiver cable supplies 10-Mb/s Manchester encoded data. These pins must be isolated with a pulse transformer. End of Packet (EOP) is detected when a positive transition has not occurred for 200 ns.</td>
</tr>
<tr>
<td>DTE</td>
<td>4</td>
<td>I</td>
<td>Data Terminal Equipment. A strapping option, which when tied high (VCC) enables generating a SQE-test signal at the end of each packet (as required for DTE applications). When DTE is tied low (VSS) the SQE test is disabled, but the collision circuit remains enabled for use in repeater applications. When the DTE is floated, an internal pull-up biases the signal high.</td>
</tr>
<tr>
<td>LID</td>
<td>5</td>
<td>I</td>
<td>Link Integrity Disable. A strapping option, which when tied high (VCC) disables the link integrity function of the TPMAU. When link integrity is enabled, the receive traffic indicator remains on when the receive twisted pair link is present.</td>
</tr>
<tr>
<td>RCV, RCV</td>
<td>6, 7</td>
<td>O</td>
<td>Receive Data Pair. A differential output pair that drives the DI pair of the AUI cable with 10-Mb/s Manchester encoded data. These pins must be isolated from the AUI transceiver cable with a pulse transformer.</td>
</tr>
<tr>
<td>CLSN, CLSN</td>
<td>8, 9</td>
<td>O</td>
<td>Collision Presence Pair. A differential output pair that drives the CI pair of the AUI cable with a 10-MHz (± 15%) signal when simultaneous activity exists on the TRMT and RD pairs. These pins must be isolated from the AUI-transceiver cable with a pulse transformer.</td>
</tr>
<tr>
<td>VCC</td>
<td>10</td>
<td>-</td>
<td>Power. Digital, 5 V.</td>
</tr>
<tr>
<td>VSS</td>
<td>11</td>
<td>-</td>
<td>Ground.</td>
</tr>
<tr>
<td>JLED</td>
<td>12</td>
<td>O</td>
<td>Jabber Indicator. Indicates that the watchdog timer has timed out and the twisted pair drivers have been disabled.</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>XTAL</td>
<td>13</td>
<td>I</td>
<td>Crystal In. A 20-MHz clock input. This signal can be driven by a 20-MHz, parallel-resonant crystal or a MOS level clock with a 60/40 duty cycle.</td>
</tr>
<tr>
<td>VSS</td>
<td>14</td>
<td>I</td>
<td>Ground. For XTAL (Pin 13) and indicator output drivers.</td>
</tr>
<tr>
<td>CLED</td>
<td>15</td>
<td>O</td>
<td>Collision Indicator. Indicates that a collision has been detected by the TP medium attachment unit.</td>
</tr>
<tr>
<td>RLED</td>
<td>16</td>
<td>O</td>
<td>Receive Indicator. Indicates that a reception from the network is in progress.</td>
</tr>
<tr>
<td>XLED</td>
<td>17</td>
<td>O</td>
<td>Transmit Indicator. Indicates that a transmission onto the network is in progress.</td>
</tr>
<tr>
<td>VCC</td>
<td>18</td>
<td></td>
<td>Power. 5 V.</td>
</tr>
<tr>
<td>HDAT, HDAT</td>
<td>19, 22</td>
<td>O</td>
<td>TP Transmit Pair Drivers. These four outputs constitute the twisted-pair drivers, which have predistortion capabilities. The HDAT/HDAT outputs generate the 10-Mb/s Manchester encoded data. The LDAT/LDAT outputs mirror the HDAT/HDAT outputs except for “fat” bit occurrences. During the second half of a “fat” bit (either high or low), the LDAT/LDAT outputs are inverted with respect to HDAT/HDAT outputs. This signal behavior reduces the amount of jitter by preventing overcharge on the twisted-pair medium.</td>
</tr>
<tr>
<td>LTTE</td>
<td>24</td>
<td>I</td>
<td>Lower TP Threshold Enable (Active Low). For 10BASE-T compatible operation this pin must be left open. But, if this pin is grounded, the TP receiver threshold is lowered by approximately 4.5 db from the nominal 10BASE-T required specification. By using this lower-threshold option and by selecting different compensation resistor values for waveconstruction circuits, a customized interface is possible for non-10BASE-T applications. However, once this lower threshold is invoked, the wiring used must not be a bundled system (e.g., 25 pair) where other services reside (e.g., voice, other 10BASE-T users, etc.).</td>
</tr>
<tr>
<td>TEST</td>
<td>25</td>
<td>I</td>
<td>Test. This pin is used for testing; it should be connected to VSS during normal operation.</td>
</tr>
<tr>
<td>RD, RD</td>
<td>26, 27</td>
<td>I</td>
<td>TP Receive Pair. The differential twisted pair receiver. The receive pair is connected to the twisted pair medium and is driven with 10-Mb/s Manchester encoded data.</td>
</tr>
<tr>
<td>VCC</td>
<td>28</td>
<td></td>
<td>Analog Power. 5 V.</td>
</tr>
</tbody>
</table>
**Figure 3. Typical System Configuration**

**Figure 4. Typical Load Output on the Outputs of RCV/RCV and CLSN/CLSN**

*Required transformers are the Pulse Engineering Inc. (PE64503), TDK Corp. (TLA-100-3E), Coilcraft (LAXET304 or LO323-A), or equivalents.*
FUNCTIONAL DESCRIPTION

Overview

The 82506TB provides the transmit, receive, and collision detection functions specified by the IEEE 802.3 committee for the 10BASE-T Draft 11 (P802.3j/D11) specification for a 10-Mb/s, CSMA/CD, twisted-pair Ethernet. The 82506TB is used as the interface between the attachment unit interface (AUI) signals and the twisted pair. Two strapping options are available.

- **Link Integrity Disable (LID).** When the LID strapping option is enabled (driven high), the link integrity function is disabled. When driven low, link pulses are transmitted on the twisted pair medium in the absence of data transmissions. In addition, the receiver expects to see link pulses in the absence of receive data. If no receive data or link pulses are received within 100 ms ± 50 ms the 82506TB will enter a link fail state. When LID is floated an internal pull-up biases the signal high.

- **Data Terminal Equipment (DTE).** When the DTE strapping option is enabled (driven high) the SQE test sequence is transmitted to the DTE after every successful transmission on the twisted pair network.

The 82506TB simplifies network management and troubleshooting by providing four status indicator LED drivers that monitor traffic on a node and report transmit, receive, collision, and jabber conditions.

Figure 3 is an example of a typical system configuration.

Transmit Path (AUI to TP)

The transmit portion of this component transverses data from the AUI to the twisted-pair analog filters. It also loops back the data to the RCV pair.

- **AUI Receiver.** The TP MAU receives transmit data from the data terminal equipment on the transmit pins (TRMT/TRMT) of the AUI-DO circuit (as defined by the IEEE802.3-1988 specification). The 82506TB then transmits the data onto the twisted-pair cable via the twisted-pair drivers. The AUI transmit inputs must be transformer coupled to the TRMT/TRMT pins. For best operation, the AUI signal should be dc biased to a common mode voltage of \( V_{CC}/2 \).

The squelch circuit rejects (filters) all signals with an amplitude less than 160 mV or a pulse width less than 20 ns. A signal with an amplitude greater than 300 mV and a pulse width greater than 75 ns is accepted and turns off the squelch filter. The squelch filter remains off until an IDL pulse is detected or until the input does not exceed the detection threshold for 500 ± 100 ns.

- **AUI Receive Signal Levels.** The receiver (TRMT/TRMT) is able to recognize differential signals as small as 300-mV peak. Internal circuitry samples the common mode voltage to provide full differential signal detection.

- **TP Driver Characteristics.** The drivers (HDAT/LDAT) output CMOS logic levels with a source resistance less than 10 Ω and maximum current rating of 25 mA dc. All TP output driver pins are driven low as a result of any of the following:

  - Reception of an IDL signal.
  - A jabber condition is detected.
  - Activation of a link failure.
  - TRMT pair input fails to cross the detection threshold for 500 ± 100 ns.

When the driver detects the end of an IDL pulse on the TRMT pair, a timer of not more than 5 bit times (BT) is started. Activity on the TRMT pair is ignored until this timer expires.

Receive Path (TP to AUI)

When a RD signal is present, the receive circuit of the 82506TB transfers data from the RD pair input to the RCV pair output.

- **AUI Driver Characteristics.** This driver differentially drives a current onto the load connected between the RCV and \( R_{CV} \) pins. The current through the load results in an output voltage between ±0.6 V and ±1.2 V measured differentially between the two pins. An external resistor (78 Ω) and capacitor (100 pF) must be connected for proper termination, as shown in Figure 4. This output is in accordance with the IEEE Spec 802.3 Sec.7.4.1 for MAUs. When the driver detects that it has finished sending an IDL pulse to the AUI it starts a timer of not more than 5 BT. Activity on the RD pair is ignored when this timer is functioning.

- **TP Receiver Threshold.** The TP receiver is connected to the output of a band limiting filter. The filter's input is transformer coupled to the twisted pair. The receiver is able to recognize differential signals as small as 350 mV peak. An external biasing circuit must provide a common mode voltage of \( V_{CC}/2 \). The differential input impedance of the RD pair is 20 kΩ ± 20%. Internal circuitry generates a dual-level bias voltage to determine proper signal level thresholds and prevent reception of spurious signals from the network (this is similar to a squelch function).
When the signal level at the RD input falls below 
\(-500 \text{ mV, with respect to the common mode}
\) voltage \(\pm 10\%\), the data path is activated and
the received signal is passed to the AUI cable.
At the beginning of a reception the bias level at
the RD input is reduced to \(-350 \text{ mV with re}
\)spect to the common mode voltage \(\pm 10\%\)

**Collision**

The collision detection portion of the 82506TB sen-
ses the simultaneous presence of data on the TRMT
and RD pins. It reacts by transmitting a 10-MHz
square wave on the CLSN pair of the AUI cable. This
signal is a periodic waveform of 10 MHz \(\pm 15\%\),
with a duty cycle no worse than 40/60 or 60/40. It is
transmitted within 9 bit times after the component
detects a collision (as specified by 10BASE-T Draft
11, Sec.14.2.1.3). If the receive pair becomes active
while the transmit pair is active, the loopback data
on the RCV pair switches from transmit data to
receive data within 13 \(\pm 3\) BT from the assertion
of the CLSN pair. If the RD pair goes active while
the TRMT pair is active a collision condition will be
detected and the SQE will continue for 7 \(\pm 2\) BT. If a
collision condition exists where the TRMT pair has
gone idle while the RD pair is still active, SQE can
continue for up to 9 BT.

The collision AUI driver differentially drives a signal
onto the load connected between the CLSN pair.

This driver differentially drives a current onto the
load connected between the CLSN and CLSN pins.
The current through the load results in an output
temperature between \(\pm 0.6\) V and \(\pm 1.2\) V measured
differentially between the two pins. An external resistor
(78 \(\Omega\)) and capacitor (100 pF) must be connected
for proper termination, as shown in Figure 4. The
output is in accordance with IEEE 802.3-1988 Sec.
7.4.1 for the AUI.

**Jabber (Watchdog Timer)**

The 82506TB supports a self-interrupt function that
protects the network from a jabbering node (i.e.,
continuous transmission). The component provides
a nominal window of 50 ms during the time a normal
data link frame can be transmitted. If the frame
length exceeds this duration, the component imme-
diately inhibits all further transmission of that frame
and activates the CLSN pair (as specified by IEEE
802.3-1988 Sec. 8.2.1.5). When activity on the
TRMT pair has ceased, the component continues to
present the CS0 signal to the CLSN pair for 0.5 s \(\pm
\)50\%. The component then resets itself and returns
to the idle state (as specified by the 10BASE-T Draft
11 14.2.1.5). The transmission of link integrity pulses
from the TP drivers is not inhibited when the TP
MAU jabber is activated and link integrity is enabled.

**SQE Test (Heartbeat)**

The SQE test begins within 11 \(\pm 5\) bit times after
the TRMT pair detects the IDL signal. The SQE test
duration is 10 \(\pm 5\) bit times. When the AUI-DO cir-
cuit has gone idle after a successful transmission
(without a collision), and the DTE input is high, the
82506TB activates the CLSN pair to simulate a colli-
sion.

**Link Integrity**

The link integrity function determines if the receive
twisted-pair link is faulty. Enabling the function (LID
tied to VSS) causes the RLED receive traffic indica-
tor to display the status of the receive twisted-pair
link. The link integrity function permits the active dis-
abling of the transmit and loopback paths within the
TP MAU component in response to a link integrity
fault. The link integrity function monitors the RD pair
for either data or link test pulses by providing a 100
ms \(\pm 50\) ms window during which data or a link test
pulse is expected. If this timer expires and the LID is
off, the RLED indicator is turned off and the compo-
nent’s transmit and loopback capabilities are dis-
abled. The 82506TB remains in a link fail state until
after a data packet is received, or until after a se-
quence of consecutive link test pulses are received.
The sequence length is between two and ten pulses.
If a pulse or receive traffic is detected with this
window, the timer is reset and the LLED indicator
remains on.

The TP MAU also transmits link test pulses onto the
transmit twisted pair link when link integrity is en-
abled. In the absence of transmit traffic, a link test
pulse is transmitted at a nominal rate of one pulse
each 16 ms \(\pm 8\) ms. If the link integrity is disabled,
the RLED indicator remains on in the absence of
receive traffic, (data and link pulses). Received link
test pulses are also ignored at the RD pair input.

**LED Status**

Four light-emitting diodes (LEDs) give the user a vi-
sual indication of the MAU's status. The 82506TB
provides the logic signals needed to drive the LEDs.

- **XLED.** The following LED values (on or off) are
  used to indicate transmission (AUI) status.
- The LED is normally on, which indicates no
  transmission is in progress.
- The LED is off when a valid packet is transmit-
ted. The duration of the off period is 100 ms \(\pm
\)10 ms. The minimum duration of the on period is
  4 ms while waiting for next valid packet trans-
mission.
• **RLED**—With LID Disabled. The following LED values (on or off) are used to indicate reception (TP) status.
  - The LED is normally on, which indicates there is no receive traffic.
  - The LED is off when a valid packet is received. The duration of the off period is 100 ms ± 10 ms. The minimum duration of the on period is 4 ms while waiting for next valid packet receive.
• **RLED**—With LID Enabled. The following LED values (on or off) are used to indicate reception (TP) status.
  - The LED is normally on, which indicates no receive traffic and successful reception of the link test pulse.
  - The LED turns off if no receive traffic or link integrity signals have been received for more than 0.5 s. This visually indicates a failure of a link segment. The LED remains off until a link test pulse, or receive traffic, is successfully detected, after which the LED is turned on with a minimum on time of 0.5 s.
  - If the link is working, the LED will be turned off when a valid data packet is received from the twisted pair. When a packet is received, the LED is turned off for a duration of 100 ms ± 10 ms, then the LED is turned back on.
  - The LED remains on for a minimum of 4 ms; it is turned off when the next packet is received.
• **CLED.** The following LED values indicate collision status.
  - The LED is normally off, which indicates no collision.
  - The LED is turned on when a collision is detected. It remains on for a nominal time of 15 ms ± 5 ms, after which it is turned off.
  - The LED may be turned back on immediately upon detection of another collision. There is no minimum off time.
  - If a collision occurs while the LED is on, the LED remains on for the nominal time following the last detected collision.
• **JLED.** The following LED values indicate the jabber status.
  - The LED is normally off, which indicates no jabber condition.
  - The LED is turned on when the watchdog timer times out, and the TP drivers are disabled. It remains on until the jabber condition is corrected.
  - The LED is turned off after the watchdog timer counts out the 0.5 s ± 0.25 s reset time.

### LED Drivers

The typical LED circuit consists of an external resistor in series with the LED and connected the $V_{CC}$.

The LED driver pulls the pin low to turn the LED on. Each LED driver can sink up to 15 mA of current, with an output impedance of less than 50 Ω.

### Clock Generation

A 20-MHz, parallel-resonant crystal is used to control the clock generation oscillator of the TP MAU. We recommend that the crystal meet the following specifications.

- Quartz crystal
- $20 \text{ MHz} \pm 0.01\%$
- Parallel resonant with a 20-pF load fundamental mode with a maximum series resistance of 25 Ω.

The crystal shunt and external capacitance should be less than 10 pF. The crystal should be connected adjacent to the 82506TB to the XTAL and $V_{SS}$ pins. The crystal shunt capacitance (CO) should not exceed 5 pF.

An external MOS-level clock can be applied to the crystal oscillator input. A resistor should be added in series with the clock source to limit the amplitude of the voltage swing seen by the pin. A 500-Ω resistor works well in most cases. If users are concerned about the duty-cycle variation caused by driving the TPMAU with a clock source, the following test can be done on the bench to empirically determine the best resistor value for the user's application.

- Place the part in dc test mode, as described in the Test Mode section of this document.
- Attach an oscilloscope to the JLED pin. This pin outputs the internal clock source.
- Alter the resistor value to obtain an optimal duty-cycle ratio. Experiments have shown that a 500-Ω resistor works well for LS TTL logic levels; CMOS logic levels need a 1-kΩ resistor.

Under no circumstances should the clock be driven straight into the TPMAU. Also, under no circumstances should the clock stop, not even briefly, once power is applied to the TPMAU. If the clock to the TPMAU is stopped, power to the TPMAU must be removed to ensure proper behavior of the TPMAU.

### Strapping Options

All strapping options are connected to internal pull-up resistors, (nominally 100 kΩ). A resistor tying a strapping option low must be able to sink 70 μA.

### Test Mode

The 82506TB enters the ac or dc test mode when the test pin (TEST) is held high. The ac test mode is activated by also holding the DTE pin high; the internal clock speeds are increased by three to reduce...
the ac test time. The dc test is activated by holding the DTE pin low while TEST is held high. During the dc test the oscillator frequency and duty cycle can be tested on the JLED pin and the three internal clocks (ACK, BCK, and CCK) can be tested on the XLED, RLED, and CLED pins respectively. The AUI driver current can be measured with a 39-Ω resistor between the receive pair pins.

Power Considerations

There are seven power connections to the TP MAU—three pairs of VCC and VSS connections and a fourth VSS pin for the XTAL oscillator. Table 2 describes which internal circuits are powered by each VCC/VSS pair.

### Table 2. INTERNAL CIRCUIT

<table>
<thead>
<tr>
<th>Pin No.</th>
<th>Internal Circuits</th>
</tr>
</thead>
<tbody>
<tr>
<td>1, 28</td>
<td>Analog Supplies. Analog signal receivers, energy detection circuits, delay lock loop, and band gap reference.</td>
</tr>
<tr>
<td>10, 11</td>
<td>AUI Output Drivers. Digital polycells, XTAL oscillator (VCC only), and LED drivers (VCC only).</td>
</tr>
<tr>
<td>14</td>
<td>GND only for XTAL oscillator and pins 15, 16, and 17.</td>
</tr>
<tr>
<td>18, 23</td>
<td>TP CMOS output drivers only.</td>
</tr>
</tbody>
</table>

**DC Characteristics** $T_A = 0$ to $70^\circ$C, $V_{CC} = 5$ V ± 5%

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Parameter Description</th>
<th>Min</th>
<th>Max</th>
<th>Units</th>
<th>Test Conditions</th>
</tr>
</thead>
<tbody>
<tr>
<td>$V_{AID}$</td>
<td>Input Differential Voltage (AUI)</td>
<td>0.300</td>
<td>1.3</td>
<td>V</td>
<td></td>
</tr>
<tr>
<td>$V_{TID}$</td>
<td>Input Differential Voltage (TP)</td>
<td>0.350</td>
<td>2.0</td>
<td>V</td>
<td></td>
</tr>
<tr>
<td>$V_{XTL}$</td>
<td>XTAL Input Low Voltage</td>
<td>0.4</td>
<td>V</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$V_{XCH}$</td>
<td>XTAL Input High Voltage</td>
<td>3.9</td>
<td>V</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$V_{IL}$</td>
<td>Input Low Voltage</td>
<td>0.2</td>
<td>V</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$V_{IH}$</td>
<td>Input High Voltage</td>
<td>4.0</td>
<td>V</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$V_{AOD}$</td>
<td>Output Differential Voltages (AUI)</td>
<td>0.600</td>
<td>1.2</td>
<td>V</td>
<td></td>
</tr>
<tr>
<td>$V_{TOL}$</td>
<td>Output Voltage Low (TP)</td>
<td>0.1</td>
<td>V</td>
<td>$V_{CC} = 5.0$ V, $R_L = 500$ Ω</td>
<td></td>
</tr>
<tr>
<td>$V_{TOH}$</td>
<td>Output Voltage High (TP)</td>
<td>4.9</td>
<td>V</td>
<td>$V_{CC} = 5.0$ V, $R_L = 500$ Ω</td>
<td></td>
</tr>
<tr>
<td>$V_{OL}$</td>
<td>Output Voltage Low</td>
<td>0.13</td>
<td>V</td>
<td>$V_{CC} = 5.0$ V, $R_L = 2000$ Ω</td>
<td></td>
</tr>
<tr>
<td>$V_{OH}$</td>
<td>Output Voltage High</td>
<td>4.87</td>
<td>V</td>
<td>$V_{CC} = 5.0$ V, $R_L = 2000$ Ω</td>
<td></td>
</tr>
<tr>
<td>$R_S$</td>
<td>TP Driver Series Impedance</td>
<td>10</td>
<td>Ω</td>
<td>$V_{CC} = 4.5$ V, $I = 25$ mA (max)</td>
<td></td>
</tr>
<tr>
<td>$I_{CC}$</td>
<td>Power Supply Current with a Traffic Load</td>
<td>145</td>
<td>mA</td>
<td>$V_{CC} = 5.00$ V</td>
<td></td>
</tr>
<tr>
<td>$PD$</td>
<td>Power Dissipation with a Traffic Load</td>
<td>0.6</td>
<td>W</td>
<td>$V_{CC} = 5.0$ V</td>
<td></td>
</tr>
</tbody>
</table>
AC Timing Conditions
1. \( T_A = 0 \) to 70°C, \( V_{CC} = 5 \text{ V} \pm 5\% \).
2. Timing measurement points are 50% points unless otherwise noted.

Clock Timing

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Parameter</th>
<th>Min</th>
<th>Max</th>
<th>Units</th>
</tr>
</thead>
<tbody>
<tr>
<td>( t_1 )</td>
<td>XTAL (or Oscillator) Frequency</td>
<td>18</td>
<td>22</td>
<td>MHz</td>
</tr>
<tr>
<td>( t_2 )</td>
<td>XTAL High and Low Times</td>
<td>22.5</td>
<td>27.5</td>
<td>ns</td>
</tr>
</tbody>
</table>

Transmit Timing

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Parameter</th>
<th>Min</th>
<th>Max</th>
<th>Units</th>
</tr>
</thead>
<tbody>
<tr>
<td>( t_3 )</td>
<td>DTE Setup and Hold Time to TRMT Pair Active</td>
<td>10</td>
<td></td>
<td>( \mu \text{s} )</td>
</tr>
<tr>
<td>( t_4 )</td>
<td>Transmit Start-up Delay</td>
<td>0</td>
<td>2</td>
<td>bits</td>
</tr>
<tr>
<td>( t_5 )</td>
<td>Transmit Steady State Delay</td>
<td>0</td>
<td>2</td>
<td>bits</td>
</tr>
<tr>
<td>( t_6 )</td>
<td>Transmit Start-up Delay Variability</td>
<td>0</td>
<td>2</td>
<td>bits</td>
</tr>
<tr>
<td>( t_7 )</td>
<td>Loopback Start-up Delay</td>
<td>0</td>
<td>5</td>
<td>bits</td>
</tr>
<tr>
<td>( t_8 )</td>
<td>Loopback Steady State Delay</td>
<td>0</td>
<td>1</td>
<td>bits</td>
</tr>
<tr>
<td>( t_9 )</td>
<td>TD* Held High at End of Packet</td>
<td>250</td>
<td>350</td>
<td>ns</td>
</tr>
<tr>
<td>( t_{10} )</td>
<td>Incremental Transmit Jitter</td>
<td>0</td>
<td>3.5</td>
<td>ns</td>
</tr>
<tr>
<td>( t_{11} )</td>
<td>TRMT Pair Return to Idle to SQE Test</td>
<td>600</td>
<td>1600</td>
<td>ns</td>
</tr>
<tr>
<td>( t_{12} )</td>
<td>SQE Test Duration</td>
<td>500</td>
<td>1500</td>
<td>ns</td>
</tr>
</tbody>
</table>

*TD represents the differential voltage between the signals HDAT and RDAT.*
Transmit Timing: Start of Packet

Transmit Timing: End of Packet
## Receive Timing

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Parameter</th>
<th>Min</th>
<th>Max</th>
<th>Units</th>
</tr>
</thead>
<tbody>
<tr>
<td>t13</td>
<td>Receive Start-up Delay</td>
<td>0</td>
<td>5</td>
<td>bits</td>
</tr>
<tr>
<td>t14</td>
<td>Receive Steady State Delay</td>
<td>0</td>
<td>2</td>
<td>bits</td>
</tr>
<tr>
<td>t15</td>
<td>Receive Start-up Delay Variability</td>
<td>0</td>
<td>2</td>
<td>bits</td>
</tr>
<tr>
<td>t16</td>
<td>Incremental Receive Jitter</td>
<td>0</td>
<td>1.5</td>
<td>ns</td>
</tr>
<tr>
<td>t17</td>
<td>RD Pair Held High at End of Packet</td>
<td>0</td>
<td>300</td>
<td>ns</td>
</tr>
</tbody>
</table>

### Receive Timing: Start of Packet

![Receive Timing: Start of Packet](image1)

### Receive Timing: End of Packet

![Receive Timing: End of Packet](image2)

## Collision Timing

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Parameter</th>
<th>Min</th>
<th>Max</th>
<th>Units</th>
</tr>
</thead>
<tbody>
<tr>
<td>t18</td>
<td>Onset of Collision to CLSN Pair Assertion</td>
<td>0</td>
<td>900</td>
<td>ns</td>
</tr>
<tr>
<td>t19</td>
<td>RCV Pair Source to RD after CLSN Assert</td>
<td>0</td>
<td>900</td>
<td>ns</td>
</tr>
<tr>
<td>t20</td>
<td>End of Collision to CLSN Pair Return to Idle</td>
<td>0</td>
<td>900</td>
<td>ns</td>
</tr>
<tr>
<td>t21</td>
<td>CLSN Pair High/Low Time</td>
<td>40</td>
<td>60</td>
<td>ns</td>
</tr>
<tr>
<td>t22</td>
<td>CLSN Pair Frequency</td>
<td>8.5</td>
<td>11.5</td>
<td>MHz</td>
</tr>
</tbody>
</table>
Collision Timing

Collision Signal Timing

Jabber Protection Timing

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Parameter</th>
<th>Min</th>
<th>Max</th>
<th>Units</th>
</tr>
</thead>
<tbody>
<tr>
<td>t23</td>
<td>Assertion of TRMT Pair to Jabber Inhibit</td>
<td>45</td>
<td>55</td>
<td>ms</td>
</tr>
<tr>
<td>t24</td>
<td>Jabber Inhibit to CLSN Assert</td>
<td>0</td>
<td>900</td>
<td>ns</td>
</tr>
<tr>
<td>t25</td>
<td>TRMT Idle to Jabber Inhibit Removed</td>
<td>250</td>
<td>750</td>
<td>ms</td>
</tr>
</tbody>
</table>
Jabber Protection Timing

Link Integrity Timing

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Parameter</th>
<th>Min</th>
<th>Max</th>
<th>Units</th>
</tr>
</thead>
<tbody>
<tr>
<td>$t_{26}$</td>
<td>Last TRMT Pair Activity to Link Test Pulse</td>
<td>8</td>
<td>24</td>
<td>ms</td>
</tr>
<tr>
<td>$t_{27}$</td>
<td>Link Test Pulse Width</td>
<td>80</td>
<td>120</td>
<td>ns</td>
</tr>
<tr>
<td>$t_{28}$</td>
<td>Last RD Activity to Link Integrity Fault Assert</td>
<td>50</td>
<td>150</td>
<td>ms</td>
</tr>
<tr>
<td>$t_{29}$</td>
<td>Minimum Idle Time Between Consecutive Leakbeat Reception</td>
<td>6</td>
<td>8</td>
<td>ms</td>
</tr>
<tr>
<td>$t_{30}$</td>
<td>Maximum Idle Time Between Consecutive Leakbeat Reception</td>
<td>24</td>
<td>150</td>
<td>ms</td>
</tr>
<tr>
<td>$t_{31}$</td>
<td>JLED Turn-on Time</td>
<td></td>
<td>10</td>
<td>μs</td>
</tr>
</tbody>
</table>
### LED Timing

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Parameter</th>
<th>Min</th>
<th>Max</th>
<th>Units</th>
</tr>
</thead>
<tbody>
<tr>
<td>t₃₂</td>
<td>RLED Fixed Off Time</td>
<td>90</td>
<td>110</td>
<td>ms</td>
</tr>
<tr>
<td>t₃₃</td>
<td>RLED Minimum on Time</td>
<td>4</td>
<td>8</td>
<td>ms</td>
</tr>
<tr>
<td>t₃₄</td>
<td>Last RD Activity to RLED Off (Link Integrity Fault)</td>
<td>50</td>
<td>150</td>
<td>ms</td>
</tr>
<tr>
<td>t₃₅</td>
<td>RLED Minimum On Time After Link Integrity Fault Correction</td>
<td>500</td>
<td>1500</td>
<td>ms</td>
</tr>
<tr>
<td>t₃₆</td>
<td>XLED Turnoff Time</td>
<td>—</td>
<td>10</td>
<td>μs</td>
</tr>
<tr>
<td>t₃₇</td>
<td>XLED Fixed Off Time</td>
<td>90</td>
<td>110</td>
<td>ms</td>
</tr>
<tr>
<td>t₃₈</td>
<td>XLED Minimum On Time</td>
<td>4</td>
<td>8</td>
<td>ms</td>
</tr>
<tr>
<td>t₃₉</td>
<td>CLED Turnoff Time</td>
<td>—</td>
<td>10</td>
<td>μs</td>
</tr>
<tr>
<td>t₄₀</td>
<td>CLED Nominal On Time</td>
<td>10</td>
<td>20</td>
<td>ms</td>
</tr>
<tr>
<td>t₄₁</td>
<td>RLED Turnoff Time</td>
<td>—</td>
<td>10</td>
<td>μs</td>
</tr>
</tbody>
</table>

### LED Timing: Transmit and Receive

![LED Timing Diagram](image)

### LED Timing: Collision

![LED Timing Diagram](image)
Outline Diagrams
Dimensions are in inches and (millimeters).

28-Pin, Plastic DIP

**NOTES:**
Meets JEDEC standards.
Index mark may be a semicircular notch or circular dimple located in the index area.
28-Pin, Plastic SOJ
Dimensions are in inches and (millimeters)

NOTE:
Index mark may be a notch, dimple, or bevel.
An 82586 Data Link Driver

CHARLES YAGER
# AN 82586 DATA LINK DRIVER

## CONTENTS

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>INTRODUCTION</td>
<td>1-339</td>
</tr>
<tr>
<td>1.0 FITTING THE SOFTWARE INTO THE OSI MODEL</td>
<td>1-339</td>
</tr>
<tr>
<td>2.0 LARGE MODEL COMPILATION</td>
<td>1-340</td>
</tr>
<tr>
<td>3.0 THE 82586 HANDLER</td>
<td>1-340</td>
</tr>
<tr>
<td>3.1 The Buffer Model</td>
<td>1-340</td>
</tr>
<tr>
<td>3.2 The Handler Interface</td>
<td>1-341</td>
</tr>
<tr>
<td>3.3 Initialization</td>
<td>1-343</td>
</tr>
<tr>
<td>3.3.1 Building the CB and RFA Pools</td>
<td>1-343</td>
</tr>
<tr>
<td>3.3.2 82586 Initialization</td>
<td>1-343</td>
</tr>
<tr>
<td>3.3.3 Self Test Diagnostics</td>
<td>1-344</td>
</tr>
<tr>
<td>3.4 Command Processing</td>
<td>1-345</td>
</tr>
<tr>
<td>3.4.1 Accessing Command Blocks</td>
<td>1-345</td>
</tr>
<tr>
<td>3.4.2 Issuing CU Commands</td>
<td>1-345</td>
</tr>
<tr>
<td>3.4.3 Interrupt Service Routine</td>
<td>1-346</td>
</tr>
<tr>
<td>3.4.4 Sending Frames</td>
<td>1-346</td>
</tr>
<tr>
<td>3.4.5 Accessing Transmit Buffers</td>
<td>1-347</td>
</tr>
<tr>
<td>3.4.6 Multicast Addresses</td>
<td>1-347</td>
</tr>
<tr>
<td>3.4.7 Resetting the 82586</td>
<td>1-348</td>
</tr>
<tr>
<td>3.5 Receive Frame Processing</td>
<td>1-349</td>
</tr>
<tr>
<td>3.5.1 Receive Interrupt Processing</td>
<td>1-349</td>
</tr>
<tr>
<td>3.5.2 Returning FDs and RBDs</td>
<td>1-349</td>
</tr>
<tr>
<td>3.5.3 Restarting the Receive Unit</td>
<td>1-349</td>
</tr>
<tr>
<td>4.0 LOGICAL LINK CONTROL</td>
<td>1-350</td>
</tr>
<tr>
<td>4.1 Adding and Deleting LSAPs</td>
<td>1-352</td>
</tr>
<tr>
<td>5.0 APPLICATION LAYER</td>
<td>1-352</td>
</tr>
<tr>
<td>5.1 Application Layer Human Interface</td>
<td>1-352</td>
</tr>
<tr>
<td>5.2 A Sample Session</td>
<td>1-353</td>
</tr>
<tr>
<td>5.3 Terminal Mode</td>
<td>1-355</td>
</tr>
<tr>
<td>5.3.1 Sending Frames</td>
<td>1-356</td>
</tr>
<tr>
<td>5.3.2 Receiving Frames</td>
<td>1-356</td>
</tr>
<tr>
<td>5.4 Monitor Mode</td>
<td>1-356</td>
</tr>
<tr>
<td>5.5 High Speed Transmit Mode</td>
<td>1-357</td>
</tr>
<tr>
<td>APPENDIX A: COMPILING, LINKING, LOCATING, AND RUNNING THE SOFTWARE ON THE ISBC 186/51 BOARD</td>
<td>1-358</td>
</tr>
</tbody>
</table>

1-338
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 is 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 application 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 I" 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.)

The C_Assy_Support module has a run time start off 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.

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

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

**Figure 5. Free CB Pool**

**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 SEGMT contains the address of the

![Figure 7. Free RFA](231421-6)
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()
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_E_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_crs_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 indicates whether the transceiver passed its self test or not. The transceiver executes a self test after each transmission. If the transceiver’s self test passed, it will activate the collision signal during the IFS time.

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() 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.

Figure 11. Flow Chart for Sending a Frame
command blocks are 18 bytes long, while the MC Setup command can be up to 16,392 bytes. Therefore a separate Multicast Address 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.lpbk_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() 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 8256” 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:

- Data Link Layer (Station Component) 00H
- Transport Layer FEH
- Network Management Layer 08H
- User Processes multiples of 4 in the range OCH < LSAP ≤ FCH

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>XID</td>
<td>Exchange ID</td>
</tr>
<tr>
<td>TEST</td>
<td>TEST</td>
<td>Remote Loopback</td>
</tr>
</tbody>
</table>

Figure 15. IEEE 802.2 Class 1, Type 1 Commands and Responses

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_886()</td>
</tr>
<tr>
<td>Add_Dsap_Address (dsap, pfunc)</td>
<td>Add a DSAP address to the active list dsap - DSAP address pfunc - pointer to the SAP function</td>
</tr>
<tr>
<td>Delete_Dsap_Address (dsap)</td>
<td>Delete a DSAP address dsap - DSAP address</td>
</tr>
<tr>
<td>Recv_Frame (pfld)</td>
<td>Receives a frame from the 82586 Handler pfd - Frame Descriptor Pointer</td>
</tr>
<tr>
<td>Station_Component_Response (pfld)</td>
<td>Generates a response to a frame addressed to the Station Component pfd - Frame Descriptor Pointer</td>
</tr>
</tbody>
</table>
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>T: Terminal Mode</th>
<th>M: Monitor Mode</th>
</tr>
</thead>
<tbody>
<tr>
<td>X: High Speed Transmit Mode</td>
<td>V: Change Transmit Statistics</td>
</tr>
<tr>
<td>P: Print All Counters</td>
<td>C: Clear All Counters</td>
</tr>
<tr>
<td>A: Add a Multicast Address</td>
<td>Z: Delete a Multicast Address</td>
</tr>
<tr>
<td>S: Change the SSAP Address</td>
<td>D: Change the DSAP Address</td>
</tr>
<tr>
<td>N: Change Destination Node Address</td>
<td>L: Print All Addresses</td>
</tr>
<tr>
<td>R: Re-Initialize the Data Link</td>
<td>B: Change the Number Base</td>
</tr>
</tbody>
</table>

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 802.2/802.3 protocol converter.

Monitor Mode - allows the station to repeatedly transmit any size frame to the cable. While in the Monitor Mode, the terminal provides a dynamic update of 6 station related parameters.

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

Change Transmit Statistics - When Transmit Statistics is on several transmit statistics are gathered during transmission. If Transmit Statistics is off, statistics are not gathered and the program jumps over the section of code in the interrupt routine which gathers these statistics. The transmission rate is slightly increase when Transmit Statistics is off.

Print All Counters - Provides current information on the following counters.

- Good frames transmitted:
- Good frames received:
- CRC errors received:
- Alignment errors received:
- Out of Resource frames:
- Receiver overrun frames:

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.
Clear All Counters - Resets all of the counters.

Add/Delete Multicast Address - Adds and Deletes Multicast Addresses.

Change SSAP Address - Deletesthe 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 -> 00AA0000179E
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 -> 00AA0000179E
Would you like to add another Multicast Address? (Y or N) -> N
This Station's Host Address is: 00AA00001868
The Address of the Destination Node is: 00AA0000179E
This Station's LSAP Address is: 20
The Address of the Destination LSAP is: 20
The following Multicast Addresses are enabled: 00AA0000179E
---
Commands are:

T - Terminal Mode
X - High Speed Transmit Mode
M - Monitor Mode
P - Print All Counters
A - Add a Multicast Address
S - Change the SSAP Address
N - Change Destination Node Address
R - Re-Initialize the Data Link

V - Change Transmit Statistics
C - Clear All Counters
Z - Delete a Multicast Address
D - Change the DSAP Address
L - Print All Addresses
B - Change the number Base

Enter a command, type H for Help

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: 0
Frames that deferred: 4

Enter a command, type H for Help

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</th>
<th># of Good Frames</th>
<th>CRC Errors</th>
<th>Alignment Errors</th>
<th>No Resource Overrun Errors</th>
<th>Receive Resource Overrun Errors</th>
</tr>
</thead>
<tbody>
<tr>
<td>Transmitted</td>
<td>Received</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>32</td>
<td>0</td>
<td>00000</td>
<td>00000</td>
<td></td>
<td>00000</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

1-354
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 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.

<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 TxInt</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 TxInt</td>
</tr>
</tbody>
</table>

Table 1. FIFO State Table

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 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.

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 TxInt</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 TxInt</td>
</tr>
</tbody>
</table>

Table 1. FIFO State Table
5.3.1 SENDING FRAMES

The Terminal Mode is entered when the TerminalMode() function is called from the Menu interface. The TerminalMode() 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 sendFrame 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 SendFrame() function is called to transmit the frame, and continues to be called until the function returns TRUE.

The loop is repeated until a carriage return 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_Frame(). 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.

![Flow Chart](231421-16)
APPENDIX A
COMPILING, LINKING, LOCATING, AND RUNNING THE SOFTWARE ON THE 186/51 BOARD

******** Instruction 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

E36–E44 OUT E43–E47 OUT
E39–E47 OUT E46–E50 OUT
E37–E45 OUT E44–E48 OUT

WIRE WRAP
E36–E47 IN E43–E50 IN
E39–E44 IN E46–E47 IN
E79–E45 IN E90–E48 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
FFFFEH 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
cc86.86 :F6:%0 LARGE ROM DEBUG DEFINE(DEBUG) include(:F6:)
exit

******* submit file for linking and locating:  *******
run
:F6:uap.obj, lclib.lib to :F6:dd1.lnk size(stack(4000h)) notype
loc86 :F6:dd1.lnk to :F6:dd1.loc
initcode (OD000h) start(begin) order(classes(data, stack, code)) &
addresses(classes(data(3000h), stack(0CB00h), code(0D0020h)))
oh86 :F6:dd1.loc to :F6:dd1.rom
exit

******* submit file for burning EPROMs using IPFS:  *******
ipps
i 86
f :F6:dd1.rom (0d0000h)
3
2
1
0 to :F6:lo.byty
y
1 to :F6:hi.byty
y
t 27128
9
c :F6:lo.byty t p
n
C :F6:hi.byty t p
n
exit
/PCD/USR/CHUCK/CSRC/DLD.H

/***************************************************************
* B2586 Structures and Constants
***************************************************************

/# general purpose constants #/
#define INUSE 0
#define EMPTY 1
#define FULL 2
#define FREE 3
#define TRUE 4
#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_LPBK_INTERNAL 2
#define FAILED_LPBK_EXTERNAL 3
#define FAILED_LPBK_TRANSCEIVER 4

/# Frame Commands #/
#define UI 0x03  /* Unnumbered Information Frame */
#define XID 0x00  /* Exchange Identification */
#define TEST 0x03  /* Remote Loopback Test */
#define F_F_BIT 0x01  /* 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**(B-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; /* B2586 bus width: 0 - 16 bits
                      1 - 8 bits */
# Intermediate System Configuration Pointer ISCP *

```c
struct ISCP {
    u_short busy;  /* set to 1 by CPU before its first CA, cleared by 82386 after reading */
    u_short offset;  /* offset of system control block */
    u_short base1;  /* base of system control block */
    u_short base2;
};
```

/* System Control Block SCB */

```c
struct SCB {
    u_short stat;  /* Status word */
    u_short cmd;  /* Command word */
    u_short cbi_offset;  /* Offset of first command block in CBL */
    u_short rfa_offset;  /* Offset of first frame descriptor in RFA */
    u_short crc_err;  /* 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 */

```c
struct CB {
    u_short stat;  /* Status of Command */
    u_short cmd;  /* Command */
    u_short link;  /* Link field */
    u_short parmi;  /* Parameters */
    u_short parm2;
    u_short parm3;
    u_short parmi;
    u_short pani;
    u_short pani;
};
```

/* Multicast Address Command Block MA_CB */

```c
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[TADD_LEN.MULTI_ADDR_CNT];  /* MC address area */
};
```

/* Transmit Buffer Descriptor TBD */

```c
struct TBD {
};
```
```
// PCG/USR/CHUCK/CSRC/DLD.H

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 TB *buff_ptr; /* not used by the 386: used by the software to save address translation routine. */

/* Transmit Buffers */
struct TB {
    char data [TBUF_SIZE];
};

/* Frame Descriptor FD */
struct FD {
    u_short stat;  /* Status Word of FD */
    u_short el_s;  /* EL and 8 bits */
    u_short link;  /* Link to next FD */
    u_short rbd_offset; /* Receive buffer descriptor offset */
    char dest_addr[ADDR_LEN]; /* Destination address */
    char src_addr[ADDR_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 386: 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;   /* ISO Data Link Command */
};

/* LSAP Address Table */
struct LAT {
    char stat;  /* INUSE or FREE */
};
```
```c
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 mc address */
};

/* general purpose flags */

struct FLAGS {
    unsigned diag_done: 1; /* diagnosis command complete */
    unsigned stat_on: 1; /* network diagnostic statistics on/off */
    unsigned reset_sena: 1; /* don't reset when this bit is set */
    unsigned reset_end: 1; /* reset when this bit is set */
    unsigned lpbk_test: 1; /* loopback test flag */
    unsigned lpbk_mode: 1; /* loopback mode on/off */
};

/* General purpose bits */
#define ELBIT 0x8000
#define EOPBIT 0x8000
#define EBIT 0x4000
#define IBIT 0x2000
#define CBIT 0x1000
#define BBIT 0x4000
#define OMBIT 0x2000

/* SCB patterns */
#define CX 0x8000
#define FR 0x4000
#define CNA 0x2000
#define RNR 0x1000
#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 MC_SETUP 0x0003
#define TRANSMIT 0x0004
#define TDR 0x0005
#define DUMP 0x0006
#define DIAGNOSE 0x0007
```
# Define Command and Status Masks

```c
#define CMD_MASK 0x0007
#define NOERRBIT 0x2000
#define COLLMASK 0x000F
#define DEFERMASK 0x0080
#define NOCSRMASK 0x0400
#define UNDERRUNMASK 0x0100
#define SOEMASK 0x0040
#define NAICOLMASK 0x0020
#define OUT_OF_RESOURCES 0x0200
```

## Configure Parameters

```c
#define FIFO_LIM 0x0800 /* use FIFO lim of 8 */
#define BYTE_CNT 0x0008
#define SRDY 0x0040
#define SAV_BP 0x0080 /* address length of 6 bytes */
#define ADDR_LEN 0x0600 /* address length of 6 bytes */
#define PREAM_LEN 0x2000 /* preamble length of 8 bytes */
#define INT_LPBCK 0x4000
#define EXT_LPBCK 0x8000
#define LIN_PRIO 0x0000 /* no priority */
#define ACR 0x0000
#define EOF_MOT 0x0008
#define IFS 0x0006 /* IFS time 9.6 usec */
#define SLOT_TIME 0x0030 /* slot time 31.2 usec */
#define RETRY_NUM 0x0000 /* retry number 15 */
#define FTM 0x0001
#define FC_DIS 0x0002
#define MANCHESTER 0x0004
#define TONO_CRS 0x0008
#define MCRC_INS 0x0010
#define CRC_16 0x0020
#define BT_STUFF 0x0040
#define PAD 0x0080
#define CRSF 0x0000 /* no carrier sense filter */
#define CRS_SRC 0x0000 /* no collision detect filter */
#define CDIF 0x0000
#define CDT_SRC 0x8000
#define MIN_FRM_LEN 0x0040 /* 64 bytes */
#define MIN_DATA_LEN MIN_FRM_LEN - 18 /* assumes Ethernet/IEEE 802.3 */
#define MAX_FRAME_SIZE 1500 - 3 /* frames with 6 bytes of address */
```

---

1-364
# Define constants for storage area */
#define CB_CNT 8  /* Number of available Command Blocks */
#define FD_CNT 16  /* Number of available Frame Descriptors */
#define RB_CNT 64  /* Number of available Receive Buffer descriptors */
#define TBD_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 "dll.h"  /* 586 Data Structures */

/* 86 Timer Addresses */
#define TIMER1_CTL 0xFF5E
#define TIMER1_CNT 0xFF58
#define TIMER2_CTL 0xFF66
#define TIMER2_CNT 0xFF60

/* external functions */

/* I/O */
int inw();  /* input word : inw(address) */
void outw();  /* output word: outw(address, value) */
void init_intv();  /* initialize the interrupt vector table */
void enable();  /* enable 80186 interrupts */
void disable();  /* disable 80186 interrupts */
extern char *Build_Ptr();
short SEGNT;  /* Data segment value */
char *NULL;  /* NULL pointer */

/* Macro 'type' of definitions */
#define CA outb(0x3CB,0)  /* the command to issue a Channel Attention */
#define ESI_LOOPBACK outb(0xCB,0)  /* put the ESI in Loopback */
#define NO_ESI_LOOPBACK outb(0xCB,8)  /* take the ESI out of Loopback */
#define EDI_80130 outb(0xEO,0x63)  /* End Of Interrupt */
#define TIMER1_EDI_801B8 outw(0xFF22,0x04)  /* EDI for Timer 1 on the 86 */
#define TIMER1_EDI_80130 outw(0xEO,0x64)  /* EDI for 86's Timer1 on the 130 */
AP-235

/PCO/UGR/CHUCK/C5R/C/DLD.C

/******** memory allocation ***********/

int Self_Test;  // used for diagnostic purposes /
unsigned temp;  // temporary storage /

#define LPBK_FRAME_SIZE 4  // loopback frame storage /
char lpbk_frame[LPBK_FRAME_SIZE] = {
  0x55, 0xAA, 0x55, 0xAA};

#define whoami_io_add 0x000F0 /* I/O address of Host Address Prom */
char whoami[ADD_LEN];    /* Ram array where host address is stored */

/* transmission statistic variables */

unsigned long good_xmit_cnt;
unsigned short underrun_cnt;
unsigned short no_crs_cnt;
unsigned long defer_cnt;
unsigned short seq_err_cnt;
unsigned short max_col_cnt;
unsigned long recv_frame_cnt;
unsigned short reset_cnt;

/* Allocate storage for structures and buffers */

struct FLAGS flags;

/* 586 structures */

/* System Configuration Pointer: Rom Initialization */
/* struct SCP scp = {0x0000, 0x0000, 0x0000, 0x1FF8, 0x0000}; */
/* struct ISCP isp;   Intermediate System Configuration Pointer */

struct SCB scb;   /* System Control Block */

struct CB cb[CB_CNTJ;  /* Command Blocks */
  /*cb_tos, begin_cb1, end_cb1; */
    /* pointer to the beginning of the free command block list (cb_tos) and the
     beginning and end of the 82586 cb1 */

struct TBD tbd[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 */

231421-23
/* pointers to the beginning and the end of the rb list */

struct RB rb(RB_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_586() function */
struct CB res_cb; /* Temporary CB for reinitializing the 586 */
struct MA_CB res_ma_cb; /* Temporary MA_CB for reloading Multicast */
/* Hardware Support Functions */

Enable_586_Int()
{
    int c;
    c = inb(0x82); /* read the 80130 interrupt mask register */
    outb(0x82, 0x00F7 & c); /* write to the 80130 interrupt mask register */
}

Disable_586_Int()
{
    int c;
    c = inb(0x82);
    outb(0x82, 0x0008 & c);
}

Set_Timeout()
{
    outw(TIMER1_CNT, 0); /* Write a 0 to Timer1 count register */
    outw(0x0F5E, 0x0E09); /* Set Enable bit in Timer1 Mode/Control register */
}

Reset_Timeout()
{
    outw(0x0F5E, 0x0E09); /* Reset Enable bit in Timer1 Mode/Control register */
}

Init_Timer() /* 1B6's Timer 2 is a prescaler for Timer 1. It clocks Timer 1 every 32.7 msec. The deadman timeout is set for 1.25 sec */
{
    outw(0x0F58, 0x000C); /* Set Timer1 Interrupt Control register */
    outw(0x0F42, 0xFFFF); /* set max count register for timer2 to OFFFH */
    outw(0x0F5A, 38); /* set max count register A for timer1 */
    outw(0x0F68, 0xC01); /* Set Timer2 Mode/Control register */
    outw(0x0F5E, 0x0E09); /* Set Timer1 Mode/Control register */
    outw(0x0F28, (inw(0x0F28) & 0xFFF)); /* Enable 1B6 Timer1 interrupt */
    outw(0x0E2, (inw(0x0E2) & 0x00EF)); /* enable 80130 interrupt from 801B6 */
}

/* end hardware support functions */

Clear_Cnt();
/PCD/USR/CHUCK/CSRC/DLD.C

<
  sbc.errs_cnt = 0; /# clear 586 error statistic counters */
  sbc.errn errs_cnt = 0;
  sbc.ren_errs = 0;
  sbc.ovr_errs = 0;
  good_seq_cnt = 0; /# init data link statistics */
  underrun_cnt = 0;
  no_crs_cnt = 0;
  defcr_cnt = 0;
  recv_frame_cnt = 0;
  reset_cnt = 0;
>

Init_586()
<
  struct ISCP *iscp;
  struct MAT *mat;
  NO_ESI_LOOPBACK; /# Done for B2501. Inactivates CRS if powered up
                  in loopback */
  ESI_LOOPBACK;
  init_intv(); /# Initialization DLD's interrupt vectors */
  Init_Time();
  flags.reset_sema = 0; /# Initialize Reset Flags */
  flags.reset_send = 0;
  flags.stat_on = 1;
  Disable_586_1nt();

  iscp = 0x000000FF ; /# Initialize the ISCP pointers*/
  pb.<preh = 1;
  iscp->offset = Offset(scb);
  iscp->base1 = BEGMT << 4;
  iscp->base2 = (BEGMT >> 12) & 0x000F ;
  pNULL = Build_Ptr(NULL); /# build a NULL pointer - 8086 type: 32 bit */
  Build_Cb(); /# init Receive Frame Area */
  Build_Cb(); /# init Command Block list */
  ma.cb.cmd = 0; /# multicast address semaphore init */
  Clear_Cnt();
  sbc.stat = 0;
  CA: /# wait for the 586 to complete initialization */
  for ( i = 0; i <= 0xFF00; i++ )

231421-25
if (scb_stat == (CX : CNA))
  break;

if (i > 0xFF00)
  Fatal("DLD:Init - Did not get an interrupt after Reset/CAN");

/* Ack the reset Interrupt */
  scb_cmd = (CX : CNA);
  Wait_Scb();
  Enable_Scb();
  scb_cbl_offset = Offset(scb(O)); /* link scb to cb and fd lists */
  scb_rfs_offset = Offset(&fd(O));

  /* move the prom bytes into whoami array */
  for (i = 0; i < ADD_LEN; i++)
    whoami((ADD_LEN - 1) - i) = inb(whoami_io_addr + i * 2);

  /* Initialize the Multicast Address Table */
  for (pemat = &mat(O); pemat <= &mat(MULTI_ADDR_CNT - 1); pemat++)
    pemat->stat = FREE;
  Configure(INTERNAL_LOOPBACK); /* Put 386 in internal loopback */
  GetAddress(); /* Set up the station address */
  /* run diagnostics */
  Test_Link();
  if (Self_Test != PASSED)
    return(Self_Test);

  Configure(NO_LOOPBACK); /* Configure the B2566 */
  return(Self_Test);

Build_Rfs()
{
  for (pfld = &fd(O); pfld <= &fd(FD_CNT - 1); pfld++) {
    pfld->stat = pfld->e_q = 0;
    pfld->link = Offset(pfld+1);
    pfld->rbd_offset = NULL;
  }
# end_fd = --pf;
/* point to &fd( FD_CNT - 1 ) */

pf->link = NULL; /* last fd link is NULL */

pf->el = ELBIT; /* last fd has EL bit set */
begin_fd = pf = &fd(0); /* point to first fd */

pf->offset = Offset(&rbd[0]); /* link first fd to first rbd */

// Build a linear linked receive buffer descriptor list */
for (prbd = &rbd[0], pbuf = &rbuf[0], prbd <= &rbd[RBD_CNT - 1], prbd++, pbuf++) {

badd = SEGMNT << 4;
badd = Offset(pbuf);

prbd->buf = badd;
prbd->bch = badd >> 16;
prbd->buf_pch = pbuf;

prbd->act_cnt = 0;
prbd->link = Offset(prbd + 1);
prbd->size = RBUF_SIZE;
}

end_rbd = --prbd;
prbd->link = NULL; /* last rbd points to NULL */
prbd->size = ELBIT; /* last rbd has el bit set */
begin_rbd = &rbd[0];
}

Build_Cb() /* Build a stack of free command blocks */
{
struct CB *pcb;
struct TBD *ptbd;
struct TB *ptbu;
unsigned long badd;

for (pcb = &cb[0], pcb <= &cb(CB_CNT - 1), pcb++) {

pcb->stat = 0;
pcb->cmd = ELBIT;
pcb->link = Offset(pcb + 1);
}

pcb = begin_cbl = end_cbl = pNULL;
bcb->link = NULL;
bcb->lact = &cb[0];

// 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 = SEGMNT << 4;
}
/PCQ/USR/CHUCK/CSRC/DLD.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 */
  /* 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(NULL);
    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_sem = 1;
    Disable_586_INT();
    if ((ptbd = tbd_tos) != NULL) {
      tbd_tos = (struct Tbd *)Build_Ptr(ptbd->link);
      ptbd->link = NULL;
    }
    Enable_586_INT();
    flags.reset_sem = 0;
    if (flags.reset_pend == 1)
      Reset_586();
    return(ptbd);
  }

  Put_Tbd(ptbd)
/PCD/USR/CHUCK/CSRC/DLD.C

struct T8D *ptbd;
{
    struct T8D *p;
    /* find the end of the tbd list returned, ptbd is the beginning */
    for (p = ptbd; p->link ! = NULL; p = (struct T8D *) Build_Ptr(p->link)) ;
    p->act_cnt = TFUF_SIZE;  /* clear EOFBIT and update size on last tbd */
    tbd_tos = ptbd;
}

GetAddress()
{
    struct CB *pcb;
    #ifdef DEBUG
        if ((pcb = Get_Cb()) == NULL)
            Fatal("dld.c - GetAddress - couldn’t get a CB\n");
    #else
        pcb = Get_Cb();
    #endif  /* DEBUG */
    bcopy((char *)&pcb->parml, &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<=0xFFF0; i++)
                if (scb.cmd == 0)
                    break;
            if (i > 0xFFF0)
                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_SBS_int();
    if (begin_cbl == pNULL) { /* if the list is inactive start CU */
        begin_cbl = end_cbl = pcb;
        cb.cb_cmd_offset = Offset(pcb);
        Wait_Scb();
        cb.cmd = CU_START;
        set_deadman_timer_for_CU */
    }
    else {
        end_cbl->link = Offset(pcb);
        end_cbl = pcb;
    }
    Enable_SBS_int();
}

ISR7() /* EDI B0130 */
{
    outb(0xEO, 0x67); /* EDI B0130 */
}

ISR6() /* EDI B0130 */
{
    Write("\nInterrupt 6\n");
    outb(0xEO, 0x66); /* EDI B0130 */
}

ISR5() /* EDI B0130 */
{
    Write("\nInterrupt 5\n");
    outb(0xEO, 0x65); /* EDI B0130 */
}

/* Deadman Timer Interrupt Service Routine */

ISR_Timeout() /* Interrupt 4 */
{
    Reset_Timeout();
    if (flags.reset_sema == 1)
        flags.reset_sema = 1;
    else
        Reset_SBS();
    TIMER1_EDI_B0186;
    TIMER1_EDI_B0130;
}

/* Interrupt 0 is Uart in UAP Module */
/* Interrupt 2 is Timer in UAP Module */
ISR() {
    Write("\nInterrupt 1\n");
    outb(0xEO, 0x61); /* EDI 80130 */
}
/* 586 Interrupt service routine: Interrupt 3 */
ISR_586() {
    u_short stat_scb;
    struct CB *pcb;
    enable(); /* nesting only the uart interrupt */
    Wait_Scb();
    scb.cmd = (stat_scb = scb.stat & (CX | CNA | FR | RNR));
    CA;
    if (stat_scb & (FR | RNR))
        Recv_Int_Processing();
    if (stat_scb & CNA) {/* end of cb processing */
        Reset_Timeout(); /* clear deadman timer */
        pcb = Build_Ptr(scb.cb1_offset);
    #ifdef DEBUG
        if (begin_cbl == pNULL){
            Bug("DLD: begin_cbl == NULL in interrupt routine\n");
        }
        if (pcb->stat & 0x0000) /* set C bit not set or B bit set in interrupt routine\n");
    #endif /* 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 \se 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 & (SQEMASK | MAXCOLMASK | COLLMASK)) == 0)
                ++sqe_err_cnt;
            if (pcb->stat & DEFERMASK)
                ++defer_cnt;
        }
```c
if (pcb->stat & NOERRBIT)
    good_emit_cnt;
else {
    if (pcb->stat & NODCRSMASK)
        no_crs_cnt;
    if (pcb->stat & UNDERRUNMASK)
        underrun_cnt;
    if (pcb->stat & MAXCOLMASK)
        max_col_cnt;
}
if (pcb->parm1 == NULL)
    Put_Tbl(Build_Ptr(pcb->parm1));
break;

case DIAGNOSE:
    flagsdiag_done = 1;
    if ((pcb->stat & NOERRBIT) == 0)
        Self_Test = FAILED_DIAGNOSE;
    break;

default:
}

#endif check to see if another command is queued */
if (pcb->link == NULL)
    begin_cbl = pNULL;
else { /* restart the CU and execute the next command on the cbl */
    begin_cbl = Build_Ptr(pcb->link);
    scb_cbl_offset = pcb->link;
    Wait_Scb();
    scb.cmd = 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;
```
/PCD/USR/CHUCK/CSRC/DLD.C

Recv_Int_Processing()
{
  struct FD *pfld; /* points to the Frame Descriptor */
  struct RBD *eq;  /* points to the last rbd for the frame */
  struct RBD *prbd; /* points to the first rbd for the frame */

  for (pfld = begin_fd; pfld != pNULL; pfld = begin_fd)
    if (pfld->state & CBIT) <
      begin_fd = (struct FD *) Build_Ptr(pfld->link);
      prbd = (struct RBD *) Build_Ptr(fld->rbd_offset);
    if (prbd != pNULL) < /* check to see if a buffer is attached */

  ifdef DEBUG
    if (prbd != begin_rd)              
      Fatal("DLD: prbd != begin_rbd in Recv_Int_Processing\n");
  endif /* DEBUG */

  endif /* DEBUG */

  if (pfld->state & OUT_OF_RESOURCES)
    Put_Free_RFA(pfld);
  else {
    /* if the DLD is in a loopback test, check the frame recv */
    if (flags.loopback == 1)
      Loopback_Check(pfld);
    else
      /* if it's a multicast address check to see if it's
      in the multicast address table, if not discard the frame */
      if ((pfld->dest_addr[0] & O1) == O1) & (!Check_Multicast(pfld))
        Put_Free_RFA(pfld);
      else
        Recv_Frame(pfld);
        ++recv_frame_cnt;
      }
    
  }

  else {
    Ru_Start(); /* If RU has gone into no resources, restart it */
    break;
  }

  Loopback_Check(pfld) /* Called by Recv_Int_Processing; checks address
  and data of potential loopback frame */
  
  struct FD *pfld;
  struct RBD *prbd;
  struct RB *pbuf;

  231421-33
if ( bcmp((char *) bpf->src_addr, &whoami[0], ADD_LEN) == 0 ) {
    Put_Free_RFA(pfd);
    return;
}

prbd = (struct RBD *) Build_Ptr(pfd->prbd_offset); /* point to receive
    buffer descriptor */
prbuf = (struct RBD *) prbd->prbuf_ptr; /* point to receive buffer */

if ( bcmp((char *) prbuf->lpbk_frame[0], LPBK_FRAME_SIZE) == 0 ) {
    Put_Free_RFA(pfd);
    return;
}

flags.lpbk_test = 1; /* passed loopback test */
Put_Free_RFA(pfd);

Check_Multicast(pfd); /* returns true if multicast address is in MAT */
struct FD *pfdf;
{
    struct MAT *pmat;
    for (pmat = &mat[0]; pmat < &mat[MULTI_ADDR_CNT - 1]; pmat++)
        if ( (pmat->stat == INUSE &
                   (bcmp((char *) bpf->dest_addr, &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 */
    Send_Lpbk_Frame(); /* internal loopback test */
    if (flags.lpbk_test == 0) {
        Self_Test = FAILED_LPKB_INTERNAL;
        flags.lpbk_mode = 0;
        return;
    }

    flags.lpbk_test = 0;
    Configure(EXTERNAL_LOOPBACK); /* external loopback test w/ ESI in lpbk */
    Send_Lpbk_Frame();
    if (flags.lpbk_test == 0) {
        Self_Test = FAILED_LPKB_EXTERNAL;

    }
/PCD/USB/CHUCK/CBRC/CLD.C

flags.lpbk_mode = 0;
return;
}
flags.lpbk_test = 0;  /* external loopback test through transceiver */
NO_EBI_LOOPBACK:
Send_Lpbk_Frame();
if (flags.lpbk_test == 0)
Self_Test = FAILED_LPBK_TRANSCEIVER;
flags.lpbk_mode = 0;  /* leave loopback mode */
}
Send_Lpbk_Frame()
{
struct TBD *ptbd;
int i;
for (i = 0; i < B; i++) {  /* send lpbk frame B times, since it's best effort delivery */

#define DEBUG
if ((ptbd = Get_Tbd()) == pNULL)
Fatal("did - Send_Lpbk_Frame - couldn't get a TBD\n");
#else ptbd = Get_Tbd();
#endif /* DEBUG */
ptbd->act_cnt = EDGBIT | LPBK_FRAME_SIZE;
bgcopy((char *)ptbd->buff_ptr, &lpbk_frame[0], LPBK_FRAME_SIZE);
while(!Send_Frame(ptbd, &whoami[0]));
}

Diagnose()
{
struct CB *pcb;

#define DEBUG
if ((pcb = Get_CB()) == pNULL)
Fatal("did - Diagnose - couldn't get a CB\n");
#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 */

231421-35
/PC0/USB/CHUCK/CAPE/DLD.C

Configure(loopFlag)
  u_short loopFlag;
  struct CB *pcb;

#define DEBUG
if ((pcb = Get_Cb()) == pNULL)
  Fatal("did - Configure - couldn't get a CB
");
else
  pcb = Get_Cb();
#endif /* DEBUG */

/* Ethernet default parameters */
pcb->param1 = 0x0800;
pcb->param2 = 0x2600 ; loopflag;
pcb->param3 = 0x6000;
pcb->param4 = 0xF200;
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 : ELBIT;

Issue_CU_Cad(pcb);

Send_Frame(ptbd, padd) /* returns false if it can't get a Command block */
struct TBD *ptbd;
char *padd;
{
  struct CB *pcb;
  u_short length;
  flags.reset_sema = 1;
  if ((pcb = Get_Cb()) == pNULL) {
    flags.reset_sema = 0;
    if (flags.reset_send == 1)
      Reset_SB6();
    return(FALSE);
  }
pcb->param1 = Offset(ptbd);
/PCD/USB/CHUCK/CBRC/DLD.C

/* move destination address to command block */
bcopy((char *)pcb->para2, (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;
length += (ptbd->act_cnt & 0x3fff); /* 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 > TBUF_SIZE */
if ((length < MIN_DATA_LEN) && /* assumes a 4 byte CRC */
    (bcmp((char *)&pad, ADD_LEN) != 0))
  ptbd->act_cnt = MIN_DATA_LEN - 1; /* add one */

pcb->pam5 = length; /* length field */

pcb->cmd = TRANSMIT | ELBIT;

Issue_CU_Cmd(pcb);
flags.reset_sense = 0;
if (flags.reset_send == 1)
  return(TRUE);
)

Add_Multicast_Address(pma) /* pma - pointer to multicast address */
char *pma; /* returning false means the Multicast address table is full */
{
  struct MAT *pma;
  flags.reset_sense = 1;
  /* if the multicast address is a duplicate of one already in the MAT,
  then return */
  for (pma = mat; pma <= &mat[MULTI_ADDR_CNT - 1]; pma++)
    if (pma->stat == INUSE &&
        bcmp(pma->addr10, (char *)&pma, ADD_LEN) == 0))
      return(TRUE);

  for (pma = mat; pma <= &mat[MULTI_ADDR_CNT - 1]; pma++)
    if (pma->stat == FREE) {
      pma->stat = INUSE;
      bcopy((char *)&pma->addr10, (char *)&pma, ADD_LEN);
      break;
    }
if (pmat > &mat[MULTI_ADDR_CNT - 1]) {
    flags.reset.sema = 0;
    if (flags.reset_send == 1)
        Reset_586();
    return(FALSE);
}

Set_Multicast_Address();
flags.reset.sema = 0;
if (flags.reset_send == 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.sema = 1;
    for (pmat = mat; pmat <= &mat[MULTI_ADDR_CNT - 1]; pmat++)
        if (pmat->stat == INUSE &&
            bcmp(pmat->addr[0], (char *) pma, ADD_LEN) == 0) {
            pmat->stat = FREE;
            break;
        }

    if (pmat > &mat[MULTI_ADDR_CNT - 1]) {
        flags.reset.sema = 0;
        if (flags.reset_send == 1)
            Reset_586();
        return(FALSE);
    }

    Set_Multicast_Address();
    flags.reset.sema = 0;
    if (flags.reset_send == 1)
        Reset_586();
    return(TRUE);
}

Set_Multicast_Address()
{
    struct MAT *pmat;
    struct MA_CB *pma_cb;
    u_short i;

    i = 0;
    pma_cb = &ma.cb;
    while (pma_cb->cmd != 0) ; /* if the MA_CB is INUSE, wait until it's free */
    pma_cb->link = NULL;
for (pma = mat; pma <= mat[MULTI_ADDR_CNT - 1]; pma++)
    if (pma->stat == INUSE) {
        bcopy(pma->mc_addr[i], &pma->addr[0], ADD_LEN);
        i += ADD_LEN;
    }
    pma->mc_cnt = i;
    pma->cmd = MC_SETUP | ELBIT;
    Issue_GU_Cmd(pma);
}

Put_Free_RFA(pfd) /* Return Frame Descriptor and Receive Buffer Descriptions to the Free Receive Frame Area */
struct FD *pfd;
struct RBD *prbd; /* points to beginning of returned RBD list */
    /* points to end 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->val.s = ELBIT;
pfd->stat = 0;
prbd = (struct RBD *) Build_Ptr(pfd->rbd_offset); /* pick up the link to the rbd */
pfd->inl = pfd->rbd_offset = NULL;
/* Disable_S86_Int(); this command is only necessary in a multitasking program. However in this single task environment this routine is originally called from isr_S86(); therefore interrupts are already disabled */
if (begin_fd == pNULL)
    begin_fd = end_fd = pfd;
else {
    end_fd->link = Offset(pfd);
    end_fd->val.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 == NULL; 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
             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_Scb_Int() if Disable_Scb_Int() is used above */

   flags.reset_sema = 0;
   if (flags.reset_sema == 1)
     Reset_Scb();

   Ru_Start();
   if ((scb.stat & RU_MASK) == RU_READY) /* if the RU is already 'ready'
      then return */
     return;

   if ((begin_fd->stat & CBIT) == CBIT)
     return;

   begin_fd->rbd_offset = Offset(begin_rbd); /* limit the beginning of the rbd
      list to the first fd */
   scb.rfa_offset = Offset(begin_fd);
   Wait_Scb();
   scb.cmd = RU_START;
   CA;

   Software_Reset()
   { scb.cmd = RESET;
     CA;
     Wait_Scb();
   }

   Issue_Reset_Cmds()
   { Wait_Scb();
     scb.cmd = CU_START;
     CA;
   }
//PCG/USB/CMIXK/CSRC/DLD.C

Wait_Scb();

outw(0xff5e, O);    /* shut off timer 1 interrupt */
outw(TIMER1_CNT, O);
outw(0xff5e, 0x0009); /* use timer 1 without interrupt as a deadman */

while ((timw(0xff5e) & 0x0020) == 0)    /* if Max Cnt bit is set before CNA is set, 586 Cmd deadlocked */
    if ((scb.stat & & CNA) == CNA)
        break;

if (scb.stat & & 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 *mat;
    u_short i;

    ++reset_cnt;
    Disable_586_INT();
    EBI_LOOPBACK();
    Software_Reset();

    scb.stat = 0;
    CA;  /* wait for the 586 to complete initialization */

    for ( i = 0; i <= 0xff00; i++)
        if (scb.stat == (CX | CNA))
            break;

    if (i > 0xff00)
        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_ch1 == pNULL)
        Fatal("DLD: begin_ch1 = NULL in Reset_586");
endif /* DEBUG */
/PC0/USR/CHUCK/CBRC/DLD.C

/* Configure the 386 */
/* Ethernet default parameters: Configure is not necessary when using
  default parameters */
res_cb.link = NULL;
res_cb.param1 = 0x0800;
res_cb.param2 = 0x2600;
res_cb.param3 = 0x6000;
res_cb.param4 = 0xF200;
res_cb.param5 = 0x0000;
res_cb.param6 = 0x0040;
res_cb.cmd = CONFIGURE | ELBIT;

scb.cb1_offset = Offset(res_cb.stat);

Issue_Reset_Cmds();

/* Set the Individual Address */
bcopy((char *) res_cb.param1, &whoami[0], ADD_LEN); /* move the prom
  address to IA cmd */
res_cb.cmd = IA | ELBIT;

Issue_Reset_Cmds();

/* load the multicast addresses */
i = res_ma_cb.stat = 0;
res_ma_cb.link = NULL;

for (pmat = smat[0]; pmat <= smat[MULTI_ADDR_CNT - 1]; pmat++)
  if ((pmat->stat == IMUSE) ||
      bcopy((char *) res_ma_cb.mc_addr[i], &pmat->addr[0], ADD_LEN))
    i += ADD_LEN;

res_ma_cb.mc_cnt = i;
res_ma_cb.cmd = MC_SETUP | ELBIT;
scb.cb1_offset = Offset(res_ma_cb.stat);

Issue_Reset_Cmds();

/* Restart the Command Unit and the Receive Unit */
flags.reset_sema = 0;
flags.reset_pend = 0;
ND_EBI_LOOPBACK;
Recv_Int_Processing();

scb.cb1_offset = begin_cb1;
Wait_Bcb();
scb_cap = CU_START;
Set_Timeout(); /* Set Deadman Timer */
Enable_386_INT();
}

/* bcopy -- byte copy routine */
bcopy(dst, src, nbytes)
char *dst, *src;
int n
bytes;
{ while (nbytes--) *dst++ = *src++;
}

/* bcmp -- byte compare */
bcmp(s1, s2, nbytes)
char *s1, *s2;
int n
bytes;
{ while (nbytes-- && *s1++ == *s2++)
return(*--s1 - *--s2);
}
/PCD/USR/CHUCK/CBRC/LLC.C

#include "dID.h"

extern char *pNULL;
extern struct TBD *Get_Tbd();
extern char *Build_Ptr();

readonly char sid_frame[XID_LENGTH] = { 0, 0, XID, 0x81, 0x01, 0};
int DSAP, SSAP, XID, iid class 1 response #/

struct LAT lat[DSAP_CNT];

Init_Lat()
{
    struct LAT *plat;
    for (plat = &lat[0]; plat <= &lat[DSAP_CNT - 1]; plat++)
        plat->stat = FREE;
    return (Init_Service());
}

/* Function for adding a new DSAP */

Add_Dsap_Address(dsap, pfunc) /* DSAP must be divisible by 2^(B-N), where
                      2^(B-N) = DSAP_CNT. (i.e. N LSBs must be 0).
                      The function will return FALSE if does not
                      meet the above requirements, or the Lsap
                      Address Table is full, or the address has
                      already been used. NULL DSAP address is
                      reserved for the Station Component */

int dsap, (*pfunc)();
{
    struct LAT *plat;
    if ((dsap << (B-DSAP_SHIFT) & 0x0FF) != 0 || dsap == 0)
        return (FALSE);
    /* Check for duplicate dsaps. */
    if ((plat = &lat[dsap >> DSAP_SHIFT])-&gt;stat == FREE) {
        plat-&gt;stat = INUSE;
        plat-&gt;p_sap_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. */

231421-44

1-387
int dsap;

lat[dsap >> DSAP_SHIFT].stat = FREE;

Recv_Frame(pfd)
 struct FD   *epfd;
 {
 struct RBD   *eprbd;
 struct FRAME_STRUCT   *epfs;
 struct LAT   *eplat;
 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 too short. */
 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
 DUPLICATEADDRESSCHECK, -> no response
 frames should be recvd */
     Station_Component_Response(pfd);
   }
   /* not addressed to Station Component. */
   /* check to see if the dsap addressed is active */
   else if ( ((pfs->dsap << (5-DSAP_SHIFT)) & 0x0FF) == 0 &&
     (eplat = &lat[(pfs->dsap >> DSAP_SHIFT)]->stat == INUSE) ) {
     (*eplat->p_plat_func)(pfd);  /* call the function associated
     with the dsap received */
     return;
   }
   /* not Free_RFA(pfd);  */ /* return the dsap if not given to the user saps */
 }

Station_Component_Response(pfd)
 struct FD   *epfd;
 {
 struct FRAME_STRUCT   *epfs, *ptfs;
 struct TBD   *ptbd, *begintbd, *eq;
 struct RBD   *eprbd;
 prbd = (struct RBD *) Build_Ptr((pfd->rbd_offset));
 pfs = (struct FRAME_STRUCT *) prbd->buff_ptr;
 switch (pfs->cmd & "P_F_BIT") {  
  case XID:  

while ((ptbd = Get_Tbd()) != pNULL);
ptbd->act_cnt = EDFBIT | XID_LENGTH;
bcopy((char *) ptbd->buff_ptr, &id_frame[O], XID_LENGTH);  

ptfs = (struct FRAME_STRUCT *) ptbd->buff_ptr;

ptfs->cmd = prfs->cmd;

ptfs->ssap = prfs->ssap | C_R_BIT;  // return the frame to the sender
ptfs->sap = 0;
while (!Send_Frame(ptbd, Build_Ptr(pfd->src_addr))); break;

} else {
  begin_ptbd = ptbd;
  ptbd->act_cnt = prbd->act_cnt;
  bcopy((char *) ptbd->buff_ptr, (char *) prbd->buff_ptr, ptbd->act_cnt & Os3FFF);
  ptbd = prbd;
}

ptfs = (struct FRAME_STRUCT *) begin_ptbd->buff_ptr;
ptfs->cmd = prfs->cmd;

ptfs->ssap = prfs->ssap | C_R_BIT;  // return the frame to the sender
ptfs->sap = 0;
while (!Send_Frame(begin_ptbd, Build_Ptr(pfd->src_addr))); break;

}
```c
#include "did.h"

/* ASCII Characters */
#define ESC 0x1B
#define LF 0x0A
#define CR 0x0D
#define BS 0x08
#define BEL 0x07
#define BP 0x20
#define DEL 0x7F
#define CTRL_C 0x03

/* Hardware */
#define CH_B_CTL 0x008E
#define CH_A_CTL 0x008C
#define CH_B_DAT 0x008A
#define CH_A_DAT 0x0089
#define UART_STAT_MSK 0x70

/* Interrupt cases for B274 */
#define UART_TX_B 0
#define UART_RECV_B 0x0B
#define UART_RECV_ERR_B 0x0C
#define EXT_STAT_INT_B 0x04
#define EXT_STAT_INT_A 0x14

c char fifo_t[256];
c char fifo_r[256];
c char wrclntel; wrclntel;
unsigned char in_fifo_t, out_fifo_t, in_fifo_r, out_fifo_r, actual;
unsigned short t_buf_stat, r_buf_stat;

c char cbuf[201]; /* Command line buffer */
c char line[255]; /* Monitor Mode display line */
unsigned char dmap, ssap, send_flag, local_echo;
char Dest_Addr[ADD_LEN];
char Multi_Addr[ADD_LEN];
int tmstat; /* terminal mode status: for leaving terminal mode */
int dhex, monitor_flag, bs_stat; /* flags */

extern struct TBD; /*Get_Tbd();
extern char *Build_Ptr();
extern struct FLAGS* flags;
extern char *id_frame[1];
extern char *whoami[1];
```
/PCO/USR/CHUCK/CSRC/UAP.C

extern struct MAT mat[];
extern struct LAT lat[];
extern char *NULL;
extern unsigned long good_smnt_cnt;
extern u_short underrun_cnt;
extern u_short no_crs_cnt;
extern unsigned long defer_cnt;
extern u_short age_err_cnt;
extern u_short max_col_cnt;
extern unsigned long recv_frame_cnt;
extern u_short reset_cnt;
extern struct SCB scb;

/ * 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]|0x0FD)
define RTS_ONFA outb(CH_A_CTL, 0x05); outb(CH_A_CTL, wra[5]|wra[5]|0x02)
define RTS_OFFA outb(CH_A_CTL, 0x00); outb(CH_A_CTL, wra[5]|wra[5]|0x0FD)
define UART_TX_D1_B outb(CH_B_CTL, 0x01); outb(CH_B_CTL, wrb[1]|wrb[1]|0x0FD)
define UART_TX_E1_B outb(CH_B_CTL, 0x01); outb(CH_B_CTL, wrb[1]|wrb[1]|0x02)
define UART_RX_D1_B outb(CH_B_CTL, 0x01); outb(CH_B_CTL, wrb[1]|wrb[1]|0x07)
define UART_RX_E1_B outb(CH_B_CTL, 0x01); outb(CH_B_CTL, wrb[1]|wrb[1]|0x10)
define RESET_TX_INT outb(CH_B_CTL, 0x28)
define EDI_B0130_B274 outb(0xE0, 0x60)
define EDI_B0130_TIMER outb(0xE0, 0x62)

Enable_Uart_Int()
{
   int c;
   c = inb(0x0E2); /* read the 80130 interrupt mask register */
   outb(0x0E2, 0x00FE | c); /* write to the 80130 interrupt mask register */
}

Disable_Uart_Int()
{
   int c;
   c = inb(0x0E2);
   outb(0x0E2, 0x0001 | c);
}

Enable_Timer_Int()
{
   int c;
   outb(0x0EA, 125);
   outb(0x0EA, 0x00); /* Timer 1 interrupts every .125 sec */
   send_flag = FALSE;
   c = inb(0x0E2); /* read the 80130 interrupt mask register */
   outb(0x0E2, 0x00FB | c); /* write to the 80130 interrupt mask register */
}

231421-48
/PCO/USR/CHUCK/CSRC/UAP.C

Disable_Timer_Int()
{
    int c;
    c = inb(0x82);
    outb(0x82, 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(pmarg, cnt, pact)
{  
    char *pmarg;
    unsigned char cnt, *pact;
    unsigned char i;
    char c, buf[200];
    for (i = c = 0; (c != CR) && (c != LF) && (i < 198); i++)
    {
        c = Ci() & 0x7F;
        if (c == BS ! c == DEL) {
            if (i > 0) {
                Co(BS); Co(SP); Co(BS);
            } else
                if (c >= SP) {
                    Co(c);
                    buf[i++] = c;
                } else
                    if ((c == CR) || (c == LF)) {
                        buf[i++] = CR;
                        buf[i++] = LF;
                    } else
                        Co(BEL);
        }
        Co(CR); Co(LF);
        if (i > cnt)
            *pact = cnt;
        else
            *pact = i;
    for (i = 0; i < *pact; i++)
        *pmarg++ = buf[i];
    return;
}
Read_Char()
{
    unsigned char i;
    Read(&cbuf[0], BUFSIZE, &actual);
    i = Bskip(&cbuf[0]);
    return(cbuf[i]);
}

Write(pmsg)
char *pmsg;
{
    while (*pmsg == '0') {
        if (*pmsg == '
')
            Coi(CR);
        Co(*pmsg++);
    }
}

Fatal(pmsg) /* write a message to the screen then stop */
char *pmsg;
{
    Write("Fatal: ");
    Write(pmsg);
    for;;
}

Bug(pmsg) /* write a message to the screen then continue */
char *pmsg;
{
    Write("Bug: ");
    Write(pmsg);
}

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 - 0x37);  
    if ('a' <= c && c <= 'f')
        return(c - 0x37);
    return(OxFF);
}

Lower_Case(c)
char c;
{
    if ('a' <= c && c <= 'z')
        return(c);
    if ('A' <= c && c <= 'Z')
        return(c + 0x20);
    return(0);
Char_To_Ascii(c, ch) /* convert char to ASCII-Hex */
unsigned char c, ch[];

unsigned char i;
for ( i = (c & 0xFF) >> 4; 
    if (i < 10)
        ch[0] = i + 0x30;
    else
        ch[0] = i + 0x37;
    i = (c & 0x0F) ;
    if (i < 10)
        ch[1] = i + 0x30;
    else
        ch[1] = i + 0x37;
    ch[2] = '\0';
}

Skip(msg) /* skip blanks */
char *msg;

for (i = 0; *msg == ' '; i++, msg++);
return(i);

Read_Int() /* Read a 16 bit Integer */
{
    u_short wd, wh, wd1, wh1, j;
    char i, done, hex, dover, hover;
    for (done = FALSE; done == FALSE) {
        for (dover = false, hex = false, dover = false;
            // Read(&cbuf[0], BD, &actual);
            i = Skip(&cbuf[0]);
            //
            for (hex = dover = hover = FALSE, wd = wh = wd1 = wh1 = 0;
                // i = Ascii_To_Char(cbuf[i])) <= 15; i++) {
                if (j > 9)
                if (hex = TRUE;
                    wd = wd10; j;
                    wh = wh16 + j;
                    if (wd < wd1)
                        dover = TRUE;
                    if (wh < wh1)
                        hover = TRUE;
                    ud1 = wd; wh1 = wh;
                }
            if (cbuf[i] == 'H' || cbuf[i] == 'h' || cbuf[i] == CR || cbuf[i] == LF || cbuf[i] == ' ')
                // (hex = TRUE & & dover = FALSE)
                if (hex = TRUE;
                    done = TRUE;
                    if (hex = FALSE & & dover = FALSE)
                        done = TRUE;
                }
if (!done) {
    Write("\n This number is too big. \n It has to be less than 65536.\n ");
    Write("\n Enter number --> ");
    return(wh);
}

Int_To_Ascii(value, base, ld, ch, width) /* convert an integer to an ASCII string */

unsigned long value;
short base, ld;
char ch [ ], ld;

{ unsigned long dw, w;
  u_short j, i;
  for (i = 0; i < width; i++) {
    u_short j;
    char ch[j];
    if (!done) Int_To_Ascii(dw, 16, \"\", &ch[0], 0);  
    else Int_To_Ascii(dw, 10, \"\", &ch[0], 10);  
    for (j = 0; ch[j] != \'\0\' && j > 0; j++)
      ch[j] = ld;  
    ch[width] = \'\0\';
  }
}

Write_Long_Int(dw, i)

unsigned long dw;
short i;

{ unsigned long dw;
  u_short j, i;
  for (i = 0; i < width; i++)
    u_short j;
    char ch[j];
    if (!done) Int_To_Ascii(dw, 16, \"\", &ch[0], 0);
    else Int_To_Ascii(dw, 10, \"\", &ch[0], 10);
    for (j = 0; ch[j] != \'\0\' && j > 0; j++)
      ch[j] = ld;
  ch[width] = \'\0\';
}

Write_Short_Int(w, i)

u_short w, i;

{ u_short j, i;
  char ch[j];
  if (!done) Int_To_Ascii(dw, 16, \"\", &ch[0], 4);
  else

```c
//PC0/USB/CHUCK/CBCR/CAP.C

Int_To_Ascii(dw, 10, &ch[0], 5);
for (; j < 0; ch[j] = '0'; j++)
    line[i] = ch[j];

Yes()
{
    char b;
    for (; ; )
    {
        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 */
    {
        int i, j;
        for (; ; )
        {
            Write(pmsg);
            Read(&cbuf[0], 30, &actual);
            for (i = skip(&cbuf[0]), i = 0; i < 2*cnt; i++, j++)
                if (i < cbuf[j] && (cbuf[j] <= '9'))
                    cbuf[i] = cbuf[j] - '0';
                else
                    if ((i < cbuf[j] && (cbuf[j] <= 'F'))
                        cbuf[i] = cbuf[j] - 0x37;
                    else
                        if ((i < cbuf[j] && (cbuf[j] <= 'F'))
                            cbuf[i] = cbuf[j] - 0x37;
                        else
                            Write(" Illegal Character\n");
                        break;
                if (i >= 2*cnt - 1)
                    break;
            for (i = 0; i <= cnt - 1; i++)
                add[(cnt - 1) - i] = cbuf[2*i] << 4 | cbuf[2*i + 1];
            Write_Addr(padd, cnt)
            char *padd, cnt;
            { unsigned char i, cnt;
                for (i = cnt > 0; i-- )
                    
```


```c
/PCD/USR/CHUCK/CSRC/UAP.c

1 = padd(cnt-1);
Char_Uc_hexi1(&c[0]);
Write(&c[0]);
>
c[0] = 'n';
c[1] = 'O';
Write(&c[0]);
>
Recv_Data_l(pfd) /* Receives the frame from the 802.2 module */

struct FD
{
    struct FRAME_STRUCT *prfs, *prfs;
    struct TBD *ptbd, *begin_pstd, *eq;
    struct RBD *prbd;
    char *prbuf;
    int cnt;
}

prbd = (struct RBD *) Build_Ptr(pfd->prbd_offset);
prfs = (struct FRAME_STRUCT *) Build_Ptr(prbd->buff_ptr);

switch (prfs->cmd & 'F_BIT') {
    case UI:
        if (monitor_flag)
            break; /* Don't put data in fifo unless in terminal mode */
        prbuf = (char *) prfs;
        prbuf = 3; /* skip over the header info and point to the data */
        cnt = 3;
        pfd->length -= 3;
        for (; prbd != NULL; cnt = 0, prbuf = (char *) prbd->buff_ptr);
        for (; cnt < (prbd->act_cnt & 0x03FF) & pfd->length > 0;
            cnt++, prbuf++, pfd->length--){
            while(!buf_stat == FULL)
                Fifo_R_In(prbuf);
            prbd = Build_Ptr(prbd->link);
        }
    #ifdef DEBUG
        if (pfd->length == 0 & prbd != NULL)
            Fatal("Uap: Recv_Data_l(pfd) ");
    #endif /* DEBUG */
    break;
    case XID:
        while ((ptbd = Get_Tbd()) == NULL);
        pcopy((char *) ptbd->buff_ptr, &id_frame[0], XID_LENGTH);
        prfs = (struct FRAME_STRUCT *) ptbd->buff_ptr;
        prfs->cmd = prfs->cmd;
        prfs->ssap = prfs->ssap | C_R_BIT; /* return the frame to the sender */
        prfs->ssap = ssap;
        while(!Send_Frame(ptbd, Build_Ptr(pfd->src_addr)));
    }

231421-54

1-397
/PCD/USR/CHUCK/CSRC/UAP.C

break;

case TEST:
    for (prbd = (struct RBD *) Build_Ptr(pfd->prbd_offset);
        q = begin_prbd = pNULL; prbd != pNULL;
        prbd = Build_Ptr(prbd->link));
    while ((ptbd = Get_Ptr()) == 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 & Ox3FFF);
        pfs = (struct FRAME_STRUCT *) begin_ptbd->buff_ptr;

        pfs->cmd = pfs->cmd;
        pfs->ssap = pfs->ssap | C_R_BIT; / since return the frame to
        pfs->ssap = ssap;
        while(!Send_Frame(begin_ptbd, 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[out_fifo_t++];

    Disable_Uart_Int();
    if (out_fifo_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_ONB;
            t_buf_stat = INUSE;
        */
        Enable_Uart_Int();
        return(c);

    Fifo_T_In(c) /* called by Uart receive interrupt */
    char c;

    c = fifo_t[in_fifo_t++];
    if (t_buf_stat == EMPTY)
t_buf_stat = INUSE;  /* start filling Transmit Buffer Descriptor */
    else  /* if there are only 20 locations left, turn off the spigot */
    if (t_buf_stat == INUSE & in_fifo_t < 20 == outfifo_t) {
RTS_OFFB;
    t_buf_stat = FULL;
}

Fifo_R_Out()  /* called by transmit interrupt */
{
    char c;
    c = fifo_r(out_fifo_r++);
    if (out_fifo_r == in_fifo_r)  /* if the fifo is empty */
        r_buf_stat = EMPTY;
    else  /* if the fifo was full and is now draining */
    if (r_buf_stat == FULL & out_fifo_r - 01 == in_fifo_r)  /*
        r_buf_stat = INUSE;
    return(c);
}

Fifo_R_In()  /* called by Read_Data() */
{
    char c;
    fifo_r(in_fifo_r++) = c;
    Disable_Uart_Int();
    if (r_buf_stat == EMPTY) {
        UART_TX_E1_B;
        Co(b);     /* prime the interrupt */
        r_buf_stat = INUSE;
    }
    else  /* if the buffer is full, indicate it */
    if (r_buf_stat == INUSE & in_fifo_r == out_fifo_r)  /*
        r_buf_stat = FULL;
    Enable_Uart_Int();
}

Isr_Uart()  
{
    int stat;
    char c;
    outb(CH_B_CTL, 2);  /* point to RR2 in 8274 */
    switch(inb(CH_B_CTL) & 0x1C) {  /* read 8274 interrupt vector and service it */
        case UART TX_B:
            if (r_buf_stat == EMPTY) {
                UART_TX_DI_B;    /* if fifo is empty disable transmitter */
                REFB_TX_INT;
            }
            else
                outb(CH_B_DAT, Fifo_R_Out());
            break;
        231421-56
case UART_RECV_B:
    outb(CH_B_CTL, 1); /* point to RRI in 8274 */
    stat = inb(CH_B_CTL);
    outb(CH_B_CTL, 0x20);
    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)
        testat = 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:
    };

    EDI_80130_8274;
    EDI_B274;
}

 isr2()
{
    send_flag = TRUE;
    outb(0xEA, 125);
    outb(0xE0, 0x00); /* Timer 1 interrupts every .125 sec */
    outb(0xED, 0x62); /* EDI 80130 */
}

231421-57

1-400
Load_Leap()
{
    int Recv_Data_len;
    for(i = 0; i < sep ; i++)
        Read_Addr("\nEnter this station's LSAP in Hex --> ", &sep, 1);
    if (Add_Leap_Address(&sep, Recv_Data_len) <
        Write("\nError: LSAP Address must be one of the following:\n20H, 40H, 60H, 80H, A0H, COH, E0H \n"));
    else break;
}

Load_Multicast()
{
    for (i = 0; i < Multi_Addr[0].ADD_LEN
        Read_Addr("\nEnter the Multicast Address in Hex --> ", &Multi_Addr[0].ADD_LEN);
    if ((Multi_Addr[0] & OxO1) == O)  Write("\nError: the LBB of the Multicast Address must be 1\n"));
    else if (!Add_Multicast_Address(&Multi_Addr[0]) (      Write("\nError: Multicast Address Table is full\n"));
        break;
    else {
        Write("\nWould you like to add another Multicast Address?\n");
        Write("\nY or N --> ");
        if (!Yes())
            break;
    }
}

Remove_Multicast()
{
    for (i = 0; i < Multi_Addr[0].ADD_LEN
        Read_Addr("\nEnter the Multicast Address that you want to delete in Hex --> ", &Multi_Addr[0].ADD_LEN);
    if ((Multi_Addr[0] & OxO1) == O)  Write("\nError: the LBB of the Multicast Address must be 1\n"));
    else if (!Delete_Multicast_Address(&Multi_Addr[0]) (      Write("\nError: that Multicast Address doesn't exist\n"));
        break;
    else {
        Write("\nWould you like to delete another Multicast Address?\n");
        Write("\nY or N --> ");
        if (!Yes())
            break;
    }
/*PCO/UBR/CHUCK/C SRC/UP. C

Print_Addresses()
{
struct MAT *pmat;
int stat;

Write("\n This Stations Host Address is: ");
Write_Addr(pmat[0], ADD_LEN);
Write("\n The Address of the Destination Node is: ");
Write_Addr(pmat[Dest_Addr], ADD_LEN);
Write("\n This Stations LSAP Address is: ");
Write_Addr(&lsap, 1);
Write("\n The Address of the Destination LSAP is: ");
Write_Addr(&lsap, 1);
stat = FALSE;
for (pmat = &mat[0]; pmat <= &mat[MULTI_ADDR_CNT - 1]; pmat++)
if ((pstat = INUSE) {
    stat = TRUE;
    break;
}
if (stat) {
    Write("\n The following Multicast Addresses are enabled: ");
    for (pmat = &mat[0]; pmat <= &mat[MULTI_ADDR_CNT - 1]; pmat++)
if ((pstat = 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 Passed Diagnostic Self Tests\\n\\n\n");
else
if (stat == FAILED_DIAGNOSTIC)
    Write("\n FAILED DIAGNOSTIC\\n\\n\n");
else
if (stat == FAILED_LPBM_INTERNAL)
    Write("\n FAILED Internal Loopback Self Test\\n\\n\n");
else
if (stat == FAILED_LPBM_EXTERNAL)
    Write("\n FAILED External Loopback Self Test\\n\\n\n");
else
if (stat == FAILED_LPBM_TRANCEIVER)
    Write("\n FAILED External Loopback Through Transceiver Self Test\\n\\n\n");
}
Init_Uap()
{
outb(0xEO, 0x31); /*initialize 80130 pic - ICW1 */
outb(0xED, 0x20); /* ICW2 */
}*/

231421-59

1-402
# Start of documentation

```c

/PCG/USB/CHUCK/CSRC/UAP.C

coutb(0xE2, 0x10);  /* ICW3 */
coutb(0xE2, 0x00);  /* ICW4 */
coutb(0xE2, 0x10);  /* ICW4 */
coutb(0xE2, 0xFF);  /* mask all interrupts */
coutb(0xFF20, 0x0020);  /* set 80186 vector base */

/* Initialize the 80130 timers for Terminal Mode */
coutb(0xEE, 0x34);
coutb(0xEB, 0x38);
coutb(0xEB, 0x0B);  /* SYSTICK set for 1 msec */
coutb(0xE, 0x70);
coutb(0xEA, 125);
coutb(0xEA, 0x00);  /* Timer 1 interrupts every .125 sec */

/* Initialize the 8254 */
coutb(CH_B_CTL, 0x10);  /* CH_B_CTL, 0x28 */
coutb(CH_B_CTL, 0x30);
coutb(CH_B_CTL, 2);  /* CH_B_CTL, wrb(2) = 0x14 */
coutb(CH_B_CTL, 1);  /* CH_B_CTL, wrb(1) = 0x15 */
coutb(CH_B_CTL, 3);  /* CH_B_CTL, wrb(3) = 0xEA */

Write("***************
Write("     82546 IEEE 802.2/802.3 Compatible Data Link Driver 
Write("     82546 IEEE 802.2/802.3 Compatible Data Link Driver 
Init_DataLink();

drop = FALSE;
monitor_flag = TRUE;

Read_Adr("\nEnter the Address of the Destination Node in Hex --> ", &Dest_Addr[0], ADD_LEN);
Load_LSAP();

Read_Adr("\nEnter the Destination Node's LSAP in Hex --> ", &saps, 1);
Write("\nDo you want to Load any Multicast Addresses? (Y or N) -->
if (Yes())
    Load_Multicast();

Print_Addresses();
}

Terminal_Mode()
{
    int frame_cnt, buf_cnt;
    struct TBD *ptbd, *begin_ptbd;
    char *pbuf, c;

    Write("\n Would you like the local echo on? (Y or N)--
    if(Yes())
```

---

1-403
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;
ED1_80130_8274;
Enable_Uart_Int();
Enable_Timer_Int();
monitor_flag = FALSE;
timestamp = TRUE;
while (timestamp) {
  for (frame_cnt = 0; frame_cnt < MAX_FRAME_SIZE; q = ptbd) {
    while ((ptbd = Get_Tbd()) == pNULL); /* get a send 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++ = dsap;
      *pbuf++ = ssap;
      *pbuf++ = UI;
      buf_cnt = 3;
    } 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 send 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)
        break;
      while (t_buf_stat == EMPTY); /* wait until fifo has data */
      if ((c = *pbuf = Fifo_T_Out())) == CR) {
        *pbuf++; ++frame_cnt;
        break;
      }
      if (c == CR) { buf_cnt < TBUF_SIZE || send_flag) { /* last buffer in list */
        pbuf = dest_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, *q, *begin_ptbd;
     char *pbuf;
   }

   i = 0x20; frame_cnt = 0;
   for (; ; q = ptbd) {
      while ((ptbd = Get_Tbd()) == pNULL); // 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 802.2
         begin_ptbd = ptbd;
         *pbuf++ = dsap; // header information */
         *pbuf++ = ssap;
         *pbuf++ = UI;
         buf_cnt = 3;
       }
      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 xmit buffer with ASCII characters */
      for (; buf_cnt < TBUF_SIZE && cnt > 0;)
      { *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 *ptbd, *begin_tbd;
     printf("Do you want this station to transmit? (Y or N) --> ");
     if (Yes())
               
231421-62
/PCD/USR/CHUCK/CSRC/UAP.C

for (smt = FALSE; smt == FALSE; ) {
    if (cnt = Read_Int;() {
        if (cnt > 2045) {
            Write("\n Enter the number of data bytes in the frame --> ");
        }
        Write("Sorry, the number has to be less than 2046\n");
        cnt = Read_Int();
    }
    else smt = TRUE;
}
else smt = FALSE;

Write("\n Hit any key to exit Monitor Mode.\n");
Write("\n# of Good Frames Frames Errors CRC Alignment No Resource Overrun\n");
Write("\n Transmitted Received Errors Errors Errors Errors\n");

for (i = 0; i < 79; i++)
    line[i] = 0x20;

line[79] = CR;
line[80] = \n;  
while (line[CH_B_CTL] & 1) {
    for (i = 0; i < 72; i++)
        line[i] = 0x20;
    Write_Long_Int(good_smint_cnt, i);
    Write_Long_Int(recv_frame_cnt, i);
    Write_Long_Int(scb_crc_err, i);
    Write_Long_Int(scb_align_err, i);
    Write_Long_Int(scb_ovr_err, i);
    Write tremend(O0);
    if (smt)
        ptd = Build_Frame(cnt);
        while(!Send_Frame(ptbd, &Dest_Addr[0]));
}

Hs_Xmit_Mode();

struct TBD *ptbd;

Write("\n Hit any key to exit High Speed Transmit Mode.\n");
hs_stat = TRUE;
EOI_B0130_B0274;
Enable_Uart_Int();

/* Execute this loop until a recv char interrupt happens at Uart */
while(figsize) {
    while((ptbd = Get_Tbd()) == pNULL); /* get a smt buffer from the data link */
    ptbd->act_cnt = EOFBIT; /* set the End Of Frame bit */
    while(!Send_Frame(ptbd, &Dest_Addr[0])); /* Send Frame */
}
}

Print_Cnt()
{
    char ch[11], base, width, width_i,
    unsigned long temp;

    (dhex) {
        width = 3;
        base = 16;
    }
    else {
        base = 10;
        width = 10;
        width_i = 5;
    }

    Write("\n\n Good frames transmitted: ");
    for (i = 1; i <= 11 - width; i++)
        Co(SP);
    Int_To_Ascii(ch[0], 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(ch[0], base, \', ', &ch[0]. width);
    for (i = width - 1; i >= 0; i--)
        Co(ch[i]);
    Write(" \n \n CRC errors received: ");
    for (i = 1; i <= 15 - width; i++)
        Co(SP);
    temp = scb_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 = scb_aln_errs;
    Int_To_Ascii(temp, base, \', ', &ch[0]. width);
    for (i = width - 1; i >= 0; i--)
        Co(ch[i]);
    Write(" Out of Resource frames: ");
    for (i = 1; i <= 12 - width; i++)
        Co(SP);
    temp = scb_rec_errs;
    Int_To_Ascii(temp, base, \', ', &ch[0]. width);
for (i = width - 1; i >= 0; i--) 
    CoCch[i]);
    Write(f" Receiver overrun frame: ");
    for (i = 1; i <= 12 - width; i++)
        Co[SP];
    temp = UVR errors;
    Int_To_Ascii(temp, base, ' ', &ch[O], width);
    for (i = width - 1; i >= 0; i--)
        CoCch[i]);
    Write(f"\n\n RXDMA Reset: ");
    for (i = 1; i <= 29 - width; i++)
        Co[SP];
    \n\n u = reset_cnt;
    Int_To_Ascii(temp, base, ' ', &ch[O], width);
    for (i = width - 1; i >= 0; i--)
        CoCch[i]);
    Write(f" Transmit under run frame: ");
    for (i = 1; i <= 11 - width; i++)
        Co[SP];
    temp = underrun_cnt;
    Int_To_Ascii(temp, base, ' ', &ch[O], width);
    for (i = width - 1; i >= 0; i--)
        CoCch[i]);
    Write(f"\n\n Lost CARR: ");
    for (i = 1; i <= 26 - width; i++)
        Co[SP];
    \n\n \n temp = sqe_err_cnt;
    Int_To_Ascii(temp, base, ' ', &ch[O], width);
    for (i = width - 1; i >= 0; i--)
        CoCch[i]);
    Write(f"\n\n Maximum retry: ");
    for (i = 1; i <= 21 - width; i++)
        Co[SP];
    temp = max_col_cnt;
    Int_To_Ascii(temp, base, ' ', &ch[O], width);
    for (i = width - 1; i >= 0; i--)
        CoCch[i]);
    Write(f" Frames that deferred: ");
    for (i = 1; i <= 15 - width; i++)
        Co[SP];
    Int_To_Ascii(defer_cnt, base, ' ', &ch[O], width);
    for (i = width - 1; i >= 0; i--)
        CoCch[i]);
    \n\n Print_Help()
    { 
        Write(f"\n\n Commands are:\n\n\n T - Terminal Mode  M - Monitor Mode\n\n\n")
    }
Write ("X - High Speed Transmit Mode");  \nWrite ("P - Print All Counters");  \nWrite ("A - Add a Multicast Address");  \nWrite ("B - Change the BSAP Address");  \nWrite ("N - Change Destination Node Address");  \nWrite ("R - Re-Initialize the Data Link");  
>  
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 'x':  
        Hs_Xmit_Mode();  
        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':
AP-235

/PCD/USB/CHUCK/CSRC/UAP.C

Delete_Dsap_Address(ssap);
Load_Dsap();
bhak;
case 'd':
Read_Adr("Enter the Destination Node's LSAP in Hex --> ", &dsap);
bhak;
case 'n':
Read_Adr("Enter the Address of the Destination Node in Hex --> ", &Dest_Addr[0], ADD_LEN);
bhak;
case 'l':
Print_Addresses();
bhak;
case 'r':
Software_Reset();
Init_DataLink();
Add_Dsap_Address(ssap, Recv_Data_1);
bhak;
case 'b':
Write("In The current base is ");
if (dhex == TRUE)
Write("Hex. Would you like to change it ? (Y or N) --> ");
else
Write("Decimal. Would you like to change it ? (Y or N) --> ");
if (Yes()) {
if (dhex == TRUE)
dhex = FALSE;
else dhex = TRUE;
}
bhak;
default:
Write("Unknown command\n");
bhak;
}
/PCG/USR/CHUCK/CSR/ASSV. ASM

name c assy support
stack segment stack 'stack'
stack top label word
stack ends

OLD_DATA segment public 'DATA'
estrn SEGMT_:word 
data segment address
OLD_DATA ends

UAP_DATA segment public 'DATA'
UAP_DATA ends

OLD_CODE segment public 'CODE'
estrn Isr_Vart_:far, Isr5Far, Isr1_far
ENDS

UAP_CODE segment public 'CODE'
estrn Isr_Vart_:far, Isr5Far, Main_:far

OLD_CODE segment public 'CODE'
estrn Isr_Vart_:far, Isr5Far, Main_:far

DO_CODE segment public 'CODE'
estrn Inw_outw_initintv enable disable Build_Ptr

arg1 equ [BP + 6]
arg2 equ [BP + 10]
assume CS:DO_CODE
assume DS:DLD_DATA

; initialization program for the 82586 data link driver

begin:

mov ax, DLD_DATA ; get base of dgroup and
mov SEGMT_ ax ; pass the segment value to the c program
mov ax, Main ; go to the c program
hlt

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/CSRC/ASSY.ASM

    pop  BP
    ret

    inb_ endp

    outb_ proc far
    push BP
    mov BP, SP
    push DX
    push AX
    mov DX, arg1
    mov AX, arg2
    out DX, AL
    pop AX
    pop DX
    mov SP, BP
    pop BP
    ret

    outb_ endp

    inw_ proc far
    push BP
    mov BP, SP
    push DX
    mov DX, arg1
    in AX, DX
    pop DX
    mov SP, BP
    pop BP
    ret

    inw_ endp

    outw_ proc far
    push BP
    mov BP, SP
    push DX
    push AX
    mov DX, arg1
    mov AX, arg2
    out DX, AX
    pop AX
    pop DX
    mov SP, BP
    pop BP
    ret

    outw_ endp

    Build_Ptr_ proc far
    push BP
    mov BP, SP
    mov DX, DL_DATA
    mov AX, arg1
    mov SP, BP
    pop BP
    ret

    Build_Ptr_ endp

    Offset_ proc far
/PCD/USR/CHUCK/CSRC/ABBY.ASM

push BP
mov BP, BP
mov AX, arg1
mov SP, BP
pop BP
ret

Offset_endp

serve_int_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 isr_586_
pop ES
pop DS
pop DI
pop SI
pop DX
pop CX
pop BX
pop AX
iret

serve_int_isr endp

serve_int_6274 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 isr_Uart_
pop ES
pop DS
pop DI
pop SI
pop DX
serve_int_timeout proc far
push AX
push BX
push CX
push DX
push SI
push DI
push DB
push ES
mov AX, DLD_DATA
mov DS, AX
mov ES, AX
call Isr_Timeout_
pop ES
pop DS
pop SI
pop DX
pop CX
pop BX
pop AX
iret
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 ES, AX
call Isr7_
pop ES
pop DS
pop SI
pop DX
pop CX
pop BX
pop AX
iret
serve_int7_isr endp
داعب

`server_int7_isr endp`

`server_int6_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 Isr6_`

`pop ES`
`pop DS`
`pop DI`
`pop SI`
`pop DX`
`pop CX`
`pop BX`
`pop AX`

`iret`

`server_int6_isr endp`

`server_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 Isr5_`

`pop ES`
`pop DS`
`pop DI`
`pop SI`
`pop DX`
`pop CX`
`pop BX`
`pop AX`

`iret`

`server_int5_isr endp`

231421-72
; serve_int2_isr proc far
    push AX
    push BX
    push CX
    push DX
    push SI
    push DI
    push DS
    push ES
    mov AX, UMP_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
    sti

; Interrupt types for the 8086/8088 COMPUTER
mov DS:word ptr 80h, offset serve_int_isr
mov DS:word ptr 82h, DS_CODE
mov DS:word ptr 84h, offset serve_int_isr
mov DS:word ptr 86h, DS_CODE
mov DS:word ptr 88h, offset serve_int_isr
mov DS:word ptr 8Ah, DS_CODE
mov DS:word ptr 8Ch, offset serve_int_isr
mov DS:word ptr 8Eh, DS_CODE
mov DS:word ptr 90h, offset serve_int_timeout
mov DS:word ptr 92h, DS_CODE
mov DS:word ptr 94h, offset serve_int_isr
mov DS:word ptr 96h, DS_CODE
mov DS:word ptr 98h, offset serve_int_isr
mov DS:word ptr 9Ah, DS_CODE
mov DS:word ptr 9Ch, offset serve_int_isr
mov DS:word ptr 9Eh, DS_CODE
pop AX
pop DS
ret

end

begin: ds:did_data, ss:stack:stktop

231421-74
Implementing StarLAN with the Intel 82588

ADI GOLBERT
DATA COMMUNICATIONS OPERATION

SHARAD GANDHI
FIELD APPLICATIONS-EUROPE
**CONTENTS**

<table>
<thead>
<tr>
<th>PAGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.0 INTRODUCTION ........................................ 1-421</td>
</tr>
<tr>
<td>1.1 StarLAN ............................................. 1-421</td>
</tr>
<tr>
<td>1.2 The 82588 ........................................... 1-421</td>
</tr>
<tr>
<td>1.3 Organization of the Application Note .................... 1-421</td>
</tr>
<tr>
<td>1.4 References ........................................... 1-421</td>
</tr>
<tr>
<td>2.0 StarLAN ................................................ 1-421</td>
</tr>
<tr>
<td>2.1 StarLAN Topology ...................................... 1-422</td>
</tr>
<tr>
<td>2.1.1 Telephone Network .................................... 1-422</td>
</tr>
<tr>
<td>2.1.2 StarLAN and the Telephone Network .................... 1-423</td>
</tr>
<tr>
<td>2.1.3 StarLAN and Ethernet ................................ 1-425</td>
</tr>
<tr>
<td>2.2 Basic StarLAN Components .............................. 1-425</td>
</tr>
<tr>
<td>2.2.1 A StarLAN Node Interface ............................ 1-426</td>
</tr>
<tr>
<td>2.2.2 StarLAN HUB ......................................... 1-426</td>
</tr>
<tr>
<td>2.2.3 StarLAN Cable ....................................... 1-428</td>
</tr>
<tr>
<td>2.3 Framing ................................................ 1-428</td>
</tr>
<tr>
<td>2.4 Signal Propagation and Collision ........................ 1-429</td>
</tr>
<tr>
<td>2.4.1 Situation #1 ....................................... 1-431</td>
</tr>
<tr>
<td>2.4.2 Situation #2 ....................................... 1-431</td>
</tr>
<tr>
<td>2.4.3 Situation #3 ....................................... 1-431</td>
</tr>
<tr>
<td>2.5 StarLAN System And Network Parameters ................... 1-431</td>
</tr>
<tr>
<td>3.0 LAN CONTROLLER FOR StarLAN .......................... 1-431</td>
</tr>
<tr>
<td>3.1 IEEE 802.3 Compatibility ................................ 1-431</td>
</tr>
<tr>
<td>3.2 Configurability of the 82588 .......................... 1-431</td>
</tr>
<tr>
<td>3.3 Clocks and Timers ..................................... 1-433</td>
</tr>
<tr>
<td>3.4 Manchester Data Encoding and Decoding .................. 1-433</td>
</tr>
<tr>
<td>3.5 Detection of the Collision Presence Signal .............. 1-434</td>
</tr>
<tr>
<td>3.5.1 Collision Detection by Code Violation ................ 1-435</td>
</tr>
<tr>
<td>3.5.2 Collision Detection by Signature (or Bit) Comparison ... 1-435</td>
</tr>
<tr>
<td>3.5.3 Additional Collision Detection Mechanisms .............. 1-436</td>
</tr>
<tr>
<td>CONTENTS</td>
</tr>
<tr>
<td>----------------------------------------------</td>
</tr>
<tr>
<td>3.6 Carrier Sensing</td>
</tr>
<tr>
<td>3.7 Squelching the Input</td>
</tr>
<tr>
<td>3.8 System Bus Interface</td>
</tr>
<tr>
<td>3.9 Debug and Diagnostic Aids</td>
</tr>
<tr>
<td>3.10 Jitter Performance</td>
</tr>
<tr>
<td>4.0 THE 82588</td>
</tr>
<tr>
<td>4.1 Transmit and Retransmit Operations</td>
</tr>
<tr>
<td>4.2 Configuring the 82588</td>
</tr>
<tr>
<td>4.3 Frame Reception</td>
</tr>
<tr>
<td>4.3.1 Multiple Buffer Frame Reception</td>
</tr>
<tr>
<td>4.4 Memory Dump of Registers</td>
</tr>
<tr>
<td>4.5 Other Operations</td>
</tr>
<tr>
<td>5.0 StarLAN NODE FOR IBM PC</td>
</tr>
<tr>
<td>5.1 Interfacing to the IBM PC I/O Channel</td>
</tr>
<tr>
<td>5.1.1 Register Access and Data Bus Interface</td>
</tr>
<tr>
<td>5.1.2 Control Port</td>
</tr>
<tr>
<td>5.1.3 Clock Generation</td>
</tr>
<tr>
<td>5.1.4 DMA Interface</td>
</tr>
<tr>
<td>5.1.5 Interrupt Controller</td>
</tr>
<tr>
<td>5.2 Serial Link Interface</td>
</tr>
<tr>
<td>5.2.1 Transmit Path</td>
</tr>
<tr>
<td>5.2.2 Idle Pattern Generation</td>
</tr>
<tr>
<td>5.3 Receive Path</td>
</tr>
<tr>
<td>5.4 80188 Interface to 82588</td>
</tr>
<tr>
<td>5.5 iSBX Interface to StarLAN</td>
</tr>
<tr>
<td>6.0 THE StarLAN HUB</td>
</tr>
<tr>
<td>6.1 A StarLAN HUB for the IBM/PC</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>CONTENTS</th>
<th>PAGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>6.1.1 HUB Input Ports</td>
<td>1-465</td>
</tr>
<tr>
<td>6.1.2 Collision Detection</td>
<td>1-466</td>
</tr>
<tr>
<td>6.1.3 The Local 82588</td>
<td>1-467</td>
</tr>
<tr>
<td>6.1.4 The Collision Presence Signal</td>
<td>1-467</td>
</tr>
<tr>
<td>6.1.5 Signal Retiming</td>
<td>1-468</td>
</tr>
<tr>
<td>6.1.6 Retiming Circuit, Theory of Operation</td>
<td>1-468</td>
</tr>
<tr>
<td>6.1.7 Retiming Circuit Implementation</td>
<td>1-469</td>
</tr>
<tr>
<td>6.1.8 Driver Circuits</td>
<td>1-471</td>
</tr>
<tr>
<td>6.1.9 Header/Intermediate HUB Switch</td>
<td>1-471</td>
</tr>
<tr>
<td>6.1.10 Jabber Function</td>
<td>1-471</td>
</tr>
<tr>
<td>6.1.11 HUB Receiver Protection Timer</td>
<td>1-472</td>
</tr>
<tr>
<td>6.1.12 HUB Reliability</td>
<td>1-472</td>
</tr>
<tr>
<td>7.0 SOFTWARE DRIVER</td>
<td>1-473</td>
</tr>
<tr>
<td>7.1 Interfacing to IBM PC</td>
<td>1-473</td>
</tr>
<tr>
<td>7.1.1 Doing I/O on IBM PC</td>
<td>1-473</td>
</tr>
<tr>
<td>7.2 Initialization and Declarations</td>
<td>1-475</td>
</tr>
<tr>
<td>7.3 General Commands</td>
<td>1-477</td>
</tr>
<tr>
<td>7.4 DMA Routines</td>
<td>1-480</td>
</tr>
<tr>
<td>7.5 Interrupt Routines</td>
<td>1-482</td>
</tr>
</tbody>
</table>

| APPENDIX A: StarLAN SIGNALS                  | 1-485 |
| APPENDIX B: 802.3 1BASE5 MULTI-POINT EXTENSION (MPE) | 1-488 |
| APPENDIX C: SINGLE DMA CHANNEL INTERFACE     | 1-490 |
| APPENDIX D: MEASURING NETWORK DELAYS WITH THE 82588 | 1-492 |
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

![Figure 1. StarLAN Topology](image)
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](image)

**Figure 2. Telephone Wiring in a Building**

![Figure 3. Coexistence of Telephone and StarLAN](image)

*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
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

Figure 6. 82588 Based StarLAN Node
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 Information field is less than 46 bytes, the remaining bytes are filled with 0.
tion field is less than 46, padding bytes are used to make the field 46 bytes long. The Length field indicates how much real data is in the Information field. The last 32 bits of the frame is the Frame Check Sequence (FCS) and contains the CRC for the frame. The CRC is calculated from the beginning of the destination address to the end of the Information field. The generating polynomial (Autodin II) used for CRC is:

$$X^{32} + X^{26} + X^{23} + X^{16} + X^{12} + X^{11} + X^{10} + X^{8} + X^{7} + X^{6} + X^{4} + X^{2} + X + 1$$

No need for Figure N.

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.
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>

Beside 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**

1-432
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 1 from 0 to 1 for 0 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 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, ..., l. 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 occurs 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 at 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>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
</tr>
</thead>
<tbody>
<tr>
<td>C</td>
<td>C</td>
<td>S</td>
<td>S</td>
<td>L</td>
<td>L</td>
<td>L</td>
<td>L</td>
<td>L</td>
<td>C</td>
<td>C</td>
<td></td>
<td></td>
</tr>
<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>
</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)

<table>
<thead>
<tr>
<th>1</th>
<th>2</th>
<th>4</th>
<th>6</th>
<th>8</th>
<th>10</th>
<th>12</th>
<th>14</th>
<th>16</th>
<th>18</th>
<th>20</th>
<th>22</th>
<th>24</th>
<th>26</th>
</tr>
</thead>
<tbody>
<tr>
<td>C</td>
<td>C</td>
<td>C</td>
<td>C</td>
<td>S</td>
<td>S</td>
<td>S</td>
<td>S</td>
<td>S</td>
<td>C</td>
<td>C</td>
<td>C</td>
<td>C</td>
<td>C</td>
</tr>
<tr>
<td>0</td>
<td>2</td>
<td>4</td>
<td>6</td>
<td>8</td>
<td>10</td>
<td>12</td>
<td>14</td>
<td>16</td>
<td>18</td>
<td>20</td>
<td>22</td>
<td>24</td>
<td>26</td>
</tr>
</tbody>
</table>

Collision also if:
- RxD stays low for 25 samples or more
- A mid cell transition is missing

A single instance of code violation can qualify as collision. The 82588 has a parameter called collision detect 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
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.

![Figure 17. Mode 0, Collision Detection](image)

![Figure 18. Carrier Sensing](image)
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](image19)

![Figure 20. Access Times](image20)
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-
4 Status registers are accessed through one read port

The pointer can be changed using a command or can be automatically incremented.

```assembly
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;
```

**READING 4 STATUS REGISTERS**

**Figure 22. Reading the Status Register**

Nostics functions. The DIAGNOSE command of the 82588 does a self-test of most of the counters and timers in the 82588 serial unit. Using the DUMP command, all the internal registers of the 82588 can be dumped into the memory. The TDR command does Time Domain Reflectometry on the network. The 82588 has two loopback modes of operation. In the internal loopback mode, the TXD line is internally connected to the RXD one. No data appears outside the chip, and the 82588 is isolated from the link. This mode enables checking of the receive and transmit machines without link interference. In the external loopback mode, the 82588 becomes a full duplex device, being able to receive its own transmitted frames. In this mode data goes through the link and all CSMA/CD mechanisms are involved.

### 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.

![Jitter Diagram](image)

**Figure 23. 82588 Jitter Performance**

### 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.

I. Prepare Transmit Data—Block in Memory
2. Program DMA Controller
3. Issue Transmit Command on the Desired Channel

<table>
<thead>
<tr>
<th>BYTE COUNT</th>
<th>DESTIN. ADDRESS</th>
</tr>
</thead>
<tbody>
<tr>
<td>INFORMATION</td>
<td></td>
</tr>
</tbody>
</table>

Transmit Data Block

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.

<table>
<thead>
<tr>
<th>TX DEF</th>
<th>MAX COLL</th>
<th>NUM. OF COLLISIONS</th>
</tr>
</thead>
<tbody>
<tr>
<td>HRT BEAT</td>
<td>LOST CRS</td>
<td>LOST CTS UNDER RUN</td>
</tr>
</tbody>
</table>

Transmit & Retransmit Results Format

At the conclusion of transmission the 82588 generates an interrupt to the CPU. The CPU can then read the status registers to find out if the transmission was successful. If a collision occurs during transmission, the 82588 aborts transmission and generates the jam sequence, as required by IEEE 802.3, and informs the CPU through interrupt and the status registers. It also starts the back-off algorithm.

To re-attempt transmission, the CPU must reinitialize the DMA controller to the start of the transmit data block and issue a RETRANSMIT command to the 82588. When the 82588 receives the retransmit command and the back-off timer has expired, it transmits again. Interrupt and the status register contents again indicate the success or failure of the (re)transmit attempt.

The main difference between transmit and retransmit commands is that retransmit does not clear the internal count for the number of collisions occurred, whereas transmit does. Moreover, retransmit takes effect only when the back-off timer has expired.

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.

<table>
<thead>
<tr>
<th>SRT</th>
<th>FRM</th>
<th>NO</th>
<th>EOF</th>
<th>RCV</th>
<th>CRC</th>
<th>ALG</th>
<th>ERR</th>
<th>ERR</th>
<th>OVER</th>
<th>RUN</th>
</tr>
</thead>
<tbody>
<tr>
<td>RECV</td>
<td>O.K.</td>
<td>.....</td>
<td>.....</td>
<td>.....</td>
<td>.....</td>
<td>.....</td>
<td>.....</td>
<td>.....</td>
<td>.....</td>
<td>.....</td>
</tr>
<tr>
<td>STATUS REG. 1</td>
<td>.....</td>
<td>.....</td>
<td>.....</td>
<td>.....</td>
<td>.....</td>
<td>.....</td>
<td>.....</td>
<td>.....</td>
<td>.....</td>
<td>.....</td>
</tr>
<tr>
<td>STATUS REG. 2</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</td>
<td>COUNT</td>
<td>.....</td>
<td>.....</td>
<td>.....</td>
<td>.....</td>
<td>.....</td>
<td>.....</td>
<td>.....</td>
<td>.....</td>
<td>.....</td>
</tr>
<tr>
<td>RECEIVE STATUS</td>
<td>.....</td>
<td>.....</td>
<td>.....</td>
<td>.....</td>
<td>.....</td>
<td>.....</td>
<td>.....</td>
<td>.....</td>
<td>.....</td>
<td>.....</td>
</tr>
</tbody>
</table>

Figure 25. Receive Operation (Single Buffer)

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.
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 X 8 = 1024 ms or approximately 1 millisec. 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.

![Figure 30. I/O Channel Diagram](image)

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).

<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>220-24F</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</td>
</tr>
<tr>
<td></td>
<td>(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</td>
</tr>
<tr>
<td></td>
<td>(Secondary)</td>
</tr>
<tr>
<td>3A0-3A9</td>
<td>Binary Synchronous Communications</td>
</tr>
<tr>
<td></td>
<td>(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</td>
</tr>
<tr>
<td></td>
<td>(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 31. I/O Address Map](image)
The signal CS\textsubscript{\textendash} decodes address 300H, it is only active when AEN is inactive meaning CPU and not DMA cycles. LDPORT\textsubscript{\textendash} 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.

### Control Port Definition

<table>
<thead>
<tr>
<th>ENDRQ1</th>
<th>ENDRQ3</th>
<th>ENINTER</th>
<th>TXEN</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>“1”</td>
<td>“1”</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Enable DMA requests.</td>
<td>Enable INTERRUPT request.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>“1” Enable the transmitter.</td>
</tr>
</tbody>
</table>

On power up all bits default to “0”.
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>

Meeting StarLAN 100 PPM Requirements

| Use Parallel Resonance Crystal |
| Recommended For Precise Frequencies |
| 82588 X-TAL Oscillator Stability ±35 PPM (0–70°C) |

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 acknowledgment 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](image-url)
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 35. 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 sig-
signal completely disables the transmitter. The other pur-
pose 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 essen-
tial; 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 en-
sure 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. Fig-
ure 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
goint 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](image-url)
FILTERING OF HIGH FREQUENCY NOISE

RECEIVER/ZERO CROSSING

VOLTAGE SQUELCH (600 mV)

TIME SQUELCH

ENVELOPE IDLE DETECT

INPUT DATA

CARRIER 3 EDGES

1.6 μs

Figure 38. Input Ports

Differential

RECEIVER INPUT

FILTERED-OUT BY SQUELCH

SQUELCH RECEIVER OUTPUT

V_{OH}

V_{OL}

ZERO CROSSING RECEIVER OUTPUT

V_{OH}

V_{OL}

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.
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 to 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 1BASE5 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 49

DCO TECHNICAL MARKETING, INTEL

OUTPUT PAIRS

DRAWN BY

BELL

DESIGNER

ADI GILBERT

FILE

HUBS.DWG

DATE

03.13.95

SHEET 5 OF 6

231422-93
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](image-url)
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:

\[ \text{CDT} = \neg (EA \land \neg EB \land \neg EC \land \neg ED \land EE \land EF) \]

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:

COLLISION DETECTION SR-FF:

\[ \text{COLLEN} = \neg (\text{CDT} \lor \text{COLLEN}) \]

\[ \text{COLLEN} = \neg (\text{RESET} \lor \text{COLLEN}) \]

\[ \text{COLLEN} = \neg (\text{CDT} \land \neg EA \land \neg EB \land \neg EC \land \neg ED \land \neg EE \land \neg EF) \]

(only EA active)

(only EB active)

(only EC active)

(only ED active)

(only EE active)

(only EF active)

(none of the inputs active)

RECEIVE DATA OUTPUT:

\[ \text{RCVDAT} = \left( (RA \land \neg EA) \land (RB \land \neg EB) \land (RC \land \neg EC) \land (RD \land \neg ED) \land (RE \land \neg EE) \land (RF \land \neg EF) \right) \]

(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 multiplexor 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.

<table>
<thead>
<tr>
<th>J</th>
<th>0</th>
<th>1</th>
<th>1</th>
<th>0</th>
<th>K</th>
<th>J</th>
<th>0</th>
<th>1</th>
<th>0</th>
<th>K</th>
</tr>
</thead>
<tbody>
<tr>
<td>2t</td>
<td>1</td>
<td>2t</td>
<td>1</td>
<td>2t</td>
<td>1</td>
<td>2t</td>
<td>t = 0.5 µs</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5 µs PERIOD</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- Collision Presence Signal (CPS) is generated by the HUB when it detects more than one input line active.
- CPS violates Manchester encoding rules—due to missing mid-cell transitions—hence is detected as a collision by the DTE (82588).

Choice of Collision Presence Signal
- It is a Manchester look-alike signal—edges are 0.5 or 1.0 µs apart.
  - Identical radiation, crosstalk and jitter characteristics
  - Eases retiming of the signal in the HUB
- It is easy to generate—1.5 TTL pack, or in a PAL

Figure 51. Collision Presence Signal
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-bit 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>±25.5</td>
</tr>
<tr>
<td>Noise</td>
<td>±25.5</td>
</tr>
<tr>
<td>Total</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: ...11110001111000000011111111111110001111100...

Input into the FIFO:

Regenerated Output:

Output: ...111110000111100000000111111110000111...

FIFO:

---

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:

PDO..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.

OUTPUTS:

LDFIFO_: Loads SHORT/LONG indications into the FIFO.

LSIN: Indicates SHORT/LONG

CNTTC: Terminal count of the reconstruction counter, indicating that reconstruction of a new bit will get started.

OR: Output of the FIFO indicating, that the FIFO is empty and that IDLE generation can get started.

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.

\[
\text{LDFIFO}_\_ = ! (PD1 \# PD2 \# PD3) \& !\text{EDD}_\_ \& \text{THRESH});
\]

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.

\[
\text{CNTPE}_\_ = ! (\text{CNTEN} \& !\text{CNTEND}) \# \text{CNTTC};
\]

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.

\[
\text{ODAT} = !\text{RESET}_\_ \# (\text{CNTPE}_\_ \& !\text{OUTDAT}) \quad (A)
\# (\text{CNTPE}_\_ \& \text{OUTDAT}) \quad (B)
\# (!\text{CNTPE}_\_ \& !\text{OR}) \quad (C)
\]

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
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 the listing 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:

```assembly
lds dx,STRING_POINTER ; load pointer to string in reg. ds:dx
mov ah,09h ; 9 = function number for string o/p
int 21h ; DOS System Call
```

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:

```plaintext
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 \"; /* */
```

1-473
```
/* Declarations for external IBM PC I/O routines */
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)

<table>
<thead>
<tr>
<th>NAME</th>
<th>DOSPROCS</th>
</tr>
</thead>
<tbody>
<tr>
<td>IGROUP</td>
<td>GROUP</td>
</tr>
<tr>
<td>CGROUP</td>
<td>GROUP</td>
</tr>
<tr>
<td>DATA</td>
<td>SEGMENT</td>
</tr>
<tr>
<td>DATA</td>
<td>ENDS</td>
</tr>
<tr>
<td>DOS</td>
<td>EQU</td>
</tr>
<tr>
<td>CODE</td>
<td>SEGMENT</td>
</tr>
<tr>
<td></td>
<td>ASSUME</td>
</tr>
</tbody>
</table>

CHAR$OUT: PROCEDURE(CHAR) EXTERNAL;
DECLARE CHAR BYTE;
END CHAR$OUT;
Outputs character to the screen.
DOS system call 2

CHAR    EQU [BP+4]   STACK

CHAROUT PROC NEAR +-------
PUBLIC CHAROUT ! CHAR ! x
PUSH BP   +-------
MOV BP,SP +-------
MOV DL,CHAR +-------
MOV AH,2  !IP hi ! x-2
INT DOS   +-------
POP BP    !BF lo ! x-3
RET 2     +-------
CHAROUT   ENDP   !BP hi ! x-4 <<<SP

KEYIN$NOECHO: PROCEDURE BYTE EXTERNAL;
END KEYIN$NOECHO;
Reads character without echoing to display

KEYIN$NOECHO PROC NEAR +-------
PUBLIC KEYIN$NOECHO
MOV AH,8  !DOS call 8
INT DOS   (DOS call 8)
RET
KEYIN$NOECHO ENDP
```

Figure 7-56. I/O Routines for IBM/PC (continued) 231422-59
7.2 Initialization and Declarations

Figure 57 shows some declarations describing what addresses the devices have and also some literals to help understand the other routines in this section.

Figure 58 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.
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
ns_588  literally 'C50Ch' /* 82588 COMMAND/STATUS */
brd_port literally 'C60h' /* DMA/INTERRUPT ENABLE PORT */
pio_mask literally 'C80h' /* 8258A MASK REGISTER */
pio_cwps literally 'C80h' /* 8258A COMMAND WORD 2 */
mae_mask literally 'CAh' /* 8257A MASK REGISTER */
dma_mode literally 'Obh' /* 8288A MODE REGISTER */
dma_fif literally 'Ohh' /* 8237A 1ST/2ND BYTE FLOP */
_dma_addr literally '02h' /* 8237A CHANNEL 1 ADDR. REG. */
_dma_bcl literally '02h' /* 8237A CHANNEL 1 BYTE COUNT */
_dma_addh literally 'O4h' /* CHANNEL 1 PAGE REGISTER */
_dma_addr literally '06h' /* 8237A CHANNEL 5 ADDR. REG. */
_dma_bcl literally '06h' /* 8237A CHANNEL 5 BYTE COUNT */
dma_addr literally '02h' /* CHANNEL 3 PAGE REGISTER */
dma_on literal 01h /* START CHANNEL 1 */
dma_on literal 03h /* START CHANNEL 3 */
dma_off literal 05h /* STOP CHANNEL 1 */
dma_off literal 07h /* STOP CHANNEL 3 */
enable_888 literally '06h' /* UNMARK INTERRUPT LEVEL 5 */
sei5_pico literally '00h' /* SPECIFIC ROI LEVEL 5 */
tx_dir literally 00h /* BASE */
rx_dir literally 00h /* ENABLE 888 INTRNR. */
wr_ptr.rd_ptrfifocont-0; /* RESET STATUS FIFO */
iaset_dma_addr - convert_2bit_addr(sia_set_buff_588(0));
cnf_dma_addr - convert_2bit_addr(config_588(0));
dap_dma_addr - convert_2bit_addr(0config_buff_588(0));
tx_dma_addr - convert_2bit_addr(0tx_buffer_588(0));
rx_dma_addr - convert_2bit_addr(0rx_buffer(1).buff());
end sys_init;

82588 initialization

tinit_588: procedure;
config_588(00) = 10; /* TO CONFIGURE ALL 10 PARAMETERS */
config_588(01) = 00;
config_588(05) = 00001000b; /* NODE 0, 8 MHz CLOCK, 1 KB/S */
config_588(03) = 00000100b; /* RECEIVE BUFFER LENGTH */
config_588(04) = 00010110b; /* NO LOOPBACK, AIDR LEN = 6, PREMABLE = 6 */
config_588(08) = 00000000b; /* DIFFERENTIAL MANCHESTER - OFF */
config_588(06) = 00b; /* ITFS - 96 TCK */
config_588(09) = 00000000b; /* SLOT TIME - 64 TCK */
config_588(05) = 00000011b; /* MAX. NO. RETRIES = 15 */
config_588(06) = 00000000b; /* MANCHESTER ENCODING */
config_588(10) = 101000100b; /* INTERNAL CTS AND COT, CSR = 0 */
config_588(11) = 64; /* KIP FRAME LENGTH = 64 BYTES = 128 BITS */
```
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)
- SET-UP
- TRANSMIT
- MULTICAST-ADDRESS (MC)
- SET-UP
- 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,6,iast_dma_addr);

    /* SET DMA CHANNEL 0 OR 1 TO TRANSFER FROM MEMORY
    TO THE 82586. iast_dma_addr VARIABLE STORES THE
    20 BIT POINTER TO THE INDIVIDUAL ADDRESS BUFFER */
    if cmd_channel then output (os_588) = 11h;
    else output(os_588) = 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_588) = 13h;
    else output(os_588) = 03h;
end config;

/*--------------------------------------------------------------------------------*/
multicast: procedure public; /* COMMAND - 03 */
    call dma_load(cmd_channel,tx_dir,14,mco_dma_addr);
    if cmd_channel then output (os_588) = 13h;
    else output(os_588) = 03h;
end multicast;

/*--------------------------------------------------------------------------------*/
transmit: procedure(buffer_len) public; /* COMMAND - 04 */
    declare buffer_len word;
    tx_buffer_588(00) = low(buffer_len);
    tx_buffer_588(01) = high(buffer_len);
    call dma_load(cmd_channel,tx_dir,1536,tx_dma_addr);
    if cmd_channel then output (os_588) = 14h;
    else output(os_588) = 04h;
end transmit;

Figure 59. General Commands
tdr: procedure public; /* COMMAND - 06 */
    if cmd_channel then output (os_588) = 16h;
    else output(os_588) = 06h;
end tdr;

/*-----------------------------------------------*/
dump_588: procedure public; /* COMMAND - 06 */
    call dma_load(cmd_channel, rx_dir, 64, &disp_dma_addr);
    if cmd_channel then output (os_588) = 16h;
    else output(os_588) = 06h;
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) = 06h;
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)=0bh;
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) = 10h;
    else output(os_588) = 06h;
end retransmit;

/*-----------------------------------------------*/
abort: procedure public; /* COMMAND - 13 */
    output(os_588)=14h;
    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-eq ptr;
ptr_20bit-shl((ptr_20bit:=wrd(l)),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; /* CHANNEL 0, 0 or 1 */
declare direction byte; /* 0=RX, 88H -> XEM; 1=TX, XEM -> 88H */
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_flg) = 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 MSB ADDRESS BYTE */
output(dma_addrb_1) = low (wrd(1)); /* 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_flg) = 0;
if direction=0 then output(dma_mode)=dma_rx_mode_3; /* SAME AS BEFORE FOR CHANNEL 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_addrb_3) = low (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.

```c
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_sp1) = ch_a_588;
output(dma_0_sp2) = 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_sp2) = wrd(0);
output(dma_0_sp3) = 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 00000001b;
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 ch 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 ch 1 */
end;
end; end 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

instr_588:procedure interrupt 13;

declare stat byte,
event byte,
(st0,st1,st2,st3) byte,
rx_st0 byte,
rx_st1 byte;

/* FOLLOWING LITERALS HAVE THE PURPOSE OF ENABLE ACTING ON EITHER CHANNEL 1 OR 3 SELECTIVELY */
declare
stop_cmd_dma literally 'if cmd_channel
then output(dma_mask)-dma_off_3;
else output(dma_mask)-dma_off_1'.
stop_rx_dma literally 'if rx_channel
then output(dma_mask)-dma_off_3;
else output(dma_mask)-dma_off_1'.
issue_rx_cmd literally 'if cmd_channel
then output(os_588)-1Ch;
else output(os_588)-00h'.
issue_tx_cmd literally 'if cmd_channel
then output(os_588)-14h;
else output(os_588)-04h'.
disable; /* DISABLE INTERRUPTS */
output(os_588)-0Fh; /* RLS 588 PTR. START 0 */
event_fifo(wr_ptr).stO-input(os_588);
/* READ 82588 STATUS */
event_fifo(wr_ptr).st1,st1-input(os_588);
/* REGISTERS. PASSING */
event_fifo(wr_ptr).st2,st2-input(os_588);
/* THEM TO THE MAIN */
event_fifo(wr_ptr).st3,st3-input(os_588);
/* PROGRAM ON THE FIFO */
wr_ptr-(wr_ptr+1) and Ofh;
/* INCREMENT FIFO COUNTERS */
fifo_cnt-(fifo_cnt+1) and Ofh;
/* GET EVENT FIELD */
stat-(st0 and 10000000b) or (st1 and 00100000b);
if (stat-00h)
then do:
/* RETRANSMIT */
call dma_load(cmd_channel,tx_dir,1636,tx_dma_addr);
issue_rx_cmd;
/* UPDATE STATISTICS */
total_tx_count=total_tx_count+1;
/*TOTAL COLL*/
bad.tx_count = bad.tx_count + 1;
end;
else do;
if in_loop /* EXECUTING TRANSMISSIONS IN LOOP */
then do;
/* RE ISSUE TRANSMIT COMMAND */
call dma_load(cmd_channel,tx_dir,1636,tx_dma_addr-
issue_tx_cmd);
total_tx_count=total_tx_count+1;
end;
end;
if (st1 and 00100000b) = 0 /* BAD TRANSMIT*/
then do:
bad_tx_count = bad_tx_count + 1;
/*_INCREMENT UNDERRUN COUNTER */
tmp-scr(tmp:st1,1);
tr_under=tr_under plus 0;
/*_INCREMENT LOST CTS COUNTER */
tmp-scr(tmp,1);
lost_cts-lost_sts plus 0;
/*_INCREMENT LOST CRS COUNTER */
tmp-scr(tmp,1);
lost_crs-lost_ots plus 0;
if (stat=00h) /* INC COLLISIONS COUNTER */
then coll_cnt[17]=coll_cnt[17]+1;
end;
end;
/*_INCREMENT DEFER COUNTER */
tmp-scr(tmp:st1,1);
end;
end;
end;
end.

Figure 62. Interrupt Routine
Figure 62. Interrupt Routine (Continued)
APPENDIX A
STARLAN SIGNALS
Figure 63. StarLAN Signals
Eye Diagram (5 Bits), DIW Cable
Manchester Encoded Signal
Transmission Distance = 0.8 Kft.

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)

LOW COST. TERMINALS ATTRACTIVE

USER INSTALLABLE INTERCONNECT

FEWER CONNECTIONS TO WIRING CLOSETS

CONNECTION OPTIONAL, NOT NEEDED FOR SMALL TOPOLOGIES

Figure 65. Multipoint Extension
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)

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)
• SCHEME IS BASED ON THE FACT THAT THE 82588 EXPECTS RX CARRIER TO BE ACTIVE AFTER 1/2 SLOT TIME

LOW = 0
HIGH = 2048

SLOT_TIME = (HIGH + LOW) / 2
CONFIGURE 82588 TRANSMIT FRAME

LOW = SLOT_TIME

Y
N

COLLISION

HIGH = SLOT_TIME

HIGH - LOW < K
K = APPROXIMATION FACTOR

Y
N

END

Figure 68. Network Delay Measurement using the 82588
Interfacing Intel 82596 LAN Coprocessors with M68000 Family Microprocessors

Some portions of this document were provided by Dr. Design of San Diego, CA
# Interfacing Intel 82596 LAN Coprocessors with M68000 Family Microprocessors

## CONTENTS

<table>
<thead>
<tr>
<th>CONTENTS</th>
<th>PAGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.0 INTRODUCTION</td>
<td>1-497</td>
</tr>
<tr>
<td>1.1 Scope</td>
<td>1-497</td>
</tr>
<tr>
<td>1.2 Fundamental Assumptions</td>
<td>1-497</td>
</tr>
<tr>
<td>2.0 GENERIC IMPLEMENTATION ISSUES</td>
<td>1-497</td>
</tr>
<tr>
<td>2.1 Block Definitions</td>
<td>1-497</td>
</tr>
<tr>
<td>2.2 Clocking</td>
<td>1-499</td>
</tr>
<tr>
<td>2.3 Reset Retiming</td>
<td>1-499</td>
</tr>
<tr>
<td>2.4 CA and PORT Generation</td>
<td>1-500</td>
</tr>
<tr>
<td>2.5 Arbitration</td>
<td>1-501</td>
</tr>
<tr>
<td>2.5.1 Refresh Requests</td>
<td>1-501</td>
</tr>
<tr>
<td>2.5.2 82596 Requests</td>
<td>1-501</td>
</tr>
<tr>
<td>2.5.3 Arbiter Implementation</td>
<td>1-502</td>
</tr>
<tr>
<td>2.6 Signal Conversion</td>
<td>1-504</td>
</tr>
<tr>
<td>2.7 Wait State and Burst Generator</td>
<td>1-505</td>
</tr>
<tr>
<td>2.7.1 General Information</td>
<td>1-505</td>
</tr>
<tr>
<td>2.7.2 Single Cycle Bus Transfers</td>
<td>1-505</td>
</tr>
<tr>
<td>2.7.3 Burst-Cycle and Multiple-Cycle Bus Transfers</td>
<td>1-505</td>
</tr>
<tr>
<td>3.0 MC68030/82596CA INTERFACE</td>
<td>1-506</td>
</tr>
<tr>
<td>3.1 Design Specifications</td>
<td>1-506</td>
</tr>
<tr>
<td>3.2 Clocking</td>
<td>1-506</td>
</tr>
<tr>
<td>3.3 Reset Retiming</td>
<td>1-506</td>
</tr>
<tr>
<td>3.4 CA and PORT Generator</td>
<td>1-506</td>
</tr>
<tr>
<td>3.5 Bus Arbiter</td>
<td>1-507</td>
</tr>
<tr>
<td>3.6 Memory Signal Conversion</td>
<td>1-507</td>
</tr>
<tr>
<td>3.7 Wait State and Burst Generator</td>
<td>1-508</td>
</tr>
<tr>
<td>3.7.1 General Information</td>
<td>1-508</td>
</tr>
<tr>
<td>3.7.2 Single Cycle Transfers</td>
<td>1-508</td>
</tr>
<tr>
<td>3.7.3 Burst Cycle Bus Transfers</td>
<td>1-508</td>
</tr>
<tr>
<td>4.0 M68000/82596DX INTERFACE</td>
<td>1-509</td>
</tr>
<tr>
<td>4.1 Design Specifications</td>
<td>1-509</td>
</tr>
<tr>
<td>4.2 Clocking</td>
<td>1-509</td>
</tr>
<tr>
<td>4.3 Reset Timing</td>
<td>1-509</td>
</tr>
<tr>
<td>4.4 CA and PORT Generator</td>
<td>1-510</td>
</tr>
<tr>
<td>4.5 Bus Arbiter</td>
<td>1-510</td>
</tr>
<tr>
<td>4.6 Memory Signal Conversion</td>
<td>1-510</td>
</tr>
<tr>
<td>4.7 Wait State Generator</td>
<td>1-511</td>
</tr>
<tr>
<td>5.0 MC68000/82596SX INTERFACE</td>
<td>1-511</td>
</tr>
<tr>
<td>5.1 Design Specifications</td>
<td>1-511</td>
</tr>
<tr>
<td>5.2 Clocking</td>
<td>1-512</td>
</tr>
<tr>
<td>5.3 Reset Retiming</td>
<td>1-512</td>
</tr>
<tr>
<td>5.4 CA and PORT Generator</td>
<td>1-512</td>
</tr>
<tr>
<td>5.5 Bus Arbiter</td>
<td>1-512</td>
</tr>
<tr>
<td>5.6 Memory Signal Conversion</td>
<td>1-512</td>
</tr>
<tr>
<td>5.7 Wait State Generator</td>
<td>1-513</td>
</tr>
<tr>
<td>APPENDIX A SCHEMATICS</td>
<td>1-514</td>
</tr>
<tr>
<td>A.1 MC68030/82596CA</td>
<td>1-515</td>
</tr>
<tr>
<td>A.2 MC68020/82596DX</td>
<td>1-516</td>
</tr>
<tr>
<td>A.3 MC68000/82596SX</td>
<td>1-517</td>
</tr>
<tr>
<td>APPENDIX B PLD EQUATIONS</td>
<td>1-518</td>
</tr>
<tr>
<td>B.1 MC68030/82596CA</td>
<td>1-519</td>
</tr>
<tr>
<td>B.2 MC68020/82596DX</td>
<td>1-530</td>
</tr>
<tr>
<td>B.3 MC68000/82596SX</td>
<td>1-539</td>
</tr>
<tr>
<td>APPENDIX C TIMING DIAGRAMS</td>
<td>1-545</td>
</tr>
<tr>
<td>C.1 MC68030/82596CA</td>
<td>1-546</td>
</tr>
<tr>
<td>C.2 MC68020/82596DX</td>
<td>1-555</td>
</tr>
<tr>
<td>C.3 MC68000/82596SX</td>
<td>1-562</td>
</tr>
<tr>
<td>APPENDIX D PARTS LIST</td>
<td>1-569</td>
</tr>
</tbody>
</table>

Throughout this document, M68000 is used as a general reference to a family of microprocessors, which includes the MC68000, MC68020, MC68030. A reference to a particular member of the family will use the MC prefix followed by the specific number. 82596 is used as a general reference to a family of LAN coprocessors—the 82596CA, 82596DX, and 82596SX. A reference to a particular member of the family will use 82596 followed by the two letter suffix.
Table of Figures and Tables

CONTENTS

FIGURES

<table>
<thead>
<tr>
<th>FIGURE</th>
<th>DESCRIPTION</th>
<th>PAGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Figure 1</td>
<td>M68000 Family and 82596 Family Interface Generic Block Diagram</td>
<td>1-498</td>
</tr>
<tr>
<td>Figure 2</td>
<td>Clock Timing Relationships</td>
<td>1-498</td>
</tr>
<tr>
<td>Figure 3</td>
<td>Reset Retiming Block Diagram</td>
<td>1-499</td>
</tr>
<tr>
<td>Figure 4</td>
<td>CA and PORT Block Diagram</td>
<td>1-500</td>
</tr>
<tr>
<td>Figure 5</td>
<td>CA and PORT State Transition Diagram</td>
<td>1-500</td>
</tr>
<tr>
<td>Figure 6</td>
<td>Arbiter State Transition Diagram</td>
<td>1-503</td>
</tr>
<tr>
<td>Figure 7</td>
<td>Arbiter Signal Timings</td>
<td>1-503</td>
</tr>
<tr>
<td>Figure 8</td>
<td>Memory Signal Conversion Block Diagram</td>
<td>1-504</td>
</tr>
<tr>
<td>Figure 9</td>
<td>Memory Cycles for M68000 Family and 82596 Family</td>
<td>1-504</td>
</tr>
<tr>
<td>Figure 10</td>
<td>Wait-State and Burst Generation Block Diagram</td>
<td>1-505</td>
</tr>
<tr>
<td>Figure 11</td>
<td>82596DX and 82596SX Reset Retiming Circuit</td>
<td>1-509</td>
</tr>
</tbody>
</table>

TABLES

<table>
<thead>
<tr>
<th>TABLE</th>
<th>DESCRIPTION</th>
<th>PAGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Table 1</td>
<td>82596-Family Parallel Bus Comparison</td>
<td>1-497</td>
</tr>
<tr>
<td>Table 2</td>
<td>Clocking Specifications</td>
<td>1-499</td>
</tr>
<tr>
<td>Table 3</td>
<td>Reset Specifications</td>
<td>1-500</td>
</tr>
<tr>
<td>Table 4</td>
<td>Arbitration Signal Input Timings</td>
<td>1-501</td>
</tr>
<tr>
<td>Table 5</td>
<td>Arbitration Signal Output Timings</td>
<td>1-502</td>
</tr>
<tr>
<td>Table 6</td>
<td>82596CA Worst-Case Reset Timing Margin</td>
<td>1-506</td>
</tr>
<tr>
<td>Table 7</td>
<td>82596DX/SX Worst-Case Reset Timing Margins</td>
<td>1-509</td>
</tr>
</tbody>
</table>
1.0 INTRODUCTION

1.1 Scope

The 82596 family of LAN coprocessors provide IEEE 802.3 MAC functions for use with 10BASE5 (Ethernet), 10BASE2 (Cheapernet), 10BASE-T (Twisted Pair Ethernet), 1BASE5 (StarLAN), and other CSMA/CD LANs with serial bit rates up to 20 Mb/s. The three members of the 82596 family differ only in the characteristics of their parallel interfaces; the FIFO and serial functions are identical. Table 1 shows the parallel bus differences.

This document describes the circuits required to interface the Intel 82596 family of LAN coprocessors with the M68000 family of microprocessors. First, general interface issues are identified and then three specific designs are provided—including the PLD equations, timing diagrams, and schematics.

1.2 Fundamental Assumptions

Each design is based on several fundamental assumptions about the memory subsystem. The circuits required to support these features are implemented in a few programmable components. If the 82596 is added to existing designs in which the required circuits are already implemented, these circuits do not have to be duplicated.

The following assumptions are made about the designs.

- The memory subsystem uses DRAM.
- Refresh request signals are asynchronous to the system clock.
- Interface logic is implemented in PLDs where possible.
- 82596 family type signals will be converted to M68000 family type signals.

2.0 GENERIC IMPLEMENTATION ISSUES

2.1 Block Definitions

Each design is broken into functional blocks. The generic block diagram is shown in Figure 1. The M68000 family and 82596 family are fixed from a functional standpoint. The designer has almost no flexibility in their connection or timing. For the purpose of these designs, the memory control block is also assumed to be fixed. It is set up only for M68000-type signals and timings. The blocks for which there is some design flexibility are grouped under the name control logic. These blocks include the following.

- Clocking. Provides the proper clock phases to the 82596 and M68000. It also provides the clock for the other blocks.
- Reset Retiming. Takes the active low RESET signal that goes to the M68000 and adjusts its timing and level to be compatible with the active high RESET for the 82596.
- CA and PORT Generation. Decodes the address lines and generates the Channel Attention (CA) and CPU Port (PORT) signals to the 82596. The system designer selects the memory addresses to be decoded to activate these signals. The amount of decode logic will vary greatly depending on the system memory map.
- Arbitration. Determines which of the three master devices has control of the local bus: the M68000, the 82596, or the refresh controller. The refresh controller has the highest priority, followed by the 82596 and then the M68000. Additional master devices are supported through simple changes to the PLD equations in this block.
- Memory Signal Conversion. Takes the 82596 control signals, such as ADS and W/R, and converts them to M68000-type control signals, such as AS, DS, and R/W.
- Wait State and Burst Generation. Generates the RDY signal to the 82596 (and BRDY for the 82596CA). It also asserts the burst request (CBREQ) to the memory controller.

<table>
<thead>
<tr>
<th>82596 Version</th>
<th>Address Bits</th>
<th>Data Bits</th>
<th>Parallel Clocking</th>
<th>Burst Access</th>
<th>Parity Pins</th>
<th>Maximum Frequency (MHz)</th>
</tr>
</thead>
<tbody>
<tr>
<td>82596CA</td>
<td>32</td>
<td>32</td>
<td>x1</td>
<td>Yes</td>
<td>Yes</td>
<td>33</td>
</tr>
<tr>
<td>82596DX</td>
<td>32</td>
<td>32</td>
<td>x2</td>
<td>No</td>
<td>No</td>
<td>33</td>
</tr>
<tr>
<td>82596SX</td>
<td>24</td>
<td>16</td>
<td>x2</td>
<td>No</td>
<td>No</td>
<td>20</td>
</tr>
</tbody>
</table>
Figure 1. General Block Diagram

Figure 2. Clock Timing Relationships
2.2 Clocking

The 82596 family uses different types of clocking. The 82596DX and 82596SX use CLK2, which is twice as fast as the internal operating frequency. The two different phases of CLK2 for every CLK1 are defined as φ1 and φ2. The rising edge of CLK1 corresponds to the rising edge of CLK2 at the beginning of φ1. The 82596CA uses CLK1, which is identical to the internal operating frequency. The 82596 clock timing relationships are shown in Figure 2.

In many cases the control logic is simplified by clocking it with CLK2, even if the CPU and 82596CA are clocked by CLK1. In some other cases it is advantageous to invert the clocking signal to the 82596, which introduces a phase shift between the devices. The clocking specifications of the 82596 and M68000 are shown in Table 2.

All the designs use 74F74 flip-flops because of their high operating frequency, low propagation delay, and wide availability. If another type of flip-flop is used the timing analysis must be modified to reflect the different specifications.

2.3 Reset Retiming

Because the 82596DX and 82596SX use CLK2, the setup and hold time specifications for the reset signal are very important. The deactivation of RESET is the only means by which the 82596DX and 82596SX determine which phases of CLK2 correspond to φ1 and φ2. Since many of the control signals are only valid at the beginning of certain phases, it is crucial that the arbitration, signal conversion, and wait state generation logic know the current phase of the clock. Failure to meet these specifications can cause improper memory accesses by the 82596. Figure 3 is the reset retiming block schematic.

![Figure 3. Reset Retiming Block](292076-3)

### Table 2. Clocking Specifications

<table>
<thead>
<tr>
<th>Component</th>
<th>Freq (MHz)</th>
<th>Clock</th>
<th>Max Rise (nanoseconds)</th>
<th>Max Fall (nanoseconds)</th>
<th>Min High</th>
<th>Min Low</th>
</tr>
</thead>
<tbody>
<tr>
<td>82596CA</td>
<td>25</td>
<td>CLK</td>
<td>3.0</td>
<td>3.0</td>
<td>14.0</td>
<td>14.0</td>
</tr>
<tr>
<td>82596CA</td>
<td>33</td>
<td>CLK</td>
<td>3.0</td>
<td>3.0</td>
<td>11.0</td>
<td>11.0</td>
</tr>
<tr>
<td>82596DX</td>
<td>25</td>
<td>CLK2</td>
<td>7.0</td>
<td>7.0</td>
<td>4.0</td>
<td>5.0</td>
</tr>
<tr>
<td>82596DX</td>
<td>33</td>
<td>CLK2</td>
<td>4.0</td>
<td>3.0</td>
<td>4.5</td>
<td>4.5</td>
</tr>
<tr>
<td>82596SX</td>
<td>16</td>
<td>CLK2</td>
<td>8.0</td>
<td>8.0</td>
<td>5.0</td>
<td>7.0</td>
</tr>
<tr>
<td>MC68030</td>
<td>25</td>
<td>CLK</td>
<td>4.0</td>
<td>4.0</td>
<td>19.0</td>
<td>19.0</td>
</tr>
<tr>
<td>MC68030</td>
<td>33</td>
<td>CLK</td>
<td>3.0</td>
<td>3.0</td>
<td>14.0</td>
<td>14.0</td>
</tr>
<tr>
<td>MC68020</td>
<td>25</td>
<td>CLK</td>
<td>4.0</td>
<td>4.0</td>
<td>19.0</td>
<td>19.0</td>
</tr>
<tr>
<td>MC68020</td>
<td>33</td>
<td>CLK</td>
<td>3.0</td>
<td>3.0</td>
<td>14.0</td>
<td>14.0</td>
</tr>
<tr>
<td>MC68000</td>
<td>16</td>
<td>CLK</td>
<td>5.0</td>
<td>5.0</td>
<td>27.0</td>
<td>27.0</td>
</tr>
</tbody>
</table>
The timings for the M68000 active low signal RESET are not compatible with the 82596 active high signal RESET. To prevent possible metastable conditions, the M68000 RESET passes through a two-stage synchronizer before going to the 82596. This will usually require two 74F74 flip-flops. Using two stages, rather than one, greatly reduces the probability of metastable conditions in the 82596. One of the stages is also used to invert the signal. Table 3 lists the relevant specifications for RESET and RESET.

### Table 3. Reset Specifications

<table>
<thead>
<tr>
<th>Component</th>
<th>Freq (MHz)</th>
<th>Setup (ns)</th>
<th>Hold (ns)</th>
</tr>
</thead>
<tbody>
<tr>
<td>82596CA</td>
<td>25</td>
<td>8.0</td>
<td>3.0</td>
</tr>
<tr>
<td>92596CA</td>
<td>33</td>
<td>10.0</td>
<td>3.0</td>
</tr>
<tr>
<td>82596DX</td>
<td>25</td>
<td>10.0</td>
<td>3.0</td>
</tr>
<tr>
<td>82596DX</td>
<td>33</td>
<td>8.0</td>
<td>3.0</td>
</tr>
<tr>
<td>82596SX</td>
<td>16</td>
<td>13.0</td>
<td>4.0</td>
</tr>
<tr>
<td>MC68030</td>
<td>25</td>
<td>ND</td>
<td>ND</td>
</tr>
<tr>
<td>MC68030</td>
<td>33</td>
<td>ND</td>
<td>ND</td>
</tr>
<tr>
<td>MC68020</td>
<td>25</td>
<td>ND</td>
<td>ND</td>
</tr>
<tr>
<td>MC68020</td>
<td>33</td>
<td>ND</td>
<td>ND</td>
</tr>
<tr>
<td>MC68000</td>
<td>16</td>
<td>ND</td>
<td>ND</td>
</tr>
</tbody>
</table>

ND = Not Defined by Motorola

#### 2.4 CA and PORT Generation

The 82596 has two inputs that do not correspond to any signals generated by the CPU: Channel Attention (CA) and CPU Port (PORT). Channel Attention is always monitored by the 82596, and the falling edge is internally latched. The 82596 responds to Channel Attention by reading the system control block command word, which is stored in memory. Several fields in this command word tell the 82596 what to do: e.g., acknowledge interrupts, change the state of the command unit, change the state of the receive unit, or load the bus throttle timers.

When the 82596 does not have the bus, it examines PORT. If it is active the value on the data bus is stored in the 82596 in a special register. The value on the 4 least significant bits (D3–D0) indicates one of sixteen functions. Only four functions are defined (functions 4 through 15 are reserved and should not be used).

1. Do an internal reset (pins D31–D4 are ignored).
2. Do a self test (the results are placed at the location specified by pins D31–D4).
3. Execute a Dump command (the results are placed at the location specified by pins D31–D4).
4. Move the system configuration pointer to the location specified on pins D31–D4.

PORT has more stringent requirements for setup and hold times than CA does. PORT must meet specific setup and hold times with respect to the clock and must also be active for at least two consecutive clocks. CA is only required to be active for two clocks without meeting specific setup or hold times; however, CA only has to be active for one clock if the setup and hold times are met.

Because the M68000 family does not support a separate I/O address space, all I/O functions must be memory-mapped. The addresses for CA and PORT can be selected by the designer. In the design examples, Section 3 through 5, part of one PLD is used to generate both signals. The number of input pins on the PLD will determine the address range limitations. If the PLD has fewer input pins than the number of address lines to be decoded, one of the PLD inputs should be connected to the output of a secondary decoder. This secondary decoder must meet a worst-case propagation delay, which is listed in the comment fields of the PLD equations for each design.

Implementing CA and PORT will usually require four macro-cells, which is about one-half of a standard PLD. Two are used for the actual output signals and two are used as a state machine to control the timing of the output signals. Figure 4 is the CA and PORT generator block diagram and Figure 5 is the state transition diagram.

![Figure 4. CA and PORT Block Diagram](292076-4)

![Figure 5. CA and PORT State Transition Diagram](292076-5)
2.5 Arbitration

All the design examples assume that the memory will use DRAMs. This means there will be at least three master devices attempting to gain access to memory: the M68000 CPU, the 82596 LAN coprocessor, and the refresh controller. The requests from the refresh controller must be given highest priority to avoid corrupting data in the DRAMs. The 82596 is given the second highest priority, so it is not forced to wait and eventually overrun or underrun. The M68000 has the lowest priority because of its internal Bus Request/Bus Grant mechanism. Because some of the M68000 family CPUs have an internal cache or instruction pipeline, they can fetch code or data internally while the 82596 or the refresh controller are using the local bus.

The M68000 family uses a three-signal arbitration scheme. A master device makes a bus request by asserting BR and waiting for the M68000 to assert BG. The master device then drives BGACK while it is using the bus. When the master device no longer needs the bus it brings BR inactive, then the M68000 drives BG inactive. Finally the master device drives BGACK inactive.

2.5.1 REFRESH REQUESTS

Refresh requests are assumed to be asynchronous with the arbitration clock; therefore, the refresh signal must be synchronized—typically with a 74F74 flip-flop. At the completion of the refresh cycle the local bus will be released to the requestor having highest priority. The flip-flop is not needed if the refresh request timing is synchronous with the arbitration clock and meets the setup and hold times of the PLD.

If a refresh request arrives while the M68000 is the active bus master, the Bus Request signal (BR) to the M68000 will be asserted. When the M68000 forces BG active the arbitration logic brings BGACK active and the refresh cycle begins. When the refresh has been completed BR goes inactive. If the 82596 is the active bus master when the refresh request arrives, the refresh cycle will not start until the 82596 has completed its transfers. BR to the M68000 will remain active until the refresh cycle has completed; BR will not deassert when the 82596 completes its transfers. If another 82596 request arrives during the refresh cycle, BR will remain active until both the refresh controller and the 82596 complete their transfers.

The designer is responsible for ensuring that enough refresh requests are made to avoid corrupting data in the DRAM. These designs assume that a refresh cycle signal goes into the memory controller and indicates that a refresh cycle is in progress. If a transparent technique is used for refreshing the DRAM, or if SRAM is used, then the arbiter can be greatly simplified.

There are several transparent DRAM refresh techniques. The most common method hides the refresh cycle as extra wait states in the normal CPU or 82596 accesses. This technique eliminates the arbitration overhead of the BR/BG (HOLD/HLDA) protocol and simplifies the arbiter logic. The main disadvantage is that the wait state generator becomes more complex.

2.5.2 82596 REQUESTS

The 82596 acquires and holds the system bus via the HOLD/HLDA handshake. It requests the bus by activating HOLD. When the arbiter gives the local bus to the 82596 it asserts the HLDA signal, which is the inverted LANCYC signal from the arbiter. Overrun conditions can occur in some external devices if the 82596 holds the bus too long. The 82596's bus throttle timers can be used to regulate bus use; the timer can be activated two ways.

- Externally. A high state on the BREQ pin starts the timer.
- Internally. A high state on the HLDA pin starts the timer.

Instead of using bus arbitration schemes, the 82596CA can be forced off the bus by activating the backoff pin (BOFF). This provides higher performance and faster refresh cycles. (The 82596DX and 82596SX do not have this backoff feature.)

Because the 82596 HOLD and HLDA signals are active high and the M68000 BR and BG are active low, the arbiter must invert the logic. In addition, the timings are not compatible. The arbitration signal timings are shown in Tables 4 and 5.

Table 4. Arbitration Signal Input Timings

<table>
<thead>
<tr>
<th>Component</th>
<th>Frequency (MHz)</th>
<th>Signal</th>
<th>Output Valid Delay (ns)</th>
</tr>
</thead>
<tbody>
<tr>
<td>82596CA</td>
<td>25</td>
<td>HOLD</td>
<td>3 to 22</td>
</tr>
<tr>
<td>82596CA</td>
<td>33</td>
<td>HOLD</td>
<td>3 to 19</td>
</tr>
<tr>
<td>82596DX</td>
<td>25</td>
<td>HOLD</td>
<td>4 to 22</td>
</tr>
<tr>
<td>82596DX</td>
<td>33</td>
<td>HOLD</td>
<td>3 to 19</td>
</tr>
<tr>
<td>82596SX</td>
<td>16</td>
<td>HOLD</td>
<td>4 to 32</td>
</tr>
<tr>
<td>MC68030</td>
<td>25</td>
<td>BG</td>
<td>0 to 20</td>
</tr>
<tr>
<td>MC68030</td>
<td>33</td>
<td>BG</td>
<td>0 to 20</td>
</tr>
<tr>
<td>MC68020</td>
<td>25</td>
<td>BG</td>
<td>0 to 20</td>
</tr>
<tr>
<td>MC68020</td>
<td>33</td>
<td>BG</td>
<td>0 to 20</td>
</tr>
<tr>
<td>MC68000</td>
<td>16</td>
<td>BG</td>
<td>0 to 40</td>
</tr>
</tbody>
</table>

1-501
2.5.3 ARBITER IMPLEMENTATION

Local bus arbitration is mostly implemented in a synchronous PLD that uses the inverted CPU clock (CLK\textsubscript{I}) as the arbitration clock. The arbiter has fixed priorities and responds to bus requests from the 82596 and the refresh controller by requesting the local bus from the M68000. The arbiter asserts the Bus Request (BR) and Bus Grant Acknowledge (BGACK) signals to the M68000, and enforces the bus arbitration protocol.

The arbiter does not immediately give the bus to the requestor. The arbiter is usually required to provide an adequate DRAM precharge time and will not release the bus until the precharge time has expired. The arbiter can be greatly simplified if other logic is used to control the precharge time. Figure 6 is the arbiter state transition diagram and the signal timings are shown in Figure 7.

<table>
<thead>
<tr>
<th>Component</th>
<th>Frequency (MHz)</th>
<th>Signal</th>
<th>Minimum Setup (ns)</th>
<th>Minimum Hold (ns)</th>
</tr>
</thead>
<tbody>
<tr>
<td>82596CA</td>
<td>25</td>
<td>HLDA</td>
<td>10</td>
<td>3</td>
</tr>
<tr>
<td>82596CA</td>
<td>33</td>
<td>HLDA</td>
<td>8</td>
<td>3</td>
</tr>
<tr>
<td>82596DX</td>
<td>25</td>
<td>HLDA</td>
<td>10</td>
<td>3</td>
</tr>
<tr>
<td>82596DX</td>
<td>33</td>
<td>HLDA</td>
<td>8</td>
<td>3</td>
</tr>
<tr>
<td>82596SX</td>
<td>16</td>
<td>HLDA</td>
<td>11</td>
<td>8</td>
</tr>
<tr>
<td>MC68030</td>
<td>25</td>
<td>BR</td>
<td>NA</td>
<td>NA</td>
</tr>
<tr>
<td>MC68030</td>
<td>25</td>
<td>BGACK</td>
<td>NA</td>
<td>NA</td>
</tr>
<tr>
<td>MC68030</td>
<td>33</td>
<td>BR</td>
<td>NA</td>
<td>NA</td>
</tr>
<tr>
<td>MC68030</td>
<td>33</td>
<td>BGACK</td>
<td>NA</td>
<td>NA</td>
</tr>
<tr>
<td>MC68020</td>
<td>25</td>
<td>BR</td>
<td>NA</td>
<td>NA</td>
</tr>
<tr>
<td>MC68020</td>
<td>25</td>
<td>BGACK</td>
<td>NA</td>
<td>NA</td>
</tr>
<tr>
<td>MC68020</td>
<td>33</td>
<td>BR</td>
<td>NA</td>
<td>NA</td>
</tr>
<tr>
<td>MC68020</td>
<td>33</td>
<td>BGACK</td>
<td>NA</td>
<td>NA</td>
</tr>
<tr>
<td>MC68000</td>
<td>16</td>
<td>BR</td>
<td>NA</td>
<td>NA</td>
</tr>
<tr>
<td>MC68000</td>
<td>16</td>
<td>BGACK</td>
<td>NA</td>
<td>NA</td>
</tr>
</tbody>
</table>
Figure 6. Arbiter State Transition Diagram

Figure 7. Arbiter Signal Timings
2.6 Signal Conversion

The memory signal conversion block modifies the 82596 bus signals to simulate M68000 signals. The new bus control signals are connected directly to the M68000's control signals and are tri-stated when the 82596 is not the bus master. This block will vary depending on which M68000 and 82596 combination is used. This block can be greatly simplified if the memory controller is capable of using both M68000 and 82596 signals and timings. The memory signal conversion block diagram is shown in Figure 8.

A single PLD generates the signals Address Strobe (AS), Data Strobe (DS), Read/Write (R/W), and Data Bus Enable (DBEN) from the 82596's signals ADS and W/R. In 32-bit designs this PLD also generates SIZ0, SIZ1, A0, and A1 from the 82596's BE0–BE3. In 16-bit designs it generates UDS and LDS from the 82596SX's A1, BHE, and BLE signals. The External Cycle Start (ECS) and Operating Cycle Start (OCS) signals are emulated with a tri-state buffer (e.g., a 74F244) enabled by LANCYC. The input that corresponds to the ECS and OCS signals is ADS from the 82596. Figure 9 shows the different types of cycles for the M68000 and 82596.

Figure 8. Memory Signal Conversion Block Diagram

Figure 9. M68000 and 82596 Cycles
2.7 Wait State and Burst Generator

2.7.1 GENERAL INFORMATION

The 82596 and M68000 combinations can use three types of bus transfers (the 82596DX and 82596SX support only a basic single-cycle transfer). More than one single transfer can occur without interruption. Each transfer requires at least two clocks and begins with ADS going active during the first clock cycle and then RDY going active in the last clock. Wait states are inserted by keeping RDY inactive. Figure 10 shows the wait state and burst generator block diagram.

![Diagram](image)

Figure 10. Wait-State and Burst Generator Block Diagram

The 82596CA supports all three types of transfers: single cycle, multiple cycle, and burst. The multiple cycle is simply several uninterrupted single-cycle transfers, with each bus cycle beginning with ADS going active during the first clock and then RDY going active in the last clock. Burst cycles can contain as many as four consecutive data transfers to four consecutive locations; however, ADS is only generated before the first data transfer. The maximum amount of data moved during a burst is 16 bytes (four 4-byte transfers). The wait state and burst generation block inserts the appropriate number of wait states during the bus cycle by driving RDY and BRDY active at the appropriate time.

The M68000 family supports a 3-clock asynchronous cycle. The MC68030 also supports a 2-clock synchronous cycle that is similar to the 82596CA burst access.

2.7.2 SINGLE-CYCLE BUS TRANSFERS

The 82596 begins a cycle after the rising edge of CLK2 (CLK for the 82596CA) by asserting ADS and driving W/R and the address lines (A31–A2) valid. The conversion PLD synchronizes ADS to the clock and generates an address strobe (NEWAS). NEWAS is asserted during the same phase of the clock that a M68000 would assert AS. NEWAS is also asserted during the second clock of the 82596 transfer.

The wait state generator delays the RDY signal to the 82596 from going active. This provides time to meet the data setup and hold specifications. The 82596 requires that data be set up a few nanoseconds before the rising edge of its clock. The M68000 also requires a setup time; however, it is usually only one nanosecond. The system designer will need to make provision for, at the least, the 82596 data setup time plus an additional 2 ns for clock skew. If this cannot be met another wait state will be needed for all 82596 memory read cycles. This can be provided by modifying the PLD equations to delay the assertion of RDY by one or more clocks at the end of a read cycle. The 82596CA asserts Burst Last (BLAST) during the second clock of the first cycle, which indicates that the transfer is complete after a single cycle (the 82596DX and 82596SX do not use BLAST).

2.7.3 BURST-CYCLE AND MULTIPLE-CYCLE BUS TRANSFERS

The 82596CA tries to burst cycles for any bus request that requires more than a single data cycle to consecutive addresses. The starting address must begin on an 8-byte boundary (xxxxxxx0h or xxxxxxx8h). The fastest burst cycle for this design assumes that 80 ns interleaved DRAMs are being used, which are fast enough to allow new data to be strobed into the 82596CA on each clock. The burst cycle requires 4 clocks for the first data strobe; however, subsequent data strobes are returned with each clock.

Burst cycles begin with the 82596CA driving a valid address and asserting ADS in the same manner as non-burst cycles. The 82596CA indicates that it is willing to enter a burst cycle by holding BLAST inactive during the second clock of the cycle. The ready logic then generates a Cache Burst Request (NEWCBREQ) signal to the memory controller. If Cache Burst Acknowledge (CBACK) is returned active it indicates that the memory can operate in burst mode. Then the ready logic waits for the Synchronous Termination (STERM) bus handshake signal, which indicates that the correct number of wait states has occurred and data is valid. When STERM is received the ready logic activates BRDY to the 82596CA, which indicates its willingness to allow a burst cycle. The 82596CA drives BLAST inactive for all but the last cycle in a burst. BLAST is driven active in the last cycle of the transfer to indicate that when either BRDY or RDY is next returned the transfer is complete. RDY is always returned in response to BLAST going active.
If the memory controller cannot perform a burst cycle \( \text{CBACK} \) will not go active and the ready logic will return \( \text{RDY} \) to the 82596CA, which indicates a non-burst multiple-cycle transfer will take place. Unlike the burst cycle, \( \text{ADS} \) will go active at the beginning of the second and all subsequent transfers in the multicycle transfer and \( \text{RDY} \) is used to end the cycle rather than \( \text{BRDY} \).

The two data acknowledge signals for the MC68030 (\( \text{DSACK}0 \) and \( \text{DSACK}1 \)) can be combined because the 82596CA only needs one \( \text{RDY} \) signal. Both \( \text{DSACK} \) signals connect to the inputs of an 74F08 AND gate.

### 3.0 MC68030/82596CA INTERFACE

#### 3.1 Design Specifications

This interface example is based on the following assumptions.

- MC68030 CPU.
- 82596CA LAN coprocessor.
- 32-bit DRAM memory with burst capability.
- DRAM refresh using CAS-before-RAS technique.
- 33 and 25 MHz operating frequencies.
- Interface logic implemented in PLDs where possible.
- 82596CA signals converted to MC68030 signal types.
- Refresh request signal asynchronous to 33 MHz clock.
- Burst accesses attempted whenever possible.

**NOTE:**

Many of the circuit elements (e.g., PLDs and flip-flops) in this design probably already exist in designs presently using the MC68030. The extra elements are provided only for completeness. The final design will probably require fewer circuit elements.

### 3.2 Clocking

This design uses a clock operating at 66 MHz. It is divided by a 74F74 flip-flop to generate two 33 MHz clocks from the Q and \( \bar{Q} \) outputs: \( \text{CLK}1 \) and \( \text{CLK} \). The MC68030 uses \( \text{CLK}1 \), but the 82596CA and arbitration logic use \( \text{CLK} \). The clock-to-output-valid delay of the 74F74 is 3.8 to 7.8 ns.

### 3.3 Reset Retiming

The 82596CA reset retiming block is the same as that shown in Figure 2. The synchronizing flip-flops are clocked by \( \text{CLK}1 \). There are two 82596CA specifications for \( \text{RESET} \) that must be met: the setup time (T23) and the hold time (T24). The worst-case margin is shown in Table 6 (all times are in nanoseconds).

### 3.4 CA and PORT Generator

The CA and PORT generation block is the same as that shown in Figure 4 and is based on a 20R4 PLD (10 ns delay at 33 MHz, 15 ns delay at 25 MHz). At either speed it is clocked by \( \text{CLK}1 \). \( \text{AS} \), the address lines, and \( \text{HIADDR} \) are examined at the rising edge of \( \text{CLK}1 \). The worst-case margin to this rising edge limits the maximum propagation delay of the secondary decoder. Each margin is calculated separately.

Because \( \text{AS} \) is generated later than the address, it is checked first. The setup time margin to the PLD's flip-flop is calculated as follows (all times are in nanoseconds).

\[
\begin{align*}
\text{CLK}1 \text{ cycle} &= \text{max MC68030 AS valid delay} \\
&\quad - \text{max \( \text{CLK}1 \) to \( \text{CLK}1 \) skew} \\
&\quad - \text{min PLD setup} \\
\text{At 33 MHz} &= 30 - 15 - 2 - 10 = 3 \text{ ns} \\
&\quad \text{(with 10 ns PLD)} \\
\text{At 25 MHz} &= 40 - 18 - 2 - 15 = 5 \text{ ns} \\
&\quad \text{(with 15 ns PLD)}
\end{align*}
\]

<table>
<thead>
<tr>
<th>Frequency (MHz)</th>
<th>Clock-to-Output Delay</th>
<th>Minimum Setup</th>
<th>Minimum Hold</th>
<th>Margin Setup</th>
<th>Margin Hold</th>
</tr>
</thead>
<tbody>
<tr>
<td>33</td>
<td>3.8 7.8</td>
<td>8.0</td>
<td>3.0</td>
<td>14.2</td>
<td>0.8</td>
</tr>
<tr>
<td>25</td>
<td>3.8 7.8</td>
<td>10.0</td>
<td>3.0</td>
<td>22.2</td>
<td>0.8</td>
</tr>
</tbody>
</table>
The address has an additional margin because it is generated almost one-half clock earlier. This additional margin is calculated as follows.

\[
\text{CLK}_I \text{ cycle} - \text{CLK}_I \text{ high time} \\quad - \text{max MC68030 address valid delay} \\
- \text{max } \text{CLK}_I \text{ to } \text{CLK}_I \text{ skew} \quad (3.2) \\
- \text{min PLD setup}
\]

At 33 MHz = 30 + 15 - 21 - 2 - 10 = 12 ns (with 10 ns PLD)

At 25 MHz = 40 + 20 - 25 - 2 - 15 = 18 ns (with 15 ns PLD)

Next, the worst-case setup and hold times to the 82596CA are calculated for CA and PORT, which have identical timings. They go active based on the rising edge of \( \text{CLK}_I \). The setup margins are calculated as follows.

\[
\text{CLK}_I \text{ cycle} - \text{max PLD output valid delay (3.3)} \\
- \text{min 82596CA input setup}
\]

At 33 MHz = 30 - 7 - 7 = 16 ns (with 10 ns PLD)

At 25 MHz = 40 - 12 - 7 = 21 ns (with 15 ns PLD)

The hold margins are calculated as follows.

\[
\text{min PLD output valid delay} - \text{min 82596CA input hold} \quad (3.4) \\
= 4 - 3 = 1 \text{ ns (at 33 and 25 MHz)}
\]

### 3.5 Bus Arbiter

The bus arbiter is implemented with a 20R8 PLD (10 ns delay at 33 MHz, 15 ns delay at 25 MHz). At either speed, it is clocked by \( \text{CLK}_I \). The worst-case flip-flop setup margins to this rising edge is calculated as follows.

\[
\text{CLK}_I \text{ cycle} - \text{max 82596CA HOLD output valid} \\
- \text{min PLD input setup} \quad (3.5)
\]

At 33 MHz = 30 - 19 - 10 = 1 ns (with 10 ns PLD)

At 25 MHz = 40 - 22 - 15 = 3 ns (with 15 ns PLD)

The signal \( \overline{BG} \) does not need to meet any setup or hold times because it is internally synchronized by the MC68030. The PLD flip-flop setup time for \( BG \) is checked next. Because \( BG \) can go active 0 ns after the falling edge of \( \text{CLK}_I \), and because there can be up to 2 ns of skew between \( \text{CLK}_I \) and \( \text{CLK}_I \), it is not completely safe to directly use \( BG \) in the arbiter PLD. Instead it is run through one of the flip-flops in the PLD to fully synchronize the signal. In the worst case, \( BG \) can go active about the same time as \( \text{CLK}_I \) goes high. Because the PLD will not be clocked until the next rising edge of \( \text{CLK}_I \), there will be at least one full clock cycle minus the PLD feedback delay for the output to reach a valid state.

The unused macro-cell in the 20R8 can be used to invert \( \text{LANCYC} \) to create \( \text{HLDA} \) to the 82596CA. The outputs of the arbiter, \( \text{HLDA} \) and \( \text{REFCYC} \), are internally synchronized at their destination, so no output timing analysis is required.

### 3.6 Memory Signal Conversion

The memory signal conversion block is implemented as shown in Section 2.6. A 20R4 PLD (10 ns delay at 33 MHz, 15 ns delay at 25 MHz) is used to convert the 82596CA-type control signals to MC68030-type control signals. When the 82596CA does not have the bus all the outputs go to a high-impedance state.

The signals \( \text{SIZI}, \text{SIZO}, \text{AI}, \text{and AO} \) are generated by using simple combinatorial decodes of \( \text{BE}_3, \text{BE}_2, \text{BE}_1, \) and \( \text{BE}_0 \). The delay will be identical to the PLD propagation delay. The signals \( \text{NEWAS}, \text{NEWDS}, \text{NEWDBEN}, \) and \( \text{NEWR/W} \) are generated by using the PLD's registered outputs, which are clocked by \( \text{CLK}_I \). Their states are determined by the state of the 82596CA's signals \( \text{ADS}, \text{W/R}, \) and \( \text{RDY} \).

An 82596CA read or write cycle starts with \( \text{ADS} \) going active based on the rising edge of \( \text{CLK}_I \). \( \text{NEWAS} \) and \( \text{NEWDBEN} \) will go active on the next rising edge of \( \text{CLK}_I \). If the cycle is a read cycle \( \text{NEWDS} \) will also go active. If it is a write cycle \( \text{NEWDS} \) will go active one clock later. In general, the signals go inactive based on \( \text{RDY} \) going active. To meet the data hold times \( \text{NEWDBEN} \) stays active one extra clock during a write cycle. \( \text{NEWR/W} \) is simply the inverted and registered \( \text{W/R} \).

The timing of the PLD is checked next. The 82596CA control signals must be valid in time to meet the setup requirements of the PLD's flip-flops. The margin is calculated as follows.

\[
\text{CLK}_I \text{ cycle} - \text{max 82596CA output delay} \\
- \text{PLD input setup} \quad (3.6)
\]

At 33 MHz = 30 - 19 - 10 = 1 ns (with 10 ns PLD)

At 25 MHz = 40 - 22 - 15 = 3 ns (with 15 ns PLD)
The 82596CA supports three types of bus transfers: single cycle, multiple cycle, and burst. Each bus cycle is at least two clocks long and begins with ADS going active during the first clock and RDY active in the last clock. A bus cycle contains one or more data transfers, each of which can be up to 32 bits. Burst cycles can contain as many as four data transfers, each of which can be up to 32 bits. Burst cycles can contain as many as four data transfers, each of which can be up to 32 bits. Burst cycles begin with the 82596CA driving a valid address and asserting ADS in the same manner as non-burst cycles. The 82596CA indicates that it is willing to enter a burst cycle by holding BLAST inactive in the second clock of the cycle. The ready logic then generates a cache burst acknowledge signal (NEWCBACK) to the memory controller. If the cache burst acknowledge signal (CBACK) is returned active it indicates that the memory can operate in burst mode. The ready logic then waits for the synchronous termination (STERM) bus handshake signal, which indicates that the correct number of wait states has occurred and data is valid. The ready logic then activates BRDY to the 82596CA to indicate its willingness to permit a burst cycle. The 82596CA drives BLAST inactive for all but the last cycle in a burst. BLAST is driven active in the last cycle of the transfer to indicate that when RDY or BRDY is next returned the transfer is complete. RDY is always returned in response to BLAST going active.

Burst cycles begin with the 82596CA driving a valid address and asserting ADS in the same manner as non-burst cycles. The 82596CA indicates that it is willing to enter a burst cycle by holding BLAST inactive in the second clock of the cycle. The ready logic then generates a cache burst request (NEWCBREQ) signal to the memory controller. If the cache burst acknowledge signal (CBACK) is returned active it indicates that the memory can operate in burst mode. The ready logic then waits for the synchronous termination (STERM) bus handshake signal, which indicates that the correct number of wait states has occurred and data is valid. The ready logic then activates BRDY to the 82596CA to indicate its willingness to permit a burst cycle. The 82596CA drives BLAST inactive for all but the last cycle in a burst. BLAST is driven active in the last cycle of the transfer to indicate that when RDY or BRDY is next returned the transfer is complete. RDY is always returned in response to BLAST going active.

If the memory controller cannot perform a burst cycle CBACK will not go active and the ready logic will return RDY to the 82596CA to indicate a nonburst multiple-cycle transfer will take place. This bus transfer is simply a sequence of two or more single cycle transfers. Unlike the burst cycles, ADS goes active during the first clock of the second through fourth data transfers. The timing margins for these cycles are identical to those for nonburst single cycle transfers.

Because the 82596CA requires only one RDY signal, the two data acknowledge signals for the MC68030 (DSACK0 and DSACK1) can be combined. Both DSACK signals connect to the inputs of an 74F08 AND gate.
4.0 MC68020/82596DX INTERFACE

4.1 Design Specifications

This interface example is based on the following assumptions.

- MC68020 CPU.
- 82596DX LAN coprocessor.
- 32-bit DRAM memory without burst capability.
- DRAM refresh using CAS-before-RAS technique.
- 33 MHz operating frequency.
- Interface logic implemented in PLDs where possible.
- 82596DX signals converted to MC68020 signal types.
- Refresh request signal asynchronous to 33 MHz clock.

NOTE:
Many of the circuit elements (e.g., PLDs and flip-flops) in this design probably already exist in designs presently using the MC68020. The extra elements are provided only for completeness. The final design will probably require fewer circuit elements.

4.2 Clocking

This design uses a clock operating at 66 MHz. The 66 MHz clock, CLK2, is directly by the 82596DX. It is divided by a 74F74 flip-flop to generate two 33 MHz clocks from the Q and Q outputs: CLK1 and CLK1. The MC68020 uses CLK1, but the arbitration logic uses CLK1. The clock-to-output-valid delay of the 74F74 is 3.8 to 7.8 ns. The rising edge of CLK1 corresponds to the rising edge of CLK2 at the beginning of φ1.

4.3 Reset Retiming

The 82596DX reset retiming block is shown in Figure 11. The synchronizing flip-flops are clocked by CLK2. There are two 82596DX specifications for RESET that must be met: the setup time (T23) and the hold time (T24). The worst-case margin is shown in Table 7.

![Figure 11. Reset Retiming Block](image)

Table 7. 82596DX/SX Worst-Case Reset Timing Margin

<table>
<thead>
<tr>
<th>82596 Frequency (MHz)</th>
<th>Clock-to-Output Delay</th>
<th>Minimum</th>
<th>Maximum</th>
<th>Setup</th>
<th>Hold</th>
<th>Margin Setup</th>
<th>Margin Hold</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>(nanoseconds)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>33</td>
<td>3.8 7.8</td>
<td>8.0</td>
<td>3.0</td>
<td>14.2</td>
<td>0.8</td>
<td></td>
<td></td>
</tr>
<tr>
<td>25</td>
<td>3.8 7.8</td>
<td>10.0</td>
<td>3.0</td>
<td>22.2</td>
<td>0.8</td>
<td></td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>3.8 7.8</td>
<td>13.0</td>
<td>4.0</td>
<td>45.2</td>
<td>-0.2</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
4.4 CA and PORT Generator

The CA and PORT generation block is the same as that shown in Figure 4 and is based on a 20R4 PLD (10 ns delay at 33 MHz, 15 ns delay at 25 MHz) clocked by CLK1. AS, the address lines, and HIADDR are examined at the rising edge of CLK1. The worst-case margin to this rising edge limits the maximum propagation delay of the secondary decoder. Each margin is calculated separately.

Because AS is generated later than the address, it is checked first. The setup time margin to the PLD's flip-flop is calculated as follows (all times are in nanoseconds).

\[
\text{CLK1 cycle} - \max \text{MC68020 AS valid delay} - \max \text{CLK1 to CLK1 skew} - \min \text{PLD setup} \tag{4.1}
\]

At 33 MHz = 30 - 15 - 2 - 10 = 3 ns
(with 10 ns PLD)

At 25 MHz = 40 - 18 - 2 - 15 = 5 ns
(with 15 ns PLD)

The address has an additional margin because it is generated almost one-half clock earlier. This additional margin is calculated as follows.

\[
\text{CLK1 cycle} + \min \text{CLK1 low time} - \max \text{MC68020 address valid delay} - \max \text{CLK1 to CLK1 skew} - \min \text{PLD setup} \tag{4.2}
\]

At 33 MHz = 30 + 15 - 21 - 2 - 10 = 12 ns
(with 10 ns PLD)

At 25 MHz = 40 + 20 - 25 - 2 - 15 = 18 ns
(with 15 ns PLD)

Next, the worst-case setup and hold times to the 82596DX are calculated for CA and PORT, which have identical timings. They go active based on the rising edge of CLK1. The setup margins are calculated as follows.

\[
\text{CLK1 cycle} - \max \text{PLD output valid delay} - \max 82596DX input setup - \max \text{CLK2 to CLK1 clock skew} \tag{4.3}
\]

At 33 MHz = 30 - 7 - 7 - 7.8 = 8.2 ns
(with 10 ns PLD)

At 25 MHz = 40 - 12 - 7 - 7.8 = 13.2 ns
(with 15 ns PLD)

The hold margins are calculated as follows.

\[
\text{PLD output valid delay} - \min 82596DX input hold + \min \text{CLK2 to CLK1 skew} \tag{4.4}
\]

\[
= 4 - 3 + 3.8 = 4.8 \text{ ns (at 33 and 25 MHz)}
\]

4.5 Bus Arbiter

The bus arbiter is implemented with a 20R8 PLD (10 ns delay at 33 MHz, 15 ns delay at 25 MHz) clocked by CLK1. The worst-case margins to this rising edge is calculated as follows.

\[
\text{CLK1 cycle} + \min \text{CLK2 to CLK1 skew} - \max 82596DX HOLD output valid - \min \text{PLD input setup} \tag{4.5}
\]

At 33 MHz = 30 + 3.8 - 19 - 10 = 4.8 ns
(with 10 ns PLD)

At 25 MHz = 40 + 3.8 - 22 - 15 = 7.8 ns
(with 15 ns PLD)

The signal BR does not need to meet any setup or hold times because it is internally synchronized by the MC68020. The PLD flip-flop setup time for BR is checked next. Because BG can go active 0 ns after the falling edge of CLK1, and because there can be up to 2 ns of skew between CLK1 and CLK1, it is not completely safe to directly use BG in the arbiter PLD. Instead it is run through one of the flip-flops in the PLD to fully synchronize the signal. In the worst case, BG can go active about the same time as CLK1 goes high. Because the arbiter's flip-flop will not be clocked until the next rising edge of CLK1, there will be a full clock cycle minus the PLD feedback delay for the output to reach a valid state.

The outputs of the arbiter, LANCYC and REFCYC, are internally synchronized at their destination, so no output timing analysis is required. An external inverter is required for LANCYC to create HLDA to the 82596DX. If the PLD has an internal inverter then this will not be required.

4.6 Memory Signal Conversion

The memory signal conversion block is implemented as shown in Section 2.6. A 20R4 PLD (10 ns delay at 33 MHz, 15 ns delay at 25 MHz) is used to convert the 82596DX-type control signals to MC68020-type control signals. When the 82596DX does not have the bus all the outputs go to a high-impedance state.
The signals SIZ1, SIZ0, A1, and AO are generated by using simple combinational decodes of BE3, BE2, BE1, and BE0. The delay will be identical to the PLD propagation delay. The signals NEWAS, NEWDS, NEWDBEN, and NEWR/W are generated by using the PLD’s registered outputs, which are clocked by CLK1. Their states are determined by the state of the 82596DX’s signals ADS, W/R, and RDY.

An 82596DX read or write cycle starts with ADS going active based on the rising edge of CLK1. NEWAS and NEWDBEN will go active on the next rising edge of CLK1. If the cycle is a read cycle NEWDS will also go active. If it is a write cycle NEWDS will go active one clock later. In general, the signals go active based on RDY going active. To meet the data hold times NEWDBEN stays active one extra clock during a write cycle. NEWR/W is simply the inverted and registered Wir.

The timing of the PLD is checked next. The 82596DX control signals must be valid in time to meet the setup requirements of the PLD’s flip-flops. The margin is calculated as follows.

\[
\text{CLK1 cycle} + \min \text{CLK2 to CLK1 skew} - \max 82596DX \text{ output delay} - \text{PLD input setup}
\]

At 33 MHz = 30 + 3.8 - 19 - 10 = 4.8 ns (with 10 ns PLD)

At 25 MHz = 40 + 3.8 - 22 - 15 = 7.8 ns (with 15 ns PLD)

NEWAS goes active based on the rising edge of CLK1, which is the same as the falling edge of CLK1. The PLD clock to output valid delay is 2 to 7 ns maximum. The skew between the clocks will be -2 to +2 ns. This translates to a 0- to 5 ns delay, which is within the MC68020 specifications of 2 to 10 ns.

ECS and OCS are generated by taking ADS and running it through a tri-state buffer (74F244) that is enabled by HLDA. When the 82596DX has the bus ECS and OCS will go active about 8 ns after ADS.

### 4.7 Wait State Generator

Each 82596DX bus cycle is at least two clocks long and begins with ADS going active during the first clock and RDY active in the last clock. The wait state block inserts the proper number of wait states during the bus cycle by delaying the RDY signal to the 82596DX. The fastest single transfer in this design requires three clocks for the 82596DX. This provides enough time to meet the data setup and hold specifications. The 82596DX requires that data be valid at least 5 ns before the rising edge of its clock. The MC68020 requires only a 1 ns setup to its clock. If this cannot be met, another wait state will be needed for all 82596DX memory read cycles. This can be done by modifying the PLD equations to delay the assertion of RDY by one or more clocks.

Because the 82596DX requires only one RDY signal, the two data acknowledge signals for the MC68020 (DSACKO and DSACK1) can be combined. Both DSACK signals connect to the inputs of an 74F08 AND gate.

### 5.0 MC68000/82596SX INTERFACE

#### 5.1 Design Specifications

This interface example is based on the following assumptions.

- MC68000 CPU.
- 82596SX LAN coprocessor.
- 16-bit DRAM memory without burst capability.
- DRAM refresh using CAS-before-RAS technique.
- 16 MHz operating frequency.
- Interface logic implemented in PLDs where possible.
- 82596SX signals converted to MC68000 signal types.
- Refresh request signal asynchronous to 16 MHz clock.

**NOTE:**

Many of the circuit elements (e.g., PLDs and flip-flops) in this design probably already exist in designs presently using the MC68000. The extra elements are provided only for completeness. The final design will probably require fewer circuit elements.
5.2 Clocking

This design uses a clock operating at 32 MHz. The 32 MHz clock, CLK2, is used directly by the 82596SX. It is divided by a 74F74 flip-flop to generate two 16 MHz clocks from the Q and Q outputs: CLK1 and CLK1. The MC68000 uses CLK1, but the arbitration logic uses CLK1. The clock-to-output-valid delay of the 74F74 is 3.8 to 7.8 ns. The rising edge of CLK1 corresponds to the rising edge of CLK2 at the beginning of \( \phi_1 \).

5.3 Reset Retiming

The 82596SX reset retiming block is shown in Figure 11. The synchronizing flip-flops are clocked by CLK2. There are two 82596SX specifications for RESET that must be met: the setup time (T23) and the hold time (T24). The worst-case margin is shown in Table 7 (all times are in nanoseconds).

5.4 CA and PORT Generator

The CA and PORT generation block is the same as that shown in Figure 4 and is based on a 20R4-15 PLD clocked by CLK1. AS, the address lines, HIADDR, LDS, and CLK1 are decoded in a combinatorial macro-cell of the 20R4. The macro-cell output is sent to the input of one of the registered macro-cells, which is clocked at the rising edge of CLK1. Since propagation delay through the PLD is much less than the CLK1 cycle time, there will be a large margin on the flip-flop setup time.

Next, the worst-case setup and hold times to the 82596SX are calculated. The 82596SX timings are identical for both CA and PORT. They go active based on the rising edge of CLK1. The setup margins are calculated as follows.

\[
\text{CLK1 cycle} - \text{max PLD output} - \min 82596SX \text{ input setup} - \max \text{CLK2 to CLK1 skew} = 66 - 12 - 11 - 7.8 = 35.2 \text{ ns}
\]

Margins are calculated as follows.

\[
\begin{align*}
&- \min \text{PLD output valid delay} - \min 82596SX \text{ input hold} + \max \text{CLK2 to CLK1 skew} \\
&= 4 - 6 + 3.8 = 1.8 \text{ ns}
\end{align*}
\]

5.5 Bus Arbiter

The bus arbiter is implemented with a 20R8-15 PLD clocked by CLK1. The worst-case margins to this rising edge is calculated as follows.

\[
\begin{align*}
\text{CLK1 cycle} - \max 82596SX \text{ output} & - \min \text{PLD input setup} \\
&+ \min \text{CLK2 to CLK1 skew} \\
&= 66 - 32 - 15 + 3.8 = 22.8 \text{ ns}
\end{align*}
\]

The signal BR does not need to meet any setup or hold times because it is internally synchronized by the MC68000. The PLD flip-flop setup time for BG is checked next. Because BG can go active 0 ns after the falling edge of CLK1, and because there can be up to 2 ns of skew between CLK1 and CLK1, it is not completely safe to directly use BG in the arbiter PLD. Instead it is run through one of the flip-flops in the PLD to fully synchronize the signal. In the worst case, BG can go active about the same time as CLK1 goes high. Because the arbiter's flip-flop will not be clocked until the next rising edge of CLK1, there will be almost 60 ns for the output to reach a valid state.

The outputs of the arbiter, LANCYC and REFNCY, are internally synchronized at their destination, so no output timing analysis is required. An inverter is required for LANCYC to create HLD to the 82596SX. If the PLD has an internal inverter then this will not be required. If not, one of the unused macrocells in the 20R8 can be used to perform the inversion.

5.6 Memory Signal Conversion

The memory signal conversion block is implemented as shown in Section 2.6. A 20R4-15 PLD is used to convert the 82596SX-type control signals to MC68000-type control signals. When the 82596SX does not have the bus all the outputs go to a high-impedance state.

The signals UD, LDS, and A1 are generated by using simple combinatorial decodes of BHE and BLE. The delay will be identical to the PLD propagation delay, which is 15 ns maximum. The signals NEWAS, NEWDS, NEWDBEN, and NEWR/W are generated by using the PLD's registered outputs, which are clocked by CLK1. Their states are determined by the state of the 82596SX's signals ADS, RDY, and W/R.

An 82596SX read or write cycle starts with ADS going active based on the rising edge of CLK2. NEWAS and NEWDBEN will go active on the next rising edge of
CLK1. If the cycle is a read cycle \( \text{NEWDS} \) will also go active. If it is a write cycle \( \text{NEWDS} \) will go active one clock later. In general, the signals go inactive based on \( \text{ROY} \) going active. To meet the data hold times \( \text{NEWDBEN} \) stays active one extra clock during a write cycle. \( \text{NEWR/W} \) is simply the inverted and registered \( W/R \).

The timing of the PLD is checked next. The 82596SX control signals must be valid in time to meet the setup requirements of the PLD's flip-flops. The margin is calculated as follows.

\[
\text{CLK1 cycle} + \min \text{CLK2 to CLK1 skew} - \max \text{82596SX output delay} - \text{PLD input setup} = 66 + 3.8 - 36 - 15 = 18.8 \text{ ns}
\]

\( \text{NEWAS} \) goes active based on the falling edge of CLK1, which is the same as the rising edge of CLK1. The PLD clock to output valid delay is 5 to 12 ns maximum. The skew between the clocks will be \(-2\) to \(+2\) ns. This translates to a 3 to 14 ns delay, which is within the MC68000 specifications of 3 to 40 ns.

5.7 Wait State Generator

The 82596SX bus cycle is at least two clocks long and begins with \( \text{ADS} \) going active during the first clock and \( \text{RDY} \) active in the last clock. The wait state generation block inserts the proper number of wait states during the bus cycle. For this design it was assumed that the DRAM would allow for zero wait state accesses. If slower DRAMs are used, wait states will need to be inserted in the DSACK and RDY generation circuits.

The fastest single cycle transfer in this design requires three clocks for the 82596SX. The wait state generator delays the \( \text{RDY} \) signal to the 82596SX. This provides enough time to meet the data setup and hold specifications. The 82596SX requires that data be valid at least 5 ns before the rising edge of its clock. The MC68000 requires only a 1 ns setup to its clock. The system designer will need to guarantee that the 82596SX has at least a 5 ns data setup to this edge, plus 2 ns for the clock skew. If this cannot be met, another wait state will be needed for all 82596 memory read cycles. This can be done by modifying the PLD equations to delay the assertion of \( \text{RDY} \) by one or more clocks.

ECS and OCS are generated by taking ADS and running it through a tri-state buffer (74F244) that is enabled by HLDA. When the 82596SX has the bus ECS and OCS will go active about 8 ns after ADS.
Each schematic includes only the logic needed to interface the 82596 to the M68000. The address and data bus connections between the two are not currently shown.
A.2 MC68020/82596DX
Several conventions are used in the PLD equations. These are designed to make the equations easier to understand. In general the equations are designed for PLD's with fixed macro-cells. If programmable macro-cells are available, then some reduction will be possible.

Pin signal name assignments are followed by a comment indicating the pin type:

- Input = I
- Combinatorial input/output = I/O
- Registered input/output = R, I/O

Names for active low signals are followed by an underscore. For example, AS__ is an active low signal and BREQ is an active high signal.

Logical operators are defined as:
- Logical AND = &
- Logical OR = #
- Logical NOT = !

Where possible, state and truth table formats are used.

General comments start with an " in the left-most column. Specific Comments appear on the same line as the individual equations or terms within the equation. If there is not room on the right side for the comment to fit on the same line, then it will be indented on the following line.
B.1 MC68030/82596CA

Module ARB FLAG ‘-R3’;
Title ‘82596CA Arbitration for Local Bus’ Rev. B 02/20/90
DOCTOR DESIGN, San Diego, CA
PLD20R8-10’;

" --------------------------------------------------- Description --------------------------------------------------- "

FOR: 82596CA / 68030 Interface

This PLD arbitrates between the CPU, the LAN Controller and
the Refresh requestor for the local bus. Refresh requests are
given highest priority, and the 82596 requests given second
highest priority. The CPU normally controls the local bus
when no requests are pending.

Requestors are granted the bus by using the inverted CPU clock
(CLK1). Since CLK1 is also used in the equations, it must
connected with a separate pin with a separate name (CCLK1_).

The refresh request (RFRQ) is assumed to be an active low
signal having the required 12 ns set-up to the inverted clock
(CLK1). If this set-up cannot be guaranteed, the request
must be synchronized through an external flip-flop, clocked
with CLK1_.

SBG_ is the synchronized 68030 Bus Grant (BG_) signal.

HLDA is the inverted LANCYC*. Due to a lack of P-terms, HLDA
will be delayed by 1 clock. If a PLD with inverter outputs is
available, then LANCYC* can be inverted and used directly as
HLDA.

The Bus Request signal, BREQ, and Backoff, BOFF, are used to
activate the Bus Throttle Timers and backoff function. The
equations are included but the outputs are always set
inactive. It is left to the system designer to define input
conditions for this signal.

The two states LAN_OFF and LAN_RF can be used if the external
circuitry cannot guarantee that the 82596 will get off the bus
in time to do refresh cycles. If these states are used, a
larger PLD will be needed to generate the BGACK to the 68030.

UNUSED INPUT PINS : 5
UNUSED OUTPUT PINS (REGISTERED) : 0
UNUSED OUTPUT PINS (COMBINATORIAL) : 0

"*****************************************************************************"
arb Device 'P20R8';

<table>
<thead>
<tr>
<th>Pin</th>
<th>&quot;I&quot;</th>
<th>&quot;R, I/O&quot;</th>
<th>&quot;R, I/O&quot;</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Mode</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>IDLE</td>
<td>[1,1,1,1,1];</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>REQ</td>
<td>[0,1,1,1,1];</td>
<td>Generic request to CPU for local bus.</td>
</tr>
<tr>
<td>4</td>
<td>RF_CYC</td>
<td>[1,0,0,1,1];</td>
<td>Refresh request has been granted.</td>
</tr>
<tr>
<td>5</td>
<td>LAN_CYC</td>
<td>[1,0,1,0,1];</td>
<td>LAN request has been granted.</td>
</tr>
<tr>
<td>6</td>
<td>PRE_CHG</td>
<td>[1,0,1,1,1];</td>
<td>Required for back-to-back cycles.</td>
</tr>
<tr>
<td>7</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>13</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>14</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>16</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>17</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>18</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>19</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>20</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>21</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>22</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>23</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>24</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

MODE = [BR_, BGACK_, RFCYC_, LANCYC_, BOFF_];

Equations

BREQ := 0;  
" Bus Throttle conditions will need to be defined by the system designer.

SBG_ = !BG_ & CCLK1_  
" Set synchronized bus grant during high clock to phase meet setup to ARB PLD.

# !SBG_ & !BG_  
" Hold until processor bus grant goes away.

# !SBG_ & !CCLK1_  
" Hold through low clock phase to meet setup.

HLDA := !LANCYC_;  
" Create HLDA from inverted LANCYC_.

292076-36
MODE := RESET & IDLE; " Initialize state machine to IDLE State on reset.

State_Diagram IN arb MODE

state IDLE : IF (!RFRQ_ # HOLD) THEN REQ
ELSE IDLE;

state REQ : CASE (!RFRQ_ & !SBG_)
(HOLD & RFRQ_ & !SBG_)
(!((RFRQ_ & !SBG_)
# (HOLD & RFRQ_ & SBG_)))
ENDCASE;

state RF_CYC : CASE (RFRQ_ & !HOLD & SBG_)
(RFRQ_ & HOLD)
(!RFRQ_)
ENDCASE;

state LAN_CYC : CASE (HOLD & RFRQ_ & !LOCK & SBG_)
(!HOLD & !RFRQ_ & !LOCK)
" 
(HOLD & RFRQ_ & !LOCK)
(HOLD & RFRQ_)
ENDCASE;

state PRE_CHG : CASE (RFRQ_ & !HOLD & !LOCK & SBG_)
(!RFRQ_ & !LOCK)
(RFRQ_ & HOLD)
(RFRQ_ & !HOLD & !LOCK & !SBG_) :PRE_CHG;
ENDCASE;

" state LAN_OFF : IF (!HOLD) THEN LAN_RF
ELSE LAN_OFF;

" state LAN_RF : IF (RFRQ_) THEN LAN_CYC
ELSE LAN_RF;

" **************************** Revision History ****************************
" 
" Rev. A 01/03/90 - KKP - First Version
" Rev. B 02/20/90 - KKP - Put BG_Synchronization into PLD.
" 
" **************************** Revision History ****************************

end ARB
Module CAPORT FLAG '-R3';
Title '82596CA Channel Attention and Port DOCTOR DESIGN, San Diego, CA PLD20R4-15';

************************ Description ************************

" FOR: 82596CA / 68030 Interface
This PLD decodes the 68030 address lines and generates the Channel Attention and PORT_ signals to the 82596. The choice of address is of address is left to the system designer.

Nine address decode lines are available. They could be connected to A31-A23. NC1, NC2, NC3, and NC4 are combinational outputs. They can be used as extra address inputs. NC5 is a standard input that is also available as an extra address term. If even more decode lines are required, then the HIADDR input is for the output of the external decoder. This decode must be done in less than 16 ns.

In the line ADDR = [A09,A08,...], the values for A09-A01 should be set high or low (inverted) for the desired range. The decode values for CA_ACC and PORT_ACC (110 and 220) are arbitrary and can be modified as needed.

S0_ and S1_ are state bits used for generating wait states for PORT_ assertion.

UNUSED INPUT PINS : 1
UNUSED OUTPUT PINS (REGISTERED) : 0
UNUSED OUTPUT PINS (COMBINATORIAL) : 4

********************************************
caport Device 'P20R4';

CLK1_ Pin 1; "I" VCC Pin 24;
A5_ Pin 2; "I" HIADDR Pin 23; "I"
A01 Pin 3; "I" NC1 Pin 22; "I/O"
A02 Pin 4; "I" NC2 Pin 21; "I/O"
A03 Pin 5; "I" CA Pin 20; "R,I/O"
A04 Pin 6; "I" S0_ Pin 19; "R,I/O"
A05 Pin 7; "I" S1_ Pin 18; "R,I/O"
A06 Pin 8; "I" PORT_ Pin 17; "R,I/O"
A07 Pin 9; "I" NC3 Pin 16; "I/O"
A08 Pin 10; "I" NC4 Pin 15; "I/O"
A09 Pin 11; "I" NC5 Pin 14; "I"
GND Pin 12; OE_ Pin 13; "I"

292076-38
"Declarations

X,C = .X, C;

ADDR = [A09, A08, A07, A06, A05, A04, A03, A02, A01, X, X];  " User defined address.

CA_ACC MACRO { (ADDR == "h110") & HIADDR & !AS_ };
PORT_ACC MACRO { (ADDR == "h220") & HIADDR & !AS_ };

MODE = [CA, PORT_, S0_, S1_];

IDLE = [1, 1, 1, 1];
PORT_SET = [1, 0, 1, 1];  " Set PORT_ to 82596.
PORT_HLD1 = [1, 0, 0, 1];  " Hold for one clock state.
PORT_HLD2 = [1, 0, 0, 0];  " Hold for a second clock state.
ACCESS_OFF = [1, 1, 0, 0];  " Deassert PORT_ and CA.
CA_SET = [0, 1, 1, 1];  " Set CA to 82596.
CA_HLD1 = [0, 1, 0, 1];  " Hold for one clock state.
CA_HLD2 = [0, 1, 0, 0];  " Hold for a second clock state.

Equations

State_Diagram IN caport MODE

state IDLE   : CASE (PORT_ACC) : PORT_SET;
               (CA_ACC) : CA_SET;
               (! (PORT_ACC # CA_ACC)) : IDLE;
               ENDCASE;

state PORT_SET : GOTO PORT_HLD1;

state PORT_HLD1 : GOTO PORT_HLD2;

state PORT_HLD2 : GOTO ACCESS_OFF;

state ACCESS_OFF : IF AS_ THEN IDLE
                   ELSE ACCESS_OFF;

state CA_SET   : GOTO CA_HLD1;

state CA_HLD1  : GOTO CA_HLD2;

state CA_HLD2  : GOTO ACCESS_OFF;

" ******************** Revision History *******************************
"  "
"  "
"  ******************************************************

end CAPORT
Module CNVRT FLAG '-R3';
Title '82596CA Signal Conversion' Rev. A 01/03/90
DOCTOR DESIGN, San Diego, CA
PLD20R4-10';

************************************************** Description **************************************************

FOR: 82596CA / 68030 Interface

This PLD converts the 82596 signals to 68030 type signals.

DELAYAS_ is generated in the RDY PLD to delay AS_ until it is
known whether a multiple or burst transfer is to take place.

A PLD 20R4 was used in this example, requiring separate
output enables (LANCYC_ and LANCYC2_ connected external to
the PLD) for the registered and latched outputs.

NEWRW_, NEWSAS_, NEWSIZ0, and NEWDBEN_ are registered outputs.
All of these signals will be enabled when the 82596 has
control of the local bus, otherwise they will be tri-stated.

The combinatorial outputs are generated using a truth table.

For completeness, default settings are included for the
impossible BE# input combinations.

UNUSED INPUT PINS : 3
UNUSED OUTPUT PINS (REGISTERED) : 0
UNUSED OUTPUT PINS (COMBINATORIAL) : 0

***********************************************************************************************************
cnvrt Device 'P20R4';

CLK1_ Pin 1; "I"  VCC Pin 24;
NC1_ Pin 2; "I"  RY_ Pin 23; "I"
DELAYAS_ Pin 3; "I"  NEWSA0 Pin 22; "I/O"
BE0_ Pin 4; "I"  NEWSIZ0 Pin 21; "I/O"
BE1_ Pin 5; "I"  NEWRW_ Pin 20; "R,I/O"
BE2_ Pin 6; "I"  NEWSAS_ Pin 19; "R,I/O"
BE3_ Pin 7; "I"  NEWSIZ1 Pin 18; "R,I/O"
WR_ Pin 8; "I"  NEWDBEN_ Pin 17; "R,I/O"
NC2_ Pin 9; "I"  NEWSIZ2_ Pin 16; "I/O"
NC3_ Pin 10; "I"  NEWA1_ Pin 15; "I/O"
NEWCBRQ_ Pin 11; "I"  LANCYC2_ Pin 14; "I"
GND Pin 12; LANCYC_ Pin 13; "I"

292076-40
Equations

\[
\begin{align*}
\text{ENABLE NEWAS} & = \text{!LANCYC}; \\
\text{ENABLE NEWDS} & = \text{!LANCYC}; \\
\text{ENABLE NEWDBEN} & = \text{!LANCYC}; \\
\text{ENABLE NEWRW} & = \text{!LANCYC}; \\
\text{ENABLE NEWSIZ0} & = \text{!LANCYC2}; \\
\text{ENABLE NEWSIZ1} & = \text{!LANCYC2}; \\
\text{ENABLE NEWA0} & = \text{!LANCYC2}; \\
\text{ENABLE NEWA1} & = \text{!LANCYC2}; \\
\end{align*}
\]

\[
\text{!NEWAS} := \\
\text{!DELAYAS} & \quad \text{"Start after BLAST valid.}\]
\[
\# \text{!NEWAS} \& \text{!NEWCBRQ} & \quad \text{"Hold through multiple/burst transfer.}\]
\[
\text{!NEWDS} := \\
\text{!WR} \& \text{!DELAYAS} & \quad \text{"Start DS with AS during read cycle.}\]
\[
\# \text{WR} \& \text{!NEWDBEN} \& \text{RDY} & \quad \text{"Delay DS by 1 clock during a write cycle.}\]
\[
\# \text{!NEWDS} \& \text{!NEWCBRQ} & \quad \text{"Hold until clock following RDY set.}\]
\[
\text{!NEWDBEN} := \\
\text{!DELAYAS} & \quad \text{"Enable data transceivers as soon as 82596 begins its cycle.}\]
\[
\# \text{!WR} \& \text{!NEWDBEN} \& \text{RDY} & \quad \text{"Hold as long as AS during read.}\]
\[
\# \text{WR} \& \text{!NEWDBEN} \& \text{!NEWAS} & \quad \text{"Longer data hold during a write.}\]
\[
\text{!NEWRW} := \text{WR} & \quad \text{"invert WR to match processor}\]
\]

"The following truth table converts the byte enable signals from the 82596 into the 68030 SIZ signals and address lines A0 and A1.

Truth_Table

\[
( [\text{BE3}, \text{BE2}, \text{BE1}, \text{BE0}, \text{LANCYC2}] \rightarrow [\text{NEWSIZ1}, \text{NEWSIZ0}, \text{NEWA1}, \text{NEWA0}] )
\]

\[
\begin{align*}
[ X , X , X , X , 1 ] & \rightarrow [ 1 , 1 , 1 , 1 , 1 ]; \\
[ 1 , 1 , 1 , 1 , 0 ] & \rightarrow [ 1 , 1 , 1 , 1 , 1 ]; \\
[ 1 , 1 , 1 , 0 , 0 ] & \rightarrow [ 0 , 1 , 1 , 1 , 1 ]; \\
[ 1 , 1 , 0 , 0 , 0 ] & \rightarrow [ 1 , 0 , 1 , 1 , 0 ]; \\
[ 1 , 0 , 0 , 0 , 0 ] & \rightarrow [ 1 , 1 , 0 , 1 , 1 ]; \\
[ 0 , 0 , 0 , 0 , 0 ] & \rightarrow [ 0 , 0 , 0 , 0 ]; \\
[ 1 , 1 , 0 , 1 , 0 ] & \rightarrow [ 0 , 1 , 1 , 0 ]; \\
[ 1 , 0 , 0 , 1 , 0 ] & \rightarrow [ 1 , 0 , 1 ]; \\
[ 0 , 0 , 0 , 1 , 0 ] & \rightarrow [ 1 , 0 , 0 ]; \\
[ 1 , 0 , 1 , 1 , 0 ] & \rightarrow [ 0 , 1 , 0 ]; \\
[ 0 , 0 , 1 , 1 , 0 ] & \rightarrow [ 1 , 0 ]; \\
[ 0 , 1 , 1 , 1 , 0 ] & \rightarrow [ 0 , 1 ]; 
\end{align*}
\]
The following BE# input combinations are illegal and will result in erroneous data transfers.

\[
\begin{align*}
[ X, 0, 1, 0, 1 ] & \rightarrow [ 1, 1, 1, 1 ]; \\
[ 0, 1, 0, X, 1 ] & \rightarrow [ 1, 1, 1, 1 ]; \\
[ 0, 1, 1, 0, 1 ] & \rightarrow [ 1, 1, 1, 1 ];
\end{align*}
\]

*************** Revision History *******************************


**************************************************************************

end CNVRT
This PLD generates the RDY_ and BRDY_ signals to the 82596. It also generates the Burst Request (CBREQ_) signal to the memory controller. It uses the 68030 signals Address Strobe (AS_), Data Strobe (DS_), Data Bus Enable (DBEN_), Data Acknowledge (DSACK_) and Synchronous Termination (STERM__). It also uses Cache Burst Acknowledge (CBACK_) from the memory controller and Burst Last (BLAST_) from the 82596. The DELAYAS_ signal is used to delay the generation of AS_ to the memory controller in order to determine whether a burst transfer is about to take place. Because CLK1_ is needed for both the flip-flop registers and in the combinatorial equations, it is connected to both pins 1 and 2. Two separate names are required in the equations (CLK1_ and CCLK1_).

The first three burst data transfers between the 82596 and the memory will be acknowledged with the BRDY_ signal. The last (fourth) burst data transfer cycle will be acknowledged with a RDY_.

This PLD must be 10 ns or faster to meet BRDY set-up to CLK1_ on 82596.

The output DELAYRDY_ is only used inside this PLD to generate a delay for the RDY_ signal to the 82596.

UNUSED INPUT PINS : 2
UNUSED OUTPUT PINS (REGISTERED) : 1
UNUSED OUTPUT PINS (COMBINATORIAL) : 1

rdy Device 'P20R4';

CLK1_  Pin 1; "I"  VCC    Pin 24;
CCLK1_ Pin 2; "I"  NC2    Pin 23; "I"
AS_    Pin 3; "I"  RDY_   Pin 22; "I/O"
DS_    Pin 4; "I"  BRDY_  Pin 21; "I/O"
STERM_ Pin 5; "I"  DELAYAS_ Pin 20; "R, I/O"
DBEN_  Pin 6; "I"  S0     Pin 19; "R, I/O"
ADS_   Pin 7; "I"  NEWCBREQ_ Pin 18; "R, I/O"
CBACK_ Pin 8; "I"  NC3    Pin 17; "R, I/O"
NC1_   Pin 9; "I"  DELAYRDY_ Pin 16; "I/O"
DSACK_ Pin 10; "I"  NC4    Pin 15; "I/O"
BLAST_ Pin 11; "I"  LANCYC_ Pin 14; "I"
GND    Pin 12; OE_    Pin 13; "I"
"Declarations

MODE = [DELAYAS, S0, NEWCBREQ];

IDLE = [1,1,1];
BLST_WT = [0,1,1];  " Wait for BLAST to determine if burst data transfer.
NO_BURST = [1,0,1];  " BLAST_ active so no burst transfer.
BRST_CYC = [0,0,1];  " BLAST_ not active so multiple or burst transfer.
STERM_1 = [0,0,0];  " Wait for acknowledge.
STERM_2 = [0,1,0];  " Wait for acknowledge.
STERM_3 = [1,1,0];  " Wait for acknowledge.

Equations

!ROY_ = !BLAST_ & !DELAYRDY_ & CCLK1_ & !A_ & CBACK_  
  " Return ROY_ whenever BLAST_ asserted.
# !STERM_ & NEWCBREQ_  
  " Fourth burst transfer or synchronous transfer.
# !RDY_ & !DBEN_  
  " Hold ROY_ until data requirement met.

!BRDY_ = !STERM_ & !NEWCBREQ_ & !CBACK_  
  " Assert BRDY_ during burst cycles
# !BRDY_ & !CCLK1_  
  " Hold so recognized on rising edge of CLK1_ to 82596.

!DELAYRDY_ =

  !DS_ & !LANCYC_ & !CCLK1_  " Delay RDY for data setup.
# !DELAYRDY_ & !A_  " Hold until end of data cycle.

State_Diagram IN rdy MODE

state IDLE : IF !ADS_ THEN BLST_WT  
  ELSE IDLE;

state BLST_WT : IF !BLAST_ THEN NO_BURST  
  ELSE BRST_CYC;

state NO_BURST : GOTO IDLE;

state BRST_CYC : CASE (!(BRDY_ & !CBACK_) ) : STERM_1;
  ((!(BRDY_ & !CBACK_) # !BLAST_ ) : IDLE;
  (BRDY_ & !BLAST_ ) : BRST_CYC;
ENDCASE;

state STERM_1 : CASE (!(BRDY_ & !CBACK_) ) : STERM_2;
  ((!(BRDY_ & !CBACK_) # !BLAST_ ) : IDLE;
  (BRDY_ & !BLAST_ ) : STERM_1;
ENDCASE;

292076-44
state STERM_2 : CASE (!BRDY_ & !CBACK_) : STERM_3;
((!BRDY_ & CBACK_) & !BLAST_) : IDLE;
(BRDY_ & BLAST_) : STERM_2;
ENDCASE;

state STERM_3 : IF (!BRDY_ & !BLAST_) THEN IDLE
ELSE STERM_3;

" ******************** Revision History **************************
"  " ***************************************************************
end RDY
B.2 MC68020/82596DX

Module ARB  FLAG '-R3';
Title '82596DX Arbitration for Local Bus'  Rev. A  01/12/90
DOCTOR DESIGN, San Diego, CA
PLD20R6-10';

"  **************************************************  Description  **************************************************
"  FOR:  82596DX / 68020 Interface
"  This PLD arbitrates between the CPU, the LAN Controller and
"  the Refresh requestor for the local bus. Refresh requests are
"  given highest priority, and the 82596DX requests given second
"  highest priority. The CPU normally controls the local bus
"  when no requests are pending. The RDY_ acknowledge signal to
"  the 82596DX is also generated in this PLD. The signal
"  DELAYRDY_ is an embedded signal used only in this PLD to
"  generate RDY_.
"  Requestors are granted the bus by using the inverted CPU clock
"  (CLK1). Because is required for both the registered and
"  combinatorial terms, CLK1 is connected to both the clock and a
"  combinatorial input. The combinatorial term is called CCLK1.
"  The refresh request (RFRQ_) is assumed to be an active low
"  signal having the required 12 ns set-up to CLK1. If this
"  set-up cannot be guaranteed, the request must be synchronized
"  through an external flip-flop, clocked with CLK1.
"  The SBG_ signal is the synchronized 68020 Bus Grant (BG_)
"  signal.
"  Because the 82596 uses and active-high HOLD, LANCYC_ is
"  inverted with an external 74F04.
"  The equations and marco-cells are allocated for the Bus
"  Request (BREQ) signal, which is used to activate the 82596DX
"  Bus Throttle Timers. In these equations it is set inactive.
"  It is left to the system designer to define input conditions
"  for this signal.
"  UNUSED INPUT PINS : 4
"  UNUSED OUTPUT PINS (COMBINATORIAL) : 0
"  UNUSED OUTPUT PINS (REGISTERED) : 0
"  **************************************************
arb Device 'P20R6';

CLK1    Pin  1; "I"   VCC    Pin  24;
CCLK1   Pin  2; "I"   NC3    Pin  23; "I"
AS_     Pin  3; "I"   DELAYRDY_ Pin 22; "I/O"
HOLD    Pin  4; "I"   BREQ   Pin  21; "R,I/O"
LOCK    Pin  5; "I"   LANCYC_ Pin  20; "R,I/O"
DS_     Pin  6; "I"   BR_    Pin  19; "R,I/O"
BG_     Pin  7; "I"   BGACK_ Pin  18; "R,I/O"
DBEN_   Pin  8; "I"   RFCYC_ Pin  17; "R,I/O"
RFRQ_   Pin  9; "I"   SBG_   Pin  16; "R,I/O"
NC1     Pin 10; "I"   RDY_   Pin  15; "I/O"
NC2     Pin 11; "I"   NC4    Pin  14; "I"
GND     Pin 12; "I"   OE_    Pin  13; "I"

MODE = [BR_,BGACK_,RFCYC_,LANCYC_];

IDLE = [1,1,1,1];
REQ = [0,1,1,1];  " Generic request to CPU for local bus.
RF_CYC = [1,0,0,1];  " Refresh request has been granted.
LAN_CYC = [1,0,1,0];  " LAN request has been granted.
PRE_CHG = [1,0,1,1];  " Required for back-to-back cycles.

Equations

BREQ := 0;  " Bus Throttle conditions will need
            " to be defined by the system designer.

!SBG_ = !BG_ & CCLK1  " Set during high phase of ARB clock to
            " meet setup time into PLD.
  # !SBG_ & !BG_  " Hold with processor bus grant.
  # !SBG_ & !CCLK1;  " Hold through low phase of clock to meet setup time.

!DELAYRDY_ =

  !DS_ & !LANCYC_ & !CCLK1  " Delay RDY_ for data set-up.
  # !DELAYRDY_ & !AS_;  " Hold until end of data cycle.

!RDY_ =

  !DELAYRDY_ & CCLK1 & !AS_  " Return RDY_ after delay while
            " data cycle still in progress.
  # !RDY_ & !DBEN_;  " Hold until end of data cycle.
State_Diagram IN arb MODE

state IDLE  : IF (!RFRQ_ & HOLD) THEN REQ
            ELSE IDLE;

state REQ  : CASE (!RFRQ_ & !SBG_)
            (HOLD & RFRQ_)
            (!((!(RFRQ_ & !SBG_)
            & (HOLD & RFRQ_ & SBG_)))):
            RF_CYC;
            RF_CYC;
            RF_CYC;
            RF_CYC;
            ENDCASE;

state RF_CYC: CASE (RFRQ_ & !HOLD)
              (RFRQ_ & HOLD)
              (!RFRQ_): IDLE;
              IDLE;
              IDLE;
              RP_CYC;
              ENDCASE;

state LAN_CYC: CASE (!HOLD & RFRQ_ & !LOCK)
              (!HOLD & !RFRQ_ & !LOCK)
              (HOLD):
              IDLE;
              !PRE_CHG;
              !LAN_CYC;
              !LAN_CYC;
              ENDCASE;

state PRE_CHG: CASE (RFRQ_ & !HOLD & !LOCK)
               (RFRQ_ & HOLD)
               (!RFRQ_ & !LOCK)
               (!SBG_ & RFRQ_ & !HOLD & !LOCK):
               IDLE;
               LAN_CYC;
               RF_CYC;
               PRE_CHG;
               ENDCASE;

" ****************************************** Revision History ******************************************
" Rev. A 01/12/90  - KKP - First Version
" Rev. B 02/20/90  - KKP - Moved BG_synchronization into PLD.
" *********************************************************************************************************************************

end ARB
Module CAPORT FLAG '-R3';
Title '82596DX Channel Attention and Port'  Rev. A  01/12/90
DOCTOR DESIGN, San Diego, CA
PLD20R4-15',

" ************************************************** Description **************************************************
"
"
FOR: 82596DX / 68020 Interface
"
"
This PLD decodes the 68020 address lines and generates the Channel Attention and PORT_ to the 82596DX. The choice of address is left to the system designer.
"
"
Nine address decode lines are available. They could be connected to A31-A23. NC1, NC2, NC3, and NC4 are combinatorial outputs. They can be used as extra address inputs. NC5 is a standard input that is also available as an extra address term. If even more decode lines are required, then the HIADDR input is for the output of the external decoder. This decode must be done in less than 22 ns.
"
"
In the line ADDR = [A09,A08,...], the values for A09-A01 should be set high or low (inverted) for the desired range.
The decode values for CA_ACC and PORT_ACC (110 and 220) are arbitrary and can be modified as needed.
"
"
S0_ AND S1_ are state bits used for generating wait states for PORT_ assertion.
"
"
UNUSED INPUT PINS : 1
UNUSED OUTPUT PINS (REGISTERED) : 0
UNUSED OUTPUT PINS (COMBINATORIAL) : 4
"
"
************************************************** caport Device 'P20R4' **************************************************

CLK1_  Pin 1; "I"  VCC  Pin 24;
A5_  Pin 2; "I"  HIADDR  Pin 23; "I"
A01  Pin 3; "I"  NC1  Pin 22; "I/O"
A02  Pin 4; "I"  NC2  Pin 21; "I/O"
A03  Pin 5; "I"  CA  Pin 20; "R, I/O"
A04  Pin 6; "I"  S0_  Pin 19; "R, I/O"
A05  Pin 7; "I"  S1_  Pin 18; "R, I/O"
A06  Pin 8; "I"  PORT_  Pin 17; "R, I/O"
A07  Pin 9; "I"  NC3  Pin 16; "I/O"
A08  Pin 10; "I"  NC4  Pin 15; "I/O"
A09  Pin 11; "I"  NC5  Pin 14; "I"
GND  Pin 12;  OE  Pin 13; "I"
" Declarations

X,C = .X,.C.;

ADDR = [A09,A08,A07,A06,A05,A04,A03,A02,A01,X,X,X];  " User defined address.

CA_ACC MACRO ( (ADDR == ^h110) & HIADDR & !AS_ );

PORT_ACC MACRO ( (ADDR == ^h220) & HIADDR & !AS_ );
MODE = [CA, PORT_, S0_, S1_];

IDLE = [1, 1, 1, 1];
PORT_SET = [1, 0, 1, 1]; /* Set PORT_ to 82596DX.
PORT_HLD1 = [1, 0, 0, 1]; /* Hold for one clock state.
PORT_HLD2 = [1, 0, 0, 0]; /* Hold for a second clock state.
ACCESS_OFF = [1, 1, 0, 0]; /* Deassert PORT_ and CA.
CA_SET = [0, 1, 1, 1]; /* Set CA to 82596DX.
CA_HLD1 = [0, 1, 0, 1]; /* Hold for one clock state.
CA_HLD2 = [0, 1, 0, 0]; /* Hold for a second clock state.

Equations

State_Diagram IN caport MODE

state IDLE : CASE (PORT_ACC)
            (CA_ACC)
            (NOT (PORT_ACC AND CA_ACC))
            (PORT_ACC OR CA_ACC)
            ENDCASE;

state PORT_SET : GOTO PORT_HLD1;
state PORT_HLD1 : GOTO PORT_HLD2;
state PORT_HLD2 : GOTO ACCESS_OFF;
state ACCESS_OFF : IF AS_ THEN IDLE
                   ELSE ACCESS_OFF;
state CA_SET : GOTO CA_HLD1;
state CA_HLD1 : GOTO CA_HLD2;
state CA_HLD2 : GOTO ACCESS_OFF;

" **************** Revision History ***********************
" ***********end CAPORT

202076-51
Module CNVRT FLAG ‘-R3’;
Title ‘82596DX Signal Conversion’ Rev. A 1/12/90
DOCTOR DESIGN, San Diego, CA
PLD20R4-10’;

"********************************************************************** Description "**********************************************************************

" FOR: 82596DX / 68020 Interface

This PLD converts the 82596DX signals to 68020 type signals.

These signals will be enabled when the 82596DX has control of the local bus (LANCYC_ is low), otherwise they will be tri-stated.

A PLD 20R4 was used in this example, requiring separate enables, LANCYC_ and LANCYC2_, which are the same signal external to the PLD. If the PLD does not require separate output enables for registered and latched outputs then this is not required.

NEWRW_, NEWAS_, NEWDS_, and NEWDBEN_ are registered outputs.
NEWSI20, NEWSI21, NEWA0, and NEWA1 are combinatorial outputs.

The combinatorial outputs are generated using a truth table.
For completeness, default settings are included for the impossible BE# input combinations.

UNUSED INPUT PINS : 4
UNUSED OUTPUT PINS (COMBINATORIAL) : 0
UNUSED OUTPUT PINS (REGISTERED) : 0

**********************************************************************
cnvrt Device 'P20R4';

CLK1_ Pin 1; "I" VCC Pin 24;
NC1 Pin 2; "I" RDY_ Pin 23; "I"
ADS_ Pin 3; "I" NEWA0 Pin 22; "I/O"
BE0_ Pin 4; "I" NEWSI20 Pin 21; "I/O"
BE1_ Pin 5; "I" NEWRW_ Pin 20; "R,I/O"
BE2_ Pin 6; "I" NEWAS_ Pin 19; "R,I/O"
BE3_ Pin 7; "I" NEWDS_ Pin 18; "R,I/O"
WR_ Pin 8; "I" NEWDBEN_ Pin 17; "R,I/O"
NC2 Pin 9; "I" NEWSI21 Pin 16; "I/O"
NC3 Pin 10; "I" NEWA1 Pin 15; "I/O"
NC4 Pin 11; "I" LANCYC2_ Pin 14; "I"
GND Pin 12; LANCYC_ Pin 13; "I"

"Declarations

X = .X.;
Equations

ENABLE NEWAS_ = !LANCYC_;
ENABLE NEWDS_ = !LANCYC_;
ENABLE NEWDBEN_ = !LANCYC_;
ENABLE NEWRW_ = !LANCYC_;
ENABLE NEWSIZ0  = !LANCYC2_;
ENABLE NEWSIZ1  = !LANCYC2_;
ENABLE NEWA0    = !LANCYC2_;
ENABLE NEWA1    = !LANCYC2_;

!NEWAS_ :=
    !ADS_ & NEWAS_ & RDY_;    ;" Start AS_ during 68020 clock low cycle.
    # !NEWAS_ & RDY_;         ;" Hold until clock following RDY_ set.

!NEWDS_ :=
    !WR_ & !ADS_ & NEWDS_ & RDY_;    ;" Start DS_ with AS_ during read.
    # WR_ & !NEWDBEN_ & RDY_;        ;" Delay DS_ by 1 clock during write.
    # !NEWDS_ & RDY_;               ;" Hold until clock following RDY_ set.

!NEWDBEN_ :=
    !ADS_ & !NEWDBEN_ & RDY_;        ;" Enable data transceivers as soon
    # !WR_ & !NEWDBEN_ & RDY_;       ;" as 82596DX begins its cycle.
    # WR_ & !NEWDBEN_ & !NEWAS_;     ;" Hold as long as AS_ during read.
    # !NEWDBEN_ & !NEWAS_            ;" Longer data hold during a write.

!NEWRW_ := WR_;  ;" Invert W/R_ to match processor.

" The following truth table converts the byte enable signals from
" the 82596DX into the 68020 SIZE signals and address lines A0
" and A1.

Truth_Table

( [BE3_,BE2_,BE1_,BE0_,LANCYC2_] -> [NEWSIZ1,NEWSIZ0,NEWA1,NEWA0] )

[ X , X , X , X , 1 ] -> [ 1 , 1 , 1 , 1 , 1 ];
[ 1 , 1 , 1 , 1 , 1 ] -> [ 1 , 1 , 1 , 1 , 1 ];
[ 1 , 1 , 1 , 0 , 0 ] -> [ 0 , 1 , 1 , 1 , 1 ];
[ 1 , 1 , 0 , 0 , 0 ] -> [ 1 , 0 , 1 , 0 , 0 ];
[ 1 , 0 , 0 , 0 , 0 ] -> [ 1 , 0 , 0 , 1 , 0 ];
[ 1 , 0 , 0 , 0 , 1 ] -> [ 0 , 0 , 1 , 0 , 1 ];
[ 1 , 0 , 0 , 0 , 0 ] -> [ 1 , 0 , 0 , 0 , 0 ];
[ 1 , 1 , 0 , 0 , 0 ] -> [ 1 , 1 , 0 , 1 , 0 ];
[ 1 , 0 , 1 , 1 , 1 ] -> [ 1 , 0 , 1 , 0 , 1 ];
[ 1 , 0 , 1 , 1 , 0 ] -> [ 1 , 0 , 1 , 0 , 0 ];
[ 0 , 0 , 1 , 1 , 1 ] -> [ 1 , 0 , 0 , 1 , 1 ];
[ 0 , 1 , 1 , 1 , 1 ] -> [ 1 , 0 , 0 , 0 , 1 ];
[ 0 , 1 , 1 , 1 , 0 ] -> [ 1 , 0 , 0 , 0 , 0 ];
The following BE# input combinations are illegal and will result in erroneous data transfers.

\[
\begin{align*}
\{ X, 0, 1, 0, 1 \} & \rightarrow \{ 1, 1, 1, 1 \}; \\
\{ 0, 1, 0, X, 1 \} & \rightarrow \{ 1, 1, 1, 1 \}; \\
\{ 0, 1, 1, 0, 1 \} & \rightarrow \{ 1, 1, 1, 1 \};
\end{align*}
\]

****************** Revision History ****************************

Rev. A 01/12/90 - KKP - First Version.

**************************************************
B.3 MC68000/82596SX

Module ARB FLAG 'R3';
Title '82596SX Arbitration for Local Bus' Rev. A 01/20/90
DOCTOR DESIGN, San Diego, CA
PLD20R8-15';

************************************************************************************
Description

This PLD arbitrates between the CPU, the LAN Controller and
the Refresh requestor for the local bus. Refresh requests are
given highest priority, and the 82596SX requests given second
highest priority. The CPU normally controls the local bus
when no requests are pending.

Requestors are granted the bus by using the 82596SX clock,
CLK1.

The refresh request (RFRQ_) is assumed to be an active low
signal having the required 12 ns set-up to the clock (CLK1).
If this set-up cannot be guaranteed, the request must be
synchronized through an external flip-flop, clocked with CLK1.

The SBG_ signal is the synchronized 68000 Bus Grant (BG_)
signal. It is be synchronized internally using a flip-flop
clocked with CLK1. Because the 82596SX uses an active high
HLDA, LANCYC_ is inverted using one of the macro-cells.

The equations and macro-cells are allocated for the Bus
Request signal, which is used to activate the Bus Throttle
Timers. It is left to the system designer to define input
conditions for this signal.

UNUSED INPUT PINS : 6
UNUSED OUTPUT PINS (COMBINATORIAL) : 0
UNUSED OUTPUT PINS (REGISTERED) : 1

************************************************************************************

arb Device 'P20R8';

<table>
<thead>
<tr>
<th>Pin</th>
<th>Name</th>
<th>Pin</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>CLKG</td>
<td>24</td>
<td>VCC</td>
</tr>
<tr>
<td>2</td>
<td>RST</td>
<td>23</td>
<td>NC6</td>
</tr>
<tr>
<td>3</td>
<td>AS_</td>
<td>22</td>
<td>BREQ</td>
</tr>
<tr>
<td>4</td>
<td>HOLD</td>
<td>21</td>
<td>HLDA</td>
</tr>
<tr>
<td>5</td>
<td>LOCK</td>
<td>20</td>
<td>LANCYC_</td>
</tr>
<tr>
<td>6</td>
<td>NC1</td>
<td>19</td>
<td>BR_</td>
</tr>
<tr>
<td>7</td>
<td>BG_</td>
<td>18</td>
<td>SBG_</td>
</tr>
<tr>
<td>8</td>
<td>NC2</td>
<td>17</td>
<td>BGACK_</td>
</tr>
<tr>
<td>9</td>
<td>RFRQ_</td>
<td>16</td>
<td>RFCYC_</td>
</tr>
<tr>
<td>10</td>
<td>NC3</td>
<td>15</td>
<td>NC7</td>
</tr>
<tr>
<td>11</td>
<td>NC4</td>
<td>14</td>
<td>NC5</td>
</tr>
<tr>
<td>12</td>
<td>GND</td>
<td>13</td>
<td>OE_</td>
</tr>
</tbody>
</table>
MODE = [BR_,BGACK_,RFCYC_,LANCYC_];
IDLE = [1,1,1,1];    " Generic request to CPU for local bus.
REQ  = [0,1,1,1];    " Refresh request has been granted.
RFCYC = [1,0,0,1];   " LAN request has been granted.
LANCYC = [1,0,1,0];  " Required for back-to-back cycles.
PRE_CHG = [1,0,1,1]

Equations

BREQ := 0;
" Bus Throttle conditions will need to be
" defined by the system designer.

HLDA := !LANCYC_;  " 82596SX requires active-high HLDA.
SBG_ := BG_;      " Synchronized Bus Grant.

MODE := RESET & IDLE;  " Initialize state machine to IDLE State on reset

State_Diagram IN arb MODE

state IDLE : IF (!RFRQ_ & HOLD) THEN REQ
            ELSE IDLE;

state REQ : CASE (!RFRQ_ & !SBG_)
            (RFRQ_ & !VENTORY_)
            (!RFRQ_ & !SBG_ & !HOLD)
            :REQ;
            ENDCASE;

state RF_CYC : CASE (RFRQ_ & !HOLD)
               (RFRQ_ & !HOLD)
               :RF_CYC;
               ENDCASE;

state LAN_CYC : CASE (HOLD & RFRQ_ & !LOCK)
                (HOLD & !RFRQ_ & !LOCK)
                (HOLD)
                :LAN_CYC;
                ENDCASE;

state PRE_CHG : CASE (RFRQ_ & !HOLD & !LOCK)
                (RFRQ_ & !HOLD)
                (RFRQ_ & !LOCK)
                (!SBG_ & RFRQ_ & !HOLD & !LOCK)
                :PRE_CHG;
                ENDCASE;

end ARB
Module CAPORT FLAG 'R3';
Title '82596SX Channel Attention and Port Rev. A 1/20/90
DOCTOR DESIGN, San Diego, CA
PLD20R4-15';

"************************ Description ************************

" FOR: 82596SX / 68000 Interface

" This PLD decodes the 68000 address lines and generates Channel
Attention and PORT_ to the 82596. The choice of address is
left to the system designer.

" Nine address decode lines are available. They could be
connected to A23-A14. NC1 is a combinatorial outputs and
could be used as extra address input. If even more decode
lines are required, then the HIADDR input is for the output
of the external decoder. This decode must be done in less
than 60 ns.

" In the line ADDR = [A09,A08,...], the values for A09-A01
should be set high or low (inverted) for the desired range.
The decode values for CA_ACC and PORT_ACC (110 and 220) are
arbitrary and can be modified as needed.

" S0_ AND S1_ are state bits used for generating wait states for
PORT_ assertion.

" UNUSED INPUT PINS : 0
" UNUSED OUTPUT PINS (COMBINATORIAL) : 1
" UNUSED OUTPUT PINS (REGISTERED) : 2

"************************ Description ************************

caport Device 'P20R4';

CLK1   Pin 1; "I"
CCLK1  Pin 2; "I"
A01    Pin 3; "I"
A02    Pin 4; "I"
A03    Pin 5; "I"
A04    Pin 6; "I"
A05    Pin 7; "I"
A06    Pin 8; "I"
A07    Pin 9; "I"
A08    Pin 10; "I"
A09    Pin 11; "I"
GND    Pin 12; "I"
VCC    Pin 24; "I"
HIADDR Pin 23; "I"
NC1    Pin 22; "I/O"
CA     Pin 21; "I/O"
NC2    Pin 20; "R,I/O"
S0_    Pin 19; "R,I/O"
S1_    Pin 18; "R,I/O"
NC3    Pin 17; "R,I/O"
PORT_ Pin 16; "I/O"
AS_    Pin 15; "I/O"
LDS_   Pin 14; "I"
OE_    Pin 13; "I"

292076-57
"Declarations

X,C = .X.,.C.;

ADDR = [A09,A08,A07,A06,A05,A04,A03,A02,A01,X,X,X]; " User defined address.

CA_ACC MACRO { (ADDR ^= 'h110) & HIADDR & !AS_ ;

PORT_ACC MACRO { (ADDR ^= 'h220) & HIADDR & !AS_ ;

MODE = [S0_,S1_];

IDLE = [ 1, 1 ];

STR_CNT = [ 0 , 1 ]; " PORT_ or CA has been sent to 82596.
CNT_1 = [ 0, 0 ]; " Hold for one clock state.
CNT_2 = [ 1, 0 ]; " Hold for a second clock state.

Equations

!CA =

!LDS_ & !AS_ & CCLK1 & CA_ACC " Start when data valid on bus.
# !CA & !(S0_ & !S1_)
# !CA & CCLK1;
" Hold for at least 2 clocks.
" Guarantee CA hold time to 82596.

!PORT_ = !LDS_ & !AS_ & CCLK1 & PORT_ACC " Start when data valid on bus.
# !PORT_ & !(S0_ & !S1_)
# !PORT_ & CCLK1;
" Hold for at least 2 clocks.
" Guarantee PORT_ hold time to 82596.

State_Diagram IN caport MODE

state IDLE : IF (!PORT_ & CA) THEN STR_CNT
ELSE IDLE;

state STR_CNT : GOTO CNT_1;

state CNT_1 : GOTO CNT_2;

state CNT_2 : IF AS_ THEN IDLE
ELSE CNT_2;

" **************** Revision History ***********************
" " ***********************

end CAPORT
Module RDY_FLAG '-R3';
Title '82596SX Ready and Signal Conversion  Rev. A  01/20/90
DOCTOR DESIGN, San Diego, CA
PLD20R4-15';

"***************************************************************************
Description***************************************************************************
"
" This PLD generates the RDY_ signal to the 82596SX. It also
" converts the 82596SX signals BHE_, BLE_, ADS_ and WR_ to the
" 68000 equivalents, UDS, LDS, AS_, and RW_ and mimics their
" timing to the memory controller.
"
" The output DELAYRDY_ is only used inside this PLD to generate
" a delay for the RDY_ signal to the 82596SX.
"
" A 20R4 was used for this example requiring a separate input
" for the combinatorial enable.
"
" UNUSED INPUT PINS : 5
" UNUSED OUTPUT PINS (COMBINATORIAL) : 0
" UNUSED OUTPUT PINS (REGISTERED) : 0
"
"***************************************************************************
rdy Device 'P20R4';
CLK1 Pin 1; "I" VCC Pin 24;
CCLK1 Pin 2; "I" NC5 Pin 23; "I"
AS_ Pin 3; "I" NEWRW_ Pin 22; "I/O"
BHE_ Pin 4; "I" NEWAS_ Pin 21; "I/O"
BLE_ Pin 5; "I" RDY_ Pin 20; "R,I/O"
WR_ Pin 6; "I" DELAYRDY_ Pin 19; "R,I/O"
NC1 Pin 7; "I" S0_ Pin 18; "R,I/O"
NC2 Pin 8; "I" DELAYAS_ Pin 17; "R,I/O"
NC3 Pin 9; "I" NEWUDS_ Pin 16; "I/O"
ADS_ Pin 10; "I" NEWLDS_ Pin 15; "I/O"
NC4 Pin 11; "I" LANCYC2_ Pin 14; "I"
GND Pin 12; LANCYC_ Pin 13; "I"

MODE = [DELAYAS_,DELAYRDY_,RDY_,S0_];

IDLE = [1,1,1,1];
STR_AS = [0,1,1,1]; "Delay AS_ until clock phase of 68000 S2.
DLY_RDY = [0,0,1,1]; "Delay RDY_ by 1 82596 clock state.
DLY_DS = [0,0,1,0]; "Delay UDS_,LDS_ during write cycle.
STR_RDY = [0,0,0,1]; "Initiate RDY_ for 68000 type 0 wait cycle.
Equations

ENABLE NEWRW_ = !LANCYC2_;  
ENABLE NEWUDS_ = !LANCYC2_;  
ENABLE NEWLDS_ = !LANCYC2_;  
ENABLE NEWAS_ = !LANCYC2_;  

!NEWRW_ = WR_;  
# !NEWRW_ & !NEWAS_;  
" Invert 82596SX signal  
" Hold write until AS_ negated  

!NEWUDS_ =  

!WR_ & !DELAYAS_ & !BHE_ & !CCLK1 & !AS_  
# !WR_ & !SO_ & !BHE_ & !CCLK1 & !AS_  
# !NEWUDS_ & !WR_ & !DELAYAS_  
# !NEWUDS_ & WR_ & RDY;  
" Start UDS_ with AS_ on read.  
" Delay UDS_ on a write.  
" Hold thru data cycle.  

!NEWUDS_ =  

!WR_ & !DELAYAS_ & !BLE_ & !CCLK1 & !AS_  
# !WR_ & !SO_ & !BLE_ & !CCLK1 & !AS_  
# !NEWUDS_ & !WR_ & !DELAYAS_  
# !NEWUDS_ & WR_ & RDY;  
" Start LDS_ with AS_ on read.  
" Delay LDS_ on a write.  
" Hold thru data cycle.  

!NEWAS_ = !DELAYAS_ & !CCLK1'  
# !NEWAS_ & !WR_ & !DELAYAS_  
# !NEWAS_ & WR_ & RDY;  

State_Diagram IN rdy MODE  

state IDLE : IF !ADS_ THEN STR AS  
ELSE IDLE;  

state STR AS : IF !WR_ THEN DLY RDY  
ELSE DLY DS;  

state DLY DS : GOTO DLY RDY;  

state DLY RDY : GOTO STR RDY;  

state STR RDY : GOTO IDLE;  

" ******************** Revision History ********************"  

"  
" ********************"  

end RDY
APPENDIX C
TIMING DIAGRAMS

The following section includes the timing diagram for each specific design. A summary of the timing specifications is also included.

C.1 MC68030/82596CA
- Block Diagram
- MC68030 and 82596CA Clock Synchronization
- MC68030 and 82596CA CA and PORT Access
- MC68030 Local Arbitration (1 page)
- 82596CA Memory Access (2 pages)
- Timing Summary

C.2 MC68020/82596DX
- Block Diagram
- MC68020 and 82596DX Clock Synchronization
- MC68020 and 82596DX CA and PORT Access
- MC68020 Local Arbitration (2 pages)
- 82596DX Memory Access
- Timing Summary

C.3 MC68000/82596SX
- Block Diagram
- MC68000 and 82596SX Clock Synchronization
- MC68000 and 82596SX CA and PORT Access
- MC68000 Local Arbitration (1 page)
- 82596SX Memory Access (2 pages)
- Timing Summary
INTERFACE BETWEEN 68030 AND 82596CA LAN CONTROLLER
This design assumed interleaved memory for burst filling with zero wait states between accesses.

Synchronization circuit is required for 68030 SYSRESET to prevent metastable on RESET to 82596.

RESET set-up time 123 requires 8 ns.
The 68030 runs off CLK1. The 82596 runs off CLK1. CA and PORT access require 2 wait states.

Ca and PORT set-up time T13,39 require 7 ns

Ca and PORT hold time T14,30 require 4 ns

DOCTOR DESIGN
San Diego, CA
68030 CA & PORT ACCESS
1/1/80 XEP 830_port CA

102x231

S0

S1

DOCTOR DESIGN
San Diego, CA
68030 CA & PORT ACCESS
1/1/80 XEP 830_port_CA

102x231

S0

S1

The 68030 runs off CLK1. The 82596 runs off CLK1. CA and PORT access require 2 wait states.
DOCTOR DESIGN
San Diego, CA
8256 Memory Access
1/3/90

Data set-up to the 68030 is 1 ns. The 82356 requires a 5 ns set-up to T2. The violates 68030 timing by 5 ns + 2 ns = 7 ns. The memory controller must guarantee this timing or a wait state will need to be inserted.

BRDY set-up = 17 ns from latest STERW active to clock rising edge = RLD delay (10 ns) + 7 ns = 17 ns lower than 19 ns. Need 2 ns. Use faster RLD or insert a wait state if STERW cannot be guaranteed 1 ns earlier.

It is assumed that the value of AS42 is incremented by the system hardware.

This design assumed interleaved memory for burst filling with zero skew.
This design assumed interleaved memory for burst filling with zero wait states between accesses.
MC68030 AND 82596CA TIMING SUMMARY FOR 33 MHz

MC68030 PARAMETERS

6 82596 puts address out 1 clock phase before 68030 SO.
6A ADS used to generate ECS and OCS before AS asserted.
7 LANCYC off + buffer off = 15 + 10 = 25 ns.
9 Derived from PLD with clock to Q delay of 8 ns.
12 Derived from PLD with clock to Q delay of 8 ns.
12A Worst case could hold ECS and OCS as long as 20 ns (82596) + delay through buffer. Note to system designer.
13 Could be a violation for AS, DS to address hold of 4 ns (82596) = 8 ns (PLD) - 2 ns (skew) = -6 ns. System designer must guarantee address hold.
14 30 ns + 30 ns - 3 ns (common path through PLD) = 57 ns.
15 30 ns - 2 ns (skew) = 28 ns.
16 Floated with LANCYC going high. Minimum 30 ns to next cycle.
17 R/W invalid 1 clock cycle after AS/DS negated.
18 Set with addresses from 82596.
20 Set with addresses from 82596.
21 R/W set 1 clock cycle before AS = 30 ns.
22 Write cycle minimum setup to DS = 30 ns + 30 ns - 8 ns (R/W through PLD) + 5 ns (DS through common PLD) = 57 ns.
23 82596 provides required time.
25 Minimum time = 30 ns (clock) - 8 ns (AS through PLD) + 4 ns (82596) = 26 ns.
25A 30 ns - 2 ns (skew) = 28 ns.
26 30 ns + 30 ns - 19 ns (82596) + 3 ns (PLD) = 44 ns.
27 Memory controller must guarantee 1 ns.
28 N/A
29 30 ns + 4 ns (82596) - 8 ns (AS through PLD) = 26 ns.
31 N/A
32 Plenty of time
33 Latched in ARB PLD

82596CA PARAMETERS

T13 30 ns - 12 ns (PLD) = 18 ns.
T14 5 ns through PLD for PORT.
T17 30 ns - 10 ns (PLD) = 20 ns.
T18 3 ns (DBEN through PLD) + 3 ns (PLD) = 6 ns.
T19 May violate by 5 ns + 2 ns (skew) - 1 ns (memory controller) = 6 ns. System designer will need to guarantee extra 2 ns setup time.
T20 3 ns (DS from PLD) + delay through memory controller.
T21 30 ns - 8 ns = 22 ns.
T23 30 ns - 9 ns - 2 ns skew = 19 ns.
T24 4 ns minimum through flip-flop.
T26 3 CLK2 cycles.
T27 30 ns + 15 ns - 18 ns (68030) = 27 ns
T28 15 ns - 12 ns (PLD) + 2 ns (68030) - 2 ns (skew) = 3 ns
T29 30 ns - 12 ns = 18 ns.
T30 Minimum 3 ns through 10, 12, or 15 ns PLD.

N/A = Not Applicable
15 ns = 1/2 clock period
30 ns = 1 clock period
INTERFACE BETWEEN 68020 AND 82596DX LAN CONTROLLER

- CLK1
- CLK1*
- AS*
- DEC
- PAL
- A(0-31)
- D(0-31)
- A(2,31)
- RFRQ*
- LOCK*
- AR8
- PAL
- LANCYC*
- HLDA
- HOLD
- RDY*
- BE(0-3)*
- ADS*, W/R
- MEMORY
- SIGNAL
- CONVERSION
- A1,A0
- MEMORY
- CONTROL
- MA(0-X)
- MEMORY
- RAS
- CAS
- OE(HI,LO)
- WE(0-3)
- INT
- 292076-23
Synchronization circuit is required for 8020 SYSRESET* to prevent metastable on RESET to 82596.

Reset set-up time latch requires 5 ns.
MC68020 AND 82596DX TIMING
SUMMARY FOR 33 MHz

MC68020 PARAMETERS

6 82596 puts address out 1 clock phase before 68020 SO.
6A ADS used to generate ECS and OCS before AS asserted.
7 LANCYC off + buffer off = 15 + 10 = 25 ns.
9 Derived from PLD with clock to Q delay of 8 ns.
12 Derived from PLD with clock to Q delay of 8 ns.
12A Worst case could hold ECS and OCS as long as 20 ns (82596) + delay through buffer. Note to system designer.
13 Could be a violation for AS, DS to address hold of 4 ns (82596) = 8 ns (PLD) - 2 ns (skew) = -6 ns. System designer must guarantee address hold.
14 30 ns + 30 ns - 3 ns (common path through PLD) = 57 ns.
15 30 ns - 2 ns (skew) = 28 ns.
16 Floated with LANCYC going high. Minimum 30 ns to next cycle.
17 R/W invalid 1 clock cycle after AS/DS negated.
18 Set with addresses from 82596.
20 Set with addresses from 82596.
21 Setting with AS could violate read cycle timing. System designer must guarantee that 5 ns setup is not required.
22 Write cycle minimum setup to DS = 30 ns - 8 ns (R/W through PLD) + 5 ns (DS through common PLD) = 27 ns. The system designer must verify that this meets memory controller timing.
23 82596 provides required time.
25 Minimum time = 30 ns (clock) - 8 ns (AS through PLD) + 4 ns (82596) = 26 ns.
25A 30 ns - 2 ns (skew) = 28 ns.
26 30 ns - 19 ns (82596) + 3 ns (PLD) = 14 ns.
27 Memory controller must guarantee 5 ns.
28 N/A
29 30 ns + 4 ns (82596) - 8 ns (AS through PLD) = 26 ns.
31 N/A

82596DX PARAMETERS

T13 30 ns - 12 ns (PLD) = 18 ns.
T14 5 ns through PLD for PORT.
T17 30 ns - 10 ns (PLD) = 20 ns.
T18 3 ns (DBEN through PLD) + 3 ns (PLD) = 6 ns.
T19 May violate by 5 ns + 2 ns (skew) - 5 ns (memory controller) = 2 ns. System designer will need to guarantee extra 2 ns setup time.
T20 3 ns (DS from PLD) + delay through memory controller.
T21 30 ns - 8 ns = 22 ns.
T22 3 ns (PLD) + external inverter.
T26 3 CLK2 cycles.
T27 30 ns + 15 ns - 18 ns (68020) = 27 ns
T29 30 ns - 12 ns = 18 ns.
T30 Minimum 3 ns through 10, 12, or 15 ns PLD.
N/A = Not Applicable
15 ns = ½ clock period
30 ns = 1 clock period
INTERFACE BETWEEN 68000 AND 82596SX LAN CONTROLLER

68030

82596SA

CLK1

A(0-23)
D(0-15)
A(1,31)

CLK1

DEC
PAL

PAL

ARB

MEMORY

CONVERSION

MEMORY
CONTROL

MEMORY

RAS CAS
WE(0-3)

CLK1

AS*

BG*

BGACK*, BR*

AS*

R/W*

UDS*, LDS*

A0

MUX

MA(0-X)

MEMORY

CONTROL

RAS CAS
WE(0-3)

292076-29
Synchronization circuit is required for 80000 SYRESET* to prevent metastable on RESET to 82596.

RESSET set-up time 123 requires 5 ns.
The 68000 runs off CLK1. The 82592 runs off CLK1. CA and PORT accesses require 3 wall states.

CA and PORT set-up time T13,39 require 19 ns.

CA and PORT hold time T14,30 require 8 ns.

82596 Write Cycle

S0015 CPU PORT access hold time T28 requires 8 ns.
DOCTOR DESIGN
San Diego, CA
88030 Local Bus Arbitration
1/20/90 kkp orb._sx

NOT SHOWN SCALE...
Actual cycles will be longer
than shown.
Address hold violation may occur from AS depending on memory configuration. System designer will need to guarantee enough address hold to memory.

Data hold set-up T19 requires 9 ns + 2 ns (skew) = 11 ns before the rising edge of T1. The 68000 requires only 7 ns. The System Designer will need to guarantee additional 4 ns.

A write cycle must be held for 1 extra clock to guarantee data hold to the memory.
MC68000 AND 82596SX TIMING
SUMMARY FOR 33 MHz

**MC68000 PARAMETERS**

6  82596 puts address out 1 clock phase before 68000 SO.
6A FC valid when 82596 arbitrates for the bus, with LANCYC valid.
7  Address goes away with LANCYC going invalid.
8  Address goes away with AS. System design must verify that this meets memory controller requirements.
9  Derived from 15 ns PLD.
11 62 ns – 36 ns (82596 address delay) + 31 ns + 5 ns (AS through PLD) = 62 ns.
12 FC valid when 82596 arbitrates for the bus, with LANCYC valid.
14 16 ns + 62 ns + 62 ns + 10 ns (2 PLDs) = 150 ns.
15 62 ns minimum.
16 Control bus held until LANCYC goes invalid.
17 AS negated to R/W negated is the minimum time through the RDY PLD. The system designer must verify that this meets memory controller requirements.
18 Maximum from clock high is 33 ns (82596) + 15 ns (PLD) + 2 ns (skew) – 31 ns (clock) = 19 ns.
20 Same as above.
21 No timing relationship is given for the 82596 between address valid and W/R low. It is assumed that if address is delayed out of the 82596, W/R will be delayed by about the same amount. The W/R signal has an additional PLD delay for inversion.
22 Minimum time = 19 ns (18 above) – 31 ns (clock) + 62 ns (clock) = 50 ns.
23 82596 outputs data immediately on a write cycle.
25 AS, DS negated 1 clock cycle before ending 82596 write to meet this parameter by 62 ns – 27 ns (2 PLD delays) – 2 ns (skew) = 33 ns.
26 82596 outputs data immediately on a write cycle.
27 Data setup to clock low for 68000 is 7 ns which could violate T19 below.
28 N/A
29 Memory controller guarantees 0 ns.
31 N/A
32 Transition time depends on flip-flop used for deriving RESET.
33 Setup to PLD.
34 Setup to PLD.
38 Synchronizing BG and generating LANCYC will be a minimum of 31 ns + 62 ns = 93 ns.
46 Minimum width low for 82596 cycle is read cycle – 4 clocks.
53 Minimum = 31 ns (clock) + 4 ns (82596) = 35 ns.

**82596SX PARAMETERS**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>T13</td>
<td>62 ns – 15 ns (PLD) = 47 ns.</td>
</tr>
<tr>
<td>T14</td>
<td>31 ns + 5 (PLD) + 36 ns.</td>
</tr>
<tr>
<td>T17</td>
<td>62 ns – 12 ns (PLD) = 50 ns.</td>
</tr>
<tr>
<td>T18</td>
<td>5 ns (PLD).</td>
</tr>
<tr>
<td>T19</td>
<td>May violate by 9 ns (82596 setup) + 2 ns (skew) – 7 ns (memory controller) = 4 ns. System designer must verify that this meets requirements.</td>
</tr>
<tr>
<td>T20</td>
<td>5 ns (DELAYAS from PLD) + 5 ns (DS through PLD) + delay through memory controller.</td>
</tr>
<tr>
<td>T21</td>
<td>62 ns – 12 ns = 50 ns.</td>
</tr>
<tr>
<td>T23</td>
<td>31 ns – 8 ns (flip-flops) – 9 ns (FF) = 14 ns.</td>
</tr>
<tr>
<td>T24</td>
<td>4 ns (FF) – 4 ns (FF) = 8 ns</td>
</tr>
<tr>
<td>T26</td>
<td>N/A</td>
</tr>
<tr>
<td>T27</td>
<td>62 ns – 50 ns (68000) – 5 ns (PORT from PLD) = 69 ns</td>
</tr>
<tr>
<td>T28</td>
<td>62 ns – 5 ns (PORT through PLD) = 57 ns,</td>
</tr>
<tr>
<td>T29</td>
<td>62 ns – 15 ns = 47 ns.</td>
</tr>
<tr>
<td>T30</td>
<td>31 ns + 5 ns (PLD) = 36 ns.</td>
</tr>
</tbody>
</table>

N/A = Not Applicable

31 ns = ½ clock period
62 ns = 1 clock period
Each parts list includes only those components that are part of the interface. The memory controller and memory components are not included.

**D.1 MC68030/82596CA**

<table>
<thead>
<tr>
<th>Quantity</th>
<th>Generic Number</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.5</td>
<td>74F74</td>
<td>Dual D Flip-Flop</td>
</tr>
<tr>
<td>3</td>
<td>20R4</td>
<td>24-pin PLD; 4 Registered Outputs</td>
</tr>
<tr>
<td>1</td>
<td>20R8</td>
<td>24-pin PLD; 8 Registered Outputs</td>
</tr>
<tr>
<td>0.5</td>
<td>74F244</td>
<td>Octal Tri-State Buffer</td>
</tr>
</tbody>
</table>

Each PLD must have no more than 10 ns propagation delay for 33 MHz design. Each PLD must have no more than 15 ns propagation delay for 25 MHz design.

**D.2 MC68020/82596DX**

<table>
<thead>
<tr>
<th>Quantity</th>
<th>Generic Number</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.5</td>
<td>74F74</td>
<td>Dual D Flip-Flop</td>
</tr>
<tr>
<td>2</td>
<td>20R4</td>
<td>24-pin PLD; 4 Registered Outputs</td>
</tr>
<tr>
<td>1</td>
<td>20R6</td>
<td>24-pin PLD; 6 Registered Outputs</td>
</tr>
<tr>
<td>0.5</td>
<td>74F244</td>
<td>Octal Tri-State Buffer</td>
</tr>
</tbody>
</table>

Each PLD must have no more than 10 ns propagation delay for 33 MHz design. Each PLD must have no more than 15 ns propagation delay for 25 MHz design.

**D.3 MC68000/82596SX**

<table>
<thead>
<tr>
<th>Quantity</th>
<th>Generic Number</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.5</td>
<td>74F74</td>
<td>Dual D Flip-Flop</td>
</tr>
<tr>
<td>2</td>
<td>20R4</td>
<td>24-pin PLD; 4 Registered Outputs</td>
</tr>
<tr>
<td>1</td>
<td>20R8</td>
<td>24-pin PLD; 8 Registered Outputs</td>
</tr>
</tbody>
</table>

Each PLD must have no more than 15 ns propagation delay for 16 MHz design.
Implementing 10Base-T Networks with Intel’s Twisted Pair Ethernet* Components and Supercomponents

WILLIAM WAGER
TECHNICAL MARKETING ENGINEER

*Ethernet is a registered trademark of Xerox Corporation.
# Implementing 10Base-T Networks with Intel's Twisted Pair Ethernet Components and Supercomponents

## CONTENTS

<table>
<thead>
<tr>
<th>CONTENTS</th>
<th>PAGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>PROLOGUE</td>
<td>1-572</td>
</tr>
<tr>
<td>1.0 INTRODUCTION</td>
<td>1-572</td>
</tr>
<tr>
<td>2.0 SYSTEM DESCRIPTION</td>
<td>1-574</td>
</tr>
<tr>
<td>2.1 Network Description</td>
<td>1-574</td>
</tr>
<tr>
<td>2.1.1 Medium Attachment Unit</td>
<td>1-574</td>
</tr>
<tr>
<td>2.1.2 Multiple Port Repeaters</td>
<td>1-575</td>
</tr>
<tr>
<td>2.1.3 Data Terminal Equipment</td>
<td>1-576</td>
</tr>
<tr>
<td>2.1.4 Link Segments</td>
<td>1-576</td>
</tr>
<tr>
<td>2.2 Integrating with Existing 802.3 Networks</td>
<td>1-576</td>
</tr>
<tr>
<td>2.3 Software Compatibility</td>
<td>1-577</td>
</tr>
<tr>
<td>3.0 NETWORK SYSTEM COMPONENT DESIGN</td>
<td>1-577</td>
</tr>
<tr>
<td>3.1 Designing a DTE Node Based on the 82521TB Serial Supercomponent</td>
<td>1-577</td>
</tr>
<tr>
<td>3.2 Designing a DTE Node Based on the 82506TB Twisted Pair MAU Chip</td>
<td>1-577</td>
</tr>
<tr>
<td>3.2.1 Host to Ethernet LAN Controller Interface</td>
<td>1-579</td>
</tr>
<tr>
<td>3.2.2 Ethernet LAN Controller to 82C501AD Interface</td>
<td>1-579</td>
</tr>
<tr>
<td>3.2.3 The 82C501AD to 82506TB Interface</td>
<td>1-579</td>
</tr>
<tr>
<td>3.2.4 The 82506TB to Analog Front-End Interface (AFE)</td>
<td>1-579</td>
</tr>
<tr>
<td>3.3 Using the 82523TB Twisted Pair Ethernet MAU to Convert Existing</td>
<td>1-579</td>
</tr>
<tr>
<td>Ethernet Nodes to 10BASE-T</td>
<td>1-579</td>
</tr>
<tr>
<td>3.4 Designing a MAU Using the 82506TB Twisted Pair MAU Chip</td>
<td>1-579</td>
</tr>
<tr>
<td>3.5 Designing a Multiple Port Repeater Based on the 82505TA MPR</td>
<td>1-579</td>
</tr>
<tr>
<td>3.5.1 82505TA to 82504TA Interface and Clock Generation</td>
<td>1-579</td>
</tr>
<tr>
<td>3.5.2 Twisted Pair Port Design</td>
<td>1-581</td>
</tr>
<tr>
<td>3.5.3 AUI Port</td>
<td>1-581</td>
</tr>
<tr>
<td>3.5.4 Port Disable Control</td>
<td>1-581</td>
</tr>
<tr>
<td>3.5.5 LED Control</td>
<td>1-581</td>
</tr>
<tr>
<td>4.0 ANALOG FRONT END</td>
<td>1-584</td>
</tr>
<tr>
<td>4.1 Preconditioning Voltage Summer</td>
<td>1-584</td>
</tr>
<tr>
<td>4.2 High-Voltage Protection</td>
<td>1-584</td>
</tr>
<tr>
<td>4.3 EMI Filter</td>
<td>1-584</td>
</tr>
<tr>
<td>4.4 Line Coupling Devices</td>
<td>1-584</td>
</tr>
<tr>
<td>4.5 Layout Considerations</td>
<td>1-586</td>
</tr>
<tr>
<td>5.0 SUMMARY</td>
<td>1-587</td>
</tr>
<tr>
<td>APPENDIX A. EQUATIONS FOR EPLD VERSION OF 10BASE-T STATE Machines</td>
<td>1-588</td>
</tr>
</tbody>
</table>
PROLOGUE

At the time of this publication, the IEEE 802.3's 10BASE-T task force is completing its work writing the Twisted Pair Ethernet standard, and the IEEE 802.3 working group has approved the standard. The draft is finished, all that remains is for the IEEE Technical Committee on Computer Communications to approve the draft and to forward it to the IEEE Standards Board, which is expected in September 1990. The work now falls to the designers of 10BASE-T products to implement the standard correctly. The objective of this application note is to aid system designers to meet this goal.

1.0 INTRODUCTION

This application note is presented as an aid to designing a twisted pair Ethernet (TPE) LAN using Intel's TPE products and Intel's family of Ethernet LAN controllers (82586, 82590, and the 82596 family). It is aimed at system designers working on TPE designs who have working knowledge of the IEEE 802.3 standards and some analog design expertise. It can be used in conjunction with application note AP-274, Implementing Ethernet/ Cheapernet with the Intel 82586, by Kiyoshi Nishide. It supersedes AP-324 Implementing Twisted Pair Ethernet with the Intel 82504TA, 82505TA, and 82521TA.

Intel has introduced the 82521TB Serial Supercomponent (SSC), the 82523TB Medium Attachment Unit (MAU), the 82505TA Multiport Repeater Controller (MPR), and the 82506TB Twisted Pair MAU Chip (TPMC). They simplify the design of Twisted Pair Ethernet LANs based on the 10BASE-T standard. The 10BASE-T network is compatible with other ANSI/IEEE 802.3 networks (e.g., 10BASE5 and 10BASE2) at the medium access control (MAC) and physical signaling sublayers. This means that a twisted pair Ethernet LAN based on these products will be software compatible with 802.3 standard networks, and can be included in mixed networks by connection through a standard attachment unit interface (AUI) port of a repeater.

A twisted pair Ethernet LAN using these products consists of several elements: data terminal equipment (DTE), medium attachment units (MAU), multiple port repeaters (MPR), and a cable plant. More complex networks, which connect to existing 802.3 networks (e.g., 10BASE5 or 10BASE2), can be constructed by using the 802.3-standard AUI port of the MPR. A typical network using all of these elements is shown in Figure 1.

Four types of DTE and MAU combinations are shown. Two are embedded MAUs (contained within the DTE) and two are external MAUs (connected to the DTE node through a standard AUI cable). The embedded MAU designs use either the 82521TB SSC or the 82506TB TP MAU chip and its associated circuitry. One external MAU is the 82523TB, the other is based on the 82506TB. The repeaters are designed around the 82505TA and also contain one 82504TA. Each repeater contains 11 twisted pair ports, with embedded MAUs, and 1 AUI port.

The cable plant consists of standard telephone wire: either 4- or 25-pair, unshielded, twisted pair (26–22 gauge). Each segment uses two twisted pairs, transmit (TD) and receive (RD), and can coexist with other services (such as standard telephone) in the same cable bundle. Each segment has a maximum length of 100 m.
Figure 1. Typical 10BASE-T Network

KEY:
DTE: Data Terminal Equipment
MAU: Medium Attachment Unit
FOIRL: Fiber Optic Interrepeater Link
TSI: Transceiver Serial Interface
AFE: Analog Front-End
SSC: Serial Supercomponent
MSC: MAU Supercomponent

REPEATER 82505TA/82504TA

REPEATER 82505TA/82504TA

TP

TPE

FOIRL

COAX

TP

Interrepeater Link

Embedded MAU

DTE 82521TB

DTE 82501AD + 82506TB + AFE

DTE 82501AD

MAU

E Net

AUI (50 m)

AUI (50 m)

100 m

100 m

Server

Ethernet coax

(500 m)
2.0 SYSTEM DESCRIPTION

2.1 Network Description

Table I compares the 10BASE-T network features with the older standards 10BASE5 and 10BASE2.

2.1.1 MEDIUM ATTACHMENT UNIT

The MAU, or transceiver, provides the circuits required to interface to the twisted pair wire. It performs the following functions: line driving with preconditioning, line receiving, collision detection, linkbeat transmission, link integrity processing, jabber protection, and signal_quality_error test.

**MAU Line Drivers.** The transmitter is designed to drive a 96-Ω ± 20% load (76 to 115 Ω) and must meet all the specifications when connected to a 100-Ω resistive load. It is dc isolated from the twisted pair by a transformer, and has a matched source impedance of 96 Ω ± 20%. It will achieve a drive level of 2.2- to 2.8-V peak differential when driving a 100-Ω load. When the driver is sending a 10-MHz data pattern all harmonics must be 27 db below the fundamental. The signal is Manchester encoded. The return loss of the transmitter is 15 db below the incident signal in the 5- to 10-MHz range whenever the source impedance of the measuring device is between 85 and 111 Ω. These specifications apply whenever power is applied to the MAU.

A preconditioning algorithm is incorporated into the transmit circuitry. This algorithm improves overall system jitter performance by reducing the amount of jitter induced by the twisted pair. The line drivers will drive full amplitude during “thin” (50 ns) pulses and the first half of “fat” (100 ns) pulses. They will reduce their drive level to approximately 33% during the second half of “fat” Manchester pulses. This prevents the twisted pair from overcharging during the fat pulses. Without this preconditioning, the overcharge would cause a delay in the zero crossing following the “fat” bit, resulting in greater induced jitter. Figure 2 shows the idealized output waveform for the preconditioned signal at the transmitter. Preconditioning the signal limits the jitter added by the MAU and a 100-m twisted pair cable to 3.5 ns (8.0 ns when the MAU is directly attached to a 100-Ω load).

The common mode to differential impedance balance of the transmitter must exceed 29 − (17 × log f/10) db, where f is the frequency in megahertz. The magnitude of the total common output voltage will be less than 50 mV peak. Additionally, the application of a 15-V peak, 10.1-MHz sine wave will not change the differential voltage by more than 100 mV or add more than 1.0 ns edge jitter for all data sequences.

**MAU Line Receivers.** The MAU line receivers are also dc isolated by a transformer. They must have a matched differential impedance such that the return loss is at least 15 db below the incident signal in the range from 5 to 10 MHz whenever the source impedance of the measuring device is between 85 and 111 Ω. It must operate properly in the presence of any valid Manchester signal with a magnitude of 0.585 to 3.1-V differential and up to ±13.5 ns of edge jitter. It detects the End of Packet (IDL signal) within 2.3 bit times. The squelch circuit rejects as noise any of the following signals: signals less than 300 mV peak in magnitude, all signals less than 2 MHz and 6.2 V_p-p, and any sinusoid of single cycle duration starting at either zero crossing and between 2 and 15 MHz and less than 6.2 V_p-p. It can tolerate a 25- V_p-p, 500-kHz square wave and add no more than 2.5 ns of edge jitter to the signal.

**Collision Detect.** The MAU detects collisions by the simultaneous occurrence of activity on transmit and receive pair. Collisions are detected by transmitting stations and repeaters. When a transmitting station detects a collision it begins the normal 802.3 collision sequence of jam, random back off, and retransmit. When a repeater detects a collision it also begins a jam on all ports and it enforces the minimum frame length of 96 bits.

<table>
<thead>
<tr>
<th>FEATURE</th>
<th>10BASE-T</th>
<th>10BASE5</th>
<th>10BASE2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Access Method</td>
<td>CSMA/CD</td>
<td>CSMA/CD</td>
<td>CSMA/CD</td>
</tr>
<tr>
<td>Data Rate</td>
<td>10 Mb/s</td>
<td>10 Mb/s</td>
<td>10 Mb/s</td>
</tr>
<tr>
<td>Controller</td>
<td>82586/82596</td>
<td>82590/82592</td>
<td>82586/82596</td>
</tr>
<tr>
<td>Software</td>
<td>Existing</td>
<td>Existing</td>
<td>Existing</td>
</tr>
<tr>
<td>Segment Length</td>
<td>100 m</td>
<td>500 m</td>
<td>185 m</td>
</tr>
<tr>
<td>Topology</td>
<td>Star</td>
<td>Bus</td>
<td>Bus</td>
</tr>
<tr>
<td>Wire</td>
<td>Unshielded TP</td>
<td>Yellow Coax</td>
<td>Thin Coax</td>
</tr>
<tr>
<td>Impedance</td>
<td>96 Ω</td>
<td>50 Ω</td>
<td>50 Ω</td>
</tr>
<tr>
<td>Connector</td>
<td>RJ-45</td>
<td>N or Piercing</td>
<td>BNC</td>
</tr>
</tbody>
</table>
Loopback. During transmission without collision, the loopback function of the 10BASE-T MAU will also route the transmitted data back to the DTE on the receive circuit. This function mimics the natural data loopback which occurs in coax MAUs.

Link Integrity. The link integrity function is a process by which the 10BASE-T MAU can determine if its receiver is properly connected to a compatible MAU. If it is not, it disables its transmitter, receiver, and loopback functions. This prevents a one-way link failure from indefinitely disrupting the network, since the carrier sense function is dependent on the receiver. When a bad link disables a MAU’s carrier sense function, it removes itself from the network.

The link integrity function is accomplished by two independent and asynchronous activities—one for the transmitter and one for the receiver. The transmitter will fill long periods of idle with link test pulses (link beats). A link beat is transmitted after every 8 to 24 ms of silence. This defines a maximum period of silence the remote receiver will experience regardless of network traffic.

The receiver monitors its circuit for data packet and link beat reception from the remote MAU transmitter. If an excessively long period of silence occurs, the MAU will disable its receiver, transmitter, and loopback functions. Link beats are still transmitted in this mode. Once data or link beat reception resumes, the MAU reenables all its functions.

2.1.2 MULTIPLE PORT REPEATER

The repeater is the central point in the star configured network. It is usually located in a telephone closet or other central wiring point. The link segments (repeater to node or repeater to repeater connections) are then made by using available twisted pairs in the existing telephone cable plant or a dedicated cable plant. The repeater must conform to the ANSI/IEEE 802.3c-1988 standard for repeaters. It can have any number of dedicated 10BASE-T, 10BASE2, FOIRL, or 10BASE5 ports, and it can have any number of attachment unit interface (AUI) ports. The AUI ports are DTE type (DB-15 female receptacle) and can be connected to any valid 802.3 MAU. All dedicated 10BASE-T ports must support the same functions as the 10BASE-T MAU and normal repeater port functions.

The repeater supports autopartitioning and jabber protection. These two features prevent faulty nodes from taking down the entire network. The autopartition algorithm monitors ports for consecutive collisions such as would happen if a coax segment (10BASE5 or 10BASE2) were left unterminated or if the Tx and Rx twisted pairs were shorted together on a 10BASE-T segment. Once identified, that port is removed from the network until the fault condition is removed; this allows the remainder of the network to operate normally.
The jabber function of the repeater monitors the length of incoming data. If it detects an abnormally long frame it breaks it into legal lengths by inserting minimum interframe spaces on its transmitted signal. This prevents any jabber condition from being repeated onto other segments. Repetition of the jabber condition would allow its own, and other MAUs, to enter a fail state due to faults at a remote location, thus preventing normal operation of the network after the fault condition is removed. With the repeater's jabber protection, network operation resumes after the fault is removed.

2.1.3 DATA TERMINAL EQUIPMENT

DTEs include the user nodes, file servers, bridges, and other entities that can originate and accept data packets on the network. DTEs contain the medium access control (MAC) and physical layer signaling (PLS) sublayers. A DTE can also contain an embedded MAU. DTEs that do not have an embedded MAU have an AUI connector. DTEs with embedded MAUs have the medium dependent interface connector for that particular MAU (RJ-45 for 10BASE-T and BNC for 10BASE2). The MAC functions are handled by the LAN controller (Intel's 82586, 82590, 82592, or the 82596 family). The PLS functions are handled by the serial interface component (82C501AD) or a combination PLS/MAU device (82504TA and 82521TB). This architecture represents a continuity of design for migration from Ethernet/Cheapernet designs to Twisted Pair Ethernet. Only the MAU part of the design needs to be updated. This is 100% software independent.

2.1.4 LINK SEGMENTS

A 10BASE-T link segment connects two twisted pair MAUs; it comprises two medium dependent interface connectors (RJ-45 and 8-pin, standard telephone plugs), two pairs of twisted pair wire (not to exceed 100 m), and a crossover. The connector's pin assignments are shown in Table 2.

### TABLE 2. MDI CONNECTOR PIN ASSIGNMENTS

<table>
<thead>
<tr>
<th>Pin</th>
<th>Signal</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Transmit Data + (TD+)</td>
</tr>
<tr>
<td>2</td>
<td>Transmit Data − (TD−)</td>
</tr>
<tr>
<td>3</td>
<td>Receive Data + (RD+)</td>
</tr>
<tr>
<td>4</td>
<td>Not Used</td>
</tr>
<tr>
<td>5</td>
<td>Not Used</td>
</tr>
<tr>
<td>6</td>
<td>Receive Data − (RD−)</td>
</tr>
<tr>
<td>7</td>
<td>Not Used</td>
</tr>
<tr>
<td>8</td>
<td>Not Used</td>
</tr>
</tbody>
</table>

The crossover function connects the TD outputs of one MAU to the RD inputs of the other. This function can be external or embedded within a MAU. If the function is embedded the signal names on the connector refer to the remote MAU. That is, pin 1 (TD+) on a MAU with an embedded crossover is connected to the Transmit Data (+) of the remote MAU and to its own Receive Data (+). The crossover function is defined by the following connections between MAU A and MAU B shown in Table 3.

### TABLE 3. MAU A AND MAU B CONNECTIONS

<table>
<thead>
<tr>
<th>Pin</th>
<th>MAU A</th>
<th>MAU B</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>TD+(A)</td>
<td>RD+(B)</td>
</tr>
<tr>
<td>2</td>
<td>TD−(A)</td>
<td>RD−(B)</td>
</tr>
<tr>
<td>3</td>
<td>RD+(A)</td>
<td>TD+(B)</td>
</tr>
<tr>
<td>6</td>
<td>RD−(A)</td>
<td>TD−(B)</td>
</tr>
</tbody>
</table>

When an embedded crossover function is used in a DTE-to-repeater connection the crossover is usually embedded in the repeater MAU. In general, repeater MAUs have an embedded crossover and DTE MAUs do not. With proper use of the crossover function, repeaters can be cascaded through twisted pair ports and two DTEs can be connected in a point-to-point network. Repeaters can be cascaded two ways. First, one or more twisted pair ports on a repeater can be designed to have a switched (optional) crossover function. This allows a DTE connection on that port when the crossover is active; or a repeater connection, when the crossover is disabled. Secondly, two twisted pair ports with embedded crossovers can be connected by using a third external crossover.

2.2 Integrating with Existing 802.3 Networks.

Because 10BASE-T networks are fully compatible with existing 802.3 networks at the medium access control and physical layer signaling sublayers, 10BASE-T networks can be integrated with existing 802.3 networks to form one large network. The IEEE standard for repeaters allows connecting different wire types in 10-Mb/s baseband networks. This is because the repeater definition stops at the AUI connection (DTE sex). The wire type is determined by the MAU attached to the AUI connector, and can vary from port to port. Optionally, a repeater can have embedded MAUs on any of its ports. The only requirement for an embedded MAU is that functionality at the medium dependent interface point (e.g., coax tap or twisted pair connector) be maintained as if the MAU were external.

The 82505TA Multiport Repeater Controller provides embedded MAUs on 11 of its 12 ports, and an AUI connection on the remaining port. This allows creating local twisted pair subnetworks connected to an Ethernet backbone. Care must be taken not to violate the following system topology rules of 802.3 networks.
• Only one active signal path is allowed between any two stations on the network.
• No more than four repeaters are allowed in the signal path between any two stations on the network.
• There is an overall limit of 1024 stations on a network (repeaters do not count as stations).

2.3 Software Compatibility

Because the 10BASE-T definition is restricted to the MAU, software is not affected. Twisted pair networks use the same LAN controller chips (82586, 82590, 82592, and the 82596 family) as current Ethernet and Cheapernet networks and are fully software compatible.

3.0 NETWORK SYSTEM COMPONENT DESIGN

The design of various 10BASE-T network system components is presented in this section. First, DTEs with embedded MAUs, then external MAUs, and lastly repeater designs.

3.1 Designing a DTE Node Based on the 82521TB Serial Supercomponent.

A design for an 82521TB based DTE node with an embedded MAU is shown in Figure 3. It includes all of the functions described in Section 2.1.1, thereby relieving the designer of those responsibilities. It is simple to use and it does not require mastering pole-zero diagrams. It is a direct interface from the Ethernet controller to the RJ-45 connector. Implementation of the Clear to Send (CTS) signal is optional.

1-577
Figure 3. 82521TB SSC Interface

Figure 4. DTE Based on 82506TB
3.2.1 HOST TO ETHERNET LAN CONTROLLER INTERFACE

The 82586 Ethernet LAN Controller interface is included in the Intel application note, AP-274.

3.2.2 THE ETHERNET LAN CONTROLLER TO 82C501AD INTERFACE

The 82C501AD to Ethernet controller interface consists of the direct connection of TxC, TxD, RxC, RxD, RTS (TEN), CRS, and CDT signals to the controller.

A 20-MHz ± 0.01% crystal, or crystal oscillator, is recommended for clocking the 82C501AD. Many crystals that meet the requirements of the 82C501AD are available commercially and are listed in the 82C501AD data sheet.

3.2.3 THE 82C501AD TO 82506TB INTERFACE

The 82C501AD to 82506TB interface is a standard AUI. It includes an isolation transformer to provide dc common mode isolation.

3.2.4 THE 82506TB TO ANALOG FRONT-END INTERFACE (AFE)

The four transmit outputs of the 82506TB (HDAT, LDAT, HDAT, and LDAT) and two receive inputs (RD and RD) are connected to the analog front-end described in Section 4. The analog front-end handles preconditioning voltage summation, EMI filtering, and isolation.

3.3 Using the 82523TB Twisted Pair Ethernet MAU to Convert Existing Ethernet Nodes to 10BASE-T

Any Ethernet node that supports the AUI (DB-15 connector) can be converted to a 10BASE-T node by using the 82523TB MAU. A standard AUI cable is used to connect the DB-15 connectors on the node and the 82523TB. The 10BASE-T cable then plugs into the RJ-45 receptacle and the node is connected. The interface is shown in Figure 5.

3.4 Designing a MAU Using the 82506TB Twisted Pair MAU Chip

Figure 6 shows a stand-alone MAU design using the 82506TB. It contains the same AUI transformer, 82506TB, analog front-end, and connector as the embedded design. The serial interface functions and MAC services are contained in a separate DTE. The stand-alone MAU functions include carrier sense, collision detection, link integrity, jabber protection, twisted pair line driving, and line receiving. The analog front-end handles the preconditioning summation, filtering, balancing, and isolation requirements of 10BASE-T.

3.5 Designing a Multiple Port Repeater Based on the 82505TA MPR

Figure 7 shows an MPR based on the 82505TA (with one 82504TA). It contains 11 twisted pair ports with embedded MAUs and 1 AUI port. The 82505TA controls the operation of the repeater in accordance with ANSI/IEEE 802.3c-1988 repeater unit specifications, which include signal retiming, automatic preamble generation, autopartitioning, and jam signal generation. The 82504TA does Manchester decoding and clock recovery while an incoming signal is active. Two addressable latches (74LS259s) are used to control the 16 LED indicators. A 4-to-16 decoder (74LS154) is used to disable the transmitter of the receiving port during transmission without contention. The twisted pair port functions contain the line drivers, the line receivers, the filter, and the isolation required for a twisted pair embedded MAU. In addition, one AUI interface is present to provide access to existing (IEEE 802.3) 10-Mb/s baseband segments.

3.5.1 82505TA TO 82504TA INTERFACE AND CLOCK GENERATION

The 82505TA to 82504TA interface, shown in Figure 4, is straightforward. It consists of six signals directly connected between the devices. The signals are TRxD, TPS, MCV, CRS, RxC, and RxD. The 82504TA performs the Manchester decoding and clock recovery for the repeater.

A single clock oscillator is recommended for clocking the 82505TA and 82504TA. The requirements are identical to those shown for the DTE design using the 82504TA.

• Frequency Tolerance. ≤0.01%
• Rise and Fall Times. ≤5 ns
• Duty cycle. 60/40% or better
• Output. TTL compatible
Figure 6. MAU Based on 82506TB

Figure 7. Repeater Design Based on 82505TA
3.5.2 TWISTED PAIR PORT DESIGN

The twisted pair port circuit, shown in Figures 8 and 9, provides the 10BASE-T functions for each twisted pair port. It uses the same AFE (see Section 4) as the 82506TB designs.

82505TA to AFE Transmit Interface. This interface converts the four transmit signals (TRMT, TRMT, PDC, and TPEN) from the 82505TA, and the port enable (PEx) signal from the port disable control, to the four transmit signals of the analog front-end (TDH, TDL, TDH, and TDL). The design shown here uses an octal line driver (74ACT244) with the drivers paired.

The circuit operates as follows. The TRMT and TRMT signal are XOR'd with the PDC signal to generate the proper logic signals for preconditioning. These signals are then gated by TXE (Transmit enable), a function of the TPE and the PEx signal. The signals are blocked whenever the repeater is idle, the Port Enable for this port is false, or the 10BASE-T state machines have disabled transmission. The gated signals then pass through the link beat generator. Link beats are broadcast during long periods of idle. The signals now pass to the 74ACT244 line drivers, through the AFE, and onto the twisted pair.

Line Receivers. The incoming receive signal passes through the AFE into a gated line receiver controlled by the squelch circuitry. The line receiver converts the received differential signal to TTL levels and feeds it to the MPR. The receiver can be designed with a zero crossing detector (e.g., NE521) and is gated with the TCSx signal.

State Machines. The state machines are required by the 10BASE-T draft standard; they handle link integrity and jabber protection. The implementation shown uses an Intel 5C180 EPLD. The equations are in iPLSII format and are in the appendix.

Squelch Circuit. The squelch circuit distinguishes noise from valid link beats and incoming data on the receive pair by detecting signals above a preset voltage level. When there is no signal on the receive pair the squelch circuit deasserts the CARR and LINK signals. Link is asserted when a signal above the threshold arrives. CARR is asserted if the required number of bits are seen. This causes TCSx to assert and the line receiver to be enabled.

3.5.3 AUI PORT

The AUI port circuitry is shown in Figure 10. It comprises interface circuits, the DO line drivers, two quad D-flip-flops (74F175s), and terminated line receivers for the DI (squelch and data) and CI (squelch only) circuits.

The CI squelch line receiver feeds the D-0 and clear inputs for one of the quad D-flip-flop circuits. When a signal greater than the squelch offset is seen, the flip-flops are cleared and AUICDT is asserted. This continues as long as CI is active. Squelch receiver output is held high during the start of idle and the flip-flops settle in sequence. After four clocks, 150 to 200 ns, the last flip-flop is set and AUICDT deasserts. It remains deasserted during the entire idle period.

The DI line receivers work in much the same way, except that activity on CI, or an active transmission, will inhibit AUICRS. The data channel on DI is processed without a voltage offset and is gated by AUICRS. This way, the least amount of jitter is added on the AUIRxD line and the data channel is not sensitive to idle noise.

The DO line drivers are controlled by TPEN and PE11; the drivers should activate when both are asserted. A voltage divider is provided after the drivers to achieve the proper driver levels.

3.5.4 PORT DISABLE CONTROL

Port disable control is handled by a 74LS154 4-to-16 decoder. During transmission without contention the address of the originating port is given to the decoder and the control line asserted. This disables the transmitter to that port. When a transmit based collision occurs the control line to the decoder is deasserted and jam is broadcast on all ports.

3.5.5 LED CONTROL

Two 8-bit, addressable latches (74LS259s) handle this function. The controller cycles through the addresses for the LEDs each 105 ms and will turn each one on or off. The three least significant address bits (L2–L0) for the LED control are fed to each 8-bit latch. The most significant address bit (L3) controls the enable line to the two packages. The LEDCTRL signal determines the state of the LED when it is strobed by LEDSTRB.
Figure 8. Repeater 10BASE-T Interface Port

Figure 9. Squelch Circuit for 10BASE-T Port
Figure 10. AUI Port
4.0 ANALOG FRONT-END

A discrete version of the analog front-end is shown in Figure II. This preliminary design filter has been extensively tested, and interoperability results were positive. The design will meet most of the 10BASE-T requirements with the following marginalities.

<table>
<thead>
<tr>
<th>10BASE-T PARAMETER</th>
<th>SPEC</th>
<th>MEASURED</th>
</tr>
</thead>
<tbody>
<tr>
<td>RX Return Loss</td>
<td>19 dB</td>
<td>17.5 dB</td>
</tr>
<tr>
<td>TX Return Loss</td>
<td>19 dB</td>
<td>17.5 dB</td>
</tr>
<tr>
<td>Squelch at 10 MHz</td>
<td>585 mV</td>
<td>730 mV</td>
</tr>
</tbody>
</table>

A fully characterized analog front-end design will be ready in October of 1990, and will be available through Intel sales offices. The analog front-end consists of two main sections: transmit and receive. The transmit section contains the preconditioning voltage summing circuit, high voltage protection, the EMI filter, and the line coupling devices. The receive section consists of the line coupling devices, the EMI filter, and high-voltage protection.

A filter pack implementation is shown in Figure 12. Many designs are using this approach but this design has not yet been tested by Intel. A final design version will also be ready in October of 1990, and will be available through Intel sales offices.

4.1 Preconditioning Voltage Summer

The twisted pair output drivers are configured into a single matched impedance differential driver with the preconditioning voltage summing circuit. This circuit is designed to give a preconditioned differential signal. During “thin” pulses, and the first half of “fat” pulses, the differential driver provides 100% drive level power. During the second half of “fat” pulses it provides only 33% drive level as required by the preconditioning algorithm.

This circuit provides a constant source impedance whenever power is applied; this controls matching the driver impedance to the twisted pair cable impedance. This will limit reflections that would result in excessive noise.

4.2 High-Voltage Protection

Protection should be provided to prevent the active devices being damaged by high-voltage transients from the twisted pair line. We recommend placing a pair of diodes on each of the four differential signals (two transmit and two receive) as shown in the analog front-end schematic of Figure II. The diodes connect to \( V_{CC} \) and \( V_{EE} \) (or ground). These should be placed at the interface between the active devices and the low pass filters so the active circuits are protected and the filter attenuates the transients.

4.3 EMI Filter

The main function of the low pass filter is to remove the high-frequency components of the transmitted signal without affecting the in-band (5 to 10 MHz) frequencies. The high frequency components can create electromagnetic interference (EMI) above the levels permitted by FCC regulations. The design should provide minimum in-band loss, ripple, and distortion while providing maximum attenuation of frequencies above 30 MHz with appropriate roll-off in the transition band.

We recommend a filter with the following characteristics:

- Filter type: 5-pole elliptical
- Pass band: 5 to 10 MHz
- Attenuation \( \geq 30 \) MHz: 27 dB
- In-band insertion loss: 1 dB
- In-band ripple: \(<0.5 \) dB (5 to 10 MHz)
- Group delay: \(<50 \) ns
- Group delay variation: \(<5 \) ns
- Impedance: 100 \( \Omega \) balanced

The impedance of the filter must be matched to both the transmitter impedance and the line impedance. Also, balance and grounding should be tightly controlled for proper operation. Due to these considerations we recommend a differential filter built symmetrically on each line of the differential pairs with the impedance matched at each end.

4.4 Line Coupling Devices

The line coupling devices include the transformers and common mode choke. The transformers provide ac coupling between the line and the circuitry while providing dc isolation. The recommended minimum isolation is 2250 \( V_{DC} \). The windings should be identical to provide proper balance between the two ends of the transformers. To provide appropriate impedance matching in the frequency range of interest, the transformers should have appropriate primary and secondary inductance (200 \( \mu \)H typical) and minimal interwinding capacitance (\(<20 \) pF).
Figure 11: Analog Front-End

Component values should be valid over the frequency range from 1 to 100 MHz.

R in Ohms (1/4 W + 1%)
C in pF (NPO + 5%)
L in uH (+5%)
D SANYO SB05-05CP or equiv.

T1, T2, and T3 are commercially available from Sprague, Valor, Pulse Engineering and others.
The common mode choke rejects common mode radio frequency and electromagnetic interference picked up from the unshielded telephone lines. It should provide 1000-V_{dc} isolation between the windings. The common mode choke has four windings, each connected with proper polarity, in series with the receive and transmit twisted pairs. The balance of the choke is very important for providing proper noise cancellation while passing through the differential signal unaffected. We recommend a common mode to differential balance of 30 dB (measured according to the 10BASE-T draft specifications) at all frequencies up to 20 MHz.

4.5 Layout Considerations

The power and ground wiring should conform to good high-frequency practice and standards to minimize switching transients and parasitic interaction between various circuits. To achieve this, the following guidelines are presented.

- Place bypass capacitors (0.1 and 0.001 μF should be interspersed) on each IC between V_{CC} and ground. They should be located close to the V_{CC} pins.
- Make power supply and ground traces as thick as possible. This will reduce high-frequency cross coupling caused by the inductance of thin traces.
- Clock traces, and other high-frequency traces, should be have a width of at least twice the separation between the trace and the nearest ground plane.
- Connect logic and chassis ground together.
- Separate and decouple all of the analog and digital power supply lines.
- Close signal paths to ground as close as possible to their sources to avoid ground loops and noise cross coupling.
- Connect all unused IC inputs (except as directed by the manufacturer) to ground or V_{CC} to avoid noise injection or parasitic oscillations of unused circuits.
- Use high-loss magnetic beads on power supply distribution lines.
- Group each of the receive and transmit circuits, but keep them separate from each other. Separate their grounds.
- Lay out all differential circuits symmetrically so parasitic effects are also symmetrical.
- Lay out the circuitry from the line connector to the active circuitry (especially the EMI filter) on a ground plane to prevent undesirable EMI effects.
5.0 SUMMARY

This application note presented several designs meeting the IEEE 802.3 10BASE-T draft standard 10BASE-T. They use standard telephone twisted pair wiring and a star configuration for cost savings and flexibility. They use the same IEEE 802.3 standard for CSMA/CD medium access control and, where applicable, the physical layer signaling. This network type is fully software compatible with, and can connect to present Ethernet or Cheapernet networks. The hardware connection is made through an 802.3 defined AUI port and by complying with the repeater standard ANSI/IEEE 802.3c-1988.

Intel has introduced four products for designing network components (DTEs and repeaters). DTE design can be done with either the 82521TA Serial Supercomponent or the 82506TB TP Transceiver Chip. The supercomponent contains all the circuitry required between the Ethernet controller and the RJ-45 connector. DTEs that support the AUI can be instantly connected to 10BASE-T networks using the 82523TB MAU supercomponent. Multiple port repeaters can be designed using the 82505TA with an 82504TA. It allows for 11 twisted pair ports and 1 AUI port.
The following set of equations for programming the 5C180 for the 10BASE-T port design of a multiple-port repeater were generated from a proven 10BASE-T design. However, they were not verified in their present state. They are intended to serve as an example to aid in the development of a complete design.

They are written in iSTATE format, a state machine compiler to be used in conjunction with IPLSII.

```plaintext
Thom Bowns/Bill Wager
Intel
Feb. 20, 1990
Repeater 180
Rev 0
5C180
5C180 for Multiple Port Repeater- Counters, State machines, etc.
OPTIONS: TURBO=ON

PART: 5C180

INPUTS: TXC', LID, WDTD, TPEN#, PEx,
        TEST, CARR, LINK, RESET

OUTPUTS: TCSx, LKB#T, TXE, WDT,
          SLOCLK, LI

  JT14_RESET, JT58_RESET, LT_RESET, SCQ01, SCQ02,
  SCQ03, SCQ04,
  SCQ05, SCQ06, SCQ07, SCQ08, SCQ09, SCQ10,
  SCQ11, SCQ12, SCQ13, DLINK, JTQ1, JTQ2,
  JTQ3, JTQ4, JTQ5, JTQ6, JTQ7, JTQ8,
  LTQ1, LTQ2, LTQ3, LTQ4, LTQ5, LCQ1, LCQ2,
  JFQ1, LIFQ1, LIFQ2

NETWORK:

  TXCn = INP (TXC')
  LID = INP (LID)
  WDTD = INP (WDTD)
  TPENn = INP (TPEN#)
  TEST = INP (TEST)
  CARRIER = INP (CARR)
  SLINK = INP (LINK)
  RESET = INP (RESET)
  PEx = INP (PEx)

  TCSx = CONF (TCSx, VCC)
  LKB#T = RONF (ILBEAt, TXCn, GND, GND, VCC)
  LKB#T = NORF (ILBEAT, TXCn, GND, GND)
  TXE = CONF (TXE, VCC)
```
% Timer reset feedback macrocells %
JT14_RESET = NOCF(JT14_RESETd)
JT58_RESET = NOCF(JT58_RESETd)
LT_RESET = NORF(LT_RESETd, TXCn, GND, GND)

% SLOCLK macrocells %
SCQ01 = NOTF(SCT01, TXCn, RESET, GND)
SCQ02 = NOTF(SCT02, TXCn, RESET, GND)
SCQ03 = NOTF(SCT03, TXCn, RESET, GND)
SCQ04 = NOTF(SCT04, TXCn, RESET, GND)
SCQ05 = NOTF(SCT05, TXCn, RESET, GND)
SCQ06 = NOTF(SCT06, TXCn, RESET, GND)
SCQ07 = NOTF(SCT07, TXCn, RESET, GND)
SCQ08 = NOTF(SCT08, TXCn, RESET, GND)
SCQ09 = NOTF(SCT09, TXCn, RESET, GND)
SCQ10 = NOTF(SCT10, TXCn, RESET, GND)
SCQ11 = NOTF(SCT11, TXCn, RESET, GND)
SCQ12 = NOTF(SCT12, TXCn, RESET, GND)
SCQ13 = NOTF(SCT13, TXCn, RESET, GND)
SLOCLK, SCQ14 = TOTF(SCT14, TXCn, RESET, GND, VCC)
SLOCK = CLKB(SCQ14)
DLINK = NORF(SLINK, TXCn, RESET, GND)

% Jabber timer macrocells %
JTQ1 = NOTF(JTT1, SLOCK, JT14_RESET, GND)
JTQ2 = NOTF(JTT2, SLOCK, JT14_RESET, GND)
JTQ3 = NOTF(JTT3, SLOCK, JT14_RESET, GND)
JTQ4 = NOTF(JTT4, SLOCK, JT58_RESET, GND)
JTQ5 = NOTF(JTT5, SLOCK, JT58_RESET, GND)
JTQ6 = NOTF(JTT6, SLOCK, JT58_RESET, GND)
JTQ7 = NOTF(JTT7, SLOCK, JT58_RESET, GND)
JTQ8 = NOTF(JTT8, SLOCK, JT58_RESET, GND)

% Link test timer macrocells %
LTQ1 = NOTF(LTT1, SLOCK, LT_RESET, GND)
LTQ2 = NOTF(LTT2, SLOCK, LT_RESET, GND)
LTQ3 = NOTF(LTT3, SLOCK, LT_RESET, GND)
LTQ4 = NOTF(LTT4, SLOCK, LT_RESET, GND)
LTQ5 = NOTF(LTT5, SLOCK, LT_RESET, GND)

% Link count macrocells %
LCQ1 = NOTF(LCT1, TXCn, RESET, GND)
LCQ2 = NOTF(LCT2, TXCn, RESET, GND)
EQUATIONS:

% Input controlled variables %
output_active = !(TPENn + !PEx);
output_idle = TPENn + !PEx;
input_active = CARRIER;
input_idle = !CARRIER;
link_test_rcvd = SLINK*!DLINK;

% Output equations %
TCSx = input_active * !disable_receiver;
ILBEAT = linkbeat_timer_done;
LBEATn = !(ILBEAT * !LKBTI);
TKE = !disable_driver * output_active;
disable_driver = disable_driver_3 + disable_driver_4;

% Miscellaneous equations %
link_count_is_3 = LCQ2 * LCQ1;
LCRESET = LINK_TEST_FAIL_RESET * !link_test_rcvd;
LCCOUNT = LINK_TEST_FAIL_RESET * link_test_rcvd +
          LINK_TEST_FAIL * link_test_rcvd *
link_test_min_timer_done;
LCT1 = LCCOUNT + LCRESET * LCQ1;
LCT2 = LCCOUNT * LCQ1 + LCRESET * LCQ2;

% SLOCLK counter equations %
SCT01 = VCC;
SCT02 = SCQ01;
SCT03 = SCT02 * SCQ02;
SCT04 = SCT03 * SCQ03;
SCT05 = SCT04 * SCQ04;
SCT06 = SCT05 * SCQ05;
SCT07 = SCT06 * SCQ06;
SCT08 = SCT07 * SCQ07;
SCT09 = SCT08 * SCQ08;
SCT10 = SCT09 * SCQ09;
SCT11 = SCT10 * SCQ10;
SCT12 = TEST + !TEST * SCT11 * SCQ11;
SCT13 = SCT12 * SCQ12;
SCT14 = SCT13 * SCQ13;

% JABBER timer equations %
transmit_timer_done = JTQ5 * NON_JABBER_OUTPUT;
unjab_timer_done = JTQ8 * JTQ7 * UNJAB_WAIT;
unjab_timer_not_done = !unjab_timer_done;
linkbeat_timer_done = JTQ3 * !NON_JABBER_OUTPUT * !LID;
JT14_RESETd = RESET + NO_OUTPUT * output_active
                + NON_JABBER_OUTPUT * output_idle;
JT58_RESETd = JT14_RESETd
                + JAB * output_idle;
JTT1 = VCC;
JTT2 = JTT1 * JTQ1;
JTT3 = JTT2 * JTQ2;
JTT4 = JTT3 * JTQ3;
JTT5 = JTT4 * JTQ4 * JTCOUNT;
JTT6 = JTT5 * JTQ5;
JTT7 = JTT6 * JTQ6;
JTT8 = JTT7 * JTQ7;
JTCOUNT = !(JTQ5 * JTQ6 * JTQ7 * JTQ8);

% LINK timer equations %
ltime loss_timer_done = LTQ5 * LTQ4 * LTQ3 * LTQ2 * LTQ1 * IDLE_TEST;
ltime test_min_timer_done =
    (LTQ3 + LTQ4 + LTQ5) * ((IDLE_TEST) + (LINK_TEST_FAIL));
ltime test_min_timer_not_done = !ltime test_min_timer_done;
ltime test_max_timer_done = LTQ5 * LINK_TEST_FAIL;
ltime resetd = IDLE_TEST * (input active
    + (ltime test_rcvd * ltime test_min_timer_done))
    + LINK_TEST_FAIL_RESET
    + LINK_TEST_FAIL * ltime test_min_timer_done
    * ltime test_rcvd
    + LINK_TEST_FAIL_EXTEND;

LTT1 = LTCOUNT;
LTT2 = LTCOUNT * LTQ1;
LTT3 = LTCOUNT * LTQ1 * LTQ2;
LTT4 = LTCOUNT * LTQ1 * LTQ2 * LTQ3;
LTT5 = LTCOUNT * LTQ1 * LTQ2 * LTQ3 * LTQ4;
LTCOUNT = !(LTQ1 * LTQ2 * LTQ3 * LTQ4 * LTQ5);
% ----------------------------------------------- %
MACHINE: JABBER_FUNCTION
CLOCK: TXCn
CLEAR: RESET

states: [ WDT JFQ1 ]  % JFQ2 is WDT also %
NO_OUTPUT [ 0 0 ]
NON_JABBER_OUTPUT [ 0 1 ]
    JAB [ 1 1 ]
UNJAB_WAIT [ 1 0 ]

NO_OUTPUT: IF output_active THEN NON_JABBER_OUTPUT

NON_JABBER_OUTPUT: % start_transmit_timer %
    IF (output_active * transmit_timer_done * !WDTD)
    THEN JAB
    IF output_idle THEN NO_OUTPUT

JAB: IF output_idle THEN UNJAB_WAIT
    IF output_active * WDTD THEN NO_OUTPUT
    ASSERT: disable_driver_3 % ASSERT WDT here %

UNJAB_WAIT: % start_unjab_timer %
    IF output_active * unjab_timer_not_done * !WDTD
    THEN JAB
    IF unjab_timer_done + WDTD THEN NO_OUTPUT
    ASSERT: disable_driver_3 % ASSERT WDT here %
% -------------------------------------------------------- %
MACHINE: LINK_INTEGRITY_FUNCTION
CLOCK: TXCn
CLEAR: RESET

STATES: [ LI LIFO1 LIFO2 ]
         [ 0   0   0 ]
IDLE_TEST [ 1   1   0 ]
         [ 1   0   0 ]
LINK_TEST_FAIL_RESET [ 0   0   1 ]
         [ 0   1   1 ]
LINK_TEST_FAIL [ 0   1   1 ]
         [ 0   1   1 ]
LINK_TEST_FAIL_EXTEND [ 0   1   1 ]
         [ 0   1   1 ]
ALIA [ 0   1   1 ]
         [ 0   1   1 ]
ALIB [ 0   1   0 ]
         [ 0   1   0 ]
ALIC [ 1   0   1 ]
         [ 1   0   1 ]
ALID [ 1   1   1 ]
         [ 1   1   1 ]

IDLE_TEST: % start_link_loss_timer %
% start_link_test_min_timer %
IF link_loss_timer_done * !LID THEN LINK_TEST_FAIL_RESET

LINK_TEST_FAIL_RESET: IF link_test_rcvd * !LID * input_idle
THEN LINK_TEST_FAIL
IF input_active + LID THEN LINK_TEST_FAIL_EXTEND
ASSERT: disable_receiver
        disable_driver_4

LINK_TEST_FAIL: % start_link_test_min_timer %
% start_link_test_max_timer %
IF input_active + link_count_is_3 + LID
THEN LINK_TEST_FAIL_EXTEND
IF (link_test_max_timer_done
  + (link_test_min_timer_not_done
     * link_test_rcvd)) * !LID * input_idle
THEN LINK_TEST_FAIL_RESET
ASSERT: disable_driver_4
        disable_receiver

LINK_TEST_FAIL_EXTEND: IF input_idle * output_idle THEN IDLE_TEST
ASSERT: disable_driver_4
        disable_receiver

ALIA: IDLE_TEST
ALIB: IDLE_TEST
ALIC: IDLE_TEST
ALID: IDLE_TEST

END$
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.

Figure 1. PC586E Block Diagram

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.
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 buses 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 Cheaper (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 Cheaper (IEEE 802.3 10BASE2) 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) Cheaver (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 though 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.

The 82501 provides the Ethernet (IEEE 802.3 10BASE5) or Cheapernet (IEEE 802.3 10BASE2) Serial Interface for the 82586 LAN Coprocessor. Major functions of the 82501 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 82501 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).

**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

**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
- 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-FFFFFh

**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-86, 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:

- 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.

<table>
<thead>
<tr>
<th>C/D</th>
<th>RD</th>
<th>WR</th>
<th>CS</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>8251A DATA → DATA BUS</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>DATA BUS → 8251A DATA</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>STATUS → DATA BUS</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>DATA BUS → CONTROL</td>
</tr>
<tr>
<td>X</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>DATA BUS → 3-STATE</td>
</tr>
<tr>
<td>X</td>
<td>X</td>
<td>X</td>
<td>1</td>
<td>DATA BUS → 3-STATE</td>
</tr>
</tbody>
</table>

**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.

1 = CONTROL/STATUS; 0 = DATA.
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 may 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.

TxD (Transmitter Ready)
This output signals the CPU that the transmitter is ready to accept a data character. The TxD 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 TxD using a Status Read operation. TxD 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 TxD 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 TxE output will go “high”. It resets upon receiving a character from CPU if the transmitter is enabled. TxE remains high when the transmitter is disabled. TxE 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”. TxE does not go low when the SYNC characters are being shifted out.
**8251A**

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 TxC frequency. In Asynchronous transmission mode, the baud rate is a fraction of the actual TxC frequency. A portion of the mode instruction selects this factor; it can be 1, $\frac{1}{16}$ or $\frac{1}{64}$ the TxC.

For Example:

If Baud Rate equals 110 Baud,
- TxC equals 110 Hz in the 1x mode.
- TxC equals 1.72 kHz in the 16x mode.
- TxC equals 7.04 kHz in the 64x mode.

The falling edge of TxC shifts the serial data out of the 8251A.

**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.

**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).
**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**

2-6
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**
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 TxD output is raised "high" to signal the CPU that the 8251A is ready to receive a data character from the CPU. This output (TxD) 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 words 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.

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
is raised (thus the previous character is lost). All of the error flags can be reset by an Error Reset Instruction. 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 characters 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 TxC output must continue at the TxC rate. If the CPU does not provide the 8251A with a data character 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.

---

*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.

**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.

---

**Figure 11. Data Format, Synchronous Mode**

---

2-12
**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. TxDY is an exception.

---

### Status Read Format

<table>
<thead>
<tr>
<th>Bit 7</th>
<th>Bit 6</th>
<th>Bit 5</th>
<th>Bit 4</th>
<th>Bit 3</th>
<th>Bit 2</th>
<th>Bit 1</th>
<th>Bit 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>EH</td>
<td>IR</td>
<td>RTS</td>
<td>ER</td>
<td>SBRK</td>
<td>RxE</td>
<td>DTR</td>
<td>TxEN</td>
</tr>
</tbody>
</table>

- **TRANSMIT ENABLE**
  - 1 = enable
  - 0 = disable

- **DATA TERMINAL READY**
  - "high" will force DTR output to zero

- **RECEIVE ENABLE**
  - 1 = enable
  - 0 = disable

- **SEND BREAK CHARACTER**
  - 1 = forces TxD "low"
  - 0 = normal operation

- **ERROR RESET**
  - 1 = reset error flags PE, OE, FE

- **REQUEST TO SEND**
  - "high" will force RTS output to zero

- **INTERNAL RESET**
  - "high" returns 8251A to Mode Instruction Format

- **ENTER HUNT MODE**
  - 1 = enable search for Sync Characters

*(HAS NO EFFECT IN ASYNC MODE)*

---

**Figure 12. Command Instruction Format**

NOTE:
Error Reset must be performed whenever RxEnable and Enter Hunt are programmed.
Note that status update can have a maximum delay of 28 clock periods from the actual event affecting the status.

### Diagram

![Status Read Format Diagram](image)

**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.
   - i.e. TxRDY status bit = DB Buffer Empty
   - TxRDY pin out = DB Buffer Empty • (CTS = 0) • (TxEN = 1)

---

**Figure 13. Status Read Format**
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 \( V_{OH} = 2.0 \text{ V} \) and \( V_{OL} = 0.8 \), 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 TxC and RxC frequencies have the following limitations with respect to CLK: For 1x Baud Rate, \( f_{TX} \) or \( f_{RX} \leq 1/(30 \ t_{CY}) \); For 16x and 64x Baud Rate, \( f_{TX} \) 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:** This is a production data sheet. The specifications are subject to change without notice.

**WARNING:** Stressing the device beyond the "Absolute Maximum Ratings" may cause permanent damage. These are stress ratings only. Operation beyond the "Operating Conditions" is not recommended and extended exposure beyond the "Operating Conditions" 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>
<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>$V_{CC}$</td>
<td>V</td>
<td></td>
</tr>
<tr>
<td>$V_{OL}$</td>
<td>Output Low Voltage</td>
<td>0.45</td>
<td>V</td>
<td></td>
<td>$I_{OL} = 2.2 \text{ mA}$</td>
</tr>
<tr>
<td>$V_{OH}$</td>
<td>Output High Voltage</td>
<td>2.4</td>
<td>V</td>
<td></td>
<td>$I_{OH} = -400 \mu\text{A}$</td>
</tr>
<tr>
<td>$I_{OFL}$</td>
<td>Output Float Leakage</td>
<td>±10</td>
<td>$\mu\text{A}$</td>
<td></td>
<td>$V_{OUT} = V_{CC}$ to 0.45V</td>
</tr>
<tr>
<td>$I_{IL}$</td>
<td>Input Leakage</td>
<td>±10</td>
<td>$\mu\text{A}$</td>
<td></td>
<td>$V_{IN} = V_{CC}$ to 0.45V</td>
</tr>
<tr>
<td>$I_{CC}$</td>
<td>Power Supply Current</td>
<td>100</td>
<td>ma</td>
<td></td>
<td>All Outputs = High</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>Unit</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>$f_c = 1 \text{ MHz}$</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>
</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></td>
<td>3, $C_L = 150 \text{ pF}$</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></td>
<td>(Note 4)</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>$t_{CY}$</td>
<td>Clock Period</td>
<td>320</td>
<td>1350</td>
<td>ns</td>
<td>(Note 5, 6)</td>
</tr>
<tr>
<td>$t_{Ph}$</td>
<td>Clock High Pulse Width</td>
<td>120</td>
<td>90</td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>$t_{Pl}$</td>
<td>Clock Low Pulse Width</td>
<td>90</td>
<td></td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>$t_{R}$, $t_F$</td>
<td>Clock Rise and Fall Time</td>
<td>20</td>
<td></td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>$t_{DTx}$</td>
<td>TxD Delay from Falling Edge of TxC</td>
<td>1</td>
<td>μs</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$t_{TX}$</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>$t_{TPW}$</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>$t_{CY}$</td>
<td></td>
</tr>
<tr>
<td></td>
<td>16x and 64x Baud Rate</td>
<td>1</td>
<td></td>
<td>$t_{CY}$</td>
<td></td>
</tr>
<tr>
<td>$t_{TPD}$</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>$t_{CY}$</td>
<td></td>
</tr>
<tr>
<td></td>
<td>16x and 64x Baud Rate</td>
<td>3</td>
<td></td>
<td>$t_{CY}$</td>
<td></td>
</tr>
<tr>
<td>$f_{Rx}$</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>$t_{RPW}$</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>$t_{CY}$</td>
<td></td>
</tr>
<tr>
<td></td>
<td>16x and 64x Baud Rate</td>
<td>1</td>
<td></td>
<td>$t_{CY}$</td>
<td></td>
</tr>
<tr>
<td>$t_{RPD}$</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>$t_{CY}$</td>
<td></td>
</tr>
<tr>
<td></td>
<td>16x and 64x Baud Rate</td>
<td>3</td>
<td></td>
<td>$t_{CY}$</td>
<td></td>
</tr>
<tr>
<td>$t_{TXRDY}$</td>
<td>TxRDY Pin Delay from Center of Last Bit</td>
<td>14</td>
<td></td>
<td>$t_{CY}$</td>
<td>(Note 7)</td>
</tr>
<tr>
<td>$t_{TXRDY}$</td>
<td>Clearing from Leading Edge of WR</td>
<td>400</td>
<td></td>
<td>ns</td>
<td>(Note 7)</td>
</tr>
<tr>
<td>$t_{RXRDY}$</td>
<td>RxRDY Pin Delay from Center of Last Bit</td>
<td>26</td>
<td></td>
<td>$t_{CY}$</td>
<td>(Note 7)</td>
</tr>
<tr>
<td>$t_{RXRDY}$</td>
<td>Clearing from Leading Edge of RD</td>
<td>400</td>
<td></td>
<td>ns</td>
<td>(Note 7)</td>
</tr>
<tr>
<td>$t_{IS}$</td>
<td>Internal SYNDET Delay from Rising Edge of RxC</td>
<td>26</td>
<td></td>
<td>$t_{CY}$</td>
<td>(Note 7)</td>
</tr>
<tr>
<td>$t_{ES}$</td>
<td>External SYNDET Set-Up Time After Rising Edge of RxC</td>
<td>16 $t_{CY}$</td>
<td>$t_{RPD}$</td>
<td>ns</td>
<td>(Note 7)</td>
</tr>
<tr>
<td>$t_{TXEMPTY}$</td>
<td>TxEMPTY Delay from Center of Last Bit</td>
<td>20</td>
<td></td>
<td>$t_{CY}$</td>
<td>(Note 7)</td>
</tr>
<tr>
<td>$t_{WC}$</td>
<td>Control Delay from Rising Edge of WRITE (TxEn, DTR, RTS)</td>
<td>8</td>
<td>$t_{CY}$</td>
<td>(Note 7)</td>
<td></td>
</tr>
<tr>
<td>$t_{CR}$</td>
<td>Control to READ Set-Up Time (DSR, CTS)</td>
<td>20</td>
<td></td>
<td>$t_{CY}$</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)

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".

WAVEFORMS

SYSTEM CLOCK INPUT
TRANSMITTER CLOCK AND DATA

Rx DATA

TxC (1x MODE)

TxC (16x Mode)

Rx DATA

TxC (1x MODE)

TxC (16x MODE)

TxDATA

RECEIVER CLOCK AND DATA

Rx DATA

Rx DATA

RxC (1x MODE)

RxC (16 MODE)

INT SAMPLING PULSE

RxC (1x MODE)

RxC (16 MODE)

INT SAMPLING PULSE

START BIT

DATA BIT

DATA BIT

(Rx BAUD COUNTER STARTS HERE)

(Rx BAUD COUNTER STARTS HERE)

8 RxC PERIODS (16x MODE)

16 RxC PERIODS (16x MODE)

3 tcy

3 tcy

3 tcy

3 tcy

t0

t0

t0

t0

t0
WAVEFORMS (Continued)

WRITE DATA CYCLE (CPU → USART)

READ DATA CYCLE (CPU ← USART)
WAVEFORMS (Continued)

WRITE CONTROL OR OUTPUT PORT CYCLE (CPU → USART)

![Diagram of write control or output port cycle](image)

**NOTES:**
1. $T_{WC}$ includes the response timing of a control byte.
2. $T_{CR}$ includes the effect of CTS on the TxENBL circuitry.

READ CONTROL OR INPUT PORT (CPU ← USART)

![Diagram of read control or input port](image)

**NOTES:**
1. $T_{WC}$ includes the response timing of a control byte.
2. $T_{CR}$ 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.
**82050 ASYNCHRONOUS COMMUNICATIONS CONTROLLER**

- Asynchronous Operation
  - 5- to 8-Bit Character Format
  - Odd-, Even-, or No-Parity Generation and Detection
  - Serial Bit Rate: DC to 56 Kb/s
- Programmable, 16-Bit Baud Rate Generator
- System Clock
  - On-Chip Crystal Oscillator
  - Externally Generated Clock
- 28-Lead DIP and PLCC Packages
- IBM PC (INS 16450/8250A) Software Compatible
- Seven I/O Pins
  - Dedicated Modem I/O
  - General Purpose I/O
- No-TTL Interface to Most Intel Processors
- Internal Diagnostics with Local Loopback
- Complete Interrupt and Status Reporting
- CHMOS III Technology Provides Increased Reliability and Reduced Power Consumption
- Line Break Generation and Detection

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 requirements. The 82050’s simpler system interface reduces TTL glue—especially for higher frequency PC bus designs. The 82050 provides 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.

![Figure 1. Block Diagram](image-url)
82050 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><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>DCD</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

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 Frequency = External Clock Frequency
   - BRG Source Clock Frequency = 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. During reset, the 82050’s system clock mode of operation is also selected by strapping options on the RTS and DTR pins (see system clock generation).

**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
interrupt source is decoded into the IIR register; the whole procedure is repeated until there are no more pending interrupt sources.

TRANSMIT
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

<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>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</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>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>01H</td>
</tr>
<tr>
<td>LCR</td>
<td>DLAB</td>
<td>Divisor</td>
<td>Latch</td>
<td>Access</td>
<td>Bit 2</td>
<td>Parity Mode</td>
<td>Parity Mode</td>
<td>Bit 1</td>
<td>Stop Bit</td>
<td>Length</td>
</tr>
<tr>
<td>MCR</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>Loopback Control Bit</td>
<td>OUT2</td>
<td>Complement</td>
<td>0</td>
<td>RTS</td>
</tr>
<tr>
<td>LSR</td>
<td>0</td>
<td>TxM</td>
<td>Status</td>
<td>0</td>
<td>TxD</td>
<td>Empty</td>
<td>Break Detected</td>
<td>Framing Error</td>
<td>Parity Error</td>
<td>Overrun Error</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.

```
<table>
<thead>
<tr>
<th>Bit</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>RXDA — RX Data Interrupt Enable</td>
</tr>
<tr>
<td>6</td>
<td>TXDE — TX Data Empty Enable</td>
</tr>
<tr>
<td>5</td>
<td>RXIE — RX Interrupt Enable</td>
</tr>
<tr>
<td>4</td>
<td>MIE — MODEM Interrupt Enable</td>
</tr>
</tbody>
</table>
```

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.

```
<table>
<thead>
<tr>
<th>Bit</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>RX Error Condition (Highest Priority)</td>
</tr>
<tr>
<td>0</td>
<td>RX Character Available</td>
</tr>
</tbody>
</table>
```

B1, B0 — Interrupt Bits, 2-1. These two bits reflect the highest priority, enabled and pending interrupt request.

1: RX Error Condition (Highest Priority)
0: RX Character Available

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>
</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.

![Line Status Register Diagram]

- **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.

![Modem Control Register Diagram]
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 VCC + 0.5V
Voltage on VCC Pin (w.r.t. VSS) ...... -0.5V to +7V
Power Dissipation .................... 300 mW

D.C. CHARACTERISTICS (TA = 0° 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>IIL</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, RTS</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>
<tr>
<td>CXTAL</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. @ IOL = 2 mA.
3. @ IOH = -0.4 mA.
4. 0 < VIN < VCC.
5. 0.45V < VOUT < (VCC - 0.45).
6. VCC = 5.5V; VIH = VCC - 0.5V (max); VOH = VCC - 0.5V (min); IOL = IOH = 0; 9.2 MHz (max).
7. VIH = 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><strong>DIVIDE BY TWO OPTION—ACTIVE</strong></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></td>
<td>(1)</td>
</tr>
<tr>
<td>TCL2CL1</td>
<td>CLK Fall Time</td>
<td>10</td>
<td></td>
<td>(1)</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**

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Parameter</th>
<th>Min</th>
<th>Max</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<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></td>
<td>(1)</td>
</tr>
<tr>
<td>TCL2CL1</td>
<td>CLK Fall Time</td>
<td>15</td>
<td></td>
<td>(1)</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 Tcy</td>
<td></td>
<td>(1)</td>
</tr>
<tr>
<td>TTLRSL</td>
<td>RTS/DTR LOW Setup to Reset Inactive</td>
<td>6 Tcy</td>
<td></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, 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></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 after RD Active</td>
<td></td>
<td>2Tcy + 65</td>
<td>(1)</td>
</tr>
<tr>
<td>TCIAD</td>
<td>Command Inactive to Active Delay</td>
<td></td>
<td>Tcy + 15</td>
<td>(2)</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>

**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></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></td>
<td>90</td>
<td></td>
</tr>
<tr>
<td>TWHDX</td>
<td>Data in Hold Time after WR Inactive</td>
<td></td>
<td>12</td>
<td></td>
</tr>
</tbody>
</table>

### Diagrams

**READ CYCLE**

- RD
- TRLRH
- TAVRL
- TRHAX
- TRLDV
- TCIAD
- TRHDZ

**WRITE CYCLE**

- WR
- TAVWL
- TWLWH
- TCIAD
- TWHAX
- TDVWH
- TWHDX
- RD
The Intel CHMOS 82510 is designed to increase system efficiency in asynchronous environments such as modems or serial ports—including expanding performance areas: MCS-51 9-bit format and high speed async. The functional support provided in the 82510 is unparalleled—two 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 two levels. The multipersonality 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 with CHMOS III technology and includes a software powerdown option.

Figure 1. Block Diagram

*IBM and PC AT are trademarks of International Business Machines.
## 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><strong>RESET</strong>: 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><strong>CHIP SELECT</strong>: 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><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>4*25</td>
<td>I/O</td>
<td><strong>DATA BUS</strong>: 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><strong>READ</strong>: 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><strong>WRITE</strong>: A low on this input pin 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><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 82510 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 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><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, 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. <em>Note</em>: 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><strong>TRANSMIT DATA</strong>: 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>R1/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. R1 - 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>
<tr>
<td>VCC</td>
<td>21</td>
<td>P</td>
<td><strong>VCC:</strong> Device power supply.</td>
</tr>
<tr>
<td>VSS</td>
<td>7</td>
<td>P</td>
<td><strong>VSS:</strong> Ground.</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>
<thead>
<tr>
<th>Table 2. 82510 Register/Block Functions</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>BANK</th>
<th>REGISTER</th>
<th>8250 MODE</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>GENERAL WORK BANK</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>GENERAL CONFIGURATION BANK</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>MODEM CONFIGURATION BANK</td>
<td></td>
</tr>
</tbody>
</table>

Figure 4. 82510 Register Architecture

2-43
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 | TMCR | — |
| 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
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 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)</td>
<td>TMIE (1-0)</td>
<td>Read TMST</td>
</tr>
<tr>
<td></td>
<td>GSR (5)</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>GSR (4)</td>
<td>GER (4)</td>
<td>Write Character to tx FIFO</td>
</tr>
<tr>
<td></td>
<td>LSR (6)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Tx 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>
</tr>
<tr>
<td>RST (7-1)</td>
<td>GSR (2)</td>
<td>GER (2)</td>
<td>Read Character</td>
</tr>
<tr>
<td>Rx Machine</td>
<td>RST/LSR (0)</td>
<td>GER (0)</td>
<td>Write 0 to LSR/RST Bit zero. Read Character</td>
</tr>
<tr>
<td></td>
<td>GSR (0)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Rx FIFO</td>
<td>LSR (5)</td>
<td>GER (1)</td>
<td>Write to FIFO Read GIR(1)</td>
</tr>
<tr>
<td></td>
<td>GSR (1)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Tx FIFO</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>
</tr>
<tr>
<td></td>
<td>GSR (3)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Modem</td>
<td></td>
<td></td>
<td></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.

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 ulan mode, the nine bit protocol of the MCS-51 family is supported and the ulan Address characters, rather than Control Characters are recognized.
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>
</tr>
</thead>
<tbody>
<tr>
<td>A) STANDARD SET</td>
</tr>
<tr>
<td>- ASCII: 000X XXXX + 0111 1111</td>
</tr>
<tr>
<td>(ASCII DEL)</td>
</tr>
<tr>
<td>(00 - 1FH + 7 FH)</td>
</tr>
<tr>
<td>OR</td>
</tr>
<tr>
<td>- EBCDIC: 00XX XXXX</td>
</tr>
<tr>
<td>(00 - 3FH)</td>
</tr>
<tr>
<td>B) User Programmed</td>
</tr>
<tr>
<td>- ACR0, ACR1 XXXX XXXX</td>
</tr>
<tr>
<td>REGISTERS</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.

\[ f_0 = \frac{f_{in}}{\text{Divisor}} \]

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 \( \frac{1}{2} \) crystal frequency or \( \frac{1}{2} \) external clock frequency when using \( \div 2 \) 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 count downs 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:

\[
\text{deviation} = \frac{1}{2 \times \text{divisor}}
\]

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>

delay between the trigger and the terminal count is given by the following equation:

\[
\text{Delay} = \text{Count} \times (\text{System Clock Period})
\]

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.
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

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. 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.
### 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) 8250A/16450</td>
<td>0 (DLAB = 0)</td>
<td>RXD</td>
<td>TXD</td>
</tr>
<tr>
<td></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/Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>TxD (33)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0  —</td>
</tr>
<tr>
<td>RxD (35)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0  —</td>
</tr>
<tr>
<td>BAL (11)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>02H</td>
</tr>
<tr>
<td>BAH (12)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>00H</td>
</tr>
<tr>
<td>GIR/BANK (21)</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>01H</td>
</tr>
<tr>
<td>LCR (2)</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>00H</td>
</tr>
<tr>
<td>MCR (32)</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>00H</td>
</tr>
<tr>
<td>LSR (22)</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>60H</td>
</tr>
<tr>
<td>MSR (27)</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>00H</td>
</tr>
<tr>
<td>ACR0 (5)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>00H</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/Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>TxD (33)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0  —</td>
</tr>
<tr>
<td>RxD (35)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0  —</td>
</tr>
<tr>
<td>RxF (24)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1  —</td>
</tr>
<tr>
<td>TxF (34)</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>01H</td>
</tr>
<tr>
<td>GIR/BANK (21)</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>01H</td>
</tr>
<tr>
<td>TMST (26)</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>30H</td>
</tr>
<tr>
<td>TMCX (31)</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>3  —</td>
</tr>
<tr>
<td>MCR (32)</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></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>Address/Control Character Match</td>
<td>Break Terminated</td>
<td>Break Detected</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>
</tr>
<tr>
<td>RCM (30)</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>5</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>TxM interrupt</td>
<td>Modem Interrupt</td>
<td>RxM Interrupt</td>
<td>Tx FIFO Interrupt</td>
<td>Rx FIFO Interrupt</td>
<td>7</td>
<td>12H</td>
</tr>
<tr>
<td>ICM (28)</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>
<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 Select</td>
<td>Loopback or Echo Mode of Operation</td>
<td>4</td>
<td>0CH</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ACR1 (6)</td>
<td>Address/Control Character</td>
<td>Address/Control Character Match</td>
<td>Interrupt Enable</td>
<td>Disable</td>
<td>Sampling Window Mode</td>
<td>Start bit Sampling Mode</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>7</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>0</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>BBH (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</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>
<td></td>
</tr>
<tr>
<td>(21)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BBCF (10)</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>
<td></td>
</tr>
<tr>
<td>PMD (15)</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>4</td>
<td>FCH</td>
<td></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>5</td>
<td>0FH</td>
<td></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>6</td>
<td>00H</td>
<td></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 wakeup 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>• Automatic</td>
<td>• RX FIFO Depth</td>
<td>• None</td>
</tr>
<tr>
<td>• Manual</td>
<td>• RX, TX Threshold</td>
<td>• Standard</td>
</tr>
<tr>
<td>Receive</td>
<td>• Clock Options</td>
<td>• ASCII</td>
</tr>
<tr>
<td>• Sampling Window Size</td>
<td>• RX, TX Clock Mode</td>
<td>• EBCDIC</td>
</tr>
<tr>
<td>• Start Bit Detection Mode</td>
<td>1X</td>
<td>• Two User Programmed</td>
</tr>
<tr>
<td>• DPLL Disable/Enable</td>
<td>16X</td>
<td></td>
</tr>
<tr>
<td>μLAN (8051)</td>
<td>• RX, TX Clock Source</td>
<td></td>
</tr>
<tr>
<td>Address Recognition</td>
<td>BRGA</td>
<td></td>
</tr>
<tr>
<td>• Manual, Semi-Automatic,</td>
<td>BRGB</td>
<td></td>
</tr>
<tr>
<td>Automatic</td>
<td>BRGA/B Operation Mode</td>
<td></td>
</tr>
<tr>
<td>Diagnostics</td>
<td>Timer</td>
<td></td>
</tr>
<tr>
<td>• Loopback</td>
<td>BRG</td>
<td></td>
</tr>
<tr>
<td>• Remote</td>
<td>BRGA/B Divide Count</td>
<td></td>
</tr>
<tr>
<td>• Local</td>
<td>BRGA/B Source</td>
<td></td>
</tr>
<tr>
<td>• Echo</td>
<td>Sys Clock</td>
<td></td>
</tr>
<tr>
<td>• Yes/No</td>
<td>SCLK Pin</td>
<td></td>
</tr>
<tr>
<td>• Disable Error Echo</td>
<td>BRGA Output (BRGB Only)</td>
<td></td>
</tr>
<tr>
<td>• Disable Control/Address</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Char. Echo</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>TX Operation</th>
<th>I/O Pins</th>
</tr>
</thead>
<tbody>
<tr>
<td>• RTS/CTS Control</td>
<td>• Select Function for Each</td>
</tr>
<tr>
<td>Manual, Semi-Automatic,</td>
<td>Multifunction Pin</td>
</tr>
<tr>
<td>Automatic</td>
<td>• Select Direction for Multi-</td>
</tr>
<tr>
<td></td>
<td>function Pin (If Applicable)</td>
</tr>
</tbody>
</table>
1. IMD—INTERNAL MODE REGISTER

This register defines the general device mode of operation. The bit functions are as follows:

7–4: Reserved

IAM: Interrupt Acknowledge Mode Bit

0 — Manual acknowledgement of pending interrupts
1 — Automatic acknowledgement of pending interrupts (upon CPU service)

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.)

RFD: Receive FIFO Depth

0 — Four Bytes
1 — One Byte

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.

ULM: uLAN Mode

0 — Normal Mode
1 — uLAN Mode

This bit, enables the 82510 to recognize and/or match address using the 9-bit MCS-51 asynchronous protocol.

LEM: Loopback/Echo Mode Select

This bit selects the mode of loopback operation, or the mode of echo operation; depending upon which operation mode is selected by the Modem Control register bit LC.

In loopback mode (Modem Control register bit LC = 1) it selects between local and remote loopback.

0 — Local Loopback
1 — Remote Loopback

In echo mode (Modem Control register bit LC = 0) it selects between echo or non-echo operation.

0 — No Echo
1 — Echo Operation
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.

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>

SBLO—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 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></td>
<td>16X</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

CL0—CL1—Character Length Bits—These bits, together with the Transmit Mode register bit NBCL, define the character length. See Table 14.

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

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

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

**NOTE:**
This register is identical in function to ACR0.
### 7. RMD—RECEIVE MACHINE MODE REGISTER

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>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.</td>
</tr>
<tr>
<td>6</td>
<td>In uLAN mode: selects the mode of address recognition.</td>
</tr>
<tr>
<td>5</td>
<td>Manual Mode—Rx Machine reports reception of any address character, via CRF bit of RECEIVE STATUS register, and writes it to the Rx FIFO.</td>
</tr>
<tr>
<td>4</td>
<td>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.)</td>
</tr>
<tr>
<td>3</td>
<td>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.</td>
</tr>
<tr>
<td>2</td>
<td>Reserved</td>
</tr>
<tr>
<td>1</td>
<td>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.)</td>
</tr>
<tr>
<td>0</td>
<td>SWM—Sampling Window Mode—This bit controls the mode of data sampling:</td>
</tr>
</tbody>
</table>

#### uCM0, uCM1

- **00**—No Standard Set Control Characters Recognized. 
- **01**—ASCII Control Characters (00H—1FH + 7FH). 
- **10**—Reserved. 
- **11**—EBCDIC Control Character Recognized (00H—3FH). 

#### DPD

- **0**—Small Window, 3/16 sampling. 
- **1**—Large Window, 7/16 sampling. 

#### SWM

- **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

<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>Rx CLOCK MODE = RxCM</td>
<td>Rx CLOCK SOURCE = RxCS</td>
<td>RESERVED</td>
<td>Tx CLOCK MODE = TxCM</td>
<td>Tx CLK SOURCE = TxCS</td>
<td>RESERVED</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**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

<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>BRGA CLOCK SOURCE = BACS</td>
<td>RESERVED</td>
<td>RESERVED</td>
<td>RESERVED</td>
<td>BAM—BRGA MODE</td>
<td>RESERVED</td>
<td></td>
</tr>
</tbody>
</table>

**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.

**BAM—BRGA Mode of Operation**—Selects between the Timer mode or the Baud Rate Generator Mode.

0—Timer Mode (in this mode the input clock source is always the system clock).
1—Baud Rate Generator Mode
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 BAGA.)

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.
14. BBH—BRGB DIVIDE COUNT MOST SIGNIFICANT BYTE

![Diagram of BBH—BRGB Divide Count High Byte](image)

This register contains the most significant byte of the BRGB divisor/count.

15. PMD—I/O PIN MODE REGISTER

![Diagram of PMD—I/O Pin Mode Register](image)

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.
   DIOF—DCD/ICLK/OUT1 Direction.
   0— Output: ICLK or OUT1 (depending on bit DIOF).
   1— Input: DCD.

   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

**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.

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.
- **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.

**BkTe**—Break Termination Interrupt Enable.
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.

TBIE—Timer B Expired Interrupt Enable—Enables Interrupt on TBEx bit of TMST.
TAIE—Timer A Expired Interrupt Enable—Enables Interrupt on TAEx 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.

**TIMERS Interrupt Request**—This bit indicates that one of the timers has expired. (See TMST)

**TxIR—Transmit Machine Interrupt Request**—Indicates that the Transmit Machine is either empty or disabled (Idle).

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, BANKO—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.

**BI2, BI1, BI0—Interrupt Bits 0–2**—These three bits reflect the highest priority enabled pending interrupt from GSR.
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 TxlR 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.
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 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.

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.

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

TBEx—Timer B Expired—When Set generates an interrupt through TIR bit of GSR. Indicates that Timer B count has expired. This bit is set via the terminal count pulse generated by the timer when it terminates counting.

TAEx—Timer A Expired—Same as TBEx except it refers to Timer A.

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.
30. RCM—RECEIVE COMMAND REGISTER

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.

31. TMCR—TIMER CONTROL REGISTER

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

STA—Start Timer A—This command triggers timer A. At terminal count a status bit is set in TMST (TAE).

STB—Start Timer B—This command triggers timer B. At terminal count a status bit is set in TMST (TBE).

TGA—Timer-A Gate—This bit serves as a gate for Timer-A operation:
32. MCR—MODEM CONTROL 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>DTR—DTR OUTPUT BIT</td>
<td>RTS—RTS OUTPUT</td>
<td>OUT1—OUT1 OUTPUT BIT</td>
<td>OUT2—OUT2 OUTPUT BIT</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

MCR—Modem Control Register

This register controls the modem output pins. With multi—function pins it affects only the pins configured as general purpose output pins. All the output pins invert the data, i.e. their output will be the complement 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.

33. TXD—TRANSMIT DATA 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>D7</td>
<td>D6</td>
<td>D5</td>
<td>D4</td>
<td>D0</td>
<td>D1</td>
<td>D2</td>
<td>D3</td>
</tr>
</tbody>
</table>

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 components (S/W Parity, Address Marker Bit) the additional data bits should be written into the TxF register. When a byte is written to this register its contents, 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 RxP 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 (TA = 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)</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>IIL</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>ICC</td>
<td>Power Supply Current</td>
<td>(6)</td>
<td>3.8</td>
<td>mA/MHz</td>
<td></td>
</tr>
<tr>
<td>IPD</td>
<td>Power Down Supply</td>
<td>(7)</td>
<td>2</td>
<td>mA</td>
<td></td>
</tr>
<tr>
<td>ISTBY</td>
<td>Standby Supply Current</td>
<td>(10)</td>
<td>500</td>
<td>μA</td>
<td></td>
</tr>
<tr>
<td>I0HR</td>
<td>RTS, DTR Strapping Current</td>
<td>(11)</td>
<td>0.4</td>
<td>mA</td>
<td></td>
</tr>
<tr>
<td>IOLR</td>
<td>RTS, DTR Strapping Current</td>
<td>(12)</td>
<td>11</td>
<td>mA</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>
<tr>
<td>CXTAL</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. @ IOL = 2 mA.
3. @ IOH = -0.4 mA.
4. 0 < VIN < VCC.
5. 0.45V < VOUT < (VCC - 0.45).
6. VCC = 5.5V; VIH = 0.5V (max); VIL = VCC - 0.5V (min); 35 mA (max); Typical value = 2.5 mA/MHz (Not Tested); Ext
1X CLK (9 MHz max); IIL = IOH = 0.
7. VCC = 5.5V; VIH = GND; VIL = VCC; IOL = IOH = 0; device at power down mode, clock running.
8. Freq = 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>Tc/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 Rate</td>
<td>4.0</td>
<td>18.432 MHz</td>
<td></td>
</tr>
<tr>
<td>DIVIDE BY TWO OPTION—INACTIVE</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Tc</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>(1)</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. Tc 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 Tcy</td>
<td></td>
<td>(1)</td>
</tr>
<tr>
<td>TTLRSL</td>
<td>RTS/DTR LOW Setup to Reset Inactive</td>
<td>6 Tcy</td>
<td></td>
<td>(2)</td>
</tr>
<tr>
<td>TRSLTX</td>
<td>RTS/DTR Low Hold after Reset Inactive</td>
<td>0 Tcy</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>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 Delay after RD Active</td>
<td>2 Tcy + 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>40</td>
<td></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>2 Tcy + 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>

**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></td>
<td>15</td>
<td>(1)</td>
</tr>
<tr>
<td>TXL2XL1</td>
<td>SCLK Fall Time</td>
<td></td>
<td>15</td>
<td>(1)</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></td>
<td>15</td>
<td>(1)</td>
</tr>
<tr>
<td>TXL2XL1</td>
<td>SCLK Fall Time</td>
<td></td>
<td>15</td>
<td>(1)</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>
</tr>
</thead>
<tbody>
<tr>
<td>TRPW</td>
<td>RXD Setup Time to SCLK High</td>
<td>250</td>
<td></td>
</tr>
<tr>
<td>TRPD</td>
<td>RXD Hold Time After SCLK High</td>
<td>250</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>
</tr>
</thead>
<tbody>
<tr>
<td>TSCLKTXD</td>
<td>TXD Valid Delay after SCLK Low</td>
<td>—</td>
<td>170</td>
</tr>
</tbody>
</table>

REMOTE LOOPBACK SPECIFICATION

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Parameter</th>
<th>Min</th>
<th>Max</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRXDTXD</td>
<td>TXD Delay after RXD</td>
<td>—</td>
<td>170</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.

**Figure 1. Block Diagram**

**Figure 2. Configuration**

- **REGISTERS**
  - TxINT RESULT
  - RxINT RESULT
  - TEST MODE

- **CONTROL LOGIC**
  - RxD
  - RxC
  - DPLL
  - RTS
  - CTS
  - CD
  - FA_{1-4}
  - FA_{5}

- **READ/ WRITE DMA/ CONTROL LOGIC**
  - TxINT
  - RxINT
  - RD
  - WR
  - A_{0-1}
  - RESET
  - CLK

- **DATA BUS BUFFER**
  - DB_{0-7}

- **CPU INTERFACE**
  - VCC
  - TX INT
  - CLK
  - RESET
  - TXDACK
  - TXDRO
  - RXDACK
  - RXDRO
  - RD
  - WR
  - CD
  - RTS
  - CTS
  - TXC
  - 32X CLK
  - 32X CLK

- **MODEM INTERFACE**
  - 210479-2
A BRIEF DESCRIPTION OF HDLC/SDLC PROTOCOLS

General

The High Level Data Link Control (HDLC) is a standard communication link protocol established by the 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 preambles 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>TxINT</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>RxINT</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>RxRDQ</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>
</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>TXDACK</td>
<td>5</td>
<td>I</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>I</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</th>
<th>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</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>Command</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>Status</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>Parameter</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>Result</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>Reset</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>TxlNT Result</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>RxlnT Result</td>
</tr>
<tr>
<td>X</td>
<td>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</td>
<td>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](image)
SYNCHRONOUS MODEM—DUPLEX OR HALF DUPLEX OPERATION

ASYNCHRONOUS MODES—DUPLEX OR HALF DUPLEX OPERATION

ASYNCHRONOUS—NO MODEMS—DUPLEX OR HALF DUPLEX
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>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>CBSY</td>
<td>CBF</td>
<td>CPBF</td>
<td>CRBF</td>
<td>RxINT</td>
<td>TxINT</td>
<td>RxPAR</td>
<td>TxPAR</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

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 RxlRA (Receiver Interrupt Result Available)
The RxlRA is set by the 8273 when an interrupt result byte is placed in the RxlNT register. It is reset after the CPU has read the RxlNT register.

Bit 0 TxlRA (Transmitter Interrupt Result Available)
The TxlRA is set by the 8273 when an interrupt result byte is placed in the TxlNT register. It is reset when the CPU has read the TxlNT 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 RxlNT 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
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
DETAILED 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).

<table>
<thead>
<tr>
<th>SET ONE-BIT DELAY (CMD CODE A4)</th>
</tr>
</thead>
<tbody>
<tr>
<td>( A_1 )</td>
</tr>
<tr>
<td>CMD:</td>
</tr>
<tr>
<td>PAR:</td>
</tr>
</tbody>
</table>
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>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</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>0</td>
<td>1</td>
<td>0</td>
<td>0</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)**

| CMD: A1 A0 D7 D6 D5 D4 D3 D2 D1 D0 |
|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|
| 0   | 0   | 0   | 1   | 0   | 0   | 0   | 0   | 0   | 1   |
| 1   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   |

Any mode switches set in CMD code 91 can be reset using this command by placing zeros in the appropriate positions.

**(D5) HDLC MODE**

In HDLC mode, a bit sequence of seven ones (01111111) is interpreted as an abort character. Otherwise, eight ones (011111111) signal an abort.

**(D4) EOP INTERRUPT MODE**

In EOP interrupt mode, an interrupt is generated whenever an EOP character (01111111) is detected by an active receiver. This mode is useful for the implementation of an SDLC loop controller in detecting the end of a message stream after a loop poll.

**(D3) TRANSmitter EARLY INTERRUPT MODE (Tx)**

The early interrupt mode is specified to indicate when the 8273 should generate an end of frame interrupt. When set, an early interrupt is generated when the last data character has been passed to the 8273. If the user software responds with another transmit command before the final flag is sent, the final flag interrupt will not be generated and a new frame will immediately begin when the current frame is complete. This permits frames to be separated by a single flag. If no additional Tx commands are provided, a final interrupt will follow.

**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)H if NRZI is set or data (55)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 Transparent Active</td>
</tr>
<tr>
<td>Transparent Active</td>
<td>Ignore Command.</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 Transparent Active</td>
</tr>
<tr>
<td>Transparent Active</td>
<td>Ignore Command.</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>A1</th>
<th>A2</th>
<th>D3</th>
<th>D4</th>
<th>D5</th>
<th>D6</th>
<th>D7</th>
<th>D8</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</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

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>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>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

PAR: 0 1 1 1 1 1 1 1

(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.

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>CMD:</th>
<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></td>
<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>

<table>
<thead>
<tr>
<th>PAR:</th>
<th>0</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>LEAST SIGNIFICANT BYTE OF THE RECEIVE BUFFER LENGTH (B0)</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>PAR:</th>
<th>0</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>MOST SIGNIFICANT BYTE OF RECEIVE BUFFER LENGTH (B1)</td>
<td></td>
</tr>
</tbody>
</table>

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.

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>TMR:</td>
<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>
</tr>
<tr>
<td>TMR:</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>0</td>
</tr>
</tbody>
</table>

An 8273 reset command is executed by outputting a (01)\textsubscript{H} followed by (00)\textsubscript{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.
SELECTIVE RECEIVE (CMD CODE C1)

<table>
<thead>
<tr>
<th>CMD:</th>
<th>0</th>
<th>0</th>
<th>1</th>
<th>1</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>PAR:</td>
<td>0</td>
<td>1</td>
<td>LEAST SIGNIFICANT BYTE OF THE RECEIVE BUFFER LENGTH (B0)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PAR:</td>
<td>0</td>
<td>1</td>
<td>MOST SIGNIFICANT BYTE OF RECEIVE BUFFER LENGTH (B1)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PAR:</td>
<td>0</td>
<td>1</td>
<td>RECEIVE FRAME ADDRESS MATCH FIELD ONE (A1)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PAR:</td>
<td>0</td>
<td>1</td>
<td>RECEIVE FRAME ADDRESS MATCH FIELD TWO (A2)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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>CMD:</th>
<th>0</th>
<th>0</th>
<th>1</th>
<th>1</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>PAR:</td>
<td>0</td>
<td>0</td>
<td>LEAST SIGNIFICANT BYTE OF THE RECEIVE BUFFER LENGTH (B0)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PAR:</td>
<td>0</td>
<td>1</td>
<td>MOST SIGNIFICANT BYTE OF RECEIVE BUFFER LENGTH (B1)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PAR:</td>
<td>0</td>
<td>1</td>
<td>RECEIVE FRAME ADDRESS MATCH FIELD ONE (A1)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PAR:</td>
<td>0</td>
<td>1</td>
<td>RECEIVE FRAME ADDRESS MATCH FIELD TWO (A2)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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)

Terminates an active receive command immediately.

<table>
<thead>
<tr>
<th>CMD:</th>
<th>0</th>
<th>0</th>
<th>1</th>
<th>1</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>1</th>
<th>0</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>PAR:</td>
<td>NONE</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Transmit Commands

The 8273 supports three transmit commands: Transmit Frame, Loop Transmit, Transmit Transparent.

TRANSMIT FRAME (CMD CODE C8)

<table>
<thead>
<tr>
<th>CMD:</th>
<th>0</th>
<th>0</th>
<th>1</th>
<th>1</th>
<th>0</th>
<th>0</th>
<th>1</th>
<th>0</th>
<th>0</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>PAR:</td>
<td>0</td>
<td>1</td>
<td>LEAST SIGNIFICANT BYTE OF FRAME LENGTH (L0)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PAR:</td>
<td>0</td>
<td>1</td>
<td>MOST SIGNIFICANT BYTE OF FRAME LENGTH (L1)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PAR:</td>
<td>0</td>
<td>1</td>
<td>ADDRESS FIELD OF TRANSMIT FRAME (A)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PAR:</td>
<td>0</td>
<td>1</td>
<td>CONTROL FIELD OF TRANSMIT FRAME (C)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>LEAST SIGNIFICANT BYTE OF FRAME LENGTH (L0)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>MOST SIGNIFICANT BYTE OF FRAME LENGTH (L1)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>ADDRESS FIELD OF TRANSMIT FRAME (A)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>CONTROL FIELD OF TRANSMIT FRAME (C)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>LEAST SIGNIFICANT BYTE OF FRAME LENGTH (L0)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>MOST SIGNIFICANT BYTE OF FRAME LENGTH (L1)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>LEAST SIGNIFICANT BYTE OF FRAME LENGTH (L0)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>MOST SIGNIFICANT BYTE OF FRAME LENGTH (L1)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>ADDRESS FIELD OF TRANSMIT FRAME (A)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>CONTROL FIELD OF TRANSMIT FRAME (C)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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>
<tr>
<td>0</td>
<td>1</td>
<td>LEAST SIGNIFICANT BYTE OF FRAME LENGTH (L0)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>MOST SIGNIFICANT BYTE OF FRAME LENGTH (L1)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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></td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>LEAST SIGNIFICANT BYTE OF FRAME LENGTH (L0)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>MOST SIGNIFICANT BYTE OF FRAME LENGTH (L1)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>ADDRESS FIELD OF TRANSMIT FRAME (A)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>CONTROL FIELD OF TRANSMIT FRAME (C)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>LEAST SIGNIFICANT BYTE OF FRAME LENGTH (L0)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>MOST SIGNIFICANT BYTE OF FRAME LENGTH (L1)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>ADDRESS FIELD OF TRANSMIT FRAME (A)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>CONTROL FIELD OF TRANSMIT FRAME (C)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>LEAST SIGNIFICANT BYTE OF FRAME LENGTH (L0)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>MOST SIGNIFICANT BYTE OF FRAME LENGTH (L1)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>ADDRESS FIELD OF TRANSMIT FRAME (A)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>CONTROL FIELD OF TRANSMIT FRAME (C)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
SET PORT B BITS (CMD CODE A3)

This command allows user defined Port B pins to be set.

<table>
<thead>
<tr>
<th>CMD:</th>
<th>A0</th>
<th>A1</th>
<th>D2</th>
<th>D3</th>
<th>D4</th>
<th>D5</th>
<th>D6</th>
<th>D7</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

(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.

<table>
<thead>
<tr>
<th>CMD:</th>
<th>A0</th>
<th>A1</th>
<th>D2</th>
<th>D3</th>
<th>D4</th>
<th>D5</th>
<th>D6</th>
<th>D7</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>

(D9) 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.

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>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Min</td>
<td>Max</td>
<td>Min</td>
</tr>
<tr>
<td></td>
<td></td>
<td>tcy</td>
<td>tcy</td>
<td>tcy</td>
</tr>
<tr>
<td>T1</td>
<td>Between Command &amp; First Parameter</td>
<td>13</td>
<td>756</td>
<td>13</td>
</tr>
<tr>
<td>T2</td>
<td>Between Consecutive Parameters</td>
<td>10</td>
<td>604</td>
<td>10</td>
</tr>
<tr>
<td>T3</td>
<td>Command Parameter Buffer Full Bit Reset after Parameter Loaded</td>
<td>10</td>
<td>604</td>
<td>10</td>
</tr>
<tr>
<td>T4</td>
<td>Command Busy Bit Reset after Last Parameter</td>
<td>128</td>
<td>702</td>
<td>128</td>
</tr>
<tr>
<td>T5</td>
<td>CPBF Bit Reset after Last Parameter</td>
<td>10</td>
<td>604</td>
<td>10</td>
</tr>
</tbody>
</table>

2-105
WAVEFORMS (Continued)

RECEIVER INTERRUPT

![Waveform Diagram]

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>T1</td>
<td>RxIRA Bit Set after RIC Read</td>
<td>Min: 18</td>
<td>Max: 29</td>
<td>Min: 18</td>
</tr>
<tr>
<td>T2</td>
<td>RxINT Goes Away after Last Int. Result Read</td>
<td>Min: 16</td>
<td>Max: 27</td>
<td>Min: 16</td>
</tr>
</tbody>
</table>

TRANSMIT INTERRUPT

![Waveform Diagram]

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>T1</td>
<td>TxINT Inactive after Int. Results Read</td>
<td>Min: 13</td>
<td>Max: 353</td>
<td>Min: 13</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:** This is a production data sheet. The specifications are subject to change without notice.

*WARNING: Stressing the device beyond the "Absolute Maximum Ratings" may cause permanent damage. These are stress ratings only. Operation beyond the "Operating Conditions" is not recommended and extended exposure beyond the "Operating Conditions" may affect device reliability.

### D.C. CHARACTERISTICS 8273 (T<sub>A</sub> = 0°C to 70°C, V<sub>CC</sub> = + 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>V&lt;sub&gt;IL&lt;/sub&gt;</td>
<td>Input Low Voltage</td>
<td>−0.5</td>
<td>0.8</td>
<td>V</td>
<td></td>
</tr>
<tr>
<td>V&lt;sub&gt;IH&lt;/sub&gt;</td>
<td>Input High Voltage</td>
<td>2.0</td>
<td>V&lt;sub&gt;CC&lt;/sub&gt; + 0.5</td>
<td>V</td>
<td></td>
</tr>
</tbody>
</table>
| V<sub>OL</sub> | Output Low Voltage | | 0.45 | V | I<sub>OL</sub> = 2.0 mA for Data Bus Pins
| | | | | I<sub>OL</sub> = 1.0 mA for Output Port Pins
| | | | | I<sub>OL</sub> = 1.6 mA for All Other Pins |
| V<sub>OH</sub> | Output High Voltage | | 2.4 | V | I<sub>OH</sub> = −200 μA for Data Bus Pins
| | | | | I<sub>OH</sub> = −100 μA for All Other Pins |
| I<sub>IL</sub> | Input Load Current | | ±10 | μA | V<sub>IN</sub> = V<sub>CC</sub> to 0V |
| I<sub>OLF</sub> | Output Leakage Current | | ±10 | μA | V<sub>OUT</sub> = V<sub>CC</sub> to 0.45V |
| I<sub>CC</sub> | V<sub>CC</sub> Supply Current | | 180 | mA | |

### CAPACITANCE 8273 (T<sub>A</sub> = 25°C, V<sub>CC</sub> = 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>C&lt;sub&gt;IN&lt;/sub&gt;</td>
<td>Input Capacitance</td>
<td></td>
<td>10</td>
<td>pF</td>
<td>t&lt;sub&gt;c&lt;/sub&gt; = 1 MHz</td>
<td></td>
</tr>
<tr>
<td>C&lt;sub&gt;I/O&lt;/sub&gt;</td>
<td>I/O Capacitance</td>
<td></td>
<td>20</td>
<td>pF</td>
<td>Unmeasured Pins Returned to GND</td>
<td></td>
</tr>
</tbody>
</table>

### A.C. CHARACTERISTICS (T<sub>A</sub> = 0°C to 70°C, V<sub>CC</sub> = + 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>t&lt;sub&gt;CY&lt;/sub&gt;</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>t&lt;sub&gt;CL&lt;/sub&gt;</td>
<td>Clock Low</td>
<td>120</td>
<td></td>
<td></td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>t&lt;sub&gt;CH&lt;/sub&gt;</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 to 70°C, *V_CC* = +5.0V ± 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>tAC</td>
<td>Select Setup to RD</td>
<td>0</td>
<td></td>
<td>ns</td>
<td>(Note 2)</td>
</tr>
<tr>
<td>tCA</td>
<td>Select Hold from RD</td>
<td>0</td>
<td></td>
<td>ns</td>
<td>(Note 2)</td>
</tr>
<tr>
<td>tRR</td>
<td>RD Pulse Width</td>
<td></td>
<td>250</td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>tAD</td>
<td>Data Delay from Address</td>
<td>300</td>
<td></td>
<td>ns</td>
<td>(Note 2)</td>
</tr>
<tr>
<td>tRD</td>
<td>Data Delay from RD</td>
<td></td>
<td>200</td>
<td>ns</td>
<td><em>C_L</em> = 150 pF, (Note 2)</td>
</tr>
<tr>
<td>tDF</td>
<td>Output Float Delay</td>
<td>20</td>
<td>100</td>
<td>ns</td>
<td><em>C_L</em> = 20 pF For Minimum; 150 pF for Maximum</td>
</tr>
<tr>
<td>tDC</td>
<td>DACK Setup to RD</td>
<td>25</td>
<td></td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>tCD</td>
<td>DACK Hold from RD</td>
<td>25</td>
<td></td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>tKD</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>tAC</td>
<td>Select Setup to WR</td>
<td>0</td>
<td></td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>tCA</td>
<td>Select Hold from WR</td>
<td>0</td>
<td></td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>tWW</td>
<td>WR Pulse Width</td>
<td></td>
<td>250</td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>tDW</td>
<td>Data Setup to WR</td>
<td></td>
<td>150</td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>tWD</td>
<td>Data Hold from WR</td>
<td></td>
<td>0</td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>tDC</td>
<td>DACK Setup to WR</td>
<td>25</td>
<td></td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>tCD</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>tCO</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></td>
<td>tCY</td>
<td></td>
</tr>
<tr>
<td>tr</td>
<td>Input Signal Rise Time</td>
<td>20</td>
<td></td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>tf</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>tCY</td>
<td></td>
</tr>
<tr>
<td>t_CY32</td>
<td>32X Clock Cycle Time</td>
<td></td>
<td>13.02×t_CY</td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>t_CL32</td>
<td>32X Clock Low Time</td>
<td></td>
<td>4×t_CY</td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>t_CH32</td>
<td>32X Clock High Time</td>
<td></td>
<td>4×t_CY</td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>t_DLL</td>
<td>DPLL Output Low</td>
<td></td>
<td>1×t_CY−50</td>
<td>ns</td>
<td></td>
</tr>
</tbody>
</table>
A.C. CHARACTERISTICS 8273 (TA = 0°C to 70°C, VCC = +5.0V ± 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>tDCL</td>
<td>Data Clock Low</td>
<td>1×tCY−50</td>
<td>ns</td>
<td></td>
<td></td>
</tr>
<tr>
<td>tDCH</td>
<td>Data Clock High</td>
<td>2×tCY</td>
<td>ns</td>
<td></td>
<td></td>
</tr>
<tr>
<td>tDCY</td>
<td>Data Clock</td>
<td>62.5×tCY</td>
<td>ns</td>
<td>(Note 3)</td>
<td></td>
</tr>
<tr>
<td>tTD</td>
<td>Transmit Data Delay</td>
<td>200</td>
<td>ns</td>
<td></td>
<td></td>
</tr>
<tr>
<td>tDS</td>
<td>Data Setup Time</td>
<td>200</td>
<td>ns</td>
<td></td>
<td></td>
</tr>
<tr>
<td>tDH</td>
<td>Data Hold Time</td>
<td>100</td>
<td>ns</td>
<td></td>
<td></td>
</tr>
<tr>
<td>tFLO</td>
<td>FLAG DET Output Low</td>
<td>8×tCY ± 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. tAD, tRD, tAC, and tCA 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 tCY 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
WAVEFORMS (Continued)

WRITE

DMA

TRANSMIT
WAVEFORMS (Continued)

RECEIVE

DPLL OUTPUT

FLAG DETECT OUTPUT
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, -51; iAPX-86, -88, -186 and -188 families, the 8237 DMA Controller, or the 8089 I/O Processor in polled, interrupt driven, or DMA 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><strong>CLOCK</strong>: System clock, TTL compatible.</td>
</tr>
<tr>
<td>RESET</td>
<td>2</td>
<td>I</td>
<td><strong>RESET</strong>: A low signal on this pin will force the MPSC to an idle state. TxA and TxB 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><strong>CARRIER DETECT (CHANNEL A)</strong>: 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 RxDA 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>RxCB</td>
<td>4</td>
<td>I</td>
<td><strong>RECEIVE CLOCK (CHANNEL B)</strong>: The serial data are shifted into the Receive Data input (RxDg) on the rising edge of the Receive Clock.</td>
</tr>
<tr>
<td>CD_B</td>
<td>5</td>
<td>I</td>
<td><strong>CARRIER DETECT (CHANNEL B)</strong>: 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 RxDB 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>CTSB</td>
<td>6</td>
<td>I</td>
<td><strong>CLEAR TO SEND (CHANNEL B)</strong>: 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>TxCB</td>
<td>7</td>
<td>I</td>
<td><strong>TRANSMIT CLOCK (CHANNEL B)</strong>: The serial data are shifted out from the Transmit Data output (TxDb) on the falling edge of the Transmit Clock.</td>
</tr>
<tr>
<td>TxD_B</td>
<td>8</td>
<td>O</td>
<td><strong>TRANSMIT DATA (CHANNEL B)</strong>: This pin transmits serial data to the communications channel (Channel B).</td>
</tr>
<tr>
<td>RxDB</td>
<td>9</td>
<td>I</td>
<td><strong>RECEIVE DATA (CHANNEL B)</strong>: This pin receives serial data from the communications channel (Channel B).</td>
</tr>
<tr>
<td>SYNDET_B/RTSB</td>
<td>10</td>
<td>I/O</td>
<td><strong>SYNCHRONOUS DETECTION (CHANNEL B)</strong>: 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). <strong>REQUEST TO SEND (CHANNEL B)</strong>: 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_B/TxDRQA</td>
<td>11</td>
<td>O</td>
<td>READY (CHANNEL B)/TRANSMITTER DMA REQUEST (CHANNEL A): In mode 0 this pin is RDY_B 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_CC</td>
<td>40</td>
<td></td>
<td>POWER: +5V Supply</td>
</tr>
<tr>
<td>CTS_A</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_A</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>TxDA</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>TxCA</td>
<td>36</td>
<td>I</td>
<td>TRANSMIT CLOCK (CHANNEL A): The serial data are shifted out from the Transmit Data output (TxDA) on the falling edge of the Transmit Clock.</td>
</tr>
<tr>
<td>RxCA</td>
<td>35</td>
<td>I</td>
<td>RECEIVE CLOCK (CHANNEL A): The serial data are shifted into the Receive Data input (RxDA) on the rising edge of the Receive Clock.</td>
</tr>
<tr>
<td>RxDA</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_A</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>
<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>RDY_A/ RxDRQA</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>DTR_A</td>
<td>31</td>
<td>O</td>
<td>DATA TERMINAL READY (CHANNEL A): General purpose output.</td>
</tr>
<tr>
<td>IPO/ TxDRQB</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 IPI. It is low only if IPI 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>IPI/ RxDRQB</td>
<td>29</td>
<td>I/O</td>
<td>INTERRUPT PRIORITY IN/RECEIVER DMA REQUEST (CHANNEL B): In modes 0 and 1, IPI is Interrupt Priority In. A low on IPI 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>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>DTRB</td>
<td>26</td>
<td>O</td>
<td>DATA TERMINAL READY (CHANNEL B): This is a general purpose output.</td>
</tr>
<tr>
<td>A0</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>A1</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 SDLC) 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 1 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 c</td>
<td>3</td>
</tr>
<tr>
<td>1 0 0 0 c c c c c c</td>
<td>4</td>
</tr>
<tr>
<td>0 0 0 c 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; D3).

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 clock options only. The data character is transmitted least significant bit first.

The serial data are shifted out from the Transmit Data (TxD) output on the falling edge of the Transmit Clock (TxC) input at a rate programmable to 1, 1/6th, 1/32nd, or 1/64th of the clock rate supplied to the TxC input.

The TxD output is held high when the transmitter has no data to send, unless, under program control, the Send Break (WR5; D4) command is issued to hold the TxD low.

If the External/Status Interrupt bit (WR1; D0) is set, the status of CD, CTS and SYNDET are monitored and, if any changes occur for a period of time greater than the minimum specified pulse width, an interrupt is generated. CTS is usually monitored using this interrupt feature (e.g., Auto Enable option).

The Transmit Buffer Empty bit (RRO; D2) is set by the MPSC when the data byte from the buffer is loaded in the transmit shift register. Data should be written to the MPSC only when the Tx buffer becomes empty to prevent overwriting.

**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 1/2 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 1/2 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)
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, \((\text{TxC})\) and is received on the rising edge of Receive Clock \((\text{RxC})\). 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-
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 (RR0; D6).

Data Transfers using the R DY signal are for software controlled data transfers such as block moves. R DY 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 T xDRQ 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 (RR0; D6) is 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 completely 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 (6 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 (RR0; 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 RDY 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) not to 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 (RR1; 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; D9) 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.

<table>
<thead>
<tr>
<th>Synchronous Mode Register Setup—SDLC/HDLC</th>
</tr>
</thead>
<tbody>
<tr>
<td>D7</td>
</tr>
<tr>
<td>----</td>
</tr>
<tr>
<td>WR3</td>
</tr>
<tr>
<td>WR4</td>
</tr>
<tr>
<td>WR5</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 DMA 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 may be programmed to issue a Send Abort command (WR0; D5, D4, D3). This command causes at least eight 1's but less than fourteen 1's to be sent before the line reverts to continuous flags.

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 01111101111110 will be detected as two flags.

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 (OFFH) 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.

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):

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's initialized to all 1's.
11  Reset Tx Underrun/End of Message Latch.
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**

<table>
<thead>
<tr>
<th>VECTOR (CH B ONLY)</th>
<th>VARIABLE VECTOR</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>RxINT/DMA DISABLE</td>
</tr>
<tr>
<td>01</td>
<td>RxINT ON FIRST CHAR OR SPECIAL CONDITION</td>
</tr>
<tr>
<td>10</td>
<td>INT ON ALL Rx CHAR (PARITY AFFECTS VECTOR) OR SPECIAL CONDITION</td>
</tr>
<tr>
<td>11</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

**WR1**

- **D0**  
  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.

- **D1**  
  Transmitter Interrupt/DMA Enable—allows the MPSC to interrupt or request a DMA transfer when the transmitter buffer becomes empty.

- **D2**  
  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 if set then the vector returned from an interrupt acknowledge is variable as shown in the Interrupt Vector Table.

**WR2**

**Channel A Only**

- **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 interrupts.
  - 1 0 | Channel A and Channel B both use DMA.
  - 1 1 | Illegal Code.

**WR1**

- **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).

**WR1**

- **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, AO = 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.

- **D6**  
  Must be Zero.

- **D7**  
  Wait Enable—enables the wait function.

**WR2**

**Channel A Only**

- **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 interrupts.
  - 1 0 | Channel A and Channel B both use DMA.
  - 1 1 | Illegal Code.

**WR1**

- **D2**  
  Priority—this bit specifies the relative priorities of the internal MPSC interrupt/DMA sources.

  - 0 | (Highest) RxA, TxA, RxB, TxB, ExTA, ExTB (Lowest).
  - 1 | (Highest) RxA, RxB, TxA, TxB, ExTA, ExTB (Lowest).

**WR2**

**Channel A Only**

- **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 XX 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 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).

D6
zero Pin 10 = RTSB
D7
one Pin 10 = SYNDTB

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>D5</th>
<th>D4</th>
<th>D3</th>
<th>IPI</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></td>
</tr>
<tr>
<td>1</td>
<td>0</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></td>
<td></td>
<td></td>
<td>2nd INTA</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>3rd INTA</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>D7</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>High Impedance</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>D0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>85 Mode 1</td>
<td>1st INTA</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>2nd INTA</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>3rd INTA</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>D7</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>High Impedance</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>D0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>86 Mode</td>
<td>1st INTA</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>2nd INTA</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>D7</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>High Impedance</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>D0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>85 Mode 2</td>
<td>1st INTA</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>2nd INTA</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>3rd INTA</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>D7</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>High Impedance</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>D0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>85 Mode 2</td>
<td>1st INTA</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>2nd INTA</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>3rd INTA</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>D7</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>High Impedance</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>D0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>86 Mode</td>
<td>1st INTA</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>2nd INTA</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>D7</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>High Impedance</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>D0</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 WR2</th>
<th>Int/DMA Mode</th>
<th>Pin Functions</th>
<th>Priority</th>
</tr>
</thead>
<tbody>
<tr>
<td>D2 D1 D0</td>
<td>Ch. A CH. B</td>
<td>RDYA/RxDRQA</td>
<td>RxA, TxA, RxB, TxB, EXT_A, EXT_B</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Pin 32</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>RDYB/TxDRQA</td>
<td>RxA, TxA, RxB, TxB, EXT_A, EXT_B</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Pin 11</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>IPI</td>
<td>RxA, TxA (DMA)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>IPO</td>
<td>RxA, TxA (DMA)</td>
</tr>
</tbody>
</table>

**NOTE:**
1. Special Receive Condition
Interrupt Vector Mode Table

<table>
<thead>
<tr>
<th>8085 Modes 8086/88 Mode</th>
<th>V₄ V₃ V₂</th>
<th>Channel</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>V₁ V₀</td>
<td></td>
<td>B</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td></td>
<td>Tx Buffer Empty</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td></td>
<td>Ext/Status Change</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td></td>
<td>Rx Char. Available</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td></td>
<td>Special Rx Condition</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>A</td>
<td>Tx Buffer Empty</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td></td>
<td>Ext/Status Change</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td></td>
<td>Rx Char. Available</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td></td>
<td>Special Rx Condition</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):

<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>

Rx ENABLE
SYNC CHAR LOAD INHIBIT
ADDR SRCH MODE (SDLC)
Rx CRC ENABLE
ENTER HUNT MODE
AUTO ENABLES

0 0 Rx 5 BITS/CHAR
0 1 Rx 7 BITS/CHAR
1 0 Rx 6 BITS/CHAR
1 1 Rx 8 BITS/CHAR
WR3
D0 Receiver Enable—A one enables the receiver to begin. This bit should be set only after the receiver has been initialized.

D1 Sync Character Load Inhibit—A one prevents the receiver from loading sync characters into the receive buffers. In SDLC, this bit must be zero.

D2 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 (OFFH) or the value loaded into WR6. This bit must be zero in non-SDLC modes.

D3 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.

D4 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.

D5 Auto Enable—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 (RRO).

D7, D6 Receive 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

WR4
D0 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.

Write Register 4 (WR4):

D1 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.

D3, D2 Stop bits-sync mode
0 0 Selects synchronous modes
0 1 Async mode, 1 stop bit/character
1 0 Async mode, 1½ stop bits/character
1 1 Async mode, 2 stop bits/character

D5, D4 Sync mode select
0 0 8-bit sync character
0 1 16-bit sync character
1 0 SDLC mode (Flag sync)
1 1 External sync mode

D7, D6 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  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):

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 (X¹⁶ + X¹⁵ + X² + 1) and a zero in this bit selects the CCITT-CRC polynomial (X¹⁶ + X¹² + X⁵ + 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

Bits to be sent must be right justified least significant bit first, e.g.:

<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>B0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>B1</td>
<td>B0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>B2</td>
<td>B1</td>
<td>B0</td>
<td>B0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>B3</td>
<td>B2</td>
<td>B1</td>
<td>B0</td>
<td>B0</td>
<td>B0</td>
</tr>
<tr>
<td>0</td>
<td>B4</td>
<td>B3</td>
<td>B2</td>
<td>B1</td>
<td>B0</td>
<td>B0</td>
<td>B0</td>
</tr>
</tbody>
</table>

Five or less mode allows transmission of one to five bits per character. The microprocessor must format the data in the following way:

- Sends one data bit
- Sends two data bits
- Sends three data bits
- Sends four data bits
- Sends five data bits
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>D5</td>
<td>D4</td>
</tr>
<tr>
<td>D3</td>
<td>D2</td>
</tr>
<tr>
<td>D1</td>
<td>D0</td>
</tr>
</tbody>
</table>

Least significant
Sync byte (Address in SDLC/HDL Mode)

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):

<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>

Most Significant
Sync byte (must be 01111110 in SDLC/HDL Mode)

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-vectored 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 \( |P| \) 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, af-
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 in a set condition following a reset (internal or external). The only command that can reset this bit is the Reset Transmit Underrun/EOM Latch command (WR0, D6 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 D2 D1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1 0 0</td>
<td>0</td>
<td>3</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>0 1 0</td>
<td>0</td>
<td>4</td>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td>1 1 0</td>
<td>0</td>
<td>5</td>
<td>0</td>
<td>4</td>
</tr>
<tr>
<td>0 0 1</td>
<td>0</td>
<td>6</td>
<td>0</td>
<td>5</td>
</tr>
<tr>
<td>1 0 1</td>
<td>0</td>
<td>7</td>
<td>0</td>
<td>6</td>
</tr>
<tr>
<td>0 1 1</td>
<td>0</td>
<td>8</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>1 1 1</td>
<td>1</td>
<td>8</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>0 0 0</td>
<td>2</td>
<td>8</td>
<td>1</td>
<td>7</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: 2
  - 0 0 1: 0
  - 0 1 0: 0
  - 0 1 1: 0
  - 1 0 0: 0
  - 1 0 1: 0
  - 1 1 0: 0
  - 1 1 1: 1

#### LAST DATA BYTE

- 8
- 6
- 4
- 8
- 3
- 7
- 5
- 8

#### RESIDUE DATA

- 8 BITS/CHAR. MODE

- **PARITY ERROR**
- **RX OVERRUN ERROR**
- **CRC/FRAMING ERROR**
- **END OF FRAME (SDLC/HDLC MODE)**
D7 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.

Read Register 2 (RR2):

![Diagram of RR2 Register]

RR2 Channel B
D7–D0 Interrupt Vector—Contains the interrupt vector programmed into WR2. If the status affects vector mode is selected (WR1; D2), 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.

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

NOTES:
1. RR0; D0 is reset automatically when the data is read.
2. RR0; 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.

Hardware Configuration, Polled Operation

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:

- Highest Priority
  - RxA
  - RxB
  - TxA
  - TxB
  - ExTA
  - ExTB

- Lowest Priority
  - RxA
  - TxA
  - RxB
  - TxB
  - ExTA
  - ExTB

The interrupt priority resolution works differently for vectored and non-vectored modes.
Interrupt Condition Grouping

- **CONDITION**
  - RECEIVE CHARACTER
  - PARITY ERROR
  - RECEIVE OVERRUN ERROR
  - FRAMING ERROR
  - END OF FRAME (SDLC ONLY)
  - FIRST DATA CHARACTER
  - FIRST NON-SYNC CHARACTER (SYNC MODES)
  - VALID ADDRESS BYTE (SDLC ONLY)
- **MODE**
  - INTERRUPT ON ALL RECEIVE CHARACTERS
  - INTERRUPT ON FIRST R CHARACTER
  - INTERRUPT ON FIRST CHARACTER
  - RECEIVE INTERRUPT
  - EXTERNAL/STATUS INTERRUPT
  - TRANSMIT INTERRUPT

---

**INTERNAL INTERRUPT REQUEST**

- **INTERNAL INTERRUPT ACCEPTED**
- **Lower Priority Interrupts Not Accepted**

---

**INTERRUPT (EXTERNAL)**

- **INTERRUPT (EXTERNAL)**
- **INTA (EXTERNAL)**
- **INTA (INTERNAL)**
- **IP0**

**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 IP1 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 IPO goes inactive. IPO and IP1 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 IP0 follows IP1.
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**

![Diagram of DMA Acknowledge Circuit]

**DMA Timing**

![Diagram of DMA Timing]

**DMA OPERATION**

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 WR2 Ch. A</th>
<th>Channel A</th>
<th>Channel B</th>
</tr>
</thead>
<tbody>
<tr>
<td>D1 D0</td>
<td>Wait Interrupt Polled</td>
<td>Wait Interrupt Polled</td>
</tr>
<tr>
<td>0 0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0 1</td>
<td>DMA Polled</td>
<td>Interrupt Polled</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.
NOTE:
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: 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.
**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:** This is a production data sheet. The specifications are subject to change without notice.

*WARNING:* Stressing the device beyond the "Absolute Maximum Ratings" may cause permanent damage. These are stress ratings only. Operation beyond the "Operating Conditions" is not recommended and extended exposure beyond the "Operating Conditions" may affect device reliability.

**D.C. CHARACTERISTICS** \( T_A = 0°C \) 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>+2.4</td>
<td></td>
<td>V</td>
<td>( I_{OH} = 200 ) μA</td>
</tr>
<tr>
<td>( I_{IL} )</td>
<td>Input Leakage Current</td>
<td>±10</td>
<td></td>
<td>μA</td>
<td>( V_{IN} = V_{CC} ) to 0V</td>
</tr>
<tr>
<td>( I_{OFL} )</td>
<td>Output Leakage Current</td>
<td>±10</td>
<td></td>
<td>μA</td>
<td>( V_{OUT} = V_{CC} ) to 0.45V</td>
</tr>
<tr>
<td>( I_{CC} )</td>
<td>( V_{CC} ) Supply Current</td>
<td>200</td>
<td></td>
<td>mA</td>
<td></td>
</tr>
</tbody>
</table>

**NOTE:**
1. For Extended Temperature EXPRESS, use MIL8274 electrical Parameters.

**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>Units</th>
<th>Test Conditions</th>
</tr>
</thead>
<tbody>
<tr>
<td>( C_{IN} )</td>
<td>Input Capacitance</td>
<td>10</td>
<td></td>
<td>pF</td>
<td>( f_o = 1 ) MHz</td>
</tr>
<tr>
<td>( C_{OUT} )</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>

2-144
### A.C. CHARACTERISTICS

$T_A = 0^\circ \text{C to } +70^\circ \text{C}; \ V_{CC} = +5\text{V} \pm 10\%$

<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></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>$t_{DF}$</td>
<td>Output Float Delay</td>
<td></td>
<td>120</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></td>
<td>250</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>$t_{PI}$</td>
<td>IPI Hold after INTA ↑</td>
<td>0</td>
<td></td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>$t_{II}$</td>
<td>INTA Pulse Width</td>
<td></td>
<td>250</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>$t_{IL}$</td>
<td>INTA ↓ to Data Output Delay</td>
<td>200</td>
<td></td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>$t_{RQ}$</td>
<td>RD or WR to DRQ ↓</td>
<td>150</td>
<td></td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>$t_{AV}$</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 RDYA or RDYB 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></td>
<td>tcy</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>$t_{TD}$</td>
<td>TxC to INT Delay</td>
<td>4</td>
<td>6</td>
<td>tcy</td>
<td></td>
</tr>
<tr>
<td>$t_{IR}$</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>$t_{IP}$</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

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".

WAVEFORMS

CLOCK CYCLE

READ CYCLE

DEVICE UNDER TEST

$C_L = 150 \text{ pF}$

$C_L$ Includes Jig Capacitance

2-146
WAVEFORMS (Continued)

WRITE CYCLE

DMA CYCLE

READ/WRITE CYCLE (SOFTWARE POLLED MODE)
NOTES:
1. INTA signal as RD signal.
2. IPI signal acts as CS signal.
WAVEFORMS (Continued)

TRANSMIT DATA CYCLE

RECEIVE DATA CYCLE

OTHER TIMING
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
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>DB0</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>DB1</td>
<td>1 2</td>
<td>I/O</td>
<td></td>
</tr>
<tr>
<td>DB2</td>
<td>39 44</td>
<td>I/O</td>
<td></td>
</tr>
<tr>
<td>DB3</td>
<td>2 3</td>
<td>I/O</td>
<td></td>
</tr>
<tr>
<td>DB4</td>
<td>38 43</td>
<td>I/O</td>
<td></td>
</tr>
<tr>
<td>DB5</td>
<td>3 4</td>
<td>I/O</td>
<td></td>
</tr>
<tr>
<td>DB6</td>
<td>37 42</td>
<td>I/O</td>
<td></td>
</tr>
<tr>
<td>DB7</td>
<td>4 5</td>
<td>I/O</td>
<td></td>
</tr>
<tr>
<td>INT</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>IEO</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>IEI</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>
</tbody>
</table>


<table>
<thead>
<tr>
<th>Symbol</th>
<th>Pin No.</th>
<th>Type</th>
<th>Name and Function</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>INTA</strong></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><strong>VCC</strong></td>
<td>9 10</td>
<td>O</td>
<td>POWER: +5V Power supply.</td>
</tr>
<tr>
<td><strong>RDY_A/REQ_A</strong></td>
<td>10 11 O</td>
<td>I/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><strong>SYNC_A</strong></td>
<td>11 12 I/O</td>
<td>I/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><strong>RTxC_A</strong></td>
<td>12 13 I</td>
<td>I</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><strong>RxD_A</strong></td>
<td>13 14</td>
<td>I</td>
<td>RECEIVE DATA: These lines receive serial data at standard TTL levels.</td>
</tr>
<tr>
<td><strong>TRxC_A</strong></td>
<td>14 15 I/O</td>
<td>I/O</td>
<td>TRANSMIT/RECEIVE CLOCKS: These pins can be programmed in several different modes of operation. TRxC 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><strong>TxD_A</strong></td>
<td>15 16</td>
<td>O</td>
<td>TRANSMIT DATA: These output signals transmit serial data at standard TTL levels</td>
</tr>
<tr>
<td><strong>DTR_A/REQ_A</strong></td>
<td>16 19 O</td>
<td>O</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><strong>TxD_B</strong></td>
<td>25 29</td>
<td>O</td>
<td></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></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>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>CD(_A)</td>
<td>19</td>
<td>22</td>
<td>I</td>
</tr>
<tr>
<td>CD(_B)</td>
<td>21</td>
<td>24</td>
<td>I</td>
</tr>
<tr>
<td>CLK</td>
<td>20</td>
<td>23</td>
<td>I</td>
</tr>
<tr>
<td>GND</td>
<td>31</td>
<td>35</td>
<td>I</td>
</tr>
<tr>
<td>D/Ć</td>
<td>32</td>
<td>37</td>
<td>I</td>
</tr>
<tr>
<td>ĆS</td>
<td>33</td>
<td>38</td>
<td>I</td>
</tr>
<tr>
<td>A/B</td>
<td>34</td>
<td>39</td>
<td>I</td>
</tr>
<tr>
<td>WR</td>
<td>35</td>
<td>40</td>
<td>I</td>
</tr>
<tr>
<td>RD</td>
<td>36</td>
<td>41</td>
<td>I</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 SDLT. This versatile device supports virtually any serial data transfer application (Terminal, Personal Computer, Peripherals, Industrial Controller, Telecommunication sytem, 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 MCS51/68, 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 modem 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 CRC) 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</td>
<td>WR0</td>
</tr>
<tr>
<td>Transmit/Receive buffer status and External status</td>
<td>CRC initialize, initialization commands for the various modes, shift right/shift left command</td>
</tr>
<tr>
<td>RR1</td>
<td>WR1</td>
</tr>
<tr>
<td>Special Receive Condition status</td>
<td>Transmit/Receive interrupt and data transfer mode definition</td>
</tr>
<tr>
<td>RR2</td>
<td>WR2</td>
</tr>
<tr>
<td>Modified interrupt vector (Channel B only)</td>
<td>Interrupt vector (accessed through either channel)</td>
</tr>
<tr>
<td>Unmodified interrupt (Channel A only)</td>
<td></td>
</tr>
<tr>
<td>RR3</td>
<td>WR3</td>
</tr>
<tr>
<td>Interrupt Pending bits (Channel A only)</td>
<td>Receive parameters and control</td>
</tr>
<tr>
<td>RR8</td>
<td>WR4</td>
</tr>
<tr>
<td>Receive buffer</td>
<td>Transmit/Receive miscellaneous parameters and modes</td>
</tr>
<tr>
<td>RR10</td>
<td>WR5</td>
</tr>
<tr>
<td>Miscellaneous status</td>
<td>Transmit parameters and controls</td>
</tr>
<tr>
<td>RR12</td>
<td>WR6</td>
</tr>
<tr>
<td>Lower byte of baud rate generator time constant</td>
<td>Sync characters or SDLC address field</td>
</tr>
<tr>
<td>RR13</td>
<td>WR7</td>
</tr>
<tr>
<td>Upper byte of baud rate generator time constant</td>
<td>Sync character or SDLC flag</td>
</tr>
<tr>
<td>RR15</td>
<td>WR8</td>
</tr>
<tr>
<td>External/Status interrupt information</td>
<td>Transmit buffer</td>
</tr>
<tr>
<td></td>
<td>WR9</td>
</tr>
<tr>
<td></td>
<td>Master interrupt control and reset (accessed through either channel)</td>
</tr>
<tr>
<td></td>
<td>WR10</td>
</tr>
<tr>
<td></td>
<td>Miscellaneous transmitter/receiver control bits</td>
</tr>
<tr>
<td></td>
<td>WR11</td>
</tr>
<tr>
<td></td>
<td>Clock Mode control</td>
</tr>
<tr>
<td></td>
<td>WR12</td>
</tr>
<tr>
<td></td>
<td>Lower Byte of baud rate generator time constant</td>
</tr>
<tr>
<td></td>
<td>WR13</td>
</tr>
<tr>
<td></td>
<td>Upper Byte of baud rate generator time constant</td>
</tr>
<tr>
<td></td>
<td>WR14</td>
</tr>
<tr>
<td></td>
<td>Miscellaneous control bits</td>
</tr>
<tr>
<td></td>
<td>WR15</td>
</tr>
<tr>
<td></td>
<td>External/Status interrupt control</td>
</tr>
</tbody>
</table>
Figure 3: Data Path

CPU I/O

I/O DATA BUFFER

INTERNAL DATA BUS

BR GENERATOR INPUT

16-BIT DOWN COUNTER

2

BR GENERATOR OUTPUT

SYNC REGISTER & ZERO DELETE

2 BITS

RECEIVE ERROR LOGIC

SYNC CRC

RECEIVE DATA

FIFO

WR SYNC REGISTER

WR REGISTER

TRANSMIT DATA

START BIT

20-BIT TRANSMIT

SHIFT REGISTER

ZERO INSERT (5 BITS)

CRC GENERATOR

TRANSMIT MUX & 2-BIT DELAY

TRANSIT CLOCK

NRZI ENCODE

FINAL TMUX

TxD

RxD

1 BIT

MUX

NRZI DECODE

SOLC-CRC

CRC CHECKER

CRC RESULT

DPLL

DPLL OUTPUT

BR GENERATOR OUTPUT

DPLL OUTPUT

TNC

RTNC

SYNC (OSCILLATOR)

CLOCK MUX

RECEIVE CLOCK

TRANSIT CLOCK

DPLL CLOCK

BR GENERATOR CLOCK

INTERNAL TxD

MUX

INTERNAL

HUNT MODE (BSYNC)

SYNC REGISTER (5 BITS)

CRC DELAY REGISTER (5 BITS)

230834-3
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 hand-shaking 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, 1/8, 1/2 or 1/4 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](230834-4)
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 000111010000001111.

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.

![Figure 5. Detecting 5- or 7-Bit Synchronous Characters](image-url)
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](image)

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.)

\[
\text{time constant} = \frac{\text{BR clock frequency}}{2 \times \text{baud rate} \times \text{clock mode}} - 2
\]
Table 3. Time Constant Values for Standard Baud Rates at BR Clock = 3.9936 MHz

<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>39354</td>
<td>—</td>
</tr>
</tbody>
</table>

DIGITAL PHASE LOCKED LOOP

The SCC contains a digital phase locked-loop (DPLL) to recover clock information from a datastream 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 \( \frac{15}{16} \) counting transition.

The 32X clock for the DPLL can be programmed to come from either the RTxXC input or the output of the baud rate generator. The DPLL output may be programmed to be echoed out of the SCC via the TRxXC 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).

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 (RTxXC and TRxXC) 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-clock operation</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NRZI</td>
<td>4 MHz</td>
<td>32</td>
<td>125 Kbps</td>
<td></td>
</tr>
<tr>
<td></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 &quot;Point High&quot; 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>1</td>
<td>(0)</td>
</tr>
<tr>
<td>Low</td>
<td>Not True</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>(1)</td>
</tr>
<tr>
<td>Low</td>
<td>Not True</td>
<td>1</td>
<td>1</td>
<td>0</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>0</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>

Table 5. Register Addressing
First write the appropriate code into WR0, 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**

**Figure 10. Write Register Formats**
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.
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: This is a production data sheet. The specifications are subject to change without notice.

*WARNING: Stressing the device beyond the "Absolute Maximum Ratings" may cause permanent damage. These are stress ratings only. Operation beyond the "Operating Conditions" is not recommended and extended exposure beyond the "Operating Conditions" 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>$V_{IL}$</td>
<td>Input Low Voltage</td>
<td>−0.3</td>
<td>+0.8</td>
<td>V</td>
<td></td>
</tr>
<tr>
<td>$V_{IH}$</td>
<td>Input High Voltage</td>
<td>+2.4</td>
<td>$V_{CC} + 0.3$</td>
<td>V</td>
<td></td>
</tr>
<tr>
<td>$V_{OL}$</td>
<td>Output Low Voltage</td>
<td>+0.45</td>
<td>V</td>
<td>$I_{OL} = 2.0 mA$</td>
<td></td>
</tr>
<tr>
<td>$V_{OH}$</td>
<td>Output High Voltage</td>
<td>+2.4</td>
<td>V</td>
<td>$I_{OH} = -250 \mu A$</td>
<td></td>
</tr>
<tr>
<td>$I_{IL}$</td>
<td>Input Leakage Current</td>
<td>±10</td>
<td>\mu A</td>
<td>0.4V to 2.4V</td>
<td></td>
</tr>
<tr>
<td>$I_{OL}$</td>
<td>Output Leakage Current</td>
<td>±10</td>
<td>\mu A</td>
<td>0.4V to 2.4V</td>
<td></td>
</tr>
<tr>
<td>$I_{CC}$</td>
<td>$V_{CC}$ Supply Current</td>
<td>250</td>
<td>mA</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

CAPACITANCE $T_C = 25°C$; $V_{CC} = GND = 0V$

<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  $T_C = 0^\circ C \text{ to } +70^\circ C; V_{CC} = +5V \pm 5\%$

READ AND WRITE TIMING

<table>
<thead>
<tr>
<th>Number</th>
<th>Symbol</th>
<th>Parameter</th>
<th>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>tCL</td>
<td>105</td>
<td>2000</td>
<td>ns</td>
</tr>
<tr>
<td>2</td>
<td>tCH</td>
<td>CLK High Time</td>
<td>tCH</td>
<td>105</td>
<td>2000</td>
<td>ns</td>
</tr>
<tr>
<td>3</td>
<td>tf</td>
<td>CLK Fall Time</td>
<td>tf</td>
<td>20</td>
<td>10</td>
<td>ns</td>
</tr>
<tr>
<td>4</td>
<td>tr</td>
<td>CLK Rise Time</td>
<td>tr</td>
<td>20</td>
<td>15</td>
<td>ns</td>
</tr>
<tr>
<td>5</td>
<td>tCY</td>
<td>CLK Cycle Time</td>
<td>tCY</td>
<td>250</td>
<td>4000</td>
<td>ns</td>
</tr>
<tr>
<td>6</td>
<td>tAW</td>
<td>Address to WR ↓ Setup Time</td>
<td>tAW</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>tWA</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>tAR</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>tRA</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>tIC</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>tIW</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>tWI</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>tIR</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>tRI</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>tCI</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>tCLW</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>tWCS</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>tCHW</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>tCLR</td>
<td>0</td>
<td>0</td>
<td>ns</td>
</tr>
<tr>
<td>20</td>
<td>tCRS</td>
<td>CS to RD ↑ Hold Time (Note 1)</td>
<td>tCRS</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>tCHR</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>tRR</td>
<td>390</td>
<td>150</td>
<td>ns</td>
</tr>
<tr>
<td>23</td>
<td>Null</td>
<td>Parameter Deleted</td>
<td>Null</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>24</td>
<td>tRDI</td>
<td>RD ↑ to Data Not Valid Delay</td>
<td>tRDI</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>tRDV</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>tDF</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

\[ \text{DEVICE UNDER TEST} \]
\[ \text{CL} - 150 \text{pF} \]

\( C_L = 150 \text{ pF} \)
\( C_L \) Includes Jig Capacitance

OPEN DRAIN TEST LOAD

Figure 14. Read and Write Timing

2-177
## 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></td>
<td></td>
<td></td>
<td>Min</td>
<td>Max</td>
<td>Min</td>
</tr>
<tr>
<td>27</td>
<td>tAD</td>
<td>Address Required Valid to Read Data Valid Delay</td>
<td>590</td>
<td>325</td>
<td>ns</td>
</tr>
<tr>
<td>28</td>
<td>TWW</td>
<td>WR Low Time</td>
<td>390</td>
<td>60</td>
<td>ns</td>
</tr>
<tr>
<td>29</td>
<td>tDW</td>
<td>Data to WR ↓ Setup Time</td>
<td>0</td>
<td>0</td>
<td>ns</td>
</tr>
<tr>
<td>30</td>
<td>tWD</td>
<td>Data to WR ↑ Hold Time</td>
<td>0</td>
<td>0</td>
<td>ns</td>
</tr>
<tr>
<td>31</td>
<td>tWRV</td>
<td>WR ↓ 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 ↓ 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 ↓ to READY/REQ Not Valid Delay</td>
<td>240</td>
<td>200</td>
<td>ns</td>
</tr>
<tr>
<td>34</td>
<td>tRRRI</td>
<td>RD ↓ 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 ↑ 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 ↑ 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 ↓ (Acknowledge) Delay (Note 5)</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 ↓ (Acknowledge) to Read Data Valid Delay</td>
<td>190</td>
<td>100</td>
<td>ns</td>
</tr>
<tr>
<td>40</td>
<td>tEI</td>
<td>IEI to RD ↓ (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 ↑ (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 ↑ to IEO Delay</td>
<td>250</td>
<td>250</td>
<td>ns</td>
</tr>
<tr>
<td>44</td>
<td>tRII</td>
<td>RD ↓ 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 ↑ to WR ↓ Delay for No Reset</td>
<td>30</td>
<td>15</td>
<td>ns</td>
</tr>
<tr>
<td>46</td>
<td>tWR</td>
<td>WR ↑ to RD ↓ 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 (Note 3)</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>trRC</td>
<td>RxC ↑ to CLK ↑ Setup Time (Notes 1, 4)</td>
<td>100</td>
<td>100</td>
<td></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></td>
</tr>
<tr>
<td>3</td>
<td>trRCR</td>
<td>RxD to RxC ↑ Hold Time (X1 Mode) (Note 1)</td>
<td>150</td>
<td>150</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>tIDRC</td>
<td>RxD to RxC ↓ Setup Time (X1 Mode) (Notes 1, 5)</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>trRCD</td>
<td>RxD to RxC ↓ Hold Time (X1 Mode) (Notes 1, 5)</td>
<td>150</td>
<td>150</td>
<td></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></td>
</tr>
<tr>
<td>7</td>
<td>trRCS</td>
<td>SYNC to RxC ↑ Hold Time (Note 1)</td>
<td>3 tCY + 200</td>
<td>3 tCY + 200</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>tITCC</td>
<td>TxC ↓ to CLK ↑ Setup Time (Notes 2, 4)</td>
<td>100</td>
<td>100</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>tITCT</td>
<td>TxC ↓ to TxD Delay (X1 Mode) (Note 2)</td>
<td>300</td>
<td>300</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>tITCD</td>
<td>TxC ↑ to TxD Delay (X1 Mode) (Notes 2, 5)</td>
<td>300</td>
<td>300</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>tITDT</td>
<td>TxD to TRxC Delay (Send Clock Echo)</td>
<td>200</td>
<td>200</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>tdDCH</td>
<td>RTxC High Time</td>
<td>180</td>
<td>150</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>tdDCL</td>
<td>RTxC Low Time</td>
<td>180</td>
<td>150</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>tdDCY</td>
<td>RTxC Cycle Time</td>
<td>4tCY</td>
<td>4tCY</td>
<td></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>trRCH</td>
<td>TRxC High Time</td>
<td>180</td>
<td>150</td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>trRCL</td>
<td>TRxC Low Time</td>
<td>180</td>
<td>150</td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>trRCY</td>
<td>TRxC Cycle Time (Note 6)</td>
<td>4tCY</td>
<td>4tCY</td>
<td></td>
</tr>
<tr>
<td>19</td>
<td>tCC</td>
<td>CD or CTS Pulse Width</td>
<td>200</td>
<td>200</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>tCSS</td>
<td>SYNC Pulse Width</td>
<td>200</td>
<td>200</td>
<td></td>
</tr>
<tr>
<td>21</td>
<td>tWRT</td>
<td>WR to RTS Valid Delay</td>
<td>6tCY</td>
<td>6tCY</td>
<td></td>
</tr>
<tr>
<td>22</td>
<td>tWDT</td>
<td>WR to DTR Valid Delay</td>
<td>5tCY</td>
<td>5tCY</td>
<td></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
## CONTENTS

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.0 INTRODUCTION</td>
<td>2-184</td>
</tr>
<tr>
<td>1.1 Goal</td>
<td>2-184</td>
</tr>
<tr>
<td>1.2 Scope</td>
<td>2-184</td>
</tr>
<tr>
<td>2.0 82510 DESCRIPTION</td>
<td>2-184</td>
</tr>
<tr>
<td>2.1 Overview</td>
<td>2-184</td>
</tr>
<tr>
<td>2.2 Bus Interface Unit (BIU)</td>
<td>2-184</td>
</tr>
<tr>
<td>2.3 Receive Machine (RXM)</td>
<td>2-184</td>
</tr>
<tr>
<td>2.4 Transmit Machine (TXM)</td>
<td>2-185</td>
</tr>
<tr>
<td>2.5 Modem Interface Module</td>
<td>2-185</td>
</tr>
<tr>
<td>2.6 Timing Unit</td>
<td>2-185</td>
</tr>
<tr>
<td>2.7 FIFOs, Rx and Tx</td>
<td>2-186</td>
</tr>
<tr>
<td>3.0 HARDWARE DESIGN</td>
<td>2-186</td>
</tr>
<tr>
<td>3.1 System Interface</td>
<td>2-186</td>
</tr>
<tr>
<td>3.2 Reset</td>
<td>2-194</td>
</tr>
<tr>
<td>3.3 System Clock Options</td>
<td>2-195</td>
</tr>
<tr>
<td>4.0 INTERRUPT BEHAVIOR</td>
<td>2-197</td>
</tr>
<tr>
<td>4.1 FIFO Usage</td>
<td>2-197</td>
</tr>
<tr>
<td>4.2 Interrupt Handling</td>
<td>2-198</td>
</tr>
<tr>
<td>4.3 Polling</td>
<td>2-205</td>
</tr>
<tr>
<td>5.0 SOFTWARE CONSIDERATIONS</td>
<td>2-205</td>
</tr>
<tr>
<td>5.1 Configuration</td>
<td>2-205</td>
</tr>
<tr>
<td>5.2 Transmit Operation</td>
<td>2-208</td>
</tr>
<tr>
<td>5.3 Data Reception</td>
<td>2-213</td>
</tr>
<tr>
<td>5.4 Timer Usage</td>
<td>2-218</td>
</tr>
<tr>
<td>6.0 82510 IMPLEMENTATION OF XMODEM</td>
<td>2-221</td>
</tr>
<tr>
<td>6.1 XMODEM Protocol</td>
<td>2-221</td>
</tr>
<tr>
<td>6.2 Software</td>
<td>2-221</td>
</tr>
<tr>
<td>6.3 Software Listings</td>
<td>2-229</td>
</tr>
</tbody>
</table>
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-
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:

- 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 A 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</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>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</td>
<td>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</td>
<td>01H</td>
</tr>
<tr>
<td>LCR</td>
<td>DLAB Divisor Latch Access bit</td>
<td>Set Break</td>
<td>Parity Mode bit 2</td>
<td>Parity Mode bit 1</td>
<td>Stop bit Length bit 0</td>
<td>Character Length bit 1</td>
<td>Character Length bit 0</td>
<td></td>
<td>3</td>
<td>00H</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</td>
<td>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</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>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>
<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</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>RxF</td>
<td>—</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></td>
<td>1</td>
<td>—</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>1</td>
<td>—</td>
<td></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</td>
<td>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</td>
<td>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>
<td>—</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</td>
<td>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>Tx FIFO Level</td>
<td>4</td>
<td>00H</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RST</td>
<td>Address/ Control Character Received</td>
<td>Address/ Control Character Match</td>
<td>Break Terminated</td>
<td>Break Detected</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>
</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>5</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 in CTS</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>Flush Tx FIFO</td>
<td>Tx Enable</td>
<td>Tx Disable</td>
<td>6</td>
<td>—</td>
</tr>
<tr>
<td>GSR</td>
<td>—</td>
<td>—</td>
<td>Timer Interrupt</td>
<td>TxM Interrupt</td>
<td>Modern Interrupt</td>
<td>RxM 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>Software Reset</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</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>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</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>ulan Mode Select</td>
<td>Loopback or Echo Mode of Operation</td>
<td>4</td>
<td>0CH</td>
</tr>
<tr>
<td>ACR1</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</td>
<td>Address/ Control Character Recognition Interrupt Enable</td>
<td>Address/ Control Character Match</td>
<td>Break Terminated 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>0</td>
<td>00H</td>
</tr>
<tr>
<td>BACF</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</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)

2-188
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 (Tavrl)
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 (Tavwl)
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 - Tclav_{max} + Tclrl_{min} - Latch Delay_{max}

Read Access Time = 2Tclcl - Tclrl_{max} - Tdvcl - Transceiver Delay_{max}

Read Active Time = Trlrh = 2Tclcl - 46

Write Cycle:

Address Valid to Write Active = Tclcl + Tcvctv_{min} - Tclav_{max} - Latch Prop. Delay_{max}

Write Active Time = Twlwh = 2Tclcl - 40

Data Valid to Write Inactive = 2Tclcl - Tcldv_{max} - Transceiver Delay_{max} + Tcvctx_{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 (8250) 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:**

\[
\text{Address Valid to Read Active} = T1 (\text{CLK period}) + T29_{\text{min}} (\text{CLK to cmd active}) - T16_{\text{max}} (\text{ALE active delay}) - \text{Latch Prop. Delay}_{\text{max}}
\]

\[
\text{Read Access to Data Valid} = 2T1 (\text{CLK period}) - T29_{\text{max}} (\text{CLK to cmd active}) - T8 (\text{Read Data Setup Time}) - \text{Transceiver Delay}_{\text{max}}
\]

**Read Active Time** = \(2T1 (\text{CLK period}) - T29_{\text{max}} (\text{CLK to cmd active}) + T30_{\text{min}} (\text{CLK to cmd inactive})\)

**Write Cycle:**

\[
\text{Address to Write Low} = T1 (\text{CLK period}) + T29_{\text{min}} (\text{CLK to cmd active}) - T16_{\text{max}} (\text{ALE active delay}) - \text{Latch Delay}_{\text{max}}
\]

\[
\text{Write Active Time} = 2T1 (\text{CLK period}) - T29_{\text{max}} (\text{CLK to cmd active}) + T30_{\text{min}} (\text{CLK to cmd inactive})
\]

\[
\text{Data to Write High} = 3T1 - T14_{\text{min}} (\text{Write Data Valid Delay}) + T30_{\text{min}} (\text{CLK to cmd inactive}) - \text{Xcvr. Delay}_{\text{max}}
\]

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:**

\[
\text{Read Access to Data Valid} = 253.25 \text{ ns}
\]

\[
82510 \text{T}_{\text{ldv}} = 308 \text{ ns}
\]

additional time reqd. = 308-253.25 = 54.75 ns

\[
\text{Read Active Width} = 269.25 \text{ ns}
\]

\[
82510 \text{T}_{\text{rh}} = 308 \text{ ns}
\]

additional time reqd. = 308-269.25 = 38.75 ns

\[
\text{Address Valid to Read Active} = 132.75 \text{ ns}
\]

\[
82510 T_{\text{AVRL}} = 7 \text{ ns}
\]

Since each additional wait state adds 62.5 ns at 16 MHz, the 82510 requires one additional wait state.
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 TAVWL = 7 ns
Write Active Time = 300.5 ns
82510 TWLWH = 231 ns
Data to Write High = 289.5 ns
82510 TDVWH = 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 cannot 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 rates generation to be done in a manner compatible with the PC software. The PC software calculates baud rates based on a source frequency of 1.8432 MHz. The 82510 system clock (18.432 MHz) is divided by two before being fed to BRG A and then is again divided by five (BRG B default). This causes the frequency to be divided by ten before being fed into BRG A. 18.432 divided by ten yields 1.8432 MHz, so in effect the BRG A is generating baud rates from a source frequency of 1.8432 MHz (which is 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>
</tr>
</thead>
<tbody>
<tr>
<td>Auto Acknowledge</td>
</tr>
<tr>
<td>All Interrupts Disabled</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>RECEIVE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stand Ctl. Char. Recogn. disabled</td>
</tr>
<tr>
<td>Digital Phase Locked Loop (DPLL) disabled</td>
</tr>
<tr>
<td>3/16 Sampling</td>
</tr>
<tr>
<td>Majority Vote Start bit</td>
</tr>
<tr>
<td>Non µlan (Normal) mode</td>
</tr>
<tr>
<td>BkD, FE, OE, PE Int. enabled</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>FIFO</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rx FIFO Depth = 1</td>
</tr>
<tr>
<td>Tx FIFO Threshold = 0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>AUTO ECHO</th>
</tr>
</thead>
<tbody>
<tr>
<td>Disabled</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>LOOP BACK</th>
</tr>
</thead>
<tbody>
<tr>
<td>Configured for Local Loopback</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>CLOCK OPTIONS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Baud Rate = 57.6K</td>
</tr>
<tr>
<td>Rx Clock = 16 x</td>
</tr>
<tr>
<td>Rx Clock Source = BRG B</td>
</tr>
<tr>
<td>Tx Clock = 16 x</td>
</tr>
<tr>
<td>Tx Clock Source = BRG B</td>
</tr>
<tr>
<td>BRG A Mode = BRG</td>
</tr>
<tr>
<td>BRG A Source = Sys. Clock</td>
</tr>
<tr>
<td>BRG B Mode = BRG</td>
</tr>
<tr>
<td>BRG B Source = BRG A Output</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>TRANSMIT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Manual Control of RTS</td>
</tr>
<tr>
<td>1 Stop Bit</td>
</tr>
<tr>
<td>No Parity</td>
</tr>
<tr>
<td>5 Bit Character</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</td>
<td>100–900 µA</td>
<td>Not Preserved, Must be Reprogrammed</td>
</tr>
<tr>
<td></td>
<td>Automatically Disabled</td>
<td>Read FLR and Issue S/W Reset</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>External Clock</td>
<td>Enable External Clock, Read FLR and Issue S/W Reset, H/W Reset</td>
<td>100–900 µA</td>
<td>Not Preserved, Must be Reprogrammed</td>
</tr>
<tr>
<td></td>
<td>Must be Disabled by User</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Idle</td>
<td>External Clock Running</td>
<td>H/W Reset Read FLR</td>
<td>1–3 mA</td>
<td>All Data Preserved, Does Not Need to be Reprogrammed</td>
</tr>
</tbody>
</table>

2-197
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):

\[
(1) \text{Character Transmission Time} = \frac{\text{Num. of Bits per Character Frame}}{\text{Baud Rate}}
\]

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):

\[
(2) \text{Int_to_service latency} < \text{FIFO Size} \times \text{Character Transmission Time}
\]

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.

Going back to equation (2):

\[
\text{Int_to_service latency} < \text{Buffer size} \times \frac{10}{\text{baud rate}}
\]

\[
\text{Int_to_service latency} = \# \text{ of Channels} \times (\# \text{ of int. sources per channel}) \times \text{Time required to service interrupt}
\]

The Time required to service interrupt has been calculated to be 100 \(\mu\)s for a slightly optimized service routine. RMX86 interrupt service time is given as 250 \(\mu\)s and for other operating systems it should be slightly higher.

\[
\text{Int_to_service latency} = 4 \times 2 \times \text{Time required to service interrupt}
\]

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 acknowledgement—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.
Figure 10. Automatic Acknowledge Mode Operation

NOTE:
Vector refers to GIR bit (3-0)
82510: Manual Ack. Mode
8259A: Edge Triggered Non AEOI

Figure 11. Manual Acknowledge Mode Operation

NOTE:
Vector refers to GIR bit (3-0)
82510: Manual Ack. Mode
8259A: Edge Triggered Non AEOI
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 Acknowledgement 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).

![Diagram of Typical Interrupt Handler]

Figure 12. Typical Interrupt Handler
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

```
USER PRIORITY
  = RX FIFO (HI)
  RX MACHINE
  + TIMER
  TX FIFO
  TX MACHINE
  MODEM (LOW)

READ GSR

(GSR AND 01H) = 01H
YES
SERVICE RX FIFO

(GSR AND 04H) = 04H
YES
SERVICE RX MACHINE

(GSR AND 20H) = 20H
NO
SERVICE TIMER

(GSR AND 12H) > 0
YES
SERVICE TX MACHINE/ TX FIFO

(GSR AND 08H) = 08H
YES
SERVICE MODEM

EXIT
```

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 indentification 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) GSR (5)</td>
<td>TMIE (1-0) GER (5)</td>
<td>Read TMST</td>
<td>Issue Status Clear (StC)</td>
</tr>
<tr>
<td>Tx Machine</td>
<td>GSR (4) LSR (6)</td>
<td>GER (4)</td>
<td>Write Character to Tx FIFO</td>
<td>Issue StC</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>
<td>Issue StC</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(s)</td>
<td>Issue StC</td>
</tr>
<tr>
<td>Tx FIFO</td>
<td>LSR (5) GSR (1)</td>
<td>GER (1)</td>
<td>Write to FIFO Read GIR</td>
<td>Issue StC</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>
<td>Issue StC</td>
</tr>
</tbody>
</table>

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, normal 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 Sources
13. Configure BACF register for BRG A mode and source.
14. Load BBL and BBH if BRGB 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
IN CLOCKS CONFIGURE
REGISTER: SELECT SOURCES
OF TX & RX CLOCKS
SELECT MODES OF
TX & RX CLOCKS

CONFIGURE BRG A CLOCK
SOURCE AND MODE OF
OPERATION VIA BRG A
CONFIGURATION REGISTER

BRG B IS
BEING USED
AS EITHER TIMER
OF BRG

NO
YES
LOAD BAL
LOAD BBH

CONFIGURE BRG B FOR
SOURCE AND MODE VIA BRG B
CONFIGURATION REGISTER
(IF BEING USED AS A BRG)

NEED TO REDEFINE
MULTIFUNCTION PINS

NO
YES
CONFIGURE DIRECTION AND
FUNCTION FOR MULTIFUNCTION
PINS VIA I/O PIN REGISTER

ENABLE INTERRUPTS
(IF NECESSARY) ON
MODEM INPUT PINS VIA
MODEM INTERRUPT
ENABLE REGISTER

ENABLE TIMER INTERRUPTS
AS NECESSARY, VIA
TIMER INTERRUPT
ENABLE REGISTER

E
NO
YES
USING INTERRUPTS

SWITCH TO
BANK THREE

MEDIUM AND
TIMING UNIT
CONFIGURATION

D
BANK ZERO
DISABLE ALL
INTERRUPTS TO CPU
ENABLE THE 82510
BLOCK INTERRUPT
SOURCES VIA GENERAL
ENABLE REGISTER
SWITCH TO
BANK ONE
GENERAL
OPERATION

E
NO
YES
USING S/W
PARITY ADDRESS OR
NINE DATA BITS
SET BITS IN THE
TRANSMIT FLAGS REGISTER
ENABLE CPU INTERRUPTIONS
(IF USING INTERRUPTS)
END OF CONFIGURATION

Figure 14. Configuration Flow Chart (Continued)
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 \( \mu \text{s} \) 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 \( \mu \text{s} \) then the Tx FIFO 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 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.

5.2.1.1 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.

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.
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.

![Rx FIFO Hysteresis](image)

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 RXD 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 RXD 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
```c
#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, lvI, 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 ((lvI = ((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

**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 Ctl-C, or XOFF and Ctl-C, can use just the Control Character Match mode by programming the registers ACRO and ACR1. If the CPU needs to process text on a line by line basis, the standard Control Character recognition capability can be used to determine when an end of line has occurred e.g. a whole line has been received when a Carriage Return (CR) or Line Feed (LF) is received by the UART.

Implementation of a character oriented asynchronous file transfer protocol can be done using both standard and specific Control Character Recognition. In such protocols most control characters such as Start of Header (SOH), can only be received during certain states, these characters can be received via Standard Control Character Recognition. A few Control Characters (e.g. abort) can be received at any stage of communication, these can be received by using the Control Character Matching capabilities of the 82510.

**5.3.3 BREAK RECEPTION**

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 theReceive 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.
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.

Figure 24. Timer use as Time Base for Transmit and Receive
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 an 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 27. Protocol Flow for Transmit Side of XMODEM
Figure 28. Protocol Flow for Receive Side of XMODEM
Figure 29. Transmit State Machine
Figure 30. Rx State Machine
START
Initialization
WHILE (NOT QUIT) {
   UPDATE STATUS ON SCREEN
   IF (KEYBOARD HIT)
      THEN PROCESS COMMAND
      PROCESS TRANSMIT STATE MACHINE
   END
   PROCESS RECEIVE STATE MACHINE
} END

Figure 31. Software Structure

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 cannot 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.
6.3 Software Listings

```c
1. #include "C:\ftp\ftp.c"
2. #include "C:\sys\syslib.h"
3. #include "C:\hello\stdio.h"
4. #include "C:\hello\stdlib.h"
5. #include "C:\hello\fcntl.h"
6. #include "C:\hello\unistd.h"
7. /**** SEPTEMBER 1986 ****/
8. /**** ****/
9. /**** 82510 XMODEM IMPLEMENTATION ****/
10. /**** ****/
11. int eof = false;  // end of file flag #/
12. int rsk = 0;
13. int tafix;
14. int rafix;
15. int exp_pkt_num = 1;  // next packet number expected by receiver */
16. int pkt;
17. int retcnt;  // Time Out counter for receiver */
18. int quit = false;
19. int key = 0;
20. int sohcnt = 0;  // # of SOH characters received */
21. int rafcnt = 0;  // # of Rx FIFO Interrupts */
22. int cccnt = 0;  // # of Ctl-Char Interrupts */
23. int ts_state = ts_idle;  // Transmitter State Variable */
24. int rs_state = rs_idle;  // Receiver State Variable */
25. int ts_cmd = inactive;  // Indicates a valid Tx Command was given */
26. int rs_cmd = inactive;  // Indicates a valid Rx Command was issued */
27. 28. /* File to be Transmitted */
29. char ts_file_name[40] = "
30. 31. /* File to be Received */
32. char rs_file_name[40] = "
33. 34. /* send_crc_req = inactive;  // request to Tx Ctl-Char */
35. int intec = 0;  // contains the CRC vector */
36. int 1;
37. char tsdata[132];  // Tx Buffer */
38. char rsbuf[128];  // Rx Buffer */
39. char rsdata[128];
40. char ts_lbuf[32000];  // Rx File Stored in this buffer */
41. 42. 43. /* ts_state variables ****/
44. 45. int ts_index;  // Pointer to the next character in the buffer */
46. 47. 48. struct packet {
49.     char head;
50.     char pack_num;
51.     char pack_cplt;
52.     char buffer[128];
53.     char chksm;
54. }
55. 56. struct packet rsbuf, tsbuf;  
57.
```

82510 XMODEM Implementation

2-229
PAGE 2  MAIN PROGRAM ftp.c 82510 XMODEM

58. /*********************************************************
59. *** tx State machine and interrupt **********
60. *** handler flags **********
61. /*********************************************************
62. /*********************************************************
63. // tx and tx fifo */
64. int tx_req =0;  /* Flag - indicates a request for transmission to
65. 82510 Interrupt Handler */
66. int ccr_to_tx = 0; /* Actual Ctl-char to Transmit */
67. int tx_byte_cnt =0; /* Total # of Bytes Transmitted */
68. int ptrs_sent =0; /* # of Packets sent */
69. 
70. /** Timer **/
71. int tx_time_cnt =0; /* Transmitter Timer Counter */
72. 
73. /** CCR **/
74. int get_ccr_rq =0; /* Flag - Request to receive Ctl-character */
75. int ccr_to_get =0; /* Received Ctl-char value */
76. 
77. /*********************************************************
78. *** RX STATE VARIABLES **********
79. /*********************************************************
80. /*********************************************************
81. char pk_chksum; /* Calculated Chksum */
82. int eot_cnt =0; /* # of EOTs Received */
83. int bad_pkt_cnt; /* # of Bad Packets Received */
84. 
85. /*********************************************************
86. *** rx state machine and Interrupt **********
87. *** handler flags **********
88. /*********************************************************
89. 
90. /*********************************************************
91. /** rx fifo **/
92. int rx_byte_cnt =0; /* # of Bytes Received */
93. 
94. /** CCR **/
95. 
96. int cti_rad_flg =0; /* Flag indicating that a Ctl-Char has been
97. received*/
98. int rx_cri_chr=0; /* Actual Ctl-char received */
99. 
100. /** Timer **/
101. int rx_time_cnt =0; /* Receive Timer Count */

82510 XMODEM Implementation (Continued)
82510 XMODEM Implementation (Continued)

2-231
PAGE 4

MAIN PROGRAM ftp.c 82510 XMODEM

146.  mov_curs (7,1);
147.  printf ("bytes received %d",txByte_cnt);
148.  mov_curs (7,30);
149.  printf ("Bytes Sent = %d",txByte_cnt);
150.  mov_curs (7,50);
151.  printf ("EOT count %d",seq_cnt);
152.  mov_curs (3,30);
153.  printf ("pkts sent = %d",pkts_sent);
154.  mov_curs (6,30);
155.  printf ("pkts sent = %d",pkts_sent);
156.  txSecs = txTime_cnt/100;
157.  RXSecs = RA_time_cnt/100;
158.  open_wind (3,1, "Rx Timer");
159.  printf (" = RX secs",rxSecs);
160.  print (" = TX secs",txSecs);
161.  if (key = 0x00) {
162.      /* If Command Issued then process the Command */
163.      if ((key = 0x00) & (0)) {
164.          quit = process_cnt();
165.      }
166.  }
167.  else
168.     {
169.  }
170.  
171.  
172.  
173.  
174.  
175.  
176.  
177.  
178.  
179.  
180.  
181.  
182.  
183.  
184.  
185.  
186.  
187.  
188.  
189.  
190.  
191.  
192.  
193.  
194.  
195.  
196.  
197.  
198.  
199.  
200.  
201.  
202.  
203.  
204.  
205.  
206.  
207.  
208.  
209.  
210.  
211.  
212.  
213.  
214.  
215.  
216.  
217.  
218.  
219.  
220.  
221.  
222.  
223.  
224.  
225.  82510 XMODEM Implementation (Continued)
/* If the Transmit Command is issued then Wait for a NAK */
if (tx_cmd == active)
{
    ts_cmd = inactive;
    put_crc_rq = active;
    ts_time_cnt = 200*60; // 60 sec. Time Out */
    ts_state = wait_NAK;
}
break;

case wait_NAK: /* Waiting for a NAK character to begin Tx */

TRANSMITTER WAITING FOR A NAK TO BEGIN

Checks For Time Out

or NAK Received

wn_status = check_wait (); /* Time Out or NAK Rcvd */

switch (wn_status) 
{
    case time_out: /* If Time Out then Abort Transmission */
    
        ts_state = tx_idle;
        beep ();
        prmsg ("Time OUT !!! receiver not ready");
        cli (tx_r, tx_c);
        open_wind (tx_r, tx_c, "NONE");
        break;

    case waiting: /* If no Time Out and no NAK Rcvd then do nothing */
        break;

    case rs_NAK: /* If NAK received then Open file and advance to
Transmit Packet State */

        fp = open (tx_file_name, "rb");
        if (fp == NULL)
            {
                beep ();
                prmsg ("ERROR !!! file does not exist");
                cli (tx_r, tx_c);
                open_wind (tx_r, tx_c, "NONE");
                ts_state = tx_idle;
            }
        else
            {
                ts_state = ts_rdy;
                txflag = mpkt; // First task for Ts
                /* Is to Prepare Packet */
                wn_status = 0; // Reset Wait_NAK Flag */
            
            } break;
        
        break; /* end wait nak */

82510 XMODEM Implementation (Continued)
case tx_rdy:
    /**************************************************************************/
    /** TX MODEM READY TO TRANSMIT   ****/
    /**************************************************************************/
    switch (tx_flag) {
        case wxpkt:
            /* Prepare Packet */
            if (tx_req == 0)
                /* Assemble Packet */
                assemb (pktst_sent,fp);
            /* Request Int. Handler */
            tx_req = pkt;
            /* Start Transmission */
            tx_flag = txmtg;
            tx_idx = 0;
            tx_l_enb ();
            /* Enable TxM and Tx FIFO */
            Interrupts */;
            break;
        case txmtg:
            if (tx_req == 0)
                /* TMSU ready to transmit */
                while (tx_flag) {
                    /* TMSU ready to transmit */
                    tx_idx = 0;
                    promg ("packet transmitted");
                    get_crc_req = active;
                    tx_time_cnt = 10000;
                    /* Wait for ACK or NAK */
                    tx_state = wait_CC;
                    /* Wait for Ctrl Character */
                    tx_flag = wxpkt;
                    tx_l_dis ();
                    /* Disable TxM and Tx FIFO */
                    Interrupts */;
                    break;
                }
                else /* Not ready to transmit */
                    promg ("transmit");
            /* Still transmitting */
            break;
            case retx:
                /* The Retransmit request is issued by the Wait_CC state */
                /* enable txm, flush tx fifo */
                tx_req = pkt;
                /* Transmit Packet pkt in buffer */
                tx_flag = txmtg;
                /* Next task - Retransmit */
                tx_l_enb ();
                /* Enable TxM and Tx FIFO */
                Interrupts */;
                break;
            break; /* End tx rdy case */
    }
PAGE 7  MAINE PROGRAM ftp.c 82510 XMODEM

2-235

82510 XMODEM Implementation (Continued)
82510 XMODEM Implementation (Continued)

```c
else
    /* If Retransmit Count Not exceeded then go back to  
Transmit stage - task is retransmit */
{
    txflag = retrans;
    ++ retrans_cnt;
    ts_state = ts_rdy;
    break;
}

case rs_ACK:
    /* ACK Received*/
    prmsg ("ACK received");
    retrans_cnt = 0;
    ++pts_sent;
    printf ("pts_sent = %lu, pts_sent");
    if (eof == false) /* If more data to transmit  
then return to might stage and ts new pkt. */
{
    txflag = mbkt;
    ts_state = ts_rdy;
    }
else
    {
    prmsg ("sending EOT"); /* if end of file, then send EOT */
    cc_to_ts = EOT;
    ts_req = tcl_chr;
    ts_l_emb();
    while (ts_req != 0); /* wait for Int. Handler  
to reset flag */
    ts_l_dis();
    get_ccr_rq (active); /* wait for Ack */
    while (get_ccr_rq == active);
    prmsg ("EOT acknowledgement received");
    if (cc_to_get == ACK) /* ACK read, Close File */
    {
        s = fclose (fp);
        abort_ts();
        prmsg ("file transmission complete");
    }
    ts_state = ts_idle; /* Return to Idle */
}
break;
} /* end wait_cc case */
break;
} /* end switch ts state */
```
AP-401

PAGE 9 MAIN PROGRAM ftp.c 82510 XMODEM

    switch (rs_state)
    {
      case rs_idle:
        /************************************************************/
        /**** RECEIVER IDLE: *******************************************/
        /**** waits for user command *********************************/
        /**** before sending NAKs *****************************************/
        /************************************************************/
      }
    }
    break;

    case rs_rdy:
    {  
      /************************************************************/
      /**** RECEIVER READY: ***************************************
      /**** sends NAK upon Time Out *******************************
      /**** or checks for SOH ************************************
      /**** or EOT ctrl-char. ***********************************
      /************************************************************/
      rsflag =wait_rx ();  /* Checks Rs Timer and returns -
      switch (rsflag)
      {
        case waiting: /* If waiting then do nothing */
        break;
      case SOH: /* If SOH received, then go into
      500. ** sofcnt;
      501. rs_state =rs_pkt;
      502. rs_time_cnt =0;  // four second time out */
      503. sofcnt =8;
      504. break;
    }
case time_out: /* If time out & not in the midst of packet reception then send NAK */
    if ((exp_pkt_num !=1) && (rx_bytes_cnt ==0))
    
    prmsg("rx time out !!!!! sending NAK");
    if (send_crc_req == inactive)
    
    crc_to_tx = NAK;
    send_crc_req = active;
  
  rx_time_out = 200*10;
break;

case EDT: /* If End Of Test rcvd. and data rcvd then send ACK and save all packets received in file */
    ++ out_cnt;
    open_wind (23,30,"End of Test");
if (exp_pkt_num ==1)
    
    if (send_crc_req == inactive) /* Send ACK */
    
    send_crc_req = active;
    crc_to_tx = ACK;
  
  rx_state = rx_idel; /* Receiver Returns to Idel */
  
  if (exp_pkt_num ==1)
    
    rxgp = open (ra_file_name,"ab");
    rwst = write (rxgp,buf,1,exp_pkt_num-1,rxgp);
    if (rwst ==0)
      
      prmsg("Write file error");
      printf("error = %d",(rwst=error(rxgp)));
    
    if (rwst ==1)
      
      prmsg("File received");
    else
      
      prmsg("Error in closing file");
    
    break;
  
  break;

/* RX */

  case rx_pkt: /* Packet reception */
  
  receive
  
  RECEIVE PACKET STATE
  
  checks for Time Out
  or 131 bytes received
  which signals the end of packet
  
  82510 XMODEM Implementation (Continued)
AP-401

82510 XMODEM Implementation (Continued)
PAGE 12  MAIN PROGRAM: ftp.c 82510 XMODEM

627. buff_copy (esp_pkt_num);
628. ++ esp_pkt_num;
629. }
630. else
631. prmsg ("old packet retransmitted");
632. }
633. }
634. else
635. es_pkt_param (); /* If error then show
636. */
637. line state = rx_rdy;
638. makint4 (); /* Enable Ctl-Chr int#*/
639. set_bank (00);
640. outp ((bpa+1),(inp(bpa+1)|ccmn));
641. set_bank (01);
642. enkint4 ();
643. send_crr_req =active; /* Send ACK */
644. ccr_to_h = ACK;
645. 646. 647. }
648. }
649. }
650. break;
651. 652. } /* end switch rx state */
653. 654. 655. } /* end else */
656. 657. } /* end while quit */
658. 659. reset $510 (); /* reset 82510 */
660. outp ((bpa+1),00); /* disable 82510 interrupts */
661. 662. 663. cmd = 0x10; /* disable 8259A interrupt */
664. 665. v=imp (0x21);
666. cmd = (v+1);cmd;
667. outp (0x21,cmd);
668. clir ();
669. code = 0;
670. _exit (code);
671. 1/* end main */
672. 231928-49

82510 XMODEM Implementation (Continued)
82510 XMODEM Implementation (Continued)
719. \texttt{init} ();
720. \texttt{//******************************************************************************}
721. \texttt{// Initialises Software and Configures the \$2510. Also sets up the interrupt ***}
722. \texttt{// *** Handler. ***}
723. \texttt{//******************************************************************************}
724. \texttt{\textcolor{red}{/*}}
725. \texttt{\textcolor{red}{*/}}
726. \texttt{\textcolor{red}{/}}
727. \texttt{\textcolor{red}{/}}
728. \texttt{\textcolor{red}{/}}
729. \texttt{\textcolor{red}{/}}
730. \texttt{\textcolor{red}{/}}
731. \texttt{\textcolor{red}{/}}
732. \texttt{\textcolor{red}{/}}
733. \texttt{\textcolor{red}{/}}
734. \texttt{\textcolor{red}{/}}
735. \texttt{\textcolor{red}{/}}
736. \texttt{\textcolor{red}{/}}
737. \texttt{\textcolor{red}{/}}
738. \texttt{\textcolor{red}{/}}
739. \texttt{\textcolor{red}{/}}
740. \texttt{\textcolor{red}{/}}
741. \texttt{\textcolor{red}{/}}
742. \texttt{\textcolor{red}{/}}
743. \texttt{\textcolor{red}{/}}
744. \texttt{\textcolor{red}{/}}
745. \texttt{\textcolor{red}{/}}
746. \texttt{\textcolor{red}{/}}
747. \texttt{\textcolor{red}{/}}
748. \texttt{\textcolor{red}{/}}
749. \texttt{\textcolor{red}{/}}
750. \texttt{\textcolor{red}{/}}
751. \texttt{\textcolor{red}{/}}
752. \texttt{\textcolor{red}{/}}
753. \texttt{\textcolor{red}{/}}
754. \texttt{\textcolor{red}{/}}
755. \texttt{\textcolor{red}{/}}
756. \texttt{\textcolor{red}{/}}
757. \texttt{\textcolor{red}{/}}
758. \texttt{\textcolor{red}{/}}
759. \texttt{\textcolor{red}{/}}
760. \texttt{\textcolor{red}{/}}
761. \texttt{\textcolor{red}{/}}
762. \texttt{\textcolor{red}{/}}
763. \texttt{\textcolor{red}{/}}
764. \texttt{\textcolor{red}{/}}
765. \texttt{\textcolor{red}{/}}
766. \texttt{\textcolor{red}{/}}
767. \texttt{\textcolor{red}{/}}
768. \texttt{\textcolor{red}{/}}
769. \texttt{\textcolor{red}{/}}
770. \texttt{\textcolor{red}{/}}
771. \texttt{\textcolor{red}{/}}
772. \texttt{\textcolor{red}{/}}
773. \texttt{\textcolor{red}{/}}
774. \texttt{\textcolor{red}{/}}
775. \texttt{\textcolor{red}{/}}
776. \texttt{\textcolor{red}{/}}
777. \texttt{\textcolor{red}{/}}

\textbf{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. }

**82510 XMODEM Implementation (Continued)**
82510 XMODEM Implementation (Continued)
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 SOM or EOT ******/  
978. /**** inform receiver ******/  
979. /***************************************************************/  
980.     switch (ccr)  
981.     {  
982.         +ccrmt;  
983.         flgs =inp (bps +5); /* read RST register to service  
984.         RsM interrupt */  
985.         flgs =inp (bps+1);  
986.         ctic =inp (bps);  
987.         if ( ((flgs & 0xff) ==0x48) /* if no errors and ctl. char */  
988.         )  
989.         {  
990.             /* then process control char. */  
991.             /* and send to tx or rx state */  
992.             switch (ctic)  
993.             {  
994.                 case NAK:  
995.                     case ACK:  
996.                         if (get_ccr_rq == active)  
997.                             /* inform transmitter that  
998.                             ctic char. received */  
999.                             get_ccr_rq =inactive;  
1000.                             ccf_to_get =ctic;  
1001.                     break;  
1002.                 case SOM:  
1003.                     case EOT:  
1004.                         if (ctic ==SOH) /* if SOM disable CCR int. */  
1005.                         {  
1006.                             set_bnk (00);  
1007.                             outp (((bps+1)7),inp(bps+1)& ccidb));  
1008.                             set_bnk (01);  
1009.                         }  
1010.                         if (rs_state == rx_rdy) /* if receiver waiting for  
1011.                         SOM and ready to recv  
1012.                         then inform receiver of  
1013.                         a valid ctrl. char. */  
1014.                         {  
1015.                             ctl_xrd_flg =active;  
1016.                             rs_ctl_chr =ctic;  
1017.                         }  
1018.                     break;  
1019.                 case:  
1020.                     }  
1021.             }  
1022.             }  
1023.     }  
1024.     }  
1025.     }  
1026.     break;  
1027. }  

82510 XMODEM Implementation (Continued)
82510 XMODEM Implementation (Continued)

1028. case rx:
1029. /***************************************************************************/
1030. 1031. /***************************************************************************/
1032. 1033. /***************************************************************************/
1034. 1035. /***************************************************************************/
1036. 1037. /***************************************************************************/
1038. 1039. /***************************************************************************/
1040. 1041. rx_time_cnt =200*5;  /* reset Rx Timer to indicate
1042. char. received before time out */
1043. 1044. while ( rxfi1 != 0 )  /* Check Rx FIFO level and read
1045. data if FIFO not empty */
1046. {  
1047. rxdat = (rx_byte_cnt) = inp (bpa);  
1048. ++ rx_byte_cnt;  
1049. -- rxfi1;  
1050. }  
1051.  
1052. outp ((bps+7),0x08);  /* issue manual acknowledge */
1053.  
1054. default :  /* if invalid source then issue a
1055. manual acknowledge */
1056.  
1057. outp ((bps+7),0x08);  
1058.  
1059. break;  
1060.  
1061. outp ((bps+1),girval);  /* Restore Original value of Bank
1062. register to return the 82510 to
1063. original Bank */
1064.  
1065. outp (ip00,00);  /* issue end of int. to 8259*/
1066.  
1067. }  
1068.  
1069. int  bank_num;
1070.  
1071. /***************************************************************************/
1072. 1073. /***************************************************************************/
1074. 1075. /***************************************************************************/
1076. 1077. /***************************************************************************/
1078. (  
1079. int  port;
1080.  
1081. bank_reg_val =bank_num * 0x20;
1082.  
1083. port = gir_addr +bpa;
1084.  
1085. )
82510 XMODEM Implementation (Continued)
case '1':
    if (!state == tx_idle)     /* Transmit Command only
        accepted if idle */
    else
        break;
    case '2':
        CLMS();
        CLI((tx_r,tx_c);
        MV_CURS(tx_r,tx_c);
        printf("file: ");    /* Get name of file to Ta */
        scanf("%s", &tx_file_name);
        CLI((rx_r,rx_c);
        open_wind((tx_r,tx_c,"transmitting");
        open_wind((tx_r,tx_c+14,tx_file_name);
        tx_cmd = active;       /* Activates flag to signal
        Transmit idle state */
    break;
    case '3':
        CLI((rx_r,rx_c);
        MV_CURR((rx_r,rx_c);
        printf("file: ");    /* Get rx file name */
        scanf("%s", &rx_file_name);
        CLI((rx_r,rx_c);
        open_wind((rx_r,rx_c,"enabled");
        open_wind((rx_r,rx_c+14,rx_file_name);
        rx_cmd = active;       /* Activates flag to signal
        rx state machine */
        break;
    case '4':
        rst510();       /* reset 82510 */
    open_net(24,30,"device reset");
        break;
    case 'r':
        rst510();
        init();        /* reinitialize 82510 */
        embint();
        beep();
        prmsg ("82510 reinitialized");
        break;
    case '1':
        exp = system("d:\micom");
        default:
        BEEP();
        prmsg ("Incorrect command, reenter");
        break;
    if (exitmsg == true)     /* if exit command issued,
        then quit program */
        return(true);
    else return(false);
    }     /* end of command processing */

82510 XMODEM Implementation (Continued)
```c
1192. int pkei_sent; /* this value is used to get the next pkt */
1193. FILE *fp;
1194. int sum = 0;
1195. int blkcnt;
1196. int st,ft;
1197. char cpkt,cpkamp;
1198. blkcnt = fread ((tspack.buffer[0],128,1,fp); /* read 128 bytes */
1199. if (blkcnt <= 0)
1200. { if (ferror(fp))
1201. { beep ();
1202. prmsg ("EOF !!!!!!!!!! ");
1203. _exit (1);
1204. }
1205. else
1206. if (ft <= 0)
1207. { beep ();
1208. if (file_error ()
1209. { beep ();
1210. prmsg ("READ ERROR !!!!!!!!!!");
1211. ts.state = ts_idle;
1212. }
1213. }
1214. }
1215. cpkt = pkei_sent + 1;
1216. tspack.pack_num = cpkt;
1217. tspack.pack_cmpi = cpkamp;
1218. tspack.pack_cmpi = tspack.pack_num;
1219. tspack.pack_num = tspack.pack_cmpi;
1220. sum = sum + tspack.buffer[i];
1221. tspack.chsum = sum % 255;
1222. for (i=0; i<128; i++)
1223. ts.buffer [i] = tspack.pack.Buffer[i];
1224. tspack.pack_num = tspack.pack_cmpi;
1225. tspack.pack_cmpi = tspack.pack_num;
1226. tspack.pack_num = tspack.pack_cmpi;
1227. tspack.pack_cmpi = tspack.pack_num;
1228. tspack.pack_num = tspack.pack_cmpi;
1229. tspack.pack_cmpi = tspack.pack_num;
1230. tspack.pack_num = tspack.pack_cmpi;
1231. tspack.pack_cmpi = tspack.pack_num;
1232. tspack.pack_num = tspack.pack_cmpi;
```

82510 XMODEM Implementation (Continued)
check_wait ()
1255. /*******************************************************************************/
1256. /**** PROCEDURE CHECK_WAIT ***/
1257. /**** checks Ts Timer, ccr_to_get and get_ccr_rq and returns: ***/
1258. /**** Time Out - Ts Timer = 0 ***/
1259. /**** ra_ACK - Ack received ***/
1260. /**** ra_NAK - Nak received ***/
1261. /**** waiting - ts Timer not expired ***/
1262. /**** */
1263. /******************************************************************************/
1264. (1
1265. if (((tx_time_cnt) && (get_ccr_rq == active))) /* if ts Timer expired
1266. and still waiting */
1267. return (time_out);
1268. /* Ctrl-Chr rcvd then */
1269. return status */
1270. /* Ctrl-Chr rcvd then */
1271. if (get_ccr_rq == inactive) { /* Ctrl-Chr rcvd then */
1272. switch (ccr_to_get) {
1273. case ACK:
1274. return (ra_ACK);
1275. break;
1276. case NAK:
1277. return (ra_NAK);
1278. break;
1279. default:
1280. return (ra_gen); /* corrupted ctrl chr */
1281. break;
1282. }
1283. return (waiting);
1284. if ((tx_time_cnt > 0) && (get_ccr_rq == active))
1285. return (waiting);
1286. /* Ctrl-Chr rcvd then */
1287. return status */

82510 XMODEM Implementation (Continued)
### MAIN PROGRAM ftp.c 82510 XMODEM

#### abort_tx ()

```c
1299. abort_tx ()
1300. /*************************************************************************
1301. *** Abort transmission, reinitialize transmitter flags
1302. ***
1303. *********************************************************
1304. */
1305. /*************************************************************************
1306. */
1307. 
1308. {  
1309.   eof = false;
1310.   txflag = mbpkt;
1311.   quit = false;
1312.   key = 0;
1313.   ts_state = ts_idle;
1314.   ts_cmd = inactive;
1315.   send_ccr_req = inactive;
1316.   tx_ind = 0;
1317.   tx_req = inactive;
1318.   ccr_to_te = 0;
1319.   tx_byte_cnt = 0;
1320.   pkts_sent = 0;
1321.   ts_time_cnt = 0;
1322.   get_ccr_flag = 0;
1323.   ccr_to_get = 0;
1324.   set_bank (00);
1325.   outp ((bpa+1),0x27);
1326.   set_bank (001);
1327.   outp((bpa+6),0x0D);
1328.   ts_state = ts_idle;
1329.   prom ("transmitter reset");
1330. }
```

#### wait_rx ()

```c
1331.  
1332.  
1333. “It /
1334. /*************************************************************************
1335. *** WAIT_RX:
1336. ***
1337. *** checks rx timer, and returns the following value:
1338. ***   SOH - SOH received
1339. ***   EOT - EOT received
1340. ***   time out - rx timer expired
1341. ***   waiting - waiting for event
1342. ***
1343. /*************************************************************************
1344. */
1345. 
1346. {  
1347.   if ((ctl_rx msg flag = active) & (rs_time_cnt != 0))
1348.     {  
1349.       ctl_rx msg flag = inactive;
1350.       return (rs_ctl Chr);
1351.     }
1352.   else
1353.     if (rs_time_cnt != 0)
1354.       return (time_out);
1355.     else
1356.       return (waiting);
```
```c
1358. chckpkt (pknun);
1359. /**************************************************************************/
1360. /** verifies the checksum and packet **/
1361. /** number of the received packet **/
1362. /** returns a status code **/
1363. /** **/
1364. /** ERR - Packet Ok **/
1365. /** EPRKBNM - Error in packet number **/
1366. /** ECKSUM - Error in Chekk Sum **/
1367. /** EPRKCMPL - Error in packet compleen **/
1368. /** **/
1369. /**************************************************************************/
1370. int pnum;
1371. (
1372. int i;
1373. int chm;
1374. int sum = 0;
1375. char cmp1,rcmp1,cpk,chrcm;
1376. 1377. cpk=pnum;
1378. 1379. if (cmp == rdata[0]) 1380. 1381. } 1382. rcmp1 = rdata [1];
1383. 1384. if (rcmp1 == "cmp1") 1385. 1386. for (i=1; i<130; i++) 1387. } 1388. chm = sum % 255;
1389. chrcm = chm;
1390. pk_chksm = chrcm;
1391. if (chrcm == rdata[130]) 1392. 1393. return (eckum);
1394. else
1395. 1396. return (epckcmp);
1397. 1398. if (rcmp1 == "cmp1") 1399. 1400. if ((rdata[0] == cpk -1) && (cpk >)) 1401. 1402. return (eold);
1403. else
1404. 1405. return (epknum);
1406. }
1407. }
```

82510 XMODEM Implementation (Continued)
82510 XMODEM Implementation (Continued)
PAGE 10  MAIN PROGRAM ftp.c 82510 XMODEM

1444. /**************************************************************/
1445. /** PROCEDURE BUF_CPY **/
1446. /** copies packet to ram buffer **/
1447. /**************************************************************/
1448. buf_cpy (pkt_id)
1449. int pkt_id;
1450. {
1451. int i;
1452. int indx = 0;
1453. for (i=0; i<128; i++) /* No overwrite of buffer */
1454. if (indx < 128) /* No overwrite of buffer */
1455. else
1456. prmsg ("file too big, cannot save in memory");
1457. 1458. } 1459.

82510 XMODEM Implementation (Continued)

2-256
82510 XMODEM Implementation (Continued)
82510 XMODEM Implementation (Continued)
82510 XMODEM Implementation (Continued)
61. OPEN_WIND (row, col, stg)
62. int row;
63. int col;
64. char stg[];
65. /******************************************************************************
66. ****** PROCEDURE OPEN_WIND
67. ******
68. ****** prints a string in reverse video
69. ****** at the given location
70. ******
71. ******
72. ******
73.******************************************************************************
74. ( 75.   MV_CURS (row, col);
76.   VON () ;
77.   printf("%s", stg);
78.   VOFF();
79. )
80. )
81. BEEP ()
82. /******************************************************************************
83. ****** PROCEDURE BEEP
84. ******
85. ****** produces a beep
86. ******
87. ******
88. ******
89. ******
90. ******
91. ******
92.******************************************************************************
93. ( 94.   int belchr = bel;
95.   putch (belchr);
96.   belf ();
97. )
98. )
99. )
100. CLL(row, col)
101. int row;
102. int col;
103. /******************************************************************************
104. ****** PROCEDURE CLL
105. ******
106. ****** clear line at given coordinate
107. ******
108. ******
109. ******
110. ******
111. ******
112. ******
113. ******
114. ( 115.   int escchr = esci;
116.   MV_CURS (row, col);
117.   putch (escchr);
118.   printf("\n");
119. )
120. )

82510 XMODEM Implementation (Continued)
PROCEDURE CLMS

**PROCEDURE CLMS**
*clear message line*

PROCEDURE CLS

**PROCEDURE CLS**

PROCEDURE MV_CURS

**PROCEDURE MV_CURS**
*move cursor to specified location*

MV_CURS (x,y)

MV_CURS (x,y)

MV_CURS (x,y)

MV_CURS (x,y)

MV_CURS (x,y)

MV_CURS (x,y)

MV_CURS (x,y)

MV_CURS (x,y)

MV_CURS (x,y)

MV_CURS (x,y)

MV_CURS (x,y)

MV_CURS (x,y)

MV_CURS (x,y)

MV_CURS (x,y)

MV_CURS (x,y)

MV_CURS (x,y)

MV_CURS (x,y)

MV_CURS (x,y)

MV_CURS (x,y)

MV_CURS (x,y)

MV_CURS (x,y)

MV_CURS (x,y)

MV_CURS (x,y)

MV_CURS (x,y)

MV_CURS (x,y)

MV_CURS (x,y)

MV_CURS (x,y)

MV_CURS (x,y)

MV_CURS (x,y)

MV_CURS (x,y)

MV_CURS (x,y)

MV_CURS (x,y)

MV_CURS (x,y)

MV_CURS (x,y)

MV_CURS (x,y)

MV_CURS (x,y)

MV_CURS (x,y)

MV_CURS (x,y)

MV_CURS (x,y)

MV_CURS (x,y)

MV_CURS (x,y)

MV_CURS (x,y)

MV_CURS (x,y)

MV_CURS (x,y)

MV_CURS (x,y)

MV_CURS (x,y)

MV_CURS (x,y)

MV_CURS (x,y)

MV_CURS (x,y)

MV_CURS (x,y)

MV_CURS (x,y)

MV_CURS (x,y)

MV_CURS (x,y)

MV_CURS (x,y)

MV_CURS (x,y)

MV_CURS (x,y)

MV_CURS (x,y)

MV_CURS (x,y)

MV_CURS (x,y)

MV_CURS (x,y)

MV_CURS (x,y)

MV_CURS (x,y)

MV_CURS (x,y)

MV_CURS (x,y)

MV_CURS (x,y)

MV_CURS (x,y)

MV_CURS (x,y)

MV_CURS (x,y)

MV_CURS (x,y)

MV_CURS (x,y)

MV_CURS (x,y)

MV_CURS (x,y)

MV_CURS (x,y)

MV_CURS (x,y)

MV_CURS (x,y)

MV_CURS (x,y)

MV_CURS (x,y)

MV_CURS (x,y)

MV_CURS (x,y)

MV_CURS (x,y)

MV_CURS (x,y)

MV_CURS (x,y)

MV_CURS (x,y)

MV_CURS (x,y)

MV_CURS (x,y)

MV_CURS (x,y)

MV_CURS (x,y)

MV_CURS (x,y)

MV_CURS (x,y)

MV_CURS (x,y)

MV_CURS (x,y)

MV_CURS (x,y)

MV_CURS (x,y)

MV_CURS (x,y)

MV_CURS (x,y)

MV_CURS (x,y)

MV_CURS (x,y)

MV_CURS (x,y)

MV_CURS (x,y)

MV_CURS (x,y)

MV_CURS (x,y)

MV_CURS (x,y)

MV_CURS (x,y)

MV_CURS (x,y)

MV_CURS (x,y)

MV_CURS (x,y)

MV_CURS (x,y)

MV_CURS (x,y)

MV_CURS (x,y)

MV_CURS (x,y)

MV_CURS (x,y)

MV_CURS (x,y)

MV_CURS (x,y)

MV_CURS (x,y)

MV_CURS (x,y)

MV_CURS (x,y)

MV_CURS (x,y)

MV_CURS (x,y)

MV_CURS (x,y)

MV_CURS (x,y)

MV_CURS (x,y)

MV_CURS (x,y)

MV_CURS (x,y)

MV_CURS (x,y)

MV_CURS (x,y)

MV_CURS (x,y)

MV_CURS (x,y)

MV_CURS (x,y)

MV_CURS (x,y)

MV_CURS (x,y)

MV_CURS (x,y)

MV_CURS (x,y)

MV_CURS (x,y)

MV_CURS (x,y)

MV_CURS (x,y)

MV_CURS (x,y)

MV_CURS (x,y)

MV_CURS (x,y)

MV_CURS (x,y)

MV_CURS (x,y)

MV_CURS (x,y)

MV_CURS (x,y)

MV_CURS (x,y)

MV_CURS (x,y)

MV_CURS (x,y)

MV_CURS (x,y)

MV_CURS (x,y)

MV_CURS (x,y)

MV_CURS (x,y)

MV_CURS (x,y)

MV_CURS (x,y)

MV_CURS (x,y)

MV_CURS (x,y)

MV_CURS (x,y)

MV_CURS (x,y)

MV_CURS (x,y)

MV_CURS (x,y)

MV_CURS (x,y)

MV_CURS (x,y)

MV_CURS (x,y)

MV_CURS (x,y)

MV_CURS (x,y)

MV_CURS (x,y)

MV_CURS (x,y)

MV_CURS (x,y)

MV_CURS (x,y)

MV_CURS (x,y)

MV_CURS (x,y)

MV_CURS (x,y)

MV_CURS (x,y)

MV_CURS (x,y)

MV_CURS (x,y)

MV_CURS (x,y)

MV_CURS (x,y)

MV_CURS (x,y)

MV_CURS (x,y)

MV_CURS (x,y)

MV_CURS (x,y)

MV_CURS (x,y)

MV_CURS (x,y)

MV_CURS (x,y)

MV_CURS (x,y)

MV_CURS (x,y)

MV_CURS (x,y)

MV_CURS (x,y)

MV_CURS (x,y)

MV_CURS (x,y)

MV_CURS (x,y)

MV_CURS (x,y)

MV_CURS (x,y)

MV_CURS (x,y)

MV_CURS (x,y)

MV_CURS (x,y)

MV_CURS (x,y)

MV_CURS (x,y)

MV_CURS (x,y)

MV_CURS (x,y)

MV_CURS (x,y)

MV_CURS (x,y)

MV_CURS (x,y)

MV_CURS (x,y)

MV_CURS (x,y)

MV_CURS (x,y)

MV_CURS (x,y)

MV_CURS (x,y)

MV_CURS (x,y)

MV_CURS (x,y)

MV_CURS (x,y)

MV_CURS (x,y)

MV_CURS (x,y)

MV_CURS (x,y)

MV_CURS (x,y)

MV_CURS (x,y)

MV_CURS (x,y)

MV_CURS (x,y)

MV_CURS (x,y)

MV_CURS (x,y)

MV_CURS (x,y)

MV_CURS (x,y)

MV_CURS (x,y)

MV_CURS (x,y)

MV_CURS (x,y)

MV_CURS (x,y)

MV_CURS (x,y)

MV_CURS (x,y)

MV_CURS (x,y)

MV_CURS (x,y)

MV_CURS (x,y)

MV_CURS (x,y)

MV_CURS (x,y)

MV_CURS (x,y)

MV_CURS (x,y)

MV_CURS (x,y)
ASM 86 INTERRUPT INIT. ihl.asm 82510 XMODEM

1. NAME ih510
2. 3. DGROUP GROUP DATA
4. DATA SEGMENT WORD PUBLIC 'DATA'
5. ASSUME DS: DGROUP
6. DATA ENDS
7. 8. ETRN isr_510:far
9. 10. _PROC SEGMENT BYTE PUBLIC 'PROG'
11. ASSUME CS:_PROC
12. 13. PUBLIC init_ih
14. PUBLIC ih510
15. 16. init_ih PROC far
17. push BP
18. push DI
19. push AX
20. push DS
21. mov DI, OFFSET ih510
22. push CS
23. pop DS
24. mov AH, 25H ;DOS vector setup call
25. mov AL, 0CH ;COM1 vector
26. INT 21H ;DOS system call
27. pop DS
28. pop AX
29. pop DI
30. pop BP
31. ret
32. init_ih ENDP
33. 34. ih510 PROC far
35. push BP
36. push AX
37. push BX
38. push CX
39. push DX
40. push SI
41. push DI
42. push DS
43. push ES
44. mov AX, DGROUP
45. mov DS, AX
46. call isr_510
47. pop ES
48. pop DS
49. pop DI
50. pop SI
51. pop DX
52. pop CX
53. pop BX
54. pop AX
55. pop BP
56. ret
57. ih510 ENDP
58. 59. _PROC ENDS
60. end ENDS

2-262
High Performance Driver for 82510

DAN GAVISH and TSVIKA KURTS
SYSTEM VALIDATION
CONTENTS

1.0 OVERVIEW ........................................ 2-265

2.0 INTRODUCTION ................................. 2-265
  2.1 CPU Load Consideration ...................... 2-265
  2.2 82510 Features Used In This Implementation .......... 2-265

3.0 THE BURST ALGORITHM ....................... 2-266
  3.1 Background .................................. 2-266
  3.2 Burst Algorithm Description .................. 2-266

4.0 SOFTWARE MODULE MAP ..................... 2-267

5.0 HARDWARE VEHICLE DESCRIPTION ............ 2-268

6.0 SOFTWARE MODULE DESCRIPTIONS .......... 2-268
  6.1 MAIN ....................................... 2-268
  6.2 The Burst Algorithm Modules .................. 2-269
  6.3 Initializations ............................... 2-271
  6.4 Interrupt Handler ............................. 2-275

APPENDIX A—PL/M SOURCE FILE .............. 2-280

APPENDIX B—82510 BASED SBX SERIAL CHANNEL .......... 2-293
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 whenever 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 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. 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.

![Diagram of the Burst Algorithm](image-url)
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.

![Image](image_url)

Figure 6. Timer Interrupt and BURST Algorithm

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 ACR0 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](image-url)

Figure 9. Wait__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-url)
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 re-checked, 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](image)
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.
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.

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.

Figure 14. TIMER Intr Service Routine

Figure 15. MODEM Intr Service Routine
APPENDIX A
PL/M SOURCE FILE

/***************************************************************************
* * *
* 8 2 5 1 0 - H I G H  P E R F O R M A N C E  D r i v e r  *
* * *
* 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 '0OH'
DECLARE BAUD 9600 LIT '003CH' 1/ /* Character configurations */
DECLARE BAUD_19200 LIT '001EH'
DECLARE BAUD_288000 LIT '0002H'
DECLARE DLAB 0 LIT '01111111B' 1/ /* Reset DLAB */
DECLARE DLAB_1 LIT '10000000B' 1/ /* Set DLAB */
DECLARE CR LIT '0DH' 1/ /* 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' 1/ /* 8 2 5 1 0 registers */
DECLARE NAS0 LIT '00000000B'
DECLARE WORK1 LIT '00100000B'
DECLARE GEN2 LIT '01000000B'
DECLARE MODM3 LIT '01100000B'
DECLARE TXD LIT 'BASE_510 + 0' 1/ /* BANK 0 - NAS */
DECLARE RXD LIT 'BASE_510 + 0'
DECLARE BAL LIT 'BASE_510 + 2'
DECLARE BAH LIT 'BASE_510 + 2'
DECLARE GER LIT 'BASE_510 + 4'
DECLARE GEN3 LIT 'BASE_510 + 4'
DECLARE BANK LIT 'BASE_510 + 4'
DECLARE LCR LIT 'BASE_510 + 6'
DECLARE LCR LIT 'BASE_510 + 6'
DECLARE MCR LIT 'BASE_510 + 8'
DECLARE MSR LIT 'BASE_510 + 10'
DECLARE MSR LIT 'BASE_510 + 12'
DECLARE ACR0 LIT 'BASE_510 +14'
DECLARE RXF LIT 'BASE_510 + 2' 1/ /* BANK 1 - WORK */
DECLARE TF 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 ICM LIT 'BASE_510 +14'
DECLARE FMD LIT 'BASE_510 + 2' 1/ /* BANK 2 - GENERAL CONFIGURE */
DECLARE TM LIT 'BASE_510 + 6'
DECLARE FMD LIT 'BASE_510 + 6'

2-280
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' ;/* BANK 3 - MODEM */
DECLARE BBL LIT 'BASE_510 + 0' ;/* DLAB=1 */
DECLARE BACF LIT 'BASE_510 + 2';
DECLARE BBH LIT 'BASE_510 + 2' ;/* DLAB=l */
DECLARE BBCF LIT 'BASE_510 + 6';
DECLARE PI1D LIT 'BASE_510 + 8';
DECLARE I1IE LIT 'BASE_510 +10';
DECLARE TI1IE LIT 'BASE_510 +12';
DECLARE OUT2 I1CR LIT 'OOOO1000B';/* Specific register bits */
DECLARE DTR MCR LIT 'OOOOOOOlB';
DECLARE DSR-I1SR LIT 'OO100000B';
DECLARE CLRSTAT ICM LIT 'OOOOO100B';
DECLARE INTR 510 LIT '21H';
DECLARE PORT-80130M LIT 'OE2H';
DECLARE EN 80130 LIT 'OFDH';
DECLARE PORT EOI LIT 'OEOH';
DECLARE COMM-EOI LIT '61H'; /* End Of Interrupt command */
DECLARE ENRTX GER LIT 'OOOOlll1B';/* Enable Interrupt bits */
DECLARE ENTX GER LIT 'OOOOOO10B';
DECLARE ENTXSTAT GER LIT 'OOOOll10B';
DECLARE ENRX GER LIT 'OOOOl101B';
DECLARE ENTIMRX GER LIT 'OO101101B';
DECLARE DISTX GER LIT 'OOOO1101B';
DECLARE DISRX-GER LIT 'OOOOOO10B';/* Disable Interrupt bits */
DECLARE DISRTX GER LIT 'OOOOOOOOB';
DECLARE TXTHRESHO FI1D LIT 'OOOOOOOOB'; /* FIFO threshold */
DECLARE RXTHRESH0-FMD LIT 'OOOOOOOOB';
DECLARE RXTHRESH3-FMD LIT 'OOl10000B';
DECLARE MASK RXOCC LIT 'Ol110000B'; /* Mask on occupancy bits */
DECLARE MASK-TXOCC LIT 'OOOOOlllB';
DECLARE ACRSTAT-RST LIT '01000000B'; /* Mask on ACR status bits */
DECLARE NONI GIR LIT 'OO100001B';/* Interrupt vector */
DECLARE MODMI GIR LIT 'OO100000B';
DECLARE TXI-GIR LIT 'OO100010B';
DECLARE RXI-GIR LIT 'OO100100B';
DECLARE STATI-GIR LIT 'OO100110B';
DECLARE TIMI-GIR LIT 'OO101010B';
DECLARE AUTOACK IMD LIT 'OOOO1000B';
DECLARE TMOD BBCF LIT 'OOOOOOOOB'; /* Timer */
DECLARE TIMBI-TMIE LIT 'OOOOOO10B';
DECLARE FIF0 IMD LIT 'OOOOOOOOB';
DECLARE STARTIMB TMCR LIT 'OO100010B';
DECLARE STARTIMB-TMST LIT 'OOOOOO10B';
DECLARE RTXCLK_BRGA CLCF LIT '01010000B';
DECLARE LOW BAUD LIT '00H'; /* BURST algorithm */
DECLARE HUNTING MODE LIT '01H';
DECLARE SINGLE_MODE LIT '02H';
DECLARE BURST_MODE LIT '03H';
DECLARE TIME_EXP LIT 'OFFFH'; /* timeout=7mS (at 18.4 Mhz) */
DECLARE WAIT_TIME LIT '0FFFH'; /* WAIT_FOR_MODEM_STATUS */
declare tx_ptr pointer public ; /* transmit buffer */
declare tx_buf based tx_ptr (3000) byte ;
declare ix_tx word public ;
declare rx_bufp(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 tx_occ byte public ;
declare rx_occ byte public ;
declare stat byte public ;
declare baud word public ;
declare temp byte public ;
declare fin byte public ;
declare selection 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 */

/* io console utilities */
include (:f1:tioph.pex)

/* setup and h/w configurations */
include (:f1:hputil.pex)

declare main label public ;

initializer : procedure 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 ; /* clear tx_chr and rx_chr */
CALL TEXT ; /* TX_PTR is a pointer to the transmitted*/ /* data */
IX_TX= OFFFH ; /* The index buffer are assigned to -1 */
IX_RX= OFFFH ;
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 */

/***********************************************************/
/* INITIALIZE_BURST */
/***********************************************************/
* input: none *
* output: Burst_Alg *
* 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" */
/* 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 *
* *
* 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(ICM) = SWRES_CMND; /* S/W reset command */
/* BANK ZERO - NAS (The default BANK) */

/* Configured the character by writing to LCR: */
/* 1 stop bit, 8 bit length, non parity */
OUTPUT(LCR)=(STPBIT_1 + CHRLN_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 */
/* recognized to terminate reception */
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, */
/* Rxfifo depth is four bytes */
OUTPUT(FMD)=(TXTHRESH0_FMD OR RXTHRESH0_FMD) ;
/* Rxfifo threshold is temporarily zero */
/* for HUNTING mode (BURST algorithm) */
/* Txfifo threshold is zero for max */
/* interrupt latency */
IF RECEIVER THEN OUTPUT(ACR1)=End_Of_File ;
/* At the Receive station EOF is */
/* recognized, the same as ACR0 */
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) */

2-284
OUTPUT(BBH) = HIGH(TIME_EXP); /* to issue interrupt when time has expired. Move to NAS bank again */
OUTPUT(BANK) = NAS0; /* Reset DLAB */
OUTPUT(LCR) = INPUT(LCR) AND DLAB_0; /* Switch to BANK THREE - MODEM */
OUTPUT(BANK) = MODEM3; /* is BRG A */
OUTPUT(TMIE) = TIMB1_TMIE; /* Enable Timer block interrupt */
/* BANK ONE - general WORK */
/* The RUNTIME bank */
OUTPUT(BANK) = WORK1; /* Issues a command to clear all status registers */
OUTPUT(ICM) = CLRSTAT_ICM;

/* Remain in WORK - THE runtime bank */

END CONFIG_82510;

************ Procedure WAIT_FOR_MODEM_STATUS ************
"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 **
************ Procedure WAIT_FOR_MODEM_STATUS ************

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 ************
** 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 **
************ Procedure INTERRUPT_HANDLER ************

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); /* Shift the Rx occupancy bit */
RX_OCC=SHR(RX_OCC,4); /* - OPTIMIZE code - */
RX_BUF(IX_RX:=IX_RX+1)=INPUT(RXD); /* Empty the Rx FIFO and store the */
RX_BUF(IX_RX:=IX_RX+1)=INPUT(RXD); /* received character in RX_BUF */
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

BURST_MODE - 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;

HUNTING_MODE - 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 ;

2-286
/*---------------------------------------------------*/
*       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(FMD)=TXTHRESH0_FMD OR RXTHRESH3_FMD;
  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) ;
  /* Empty the Rx FIFO and store the */
  /* received character in RX_BUF */
  RX_BUF(IX_TX:=IX_TX+1)=INPUT(RXD) ;
END ;
/* End of Rx fifo interrupt */

/*************************************************************************
* TxFIFO_INTR                                                      *
*************************************************************************/
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 transmit */
      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 */

2-287
**STATUS_INTR**

- **input**: none
- **output**: Finish Rx
- **function**: service Status interrupt
- **Receive station**: EOF terminate the reception
- **Transmit station**: X_Off Disable the transmission
- **called by**: INTERRUPT HANDLER
- **calling**: none

*flowchart: figure 13 description: paragraph 6.4.4*

```c
else if intr_vec=STAT_I GIR then do ;

stat=input(rst) ; /* Get the current RST status */

rx_occ=input(fld) ; /* Rx fifo level occupancy */

rx_occ=shr(rx_occ,4) ;

while (rx_occ>0 and (not fin_rx));

rx_occ=rx_occ-1 ; /* First, empty Rx FIFO */

rx_chr=input(rxd) ;

if receiver then

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 ;

else if ((stat and acrstat_rst) <> 0) then do ;

output(bank) = nas0 ; /* If End_Of_Line was recognized, */

output(ger) = disrx_ger ;

output(bank) = work1 ; /* Disable 82510-interrupts and the */

fin_rx = true ; /* Reception */

end ;

else if ((stat and errchr_rst) <> 0) then do ;

write('** ERROR in character Status ',0)) ;

error_char_handler ;

if burst_algo=burst_mode then do ;

/* In BURST mode do: */

temp = input(tmst) ; /* Reset timer status */

output(tmcr) = startimb tmcr;

end ; /* Restart TIMER */

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 ;
        /* Store the received character in RX_buf*/
    END ;
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 ;
        /* Store the received character in RX_buf*/
END ;
MODULE 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 *

ELSE IF INTR_VEC=MODMI_GIR THEN DO;
   STAT=INPUT(MSR); /* Get MODEM status */
   CALL ERROR_MODEM_HANDLER; /* Handel Modem Errors handshake */
   END; /* End of MODEM interrupt */

OUTPUT(PORT_EOI)=COMM_EOI; /* Write End_Of Interrupt command to the */
   /* PIC (8259A) */
END INTR_HANDLER;

/* Procedure  ERROR_MODEM_HANDLER */
ERROR_MODEM_HANDLER: PROCEDURE PUBLIC;
MODEM_HANDSHAKE = FALSE; /* Flag indicates that an Error occurred */
   /* in Modem */
END ERROR_MODEM_HANDLER;

/* Procedure  ERROR_CHAR_HANDLER */
ERROR_CHAR_HANDLER: PROCEDURE PUBLIC;
RX_ERROR = TRUE; /* Flag indicates that an Error occurred */
   /* during Reception */
OUTPUT(BANK) = NASO; /* 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;
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);
  IF FIN TX THEN CALL WRITELN(@("Transmission END ED",0));
  IF FIN RX THEN CALL WRITELN(@("Reception END ED",0));
END;
OUTPUT(BANK)=NASO;
OUTPUT(GER)=DISRTX_GER;
OUTPUT(BANK)=WORK1;
IF RECEIVER THEN
  CALL WRITELN(@("The Received Message:\",0));
  CALL DISPTEXT(@RX_BUF);
END;
ELSE IF NOT FIN TX THEN
  CALL WRITELN(@("** ERROR -THE Transmission NOT ended successfully",0));
END LOOP;

TEXT: PROCEDURE PUBLIC;
TX_PTR=@("","CR,LF,
  "ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789abcdefghijklmnopqrstuvwxyz0123456789",
  "ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789abcdefghijklmnopqrstuvwxyz0123456789",
  "ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789abcdefghijklmnopqrstuvwxyz0123456789",
  "ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789abcdefghijklmnopqrstuvwxyz0123456789",
  "ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789abcdefghijklmnopqrstuvwxyz0123456789",
  "ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789abcdefghijklmnopqrstuvwxyz0123456789",
  "ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789abcdefghijklmnopqrstuvwxyz0123456789",
  "ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789abcdefghijklmnopqrstuvwxyz0123456789",
  "ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789abcdefghijklmnopqrstuvwxyz0123456789",
  "ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789abcdefghijklmnopqrstuvwxyz0123456789",
  "ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789abcdefghijklmnopqrstuvwxyz0123456789"),
  "End_Of_File,O) ;
  /* 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(@('------------------------------------------------',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 ;
      /* 2 - Receive station */
      RECEIVER=TRUE ;
      CALL INITIALIZATIONS ;
      CALL LOOP ;
   END ;
   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 a 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.
Using the 8273 SDLC/HDLC Protocol Controller

JOHN BEASTON
MICROCOMPUTER APPLICATIONS

Order Number: 611001-001
<table>
<thead>
<tr>
<th>CONTENTS</th>
<th>PAGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>INTRODUCTION</td>
<td>2-297</td>
</tr>
<tr>
<td>SDLC/HDLC OVERVIEW</td>
<td>2-297</td>
</tr>
<tr>
<td>BASIC 8273 OPERATION</td>
<td>2-300</td>
</tr>
<tr>
<td>HARDWARE ASPECTS OF THE 8273</td>
<td></td>
</tr>
<tr>
<td>CPU Interface</td>
<td>2-301</td>
</tr>
<tr>
<td>Modem Interface</td>
<td>2-305</td>
</tr>
<tr>
<td>SOFTWARE ASPECTS OF THE 8273</td>
<td></td>
</tr>
<tr>
<td>Command Phase Software</td>
<td>2-309</td>
</tr>
<tr>
<td>Execution Phase Software</td>
<td>2-310</td>
</tr>
<tr>
<td>Result Phase Software</td>
<td>2-311</td>
</tr>
<tr>
<td>8273 COMMAND DESCRIPTION</td>
<td></td>
</tr>
<tr>
<td>Initialization/Configuration Commands</td>
<td>2-313</td>
</tr>
<tr>
<td>Operating Mode Register</td>
<td>2-313</td>
</tr>
<tr>
<td>Serial I/O Mode Register</td>
<td>2-313</td>
</tr>
<tr>
<td>Data Transfer Mode Register</td>
<td>2-313</td>
</tr>
<tr>
<td>One Bit Delay Register</td>
<td>2-314</td>
</tr>
<tr>
<td>Receive Commands</td>
<td>2-314</td>
</tr>
<tr>
<td>General Receive</td>
<td>2-314</td>
</tr>
<tr>
<td>Selective Receive</td>
<td>2-314</td>
</tr>
<tr>
<td>Selective Loop Receive</td>
<td>2-314</td>
</tr>
<tr>
<td>Receive Disable</td>
<td>2-315</td>
</tr>
<tr>
<td>Transmit Commands</td>
<td>2-317</td>
</tr>
<tr>
<td>Transmit Frame</td>
<td>2-317</td>
</tr>
<tr>
<td>Loop Transmit</td>
<td>2-317</td>
</tr>
<tr>
<td>Transmit Transparent</td>
<td>2-317</td>
</tr>
<tr>
<td>Abort Commands</td>
<td>2-317</td>
</tr>
<tr>
<td>Reset Commands</td>
<td>2-318</td>
</tr>
<tr>
<td>Modem Control Commands</td>
<td>2-319</td>
</tr>
<tr>
<td>HDLC CONSIDERATIONS</td>
<td>2-319</td>
</tr>
<tr>
<td>LOOP CONFIGURATIONS</td>
<td>2-320</td>
</tr>
<tr>
<td>APPLICATION EXAMPLE</td>
<td>2-324</td>
</tr>
<tr>
<td>CONCLUSION</td>
<td>2-331</td>
</tr>
<tr>
<td>APPENDIX A</td>
<td>2-332</td>
</tr>
</tbody>
</table>
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 CPU (Central Processing Unit) intervention. For the user this means the CPU has time to take on additional tasks. 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 loop back 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>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Opening Flag</td>
<td>01111110</td>
</tr>
<tr>
<td>Address Field (A)</td>
<td>8 Bits</td>
</tr>
<tr>
<td>Control Field (C)</td>
<td>8 Bits</td>
</tr>
<tr>
<td>Information Field (I)</td>
<td>Any Length 0 to N Bits</td>
</tr>
<tr>
<td>Frame Check Sequence (FCS)</td>
<td>16 Bits</td>
</tr>
<tr>
<td>Closing Flag</td>
<td>01111110</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 polls 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 zeroes 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.

All of the previous discussion has applied to SDLC on either point-to-point or multi-point data networks. SDLC (but not HDLC) also includes specification for a loop configuration. Figure 3 compares these three configurations. IBM uses this loop configuration in its 3650 Retail Store System. It consists of a single loop controller station with one or more down-loop secondary stations. Communications on a loop rely on the secondary stations repeating a received message down loop with a delay of one bit time. The reason for the one bit delay will be evident shortly.

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 terminates its repeater function and inserts its message (with multiple preceding flags if necessary). After the closing flag, 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**

**Figure 6. 8273 Block Diagram**

2-301
**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 all received frames are transferred to memory, and Selective 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 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-
Converting 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. 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 D5 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 D5 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 D1 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. D6 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.

![Figure 15. Transmit/Receive Timing](image)

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 (Rx'C) 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 $32 \times \text{CLK}$ 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 ($A_1$, $B_1$, $B_2$, or $A_2$) the data edge falls in. (Each quadrant represents $8 \times 32 \times \text{CLK}$ times.) For example, if the edge is detected in quadrant $A_1$, 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 $A_1$ 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 $B_2$ 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 $A_1$ or $A_2$ and 8-bit times adjusting through $B_1$ or $B_2$.

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 Is 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.

Figure 17. Serial Data Timing Configuration

**Figure 18. Status Register Format**
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.
AP-36

;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 ;ROTATE CBSY INTO CARRY
       JC 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

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

```assembly
;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 all 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

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

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>B0, B1</td>
<td>LSB and MSB of Receive Buffer Length</td>
</tr>
<tr>
<td>R0, R1</td>
<td>LSB and MSB ofReceived Frame Length</td>
</tr>
<tr>
<td>L0, L1</td>
<td>LSB and MSB of Transmit Frame Length</td>
</tr>
<tr>
<td>A1, A2</td>
<td>Match Addresses for Selective Receive</td>
</tr>
<tr>
<td>RIC</td>
<td>Receiver Interrupt Result Code</td>
</tr>
<tr>
<td>TIC</td>
<td>Transmitter Interrupt Result Code</td>
</tr>
<tr>
<td>A</td>
<td>Address Field of Received Frame</td>
</tr>
<tr>
<td>C</td>
<td>Control Field of Received Frame</td>
</tr>
</tbody>
</table>
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 Is 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 Is (HDLC Abort) are received by an active receiver. When reset, an SDLC Abort (8 Is) will cause an interrupt.

D4: EOP Interrupt—Reception of an EOP character (0 followed by 7 Is) 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 data character 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 prefaces 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.
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.

![Figure 25. Serial I/O Mode Register](image)

Figure 25. Serial I/O Mode Register

**One Bit Delay Register (Figure 27)**

D7:  **One Bit Delay**—When set, the 8273 retransmits the received data stream one bit delayed. This mode is entered and exited at a received character boundary. When reset, the transmitted and received data are independent. This mode is utilized for loop operation and is discussed in a later section.

D6–D0:  **Not Used**—These bit must be 0 for the Set command and 1 for the Reset command.

![Figure 26. Data Transfer Mode Register](image)

Figure 26. Data Transfer Mode Register

![Figure 27. One Bit Delay Mode Register](image)

Figure 27. One Bit Delay Mode Register

Figure 28 shows the Set and Reset commands associated with the above registers. The mask which sets or resets the desired bits is treated as a single parameter. These commands do not interrupt nor provide results during the Result phase. After reset, the 8273 defaults to all of these bits reset.

<table>
<thead>
<tr>
<th>Register</th>
<th>Command</th>
<th>Hex Code</th>
<th>Parameter</th>
</tr>
</thead>
<tbody>
<tr>
<td>One Bit Delay Mode</td>
<td>Set</td>
<td>A4</td>
<td>Set Mask</td>
</tr>
<tr>
<td></td>
<td>Reset</td>
<td>64</td>
<td>Reset Mask</td>
</tr>
<tr>
<td>Data Transfer Mode</td>
<td>Set</td>
<td>97</td>
<td>Set Mask</td>
</tr>
<tr>
<td></td>
<td>Reset</td>
<td>57</td>
<td>Reset Mask</td>
</tr>
<tr>
<td>Operating Mode</td>
<td>Set</td>
<td>91</td>
<td>Set Mask</td>
</tr>
<tr>
<td></td>
<td>Reset</td>
<td>51</td>
<td>Reset Mask</td>
</tr>
<tr>
<td>Serial I/O Mode</td>
<td>Set</td>
<td>A0</td>
<td>Set Mask</td>
</tr>
<tr>
<td></td>
<td>Reset</td>
<td>60</td>
<td>Reset Mask</td>
</tr>
</tbody>
</table>

Figure 28. Initialization/Configuration Command Summary

**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* RxI/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, A1, A2</td>
<td>RIC, R0, R1, A, C</td>
</tr>
<tr>
<td>Selective Loop Receive</td>
<td>C2</td>
<td>B0, B1, A1, 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</th>
<th>Receiver Interrupt Result Code</th>
<th>Rx Status After INT</th>
</tr>
</thead>
<tbody>
<tr>
<td>*</td>
<td>000000 A1 Match or General Receive</td>
<td>Active</td>
</tr>
<tr>
<td>*</td>
<td>000001 A2 Match</td>
<td>Active</td>
</tr>
<tr>
<td>000</td>
<td>000111 CRC Error</td>
<td>Active</td>
</tr>
<tr>
<td>000</td>
<td>001000 Abort Detected</td>
<td>Active</td>
</tr>
<tr>
<td>000</td>
<td>001010 Idle Detected</td>
<td>Disabled</td>
</tr>
<tr>
<td>000</td>
<td>001110 EOP Detected</td>
<td>Disabled</td>
</tr>
<tr>
<td>000</td>
<td>010000 Frame &lt; 32 Bits</td>
<td>Active</td>
</tr>
<tr>
<td>000</td>
<td>010010 DMA Overrun</td>
<td>Disabled</td>
</tr>
<tr>
<td>000</td>
<td>010100 Memory Buffer Overflow</td>
<td>Disabled</td>
</tr>
<tr>
<td>000</td>
<td>010110 Carrier Detect Failure</td>
<td>Disabled</td>
</tr>
<tr>
<td>000</td>
<td>011110 Receiver Interrupt Overrun</td>
<td>Disabled</td>
</tr>
</tbody>
</table>

* *D7–D5: Partial Byte Received

<table>
<thead>
<tr>
<th>D7–D0</th>
<th>Human Description</th>
<th>Binary Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>111</td>
<td>All 8 Bits of Last Byte</td>
<td>D0–D5</td>
</tr>
<tr>
<td>000</td>
<td>D0</td>
<td>D0–D0</td>
</tr>
<tr>
<td>100</td>
<td>D1–D0</td>
<td>D1–D0</td>
</tr>
<tr>
<td>010</td>
<td>D2–D0</td>
<td>D2–D0</td>
</tr>
<tr>
<td>110</td>
<td>D3–D0</td>
<td>D3–D0</td>
</tr>
<tr>
<td>001</td>
<td>D4–D0</td>
<td>D4–D0</td>
</tr>
<tr>
<td>101</td>
<td>D5–D0</td>
<td>D5–D0</td>
</tr>
<tr>
<td>011</td>
<td>D6–D0</td>
<td>D6–D0</td>
</tr>
</tbody>
</table>

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 reconditioning the receiver (resetting B0 and B1) is 20 bytes. Thus, it is common practice to recommend 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 Bo, 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 recommended 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 RX/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–D3) 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, control and information 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 Bi-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 not 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 is 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 Txl/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>CO/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>
<thead>
<tr>
<th>RIC D₇-D₀</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.

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.

<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></td>
<td>Read</td>
<td>23</td>
<td>None</td>
<td>Port Value</td>
</tr>
<tr>
<td>B Output</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
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.)
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-
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**

[Diagram of Loop Controller Flowchart showing the process flow with decision points and flowchart symbols.]
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 (1) 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 signals 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 1s 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
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 (B0 = 00, B1 = 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 Lo and Li 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 ODH 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.

**Figure 43. Sample 8273 Monitor I/O**

Figures 44 through 51 show the flowcharts used for the 8273 program development. The actual program listing is included as Appendix A. Figure 44 is the main status poll loop. After all devices are initialized and a prompt character displayed, a loop is entered at LOOPIT. This loop checks for a change of status in the result buffer or if a keyboard character has been received by the 8251 or if a poll frame has been received. If any of these conditions are met, the program branches to the appropriate routine. Otherwise, the loop is traversed again.

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 DISPLY routine.

DISPLY 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 L0, L1. 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 ‘+’. 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

ASSEMBLE: F1.RAY73.SRC

ISIS-II 0880/0885 MACRO ASSEMBLER, X188 MODULE PAGE 1

LOC OBJ SEQ SOURCE STATEMENT

1 INOPAGING MODES NOCOND
   0000 2 TRUE EQU 00H ; 00 FOR RAYTHEON
       3 ; ; FF FOR SELF-TEST
   0000 4 TRUE1 EQU 00H ; 00 FOR NORMAL RESPONSE
       5 ; ; FF FOR LOOP RESPONSE
   0000 6 IDENT EQU 00H ; 00 FOR NO DEMO
       7 ; ; FF FOR DEMO
   8 ; ;
   9 ;
   10 ; GENERAL 6273 MONITOR WITH RAYTHEON POLL MODE ADDED
   11 ;
   12 ;
   13 ; COMMAND SUPPORTED ARE:
   14 ; RS - RESET SERIAL I/O MODE
   15 ; SS - SET SERIAL I/O MODE
   16 ; RO - RESET OPERATING MODE
   17 ; SO - SET OPERATING MODE
   18 ; RD - RECEIVER DISABLE
   19 ; GR - GENERAL RECEIVE
   20 ; SR - SELECTIVE RECEIVE
   21 ; TF - TRANSMIT FRAME
   22 ; RF - ABORT FRAME
   23 ; SP - SET PORT B
   24 ; RP - RESET PORT B
   25 ; RB - RESET ONE BIT DELAY (PAR = 7F)
   26 ; SB - SET ONE BIT DELAY (PAR = 08)
   27 ; SL - SELECTIVE LOOP RECEIVE
   28 ; TL - TRANSMIT LOOP
   29 ; Z - CHANGE MODES FLIP/FLOP
   30 ;
   31 ;******************************************************
   32 ;
   33 ; NOTE: 'SET' COMMANDS IMPLEMENT LOGICAL 'OR' FUNCTIONS
   34 ; 'RESET' COMMANDS IMPLEMENT LOGICAL 'AND' FUNCTIONS
   35 ;******************************************************
   36 ;
   37 ;******************************************************
   38 ;
   39 ; BUFFERED MODE MUST BE SELECTED WHEN SELECTIVE RECEIVE IS USED.
   40 ;
   41 ; COMMAND FORMAT IS: 'COMMAND (2 LETTERS) 'PAR #1' 'PAR #2' ETC.
   42 ;
   43 ;******************************************************
   44 ;
   45 ;******************************************************
   46 ; THE TRANSMIT FRAME COMMAND FORMAT IS: 'TF' 'A' 'C' 'BUFFER CONTENTS'.
   47 ;
   48 ; NO LENGTH COUNT IS NEEDED. BUFFER CONTENTS IS ENDED WITH A CR.
   49 ;
   50 ;******************************************************
   51 ;
   52 ;******************************************************
   53 ;
   54 ;******************************************************
   55 ; PULLED MODE: WHEN PULLED MODE IS SELECTED (DENOTED BY A '+0' PROMPT), IF

2-332
A SNMP-P or RR(8)-P IS RECEIVED. A RESPONSE FRAME OF NSA-F OR RR(8)-F IS TRANSMITTED. OTHER COMMANDS OPERATE NORMALLY.

090 67 STAT7 EQU 90H ; STATUS REGISTER
090 68 COM7 EQU 90H ; COMMAND REGISTER
091 69 PAR7 EQU 91H ; PARAMETER REGISTER
091 70 RES7 EQU 91H ; RESULT REGISTER
092 71 TXI7 EQU 92H ; TX INTERRUPT RESULT REGISTER
093 72 RXI7 EQU 93H ; RX INTERRUPT RESULT REGISTER
093 73 TEST7 EQU 93H ; TEST MODE REGISTER
094 74 DBF EQU 94H ; PARAMETER BUFFER FULL BIT
094 75 TXINT EQU 95H ; TX INTERRUPT BIT IN STATUS REGISTER
095 76 RXINT EQU 96H ; RX INTERRUPT BIT IN STATUS REGISTER
095 77 TXR EQU 97H ; TX INT RESULT AVAILABLE BIT
096 78 RXR EQU 98H ; RX INT RESULT AVAILABLE BIT
098 79 ; 8253 EQUATES
098 80 62 MODES EQU 60H ; 6253 MODE WORD REGISTER
098 81 63 CNT8 EQU 61H ; COUNTER 0 REGISTER
098 82 64 CNT18 EQU 62H ; COUNTER 1 REGISTER
098 83 65 CNT28 EQU 63H ; COUNTER 2 REGISTER
099 84 66 CNT38 EQU 64H ; CONSOLE BAUD RATE (2400)
099 85 67 MONT0 EQU 65H ; MODE FOR COUNTER 0
099 86 68 MONT2 EQU 66H ; MODE FOR COUNTER 2
100 69 LIN8 EQU 67H ; 8273 BAUD RATE LSB ADDR
100 70 LIN9 EQU 68H ; 8273 BAUD RATE MSB ADDR
101 92 BAUD RATE TABLE:
102 | | 9000 2E 00 | 4000 5C 00 | 2400 69 00 | 1200 72 01 | 600 75 02 | 300 89 05
103 | | 9000 2E 00 | 4000 5C 00 | 2400 69 00 | 1200 72 01 | 600 75 02 | 300 89 05
104 | | 9000 2E 00 | 4000 5C 00 | 2400 69 00 | 1200 72 01 | 600 75 02 | 300 89 05
105 | | 9000 2E 00 | 4000 5C 00 | 2400 69 00 | 1200 72 01 | 600 75 02 | 300 89 05
106 | | 9000 2E 00 | 4000 5C 00 | 2400 69 00 | 1200 72 01 | 600 75 02 | 300 89 05
107 | | 9000 2E 00 | 4000 5C 00 | 2400 69 00 | 1200 72 01 | 600 75 02 | 300 89 05
108 | | 9000 2E 00 | 4000 5C 00 | 2400 69 00 | 1200 72 01 | 600 75 02 | 300 89 05
109 | | 9000 2E 00 | 4000 5C 00 | 2400 69 00 | 1200 72 01 | 600 75 02 | 300 89 05
110 | | 9000 2E 00 | 4000 5C 00 | 2400 69 00 | 1200 72 01 | 600 75 02 | 300 89 05
111 | | 9000 2E 00 | 4000 5C 00 | 2400 69 00 | 1200 72 01 | 600 75 02 | 300 89 05
112 | | 9000 2E 00 | 4000 5C 00 | 2400 69 00 | 1200 72 01 | 600 75 02 | 300 89 05
113 | | 9000 2E 00 | 4000 5C 00 | 2400 69 00 | 1200 72 01 | 600 75 02 | 300 89 05
114 | | 9000 2E 00 | 4000 5C 00 | 2400 69 00 | 1200 72 01 | 600 75 02 | 300 89 05
115 | | 9000 2E 00 | 4000 5C 00 | 2400 69 00 | 1200 72 01 | 600 75 02 | 300 89 05
116 | | 9000 2E 00 | 4000 5C 00 | 2400 69 00 | 1200 72 01 | 600 75 02 | 300 89 05
117 ;
118 ; 8251A EQUATES
119 :
0089 818 CNTLS1 EQU 09H ; CONTROL WORD REGISTER
0089 812 STATS1 EQU 08H ; STATUS REGISTER
0089 812 TXD51 EQU 08H ; TX DATA REGISTER
0089 812 RXD51 EQU 09H ; RX DATA REGISTER
0089 814 MODE51 EQU 0CEH ; MODE 16K 2 STOP 1 NO PARITY
0089 815 CMD51 EQU 27H ; COMMAND: ENABLE TX/RX
0089 816 RDY EQU 02H ; RXD BIT
127 ;
128 ; MONITOR SUBROUTINE EQUATES
129 :
065F 818 GETCH EQU 061FH ; GET CHR FROM KEYBOARD, ASCII IN CH
05F8 813 ECH(X) EQU 05F8H ; ECHO CHR TO DISPLAY
075E 812 VALDG EQU 075EH ; CHECK IF VALID DIGIT, CARRY SET IF VALID
05BB 813 CAVSN EQU 05BBH ; CONVERTS ASCII TO HEX
05EB 814 COLF EQU 05EBH ; DISPLAY CR, HENCE LF TOO
06C7 815 MNOUT EQU 06C7H ; CONVERT BYTE TO 2 ASCII CHAR AND DISPLAY
126 ;
127 ; MISC EQUATES
128 :
2000 819 STKSTR EQU 20C8H ; STACK START
0003 814 CNTLC EQU 03H ; CNTL-C EQUIVALENT
0004 814 MONTOR EQU 0000H ; MONITOR
0006 814 CHBUFS EQU 2088H ; START OF COMMAND BUFFER
0200 814 CMDBUF EQU 2088H ; POLLE MODE SPECIAL TX COMMAND BUFFER
0000 814 CR EQU 00H ; ASCII CR
000A 814 LF EQU 0AH ; ASCII LF
2004 814 BST75 EQU 2004H ; BST75.JUMP ADDRESS
20CE 814 BST65 EQU 203CH ; BST65.JUMP ADDRESS
2010 814 LDADD EQU 2010H ; RESULT BUFFER LOAD POINTER STORAGE
2013 814 CARDR EQU 2013H ; RESULT BUFFER CONSOLE POINTER STORAGE
2080 814 RESBUF EQU 2080H ; RESULT BUFFER START - 255 BYTES
0083 815 SNRP EQU 53H ; SNRP-P CONTROL CODE
0011 815 RRPP EQU 11H ; RR(0)-P CONTROL CODE
0073 815 NRSP EQU 73H ; NRSP-F CONTROL CODE
0011 815 RRFP EQU 11H ; RR(0)-F CONTROL CODE
2015 814 PAFT EQU 2015H ; P AFT STORAGE
2016 814 POLIN EQU 2016H ; POLL MODE SELECTION INDICATOR
2027 815 DEMODE EQU 2827H ; DEMODE INDICATOR
161 ;
162 ;******************************************************************************
163 ; RAM STORAGE DEFINITIONS:
165 ; LOC \ DEF
166 ;
167 ; 2000-208F COMMAND BUFFER
168 ; 2080-208F RESULT BUFFER LOAD POINTER
169 ; 2012-2014 RESULT BUFFER CONSOLE POINTER
170 ; 2015 PROMPT CHARACTER STORAGE
171 ; 2016 POLL MODE INDICATOR
172 ; 2017 BRAID RATE LSB FOR SELF-TEST
173 ; 2018 BRAID RATE MSB FOR SELF-TEST
177 ; 2019 SPARE
179 ; 2020-2026 RESPONSE COMMAND BUFFER FOR POLL MODE
180 ; 2080-28FF RESULT BUFFER
181 ;
182 ;******************************************************************************
611001-47

2-334
AP-36

183;
184: PROGRAM START
185;
186: INITIALIZE 8253, 8257, 8251A, AND RESET 8273.
187: ALSO SET NORMAL MODE AND PRINT SIGNON MESSAGE
188;
0000 120 000H
190
0000 310620 191 START: LXI SP, STKSRT ; INITIALIZE SP
0003 3E36 192 MVI A, MONT0 ; 8253 MODE SET
0005 0338 193 OUT MODE3 ; 8253 MODE PORT
0007 3A1720 194 LDA LXBR1 ; GET 8273 BAUD RATE LSB
0009 033C 195 OUT CNTBS3 ; USING COUNTER 0 AS BAUD RATE GEN
000C 3A1620 196 LDA LXBR2 ; GET 8273 BAUD RATE MSB
000F 0339 197 OUT CNTBS3 ; COUNTER 0
0011 CO1080 198 CALL RXDMA ; INITIALIZE 8257 RX DMA CHANNEL
0014 CO3D80 199 CALL TXDMA ; INITIALIZE 8257 TX DMA CHANNEL
0017 3E81 200 MVI A, 00H ; OUTPUT 1 FOLLOWED BY A 0
0019 0332 201 OUT TEST73 ; TO TEST MODE REGISTER
001B 3E80 202 MVI A, 00H ; TO RESET THE 8273
001D 0332 203 OUT TEST73
001F 3E82 204 MVI A, 00H ; NORMAL MODE PROMPT CH
0021 321520 205 STR JMPF ; PUT IN STORAGE
0023 3E80 206 MVI A, 00H ; TX POLL RESPONSE INDICATOR
0025 321620 207 STR POLIN ; 0 MEANS NO SPECIAL TA
0027 322220 208 STR DEMODE ; CLEAR DEMODE
0029 21030C 209 LXI H, SIGNON ; SIGNON MESSAGE ADDR
002F C020AC 210 CALL TMS50 ; DISPLAY SIGNON
211 ;
212 ; MONITOR USES JUMPS IN RAM TO DIRECT INTERRUPTS
213 ;
0032 210420 214 LXI H, RST75 ; RST7.5 JUMP LOCATION USED BY MONITOR
0035 01000C 215 LXI B, RXI ; ADDRESS OF RX INT ROUTINE
0038 36C3 216 MVI M, BCCH ; LOAD 'JMP'_OPCODE
003A 23 217 INX H ; INC POINTER
003C 71 218 MVI M, C ; LOAD RXI LSB
003E 23 219 INX H ; INC POINTER
0040 70 220 MVI M, B ; LOAD RXI MSB
0043 21CE28 221 LXI H, RST65 ; RST6.5 JUMP LOCATION USED BY MONITOR
0044 01EBC 222 LXI B, TXI ; ADDRESS OF TX INT ROUTINE
0046 36C3 223 MVI M, BCCH ; LOAD 'JMP'_OPCODE
0048 23 224 INX H ; INC POINTER
004A 71 225 MVI M, C ; LOAD TXI LSB
004C 23 226 INX H ; INC POINTER
004E 70 227 MVI M, B ; LOAD TXI MSB
0050 3E18 228 MVI A, 18H ; SET TO RESET INTERRUPTS
0052 38 229 SIM ; RESET INTERRUPTS
0054 F8 230 EI ; ENABLE INTERRUPTS
231 ;
232 ; INITIALIZED BUFFER POINTER
233 ;
005E 210028 234 LXI H, RESBUF ; SET RESULT BUFFER POINTERS
0061 221320 235 SHLD CNDIR ; RESULT CONSOLE POINTER
0064 221820 236 SHLD LOHR ; RESULT LOAD POINTER
237 ;
242 ; MAIN PROGRAM LOOP - CHECKS FOR CHANGE IN RESULT POINTERS, USART STATUS.
243 ; OR POLL STATUS
0857 CDFB85 245 CMDREC: CALL CRLF ;DISPLAY CR
0859 3R1520 246 LDA PRMT ;GET CURRENT PROMPT CHR
0859 4F 247 MOV C,A ;MOVE TO C
0859 CDFB85 248 CALL ECHO ;DISPLAY IT
0861 3R1520 249 LOOPIT: LHD CARDR ;GET CONSOLE POINTER
0864 7D 250 MOV A,L ;SAVE POINTER LSB
0865 3R1520 251 LHD LOADR ;GET LOAD POINTER
0868 BD 252 CMP L ;SAME LSB?
0869 C2398A 253 JNZ DISPY ;NO, RESULTS NEED DISPLAYING
086C BD89 259 IN STATSI ;YES, CHECK KEYBOARD
086E 6802 260 ANI ROV ;CHR RECEIVED?
0870 C27098 261 JNZ GETCMD ;CHR MUST BE CHAR SO GO GET IT
0872 341628 262 LDA POLIN ;GET POLL MODE STATUS
0875 87 263 ANA A ;IS IT B?
0877 C24089 264 JNZ TOPOL ;NO, THEN POLL OCCURRED
0879 C36188 265 JMP LOOPIT ;YES, TRY AGAIN
087B 267 ;
087D CDFB86 268 ;COMMAND RECOGNIZER ROUTINE
087F CDF885 269 ;
0883 79 270 ;
0884 FE52 273 MOV A,C ;SETUP FOR COMPARE
0886 CDFB88 275 JZ ROWN ;GET MORE
0888 FE53 276 CPI 'S' ;??
0889 C07B88 277 JZ SDWN ;GET MORE
088E FE47 278 CPI 'B' ;??
0890 CDFB88 279 JZ SDWN ;GET MORE
0893 FE54 280 CPI 'T' ;??
0895 C0B89 281 JZ TOWN ;GET MORE
0898 FE41 282 CPI 'A' ;??
0899 C02289 283 JZ ROWN ;GET MORE
089D FE38 284 CPI 'Z' ;??
089F C01689 285 JZ CMDRE ;YES, GO CHANGE MODE
08A2 FE83 290 CPI CNTLC ;CNTL-C?
08A4 C06889 291 JZ MONTOR ;EXIT TO MONITOR
08A7 0E3F 292 ILLEG: MVI C, '?' ;PRINT ?
08A9 CDFB85 293 CALL ECHO ;DISPLAY IT
08B0 C35788 294 JMP CMDREC ;LOOP FOR COMMAND
08B3 CDFB86 295 ;
08B9 CDF885 296 ;CALL GETCH ;GET NEXT CHR
08B9 CDFB85 297 ;CALL ECHO ;ECHO IT
08B9 79 298 MOV A,C ;SETUP FOR COMPARE
08BE FE4F 299 CPI 'O' ;??
08BF C05089 300 JZ RCOND ;RO COMMAND
08B9 FE53 301 CPI 'S' ;??
08BB C06789 302 JZ ROCMD ;RS COMMAND
08B9 FE44 303 CPI 'b' ;??
08C2 CR7189 304 JZ RCOND ;RO COMMAND
08C5 FE50 305 CPI 'p' ;??
08C7 C0D89 306 JZ RCOND ;RP COMMAND
08CA FE52 307 CPI 'R' ;??
08CB C00888 308 JZ START ;START OVER
08CF FE42 309 CPI 'b' ;??
08D1 CR7089 310 JZ ROCOND ;RB COMMAND

611001-49

2-336
004 C3A788 311 JMP ILLEG ; ILLEGAL, TRY AGAIN

007 D0F86 313 DOW: CALL GETCH ; GET NEXT CHR
00A D0F85 314 CALL ECHO ; ECHO IT
00D 70 315 MOV A, B ; SETUP FOR COMPARE
00F FE4F 316 CPI 'O'; GT
018 C3A509 317 JZ SCMD ; SS COMMAND
018 FE33 318 CPI 'S'; ST
018 C3A009 319 JZ SCMD ; SS COMMAND
018 FE52 320 CPI 'R' ; R?
01B C3A009 321 JZ SCMD ; SR COMMAND
01E FE30 322 CPI 'P'; SP COMMAND
022 C32E99 323 JZ SCMD ; SP COMMAND
027 FE42 324 CPI 'B'; B?
024 C3A509 325 JZ SCMD ; SB COMMAND
02D FE4C 326 CPI 'L'; L?
029 C3A009 327 JZ SCMD ; SL COMMAND
02F C3A788 328 JMP ILLEG ; ILLEGAL, TRY AGAIN

03F C3A788 331 JMP ILLEG ; ILLEGAL, TRY AGAIN
03C C3A086 333 CALL GETCH ; GET NEXT CHR
03D 70 334 MOV A, B ; SETUP FOR COMPARE
03D FE46 335 CPI 'F' ; F?
03B C3A086 336 CALL ECHO ; ECHO IT
03A FE4C 337 CPI 'L'; L?
03A C3A009 338 CALL GETCH ; GET NEXT CHR
031 C3A788 339 JMP ILLEG ; ILLEGAL, TRY AGAIN

040 C3A086 340 CALL GETCH ; GET NEXT CHR
043 70 341 MOV A, B ; SETUP FOR COMPARE
046 FE46 342 CPI 'F' ; F?
048 C3A086 343 CALL ECHO ; ECHO IT
04B C3A009 344 CALL GETCH ; GET NEXT CHR
04D FE4C 345 CPI 'L'; L?
04A C3A009 346 CALL GETCH ; GET NEXT CHR
04E C3A788 347 JMP ILLEG ; ILLEGAL, TRY AGAIN

051 C3A086 348 CALL GETCH ; GET NEXT CHR
054 70 349 MOV A, B ; SETUP FOR COMPARE
057 FE46 350 CPI 'F' ; F?
059 C3A086 351 CALL ECHO ; ECHO IT
05C 70 352 MOV A, B ; SETUP FOR COMPARE
05F FE46 353 CPI 'F' ; F?
062 C3A009 354 CALL GETCH ; GET NEXT CHR
064 FE4C 355 CPI 'L'; L?
061 C3A009 356 CALL GETCH ; GET NEXT CHR
065 FE4C 357 CPI 'L'; L?

068 C3A086 358 CALL GETCH ; GET NEXT CHR
06B C3A086 359 CALL GETCH ; GET NEXT CHR
06E C3A086 360 CALL GETCH ; GET NEXT CHR
071 C3A086 361 CALL GETCH ; GET NEXT CHR

091 F3 359 CNCODE: DI ; DISABLE INTERRUPTS
092 341520 356 LDA PROMPT ; GET CURRENT PROMPT
095 FE2D 357 CPI '-=' ; NORMAL MODE?
097 C243B 358 JNZ SH ; NO, CHANGE IT
09A 3E89 359 MOV A, 4 ; NEW PROMPT
09B 321530 360 STR PROMPT ; STORE NEW PROMPT
09F FB 363 EI ; ENABLE INTERRUPTS
09E C35708 366 JMP CMREC ; RETURN TO LOOP
09D 3E2D 367 SH: MOV A, 4 ; NEW PROMPT CHR
09C 321530 368 STR PROMPT ; STORE IT
09A 3E89 369 MOV A, 4 ; NEW PROMPT CHR
09B C35708 370 JMP CMREC ; RETURN TO LOOP

354 ; 355 ; 356 ;

093 F3 359 CNCODE: DI ; DISABLE INTERRUPTS
094 341520 356 LDA PROMPT ; GET CURRENT PROMPT
097 FE2D 357 CPI '-=' ; NORMAL MODE?
099 C243B 358 JNZ SH ; NO, CHANGE IT
09B 3E89 359 MOV A, 4 ; NEW PROMPT
09C 321530 360 STR PROMPT ; STORE NEW PROMPT
09F FB 363 EI ; ENABLE INTERRUPTS
09E C35708 366 JMP CMREC ; RETURN TO LOOP
09D 3E2D 367 SH: MOV A, 4 ; NEW PROMPT CHR
09C 321530 368 STR PROMPT ; STORE IT
09A 3E89 369 MOV A, 4 ; NEW PROMPT CHR
09B C35708 370 JMP CMREC ; RETURN TO LOOP

371 ; 372 ;
373 TRANSMIT ANSWER TO POLL SETUP
374
094C 3E00 382 XTPOL: MVI A.00H CLEAR POLL INDICATOR
094E 221620 394 STA POLIN INDICATOR ADDR
0951 261288 395 LXI H.LOOPF SETUP STACK FOR COMMAND OUTPUT
0954 23 396 PUSH H INPUT RETURRN TO CMODE ON STACK
0955 0684 397 MVI B.04H GET # OF PARAMETERS READY
0957 212020 398 LXI H.CHDP1 POINT TO SPECIAL BUFFER
095A C3780 399 JMP COM2 JUMP TO COMMAND OUTPUTER
390
391
392
393 COMMAND IMPLEMENTING Routines
394
395
396 A0 - RESET OPERATING MODE
397
095D 0A01 398 R0CMD: MVI B.01H # OF PARAMETERS
095F 0E31 399 MVI C.51H COMMAND
0961 C3E0A 400 CALL COM GET PARAMETERS AND ISSUE COMMAND
0964 C3780 401 JMP CMOREC GET NEXT COMMAND
402
403 :A5 - DROP SERIAL I/O MODE COMMAND
404
0967 0A01 405 R0CMD: MVI B.01H # OF PARAMETERS
0969 0E5A 406 MVI C.6AH COMMAND
096B C3E0A 407 CALL CM GET PARAMETERS AND ISSUE COMMAND
096E C3780 408 JMP CMOREC GET NEXT COMMAND
409
410 A6 - RECEIVER DISABLE COMMAND
411
0971 0B00 412 R0CMD: MVI B.00H # OF PARAMETERS
0973 0E35 413 MVI C.05H COMMAND
0975 C3E0A 414 CALL CM GET ISSUE COMMAND
0978 C3780 415 JMP CMOREC GET NEXT COMMAND
416
417 A7 - RESET ONE BIT DELAY COMMAND
418
0978 0A01 419 R0CMD: MVI B.01H # OF PARAMETERS
097D 0E5A 420 MVI C.6AH COMMAND
097F C3E0A 421 CALL CM GET PARAMETER AND ISSUE COMMAND
0982 C3780 422 JMP CMOREC GET NEXT COMMAND
423
424 AS - SET ONE BIT DELAY COMMAND
425
0985 0A01 426 R0CMD: MVI B.01H # OF PARAMETERS
0987 0E5A 427 MVI C.6AH COMMAND
0989 C3E0A 428 CALL CM GET PARAMETER AND ISSUE COMMAND
098C C3780 429 JMP CMOREC GET NEXT COMMAND
430
431 SL - SELECTIVE LOOP RECEIVE COMMAND
432
098F 0A04 433 SLCMD: MVI B.04H # OF PARAMETERS
0991 0E52 434 MVI C.02H COMMAND
0993 C3E0A 435 CALL CM GET PARAMETERS AND ISSUE COMMAND
0996 C3780 436 JMP CMOREC GET NEXT COMMAND
437
438 TL - TRANSMIT LOOP COMMAND

611001-51
439;
0999 210020 440 TCM: LXI H,CHEBUF ; SET COMMAND BUFFER POINTER
099C 0682 441 MVI B,02H ; LOAD PARAMETER COUNTER
099E 300A 442 MVI M,0BH ; LOAD COMMAND INTO BUFFER
09A0 210020 443 LXI H,CHEBUF+2 ; POINT AT ADDR AND CNTL POSITIONS
09A3 C3500 444 JMP TFCMD ; FINISH OFF COMMAND IN TF ROUTINE
445;
446:50 - SET OPERATING MODE COMMAND
447;
09A6 0681 448 50CMOD: MVI B,01H ; # OF PARAMETERS
09A8 0E91 449 MVI C,05H ; COMMAND
09A9 C0350A 450 CALL CMD ; GET PARAMETER AND ISSUE COMMAND
09AD C35708 451 JMP CMDREC ; GET NEXT COMMAND
452;
453:55 - SET SERIAL I/O COMMAND
454;
09BA 0681 455 55CMOD: MVI B,05H ; # OF PARAMETERS
09BB 0E90 456 MVI C,0AH ; COMMAND
09BC C0350A 457 CALL CMD ; GET PARAMETER AND ISSUE COMMAND
09BD C35708 458 JMP CMDREC ; GET NEXT COMMAND
459;
460:SR - SELECTIVE RECEIVE COMMAND
461;
09BE 0684 462 80CMOD: MVI B,04H ; NO PARAMETERS
09BF 0E91 463 MVI C,05H ; COMMAND
09C0 C0350A 464 CALL CMD ; GET PARAMETERS AND ISSUE COMMAND
09C1 C35708 465 JMP CMDREC ; GET NEXT COMMAND
466;
467:GR - GENERAL RECEIVE COMMAND
468;
09C4 0682 469 GRCMOD: MVI B,02H ; NO PARAMETERS
09C6 0E90 470 MVI C,0AH ; COMMAND
09C8 C0350A 471 CALL CMD ; ISSUE COMMAND
09CA C35708 472 JMP CMDREC ; GET NEXT COMMAND
473;
474:AF - ABORT FRAME COMMAND
475;
09CD 0680 476 A0CMOD: MVI B,00H ; NO PARAMETERS
09CF 0E90 477 MVI C,0AH ; COMMAND
09D0 C0350A 478 CALL CMD ; ISSUE COMMAND
09D2 C35708 479 JMP CMDREC ; GET NEXT COMMAND
480;
481:RP - RESET PORT COMMAND
482;
09D4 0681 483 RPCMD: MVI B,01H ; # OF PARAMETERS
09D5 0E93 484 MVI C,63H ; COMMAND
09DC C0350A 485 CALL CMD ; GET PARAMETER AND ISSUE COMMAND
09DF C35708 486 JMP CMDREC ; GET NEXT COMMAND
487;
488:SP - SET PORT COMMAND
489;
09E0 0681 489 SPCMD: MVI B,01H ; # OF PARAMETERS
09E2 0E93 490 MVI C,63H ; COMMAND
09ED C0350A 491 CALL CMD ; GET PARAMETER AND ISSUE COMMAND
09EF C35708 492 JMP CMDREC ; GET NEXT COMMAND
494;
495:TF - TRANSMIT FRAME COMMAND
496;
611001-52
89EC 218829 497
TFCII1: MOY A,R ; LOAD POINT AT ADR AND CTRL POSITIONS
89EF 9692 498 MY
tfl(1 CII)IItF
89F1 36C8 499 /W
tfl(1 CIUlT
89F3 219229 599 LX
tfl(1 CIUH, CI'I>8UF+2 ; SET BUFFER POINTER
89F6 78 501 TFCMD: MOV A,B ; GET TEST PARAMETER COUNT
89F7 87 502 ANA A ; IS IT O?
89F8 CA78R 563 JZ TXBUFL ; YES, LOAD TX DATA BUFFER
89F8 CD0AR 564 CALL PARAMETER
89FE DB18829 585 J? CII)IItF ; ILLEGAL CHAR RETURNED
8AF1 23 586 INK H ; INC COMMAND BUFFER POINTER
8AF2 85 587 DCR B ; DEC PARAMETER COUNTER
8AF3 77 588 MOV M,A ; LOAD PARAMETER INTO COMMAND BUFFER
8AF4 CF089 589 JMP TFCMD ; GET NEXT PARAMETER
158
8AE7 218888 511 TXBUFL: LXI H,TXBUF ; LOAD TX DATA BUFFER POINTER
8AF0 818888 512 LXI B,0000H ; CLEAR BC - BYTE COUNTER
8AE0 C5 513 TXBUFL: PUSH B ; SAME BYTE COUNTER
8AE0 CD0AR 514 CALL PARAMETER ; GET DATA ALIAS PARAMETER
8AE1 401888 515 JC ENDCII) ; MAYBE END IF ILLEGAL
8AE1 77 516 MOV M,A ; LOAD DATA BYTE INTO BUFFER
8AE5 23 517 INK H ; INC BUFFER POINTER
8AE6 C1 518 POP B ; RESTORE BYTE COUNTER
8AE7 83 519 INX ; INC BYTE COUNTER
8AE8 38088 520 JMP TXBUFL ; GET NEXT DATA
8AF3 FE00 521 ENDCII) ; CPI CR ; RETURNED ILLEGAL CHAR CR?
8AF6 CD0AR 522 JZ TXBUFL ; YES, THEN TX BUFFER FULL
8AF8 C1 523 POP B ; RESTORE B TO SAVE STACK
8AF9 C3780 524 JMP ILLEG ; ILLEGAL CHAR
8AFD C1 525 TXBUFL: POP B ; RESTORE BYTE COUNTER
8AFD 218120 526 LXI H,TXBUF+1 ; POINT INTO COMMAND BUFFER
8AE0 71 527 MOV M,C ; STORE BYTE INTO LSB
8AE0 23 528 INX H ; INC POINTER
8AE2 70 529 MOV M,B ; STORE BYTE INTO MSB
8AE4 86084 530 LXI B,0000H ; LOAD PARAMETER COUNT INTO B
8AE6 2188829 531 LXI H,TFRET ; GET RETURN ADR FOR THIS ROUTINE
8AE8 C5 532 PUSH B ; PUSH ONCE
8AE9 E3 533 XTHL ; RETURN ON STACK
8AEF C5 534 PUSH B ; PUSH IT SO CMOVE CAN USE IT
8AF3 CF089 535 JMP CMOVE ; ISSUE COMMAND
8AF6 C5780 536 TFRET: JMP CMDREC ; GET NEXT COMMAND
537 ;
539 ; ROUTINE TO DISPLAY RESULT IN RESULT BUFFER WHEN LOAD AND CONSOLE
540 ; POINTERS ARE DIFFERENT.
541 ;
542 ;
8AE9 16985 543 DISPY: MV1 D,05H ; AND IS RESULT COUNTER
8AF8 218320 544 LHLD CMDRA ; GET CONSOLE POINTER
8AFE E5 545 PUSH H ; SAVE IT
8AF3 7E 546 MOV A,M ; GET RESULT IC
8AF4 E6F 547 ANI $FH ; LIMIT TO RESULT CODE
8AF2 F2BC 548 CPI $CH ; TEST IF RX OR TX SOURCE
8AF4 D628829 549 JC RXSORC ; CARRY, THEN RX SOURCE
8AF7 21C38C 550 TXSORC: LXI H,TEXMSG ; TX INT MESSAGE
8AF4 C928C 551 CALL TXMSG ; DISPLAY IT
8AF4 E1 552 DISPY: POP H ; RESTORE CONSOLE POINTER
8AF6 7E 553 DISPY: MOV A,M ; GET RESULT
8AF6 CD076C 554 CALL INOUT ; CONVERT AND DISPLAY

611001-53

2-340
AP-36

0052 BE20 555 MVI C,’ ; SP CHR
0054 DF85 556 CALL ECHO ; DISPLAY IT
0057 2C 557 INR L ; INC BUFFER POINTER
0058 15 558 DCR D ; DEC RESULT COUNTER
0059 C4E0R 559 JNZ DI5PV1 ; NOT DONE
005C 22320 560 SHLD CNDR ; UPDATE CONSOLE POINTER
005F C5780 561 JMP CNDRREC ; RETURN TO LOOP
562 ;
563 ;
564 ; RECEIVER SOURCE - DISPLAY RESULTS AND RECEIVE BUFFER CONTENTS
565 ;
566 ;
0062 2188C 567 RXS0REC: LXI H,RXMSG ; RX INT MESSAGE ADDR
0065 DF28C 568 CALL TMXSG ; DISPLAY MESSAGE
0068 E1 569 POP H ; RESTORE CONSOLE POINTER
0069 7E 570 RXS1: MOV A,M ; RETRIEVE RESULT FROM BUFFER
006A DC76E 571 CALL MNOUT ; CONVERT AND DISPLAY IT
006D BE20 572 MVI C,’ ; ASCII SP
006F DF85 573 CALL ECHO ; DISPLAY IT
0072 2C 574 INR L ; INC CONSOLE POINTER
0073 15 575 DCR D ; DEC RESULT COUNTER
0074 7R 576 MOV A.D ; GET SET TO TEST COUNTER
0075 EB4 577 CPI B4H ; IS THE RESULT R8?
0077 CA26E 578 JZ RIPT ; YES, GO SAVE IT
0079 EB3 579 CPI B3H ; IS THE RESULT R1?
007C CA7BA 580 JZ RIPT ; YES, GO SAVE IT
007F 7F 581 RXS2: ANA A ; TEST RESULT COUNTER
0080 C260A 582 JNZ RXS1 ; NOT DONE YET, GET NEXT RESULT
0083 22328 583 SHLD CNDR ; DONE, SO UPDATE CONSOLE POINTER
0086 DF85 584 CALL CRLF ; DISPLAY CR
0089 21882 585 LXI H,RXBUF ; POINT AT RX BUFFER
008C 7E 586 POP B ; RETRIEVE RECEIVED COUNT
008D 7B 587 RXS3: MOV A,B ; IS COUNT R7?
008E B1 588 ORR C ;
008F C95708 589 JZ CNDRREC ; YES, GO BACK TO LOOP
0092 7E 590 MOV A.M ; NO, GET CHR
0093 C5 591 PUSH B ; SAVE BC
0094 DC76E 592 CALL MNOUT ; CONVERT AND DISPLAY CHR
0097 BE20 593 MVI C,’ ; ASCII SP
0099 DF85 594 CALL ECHO ; DISPLAY IT TO SEPARATE DATA
009C C1 595 POP B ; RESTORE BC
009D BB 596 DCX B ; DEC COUNT
009E 23 597 INX H ; INC POINTER
009F C380AR 598 JMP RXS3 ; GET NEXT CHR
599 ;
00A2 4E 59A RIPT: MOV C.M ; GET RB FOR RESULT BUFFER
00A3 C5 59B PUSH B ; SAVE IT
00A4 C7F08 59C JMP RXS2 ; RETURN
59D ;
00A7 C1 59E RIPT: POP B ; GET RB
00A8 46 59F MOV B,M ; GET RX FOR RESULT BUFFER
00A9 C5 5A0 PUSH B ; SAVE IT
00AA C7F08 5A1 JMP RXS2
5A2 ;
5A3 ;
5A4 ;
5A5 ;
5A6 ;
5A7 ;
5A8 ;
5A9 ;
5AA ;
5AB ;
5AC ;
5AD ;
5AE ;
5AF ;
600 ;
601 ;
602 ;
603 ;
604 ;
605 ;
606 ;
607 ;
608 ;
609 ;
60A ;
60B ;
60C ;
60D ;
60E ;
60F ;
610 ;
611 ; PARAMETER INPUT - PARAMETER RETURNED IN E REGISTER
612 ;
<table>
<thead>
<tr>
<th>Off</th>
<th>Hex</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>00F0</td>
<td>671</td>
<td>IN STR73</td>
</tr>
<tr>
<td>00F0</td>
<td>672</td>
<td>RLC</td>
</tr>
<tr>
<td>00F0</td>
<td>673</td>
<td>JC</td>
</tr>
<tr>
<td>00F0</td>
<td>674</td>
<td>MOV A,M</td>
</tr>
<tr>
<td>00F0</td>
<td>675</td>
<td>OUT STR73</td>
</tr>
<tr>
<td>00F0</td>
<td>676</td>
<td>PAR1: MOV A,B</td>
</tr>
<tr>
<td>00F0</td>
<td>677</td>
<td>ANA R</td>
</tr>
<tr>
<td>00F0</td>
<td>678</td>
<td>RZ</td>
</tr>
<tr>
<td>00F0</td>
<td>679</td>
<td>INX H</td>
</tr>
<tr>
<td>00F0</td>
<td>680</td>
<td>DCR B</td>
</tr>
<tr>
<td>00F0</td>
<td>681</td>
<td>IN STR73</td>
</tr>
<tr>
<td>00F0</td>
<td>682</td>
<td>ANI CPBF</td>
</tr>
<tr>
<td>00F0</td>
<td>683</td>
<td>JNZ PAR2</td>
</tr>
<tr>
<td>00F0</td>
<td>684</td>
<td>MOV A,M</td>
</tr>
<tr>
<td>00F0</td>
<td>685</td>
<td>OUT PAR72</td>
</tr>
<tr>
<td>00F0</td>
<td>686</td>
<td>JMP PAR1</td>
</tr>
<tr>
<td>00F0</td>
<td>687</td>
<td>READ STATUS</td>
</tr>
<tr>
<td>00F0</td>
<td>688</td>
<td>DCR B</td>
</tr>
<tr>
<td>00F0</td>
<td>689</td>
<td>INITIALIZE AND ENABLE RX DMA CHANNEL</td>
</tr>
<tr>
<td>00F0</td>
<td>690</td>
<td></td>
</tr>
<tr>
<td>00F0</td>
<td>691</td>
<td></td>
</tr>
<tr>
<td>00F0</td>
<td>692</td>
<td>RODMA: MVI A,DTOMA</td>
</tr>
<tr>
<td>00F0</td>
<td>693</td>
<td>OUT MODE7: 8257 MODE PORT</td>
</tr>
<tr>
<td>00F0</td>
<td>694</td>
<td>LIX B:ROBUF</td>
</tr>
<tr>
<td>00F0</td>
<td>695</td>
<td>MOV A,C</td>
</tr>
<tr>
<td>00F0</td>
<td>696</td>
<td>OUT CHANOR</td>
</tr>
<tr>
<td>00F0</td>
<td>697</td>
<td>MOV A,B</td>
</tr>
<tr>
<td>00F0</td>
<td>698</td>
<td>OUT CHANOR</td>
</tr>
<tr>
<td>00F0</td>
<td>699</td>
<td>LIX B:RTXC</td>
</tr>
<tr>
<td>00F0</td>
<td>700</td>
<td>MOV A,C</td>
</tr>
<tr>
<td>00F0</td>
<td>701</td>
<td>OUT CHANOR</td>
</tr>
<tr>
<td>00F0</td>
<td>702</td>
<td>MOV A,B</td>
</tr>
<tr>
<td>00F0</td>
<td>703</td>
<td>OUT CHANOR</td>
</tr>
<tr>
<td>00F0</td>
<td>704</td>
<td>MVI A,ENDMA</td>
</tr>
<tr>
<td>00F0</td>
<td>705</td>
<td>OUT MODE7: 8257 MODE PORT</td>
</tr>
<tr>
<td>00F0</td>
<td>706</td>
<td>RET</td>
</tr>
<tr>
<td>00F0</td>
<td>707</td>
<td></td>
</tr>
<tr>
<td>00F0</td>
<td>708</td>
<td></td>
</tr>
<tr>
<td>00F0</td>
<td>709</td>
<td>INITIALIZE AND ENABLE TX DMA CHANNEL</td>
</tr>
<tr>
<td>70A</td>
<td></td>
<td></td>
</tr>
<tr>
<td>70A</td>
<td></td>
<td></td>
</tr>
<tr>
<td>00F0</td>
<td>710</td>
<td>TDXMA: MVI A,DTOMA</td>
</tr>
<tr>
<td>00F0</td>
<td>711</td>
<td>OUT MODE7: 8257 MODE PORT</td>
</tr>
<tr>
<td>00F0</td>
<td>712</td>
<td>LIX B:TXTC</td>
</tr>
<tr>
<td>00F0</td>
<td>713</td>
<td>MOV A,C</td>
</tr>
<tr>
<td>00F0</td>
<td>714</td>
<td>OUT CHANOR</td>
</tr>
<tr>
<td>00F0</td>
<td>715</td>
<td>MOV A,B</td>
</tr>
<tr>
<td>00F0</td>
<td>716</td>
<td>OUT CHANOR</td>
</tr>
<tr>
<td>00F0</td>
<td>717</td>
<td>MOV A,B</td>
</tr>
<tr>
<td>00F0</td>
<td>718</td>
<td>OUT CHANOR</td>
</tr>
<tr>
<td>00F0</td>
<td>719</td>
<td>LIX B:TMAL:</td>
</tr>
<tr>
<td>00F0</td>
<td>71A</td>
<td>OUT CHANOR</td>
</tr>
<tr>
<td>00F0</td>
<td>71B</td>
<td>MOV A,C</td>
</tr>
<tr>
<td>00F0</td>
<td>71C</td>
<td>OUT CHANOR</td>
</tr>
<tr>
<td>00F0</td>
<td>71D</td>
<td>OUT CHANOR</td>
</tr>
<tr>
<td>00F0</td>
<td>71E</td>
<td>MOV A,B</td>
</tr>
<tr>
<td>00F0</td>
<td>71F</td>
<td>OUT CHANOR</td>
</tr>
<tr>
<td>00F0</td>
<td>720</td>
<td>MVI A,ENDMA</td>
</tr>
<tr>
<td>00F0</td>
<td>721</td>
<td>OUT MODE7: 8257 MODE PORT</td>
</tr>
<tr>
<td>00F0</td>
<td>722</td>
<td>RET</td>
</tr>
<tr>
<td>00F0</td>
<td>723</td>
<td></td>
</tr>
<tr>
<td>00F0</td>
<td>724</td>
<td></td>
</tr>
</tbody>
</table>

**Note:** The code snippet appears to be related to a microcontroller program with instructions for handling DMA channels for both RX and TX. The instructions show how to initialize, enable, and manage DMA transfers for receiving and transmitting data.
729 ; INTERRUPT PROCESSING SECTION
730 ;
731 ORG 0080H
732 ;
733 ; RECEIVER INTERRUPT - RST 7.5 (LOC 3CH)
734 ;
735 ;
736 RXI: 736 PUSH H 737 SAVE HL
738 POP PSW 739 SAVE PSW
740 PUSH B 741 SAVE BC
742 PUSH D 743 SAVE DE
744 MI 745 A.DORMA 746 DISROUT RX DMA
747 OUT MODEST 748 R8257 MODE PORT
749 RXI: 750 MI 751 A.18H 752 RST 7.5 RST 7.5 (LOC 3CH)
753 SIM 754 D, 0AH 755 AND IS RESULT COUNTER
756 2AH 757 HLDE LPADDR 758 GET LOAD POINTER
759 E5 760 RXI: 761 PUSH H 762 SAVE IT
763 E5 764 RXI: 765 PUSH H 766 SAVE IT AGAIN
767 45 768 MVI B,L 769 SAVE LSB
770 2AH 771 HLDE CNHDR 772 GET CONSOLE POINTER
773 84 774 RXI: 775 INR B 776 BUMP LOAD POINTER LSB
777 78 778 RXI: 779 MOV A,B 780 GET SET TO TEST
781 8D 782 POP L 783 LOAD-CONSOLE?
784 DE40H 785 JZ RXI: 786 YES, BUFFER FULL
787 15 788 DCR D 789 DECODE COUNTER
789 C2160C 790 JNZ RXI: 791 NOT DONE. TRY AGAIN
791 1685 792 RXI: 793 MVI D, 08H 794 RESET COUNTER
795 F1 796 POP H 797 RESTORE L0AD POINTER
798 89H 799 RXI: 79A IN STRAT73 79B READ STATUS
79C E8H 79D RXI: 79E ANI RXINT 79F TEST RX INT BIT
800 0330C 801 RXI: 802 JZ RX1: 803 DONE, GO FINISH UP
804 09H 805 RXI: 806 IN STRAT73 807 READ STATUS AGAIN
808 E8H 809 RXI: 80A RXIA 80B IS RESULT READY?
80C 2928C 80D RXI: 80E JNZ RX1: 80F NOT TEST AGAIN
810 89H 811 RXI: 812 IN RX73 813 YES, READ RESULT
814 77H 815 RXI: 816 MOV A,R 817 STORE IN BUFFER
818 2C 819 RXI: 81A INC BUFFER POINTER
81B 15 81C RXI: 81D DCR D 81E DEC COUNTER
81F C328C 820 RXI: 821 JMP RX1: 822 GET MORE RESULTS
823 9AH 824 RXI: 825 MOV A,D 826 GET SET TO TEST
826 87H 827 RXI: 828 ANA A 829 ALL RESULTS?
82A C450C 82B RXI: 82C JZ RX1: 82D YES, SO FINISH UP
82E 3680H 82F RXI: 830 MVI A,80H 831 NO LOAD & STIL DONE
832 8C 833 RXI: 834 INR L 835 BUMP POINTER
836 15 837 RXI: 838 DCR D 839 DEC COUNTER
839 C330C 840 RXI: 841 JMP RX1: 842 GO AGAIN
843 22182H 844 RXI: 845 SHLD LPADDR 846 UPDATE LOAD POINTER
847 3622H 848 RXI: 849 LDS A 850 PRINT 851 GET MODE INDICATOR
84B 2F66H 852 RXI: 853 CPI 854 NORMAL MODE?
855 C3350C 856 RXI: 857 JZ RX1: 858 YES, CLEAN UP BEFORE RETURN
79B ; POLL MODE SO CHECK CONTROL BYTE
79D ; IF CONTROL IS A POLL, SET UP SPECIAL TX COMMAND BUFFER
79F ; AND RETURN WITH POLL INDICATOR NOT 0
801 ;
802 ;
804 ; GET PREVIOUS LOAD ADDR POINTER
807 ; GET IC BYTE FROM BUFFER
611001-57
0C52 E61E 787 ANI 10H ; LOOK AT GOOD FRAME BITS
0C54 C2890C 788 INZ RXIS ; IF NOT 0, INTERRUPT WASN'T FROM A GOOD FRAME
0C57 2C 789 INR L ; BYPASS R0 AND RL IN BUFFER
0C59 2C 790 INR L
0C59 2C 791 INR L
0C59 2C 792 MOV D,M ; GET ADR BYTE AND SAVE IT IN D
0C59 2C 793 INR L
0C5C 7E 794 MOV A,M ; GET CNTL BYTE FROM BUFFER
0C5D FE93 795 CPI SNMP ; WAS IT SNMP-P?
0C5F CAS60C 796 JZ TL ; YES, GO SET RESPONSE
0C62 FE11 797 CPI RR8P ; WAS IT RR8(0)-P?
0C64 C2890C 798 INZ RXIS ; YES, GO SET RESPONSE, OTHERWISE RETURN
0C67 1E11 799 MV1 E.RR8F ; RR8(0)-P SO SET RESPONSE TO RR8(0)-F
0C69 C360C 800 JMP TXRET ; GO FINISH LOADING SPECIAL BUFFER
0C6A 1E73 801 TL ; MV1 E.NSAF ; SNMP-P SO SET RESPONSE TO NSA-F
0C6E 213020 802 TLRET; LXI H,CNS6F1 ; SPECIAL BUFFER ADDR
0C71 3E28 806 MV1 M.0CBH ; LOAD TX FRAME COMMAND
0C73 23 808 INX H ; INC POINTER
0C74 3E80 809 MV1 M.0BH ; LB=0
0C76 23 810 INX H ; INC POINTER
0C77 3E80 811 MV1 M.0BH ; LB=0
0C79 23 812 INX H ; INC POINTER
0C7A 72 813 MOV M.D ; LOAD RXD ADDR BYTE
0C7C 72 814 INX H ; INC POINTER
0C7E 72 815 MOV M.E ; LOAD RESPONSE CNTL BYTE
0C7F 3E81 816 MV1 A.RH ; SET POLL INDICATOR NOT 0
0C80 321529 817 STR POLIN ; LOAD POLL INDICATOR
0C82 C360C 818 JMP RXIS ; RETURN
0C83 E1 820 RX16; POP H ; CLEAN UP STACK IF NORMAL MODE
0C86 C360C 821 JMP RXIS ; RETURN
0C89 COLAB8 823 RX15; CALL RXMAP ; RESET DMR CHANNEL
0C8C DL 824 POP D ; RESTORE REGISTERS
0C8D CL 825 POP B
0C8E FL 826 POP PSW
0C8F EL 827 POP H
0C90 FB 828 EI ; ENABLE INTERRUPTS
0C91 C9 829 RET ; RETURN
0C92 830
0C92 831
0C92 832 ; MESSAGE TYPER - ASSUMES MESSAGE STARTS AT HL
0C93 C5 833
0C93 7E 834
0C94 23 835 TMS5; PUSH B ;SAVE BC
0C95 2C 836 TMS52; MOV A,M ; GET ASCII CHAR
0C96 23 837 INX H ; INC POINTER
0C97 FEFF 838 CPI OFFH ; STOP?
0C99 CAR10C 839 JZ TMS51 ; YES, GET SET FOR EXIT
0C9A 4F 840 MOV C.A ; SET UP FOR DISPLAY
0C9B 0DF805 841 CALL ECHO ; DISPLAY CHAR
0C9C C3930C 842 JMP TMS52 ; GET NEXT CHR
0C9D CL 843 TMS51; POP B ; RESTORE BC
0C9E C9 844 RET ; RETURN
0C9F 845
0C9F 846
0C9F 847 ; SIGNAL MESSAGE
0C9F 848

611001-58
008B 00 049 SIGNORE: DB CR,'8273 MONITOR V1.1':CR,0FFH
008C 00 30327733
008D 00 20404F4F
008E 00 00544F52
008F 00 28285631
0090 00 2E31
0091 00 0077 FF
0092 0850 ;
0093 0851 ;
0094 0852 ;
0095 0853 ;RECEIVER INTERRUPT MESSAGES
0096 0854 ;
0097 0098 00 0086 RXMSG: DB CR,'RX INT - ':0FFH
0099 009A 00 007B 2582049
009A 00 009B 00 4E54202D
009B 00 009C 00 20
009C 00 009D 00 FF
009D 009E 00 0077 ;TRANSMITTER INTERRUPT MESSAGES
009F 00 00A0 ;
00A0 00 00A1 00 0088 TXMSG: DB CR,'TX INT - ':0FFH
00A1 00A2 00 00A3 00 4E54202D
00A3 00 00A4 00 20
00A4 00 00A5 00 FF
00A5 00A6 00 0077 ;TRANSMITTER INTERRUPT ROUTINE
00A7 00 00A8 ;
00A8 00 00A9 00 0065 TXI: PUSH H ;SAVE HL
00A9 00 00AA 00 0066 PUSH PSW ;SAVE PSW
00AA 00 00AB 00 0067 PUSH B ;SAVE BC
00AB 00 00AC 00 0068 PUSH D ;SAVE DE
00AC 00 00AD 00 0069 MVI A,DT1MR ;DISABLE TX DMA
00AD 00 00AE 00 006A DOUT MODE37 ;8257 MODE PORT
00AE 00 00AF 00 006B MVI D,0AH ;SET COUNTER
00AF 00 00B0 00 006C LHLD LOADR ;GET LOAD POINTER
00B0 00 00B1 00 006D PUSH H ;SAVE IT
00B1 00 00B2 00 006E PUSH D ;SAVE DE
00B2 00 00B3 00 006F MVI A,8B8H ;DISABLE RX DMA
00B3 00 00B4 00 0070 DOUT MODE37 ;8257 MODE PORT
00B4 00 00B5 00 0071 MVI D,0AH ;SET COUNTER
00B5 00 00B6 00 0072 LHLD LOADR ;GET LOAD POINTER
00B6 00 00B7 00 0073 PUSH D ;SAVE DE
00B7 00 00B8 00 0074 MVI B,L ;SAVE LSB IN B
00B8 00 00B9 00 0075 LHLD LOADR ;GET CONSOLE POINTER
00B9 00 00BA 00 0076 TXII: INR B ;INC POINTER
00BA 00 00BB 00 0077 MOV A,B ;GET SET TO TEST
00BB 00 00BC 00 0078 CMP L ;LOAD=CONSOLE?
00BC 00 00BD 00 0079 DOR D ;YES: BUFFER FULL
00BD 00 00BE 00 007A JZ BUFILL ;NO: TEST NEXT LOCATION.
00BE 00 00BF 00 007B DCR D ;NO: TEST NEXT LOCATION.
00BF 00 00C0 00 007C 2E00C Inz TXII ;TRY AGAIN
00C0 00 00C1 00 007D POP H ;RESTORE LOAD POINTER
00C1 00 00C2 00 007E 00 00C3 00 D7 IN TXIR73 ;READ RESULT
00C3 00 00C4 00 007F 00 00C5 00 084 MOV M,A ;STORE IN BUFFER
00C5 00 00C6 00 0085 INR L ;INR POINTER
00C6 00 00C7 00 0086 MOV M,00H ;EXTRA RESULT SPOTS 0
00C7 00 00C8 00 0087 INR L
00C8 00 00C9 00 0088 MOV M,00H
00C9 00 00CA 00 0089 INR L
00CA 00 00CB 00 008A MOV M,00H
00CB 00 00CC 00 008B MOV M,00H
00CC 00 00CD 00 008C INR L
611001-59

2-346
AP-36

PUBLIC SYMBOLS

EXTERNAL SYMBOLS

USER SYMBOLS

EXTERNAL SYMBOLS

ASSEMBLY COMPLETE. NO ERRORS

611001-60
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-ASYNCRONOUS 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, 1 1/2, 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). Modern 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.

Table 2. 8274 Addressing

<table>
<thead>
<tr>
<th>CS A1 A0</th>
<th>Read Operation</th>
<th>Write Operation</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 0 0</td>
<td>Ch. A Data Read</td>
<td>Ch. A Data Write</td>
</tr>
<tr>
<td>0 1 0</td>
<td>Ch. A Status Read</td>
<td>Ch. A Command/Parameter</td>
</tr>
<tr>
<td>0 0 1</td>
<td>Ch. B Data Read</td>
<td>Ch. B Data Write</td>
</tr>
<tr>
<td>0 1 1</td>
<td>Ch. B Status Read</td>
<td>Ch. B Command/Parameter</td>
</tr>
<tr>
<td>1 X X</td>
<td>High Impedance</td>
<td>High Impedance</td>
</tr>
</tbody>
</table>

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.
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 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 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 0 c</td>
</tr>
<tr>
<td>2</td>
<td>1 1 1 0 0 0 0 c c c</td>
</tr>
<tr>
<td>3</td>
<td>1 1 0 0 0 0 c c c c</td>
</tr>
<tr>
<td>4</td>
<td>1 0 0 0 c c c c c</td>
</tr>
<tr>
<td>5</td>
<td>0 0 0 c c c c c</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.

```pascal
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
1 stop bit        and receiver
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. MPSCPOLLRCVCHARACTER—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 (RECEIVEERROR) is called. If no receive errors have been detected, the character is input from the 8274 data port and returned to the calling program. MPSCPOLLRCVCHARACTER requires three parameters—the address of the 8274 channel data port (data$port), the address of the 8274 channel command port (cmd$port), and the address of a byte variable in which to store the received character (character$ptr).

2. MPSCPOLLTRANSCARACTHER—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. MPSCPOLLTRANSLCHARACTER requires three parameters—the address of the 8274 channel data port (data$port), the address of the 8274 channel command port (cmd$port), and the character of data that is to be transmitted (character).

3. RECEIVEERROR—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. RECEIVEERROR requires two parameters—the address of the affected 8274 command port (cmd$port) and the error status (status) from 8274 register RR1.

Interrupt-Driven Operation

In an interrupt-driven environment, all receiver operations are reported to the system processor by means of interrupts. Once a character has been received and assembled, the MPSC interrupts the system processor. The system processor must then read the character from the MPSC data buffer and clear the current interrupt. During transmission, the system processor starts 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, SYNDT, 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) 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>8086 Interrupt Type</th>
<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>8085 Interrupt Location</th>
<th>Interrupt Condition</th>
</tr>
</thead>
<tbody>
<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>0</td>
<td>V7</td>
<td>V6</td>
<td>V5</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>V1</td>
<td>V0</td>
<td>Channel B Transmitter Buffer Empty</td>
<td></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>1</td>
<td>1</td>
<td>V7</td>
<td>V6</td>
<td>V5</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>V1</td>
<td>V0</td>
<td>Channel B External/Status Change</td>
<td></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>0</td>
<td>V7</td>
<td>V6</td>
<td>V5</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>V1</td>
<td>V0</td>
<td>Channel B Receiver Character Available</td>
<td></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>1</td>
<td>V7</td>
<td>V6</td>
<td>V5</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>V1</td>
<td>V0</td>
<td>Channel B Receive Error</td>
<td></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>0</td>
<td>V7</td>
<td>V6</td>
<td>V5</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>V1</td>
<td>V0</td>
<td>Channel A Transmitter Buffer Empty</td>
<td></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>1</td>
<td>V7</td>
<td>V6</td>
<td>V5</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>V1</td>
<td>V0</td>
<td>Channel A External/Status Change</td>
<td></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>0</td>
<td>V7</td>
<td>V6</td>
<td>V5</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>V1</td>
<td>V0</td>
<td>Channel A Receiver Character Available</td>
<td></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>1</td>
<td>V7</td>
<td>V6</td>
<td>V5</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>V1</td>
<td>V0</td>
<td>Channel A Receive Error</td>
<td></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$CHARACTER$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$CHARACTER$INT).

3. **MPSC$TRANSMIT$CHARACTER$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$RECEIVE$CHARACTER$INT**—This procedure is executed when a character has been assembled by the MPSC and the MPSC has issued a character-available interrupt. If no input buffer has been set up by RECEIVE$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. **RECEIVE$ERROR$INT**—This procedure is executed when a receive error is detected. First, the error conditions are read from RRI 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$STATUS$CHANGE$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):

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 Note used in asynchronous modes.
Command 2 Reset External/Status Interrupts—resets the latched status bits of RR0 and reenables 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 Pending—if the Transmit Interrupt mode is selected, the MPSC automatically interrupts data when the transmit buffer becomes empty. When there are no more characters to be sent, issuing this command prevents further transmitter interrupts 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.

Write Register 1 (WR1):

D2 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.

D1 Transmitter Interrupt/DMA Enable—allows the MPSC to interrupt or request a DMA transfer when the transmitter buffer becomes empty.

D2 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.
Write Register 1 (WR1):

### 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, C5 = 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.

### D6
Must be Zero.

### D7
Wait Enable—enables the wait function.

Write Register 2 (WR2): Channel A

### MSB

- **D7**: 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).

### LSB

- **D0**: Note: If EXT Interrupt Enable (WR1; D0) is set.
- **0 0**: BOTH INTERRUPT
- **0 1**: A DMA, B INT
- **1 0**: BOTH DMA
- **1 1**: ILLEGAL

- **RxA > RxB > TxA > TxB > ExTA > ExTB**:
- **0**: NON VECTORED INTERRUPT
- **1**: VECTORED INTERRUPT

**NOTE**: External Status Interrupt—only if EXT Interrupt Enable (WR1; D0) is set.

**210311-7**
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 = SYNDET.

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.
Write Register 4 (WR4):

- **D0** 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.

- **D1** 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.

- **D3,D2** Stop Bits.
  - 0 0 Selects synchronous modes.
  - 0 1 Async mode, 1 stop bit/character.
  - 1 0 Async mode, 1½ stop bits/character.
  - 1 1 Async mode, 2 stop bits/character.

- **D7,D6** 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.
  - 0 0 Clock rate = Data rate \( \times 1 \).
  - 0 1 Clock rate = Data rate \( \times 16 \).
  - 1 0 Clock rate = Data rate \( \times 32 \).
  - 1 1 Clock rate = Data rate \( \times 64 \).

Write Register 5 (WR5):

- **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). When the RTS bit is reset in asynchronous mode, the signal does not go inactive until the transmitter is empty.

- **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 5 or less bits/character.
  - 0 1 Transmit 7 bits/character.
  - 1 0 Transmit 6 bits/character.
  - 1 1 Transmit 8 bits/character.

Bits to be sent must be right justified, least-significant bit first, e.g.:

- D7 D6 D5 D4 D3 D2 D1 D0
  - 0 0 B5 B4 B3 B2 B1 B0
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 EOI 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.

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.

Read Register 2 (RR2):

res
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$enable byte,
tx$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$POLL$SRC$V$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; end;

/* check for errors in received character */
output(cmd$port)=1; /* point to RRI */
if (status:=input(cmd$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; /* error return - no character avail */
end;
else do;
  character=input(data$port);
  return OFFH; /* good return - character avail */
end;
end MPSC$POLL$SRC$V$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; end;

/* 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)=30H; /* 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 for
      interrupt service */
    tx$buffer$ptr=buf$ptr;
    transmit=length=buf$length;
    transmit$status=not$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$complete do;
      if transmit$status = complete
        then return false;
      else return true;
    end;
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;
      if receive$status = complete
        then return true;
      else return false;
    end;
end RECEIVE$BUFFER;
## APPENDIX C
**INTERRUPT-DRIVEN TRANSMIT/RECEIVE SOFTWARE**

```c
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',
line$feed literally '0AH',
not$complete literally '0',
complete literally 'OFFH',
overrun literally '1',
channel$reset literally '18H',
error$reset literally '30H',
reset$ext$status literally '10H';
```

---

2-370
MPSC$INT$INIT: procedure (clock$rate, stop$bits, parity$type, parity$enable, 
rx$char$length, tx$enable, auto$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$en byte,  /* 1-bit Rx interrupt enable */
stat$aff$vector byte,  /* 2-bit Rx interrupt enable/mode */
config byte,  /* 2-bit system config - int/DMA */
priority byte,  /* 1-bit priority flag */
vector$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$port)=l4H;  /* point to WR4 */
/* set clock rate, stop bits, and parity information */
output(b$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$port)=l5H; /* point to WR5 */
/* set up transmitter parameters */
output(b$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;
MPSC$RECEIVE$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$index+1;
            /* update receive buffer index */
        end;
        /* check for line feed to end line */
        if character = line$feed
            then do; receive$length=receive$index; receive$status=complete; end;
            end;
    end;

end MPSC$RECEIVE$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;

end MPSC$TRANSMIT$CHARACTER$INT;

RECEIVE$ERROR$INT: procedure interrupt 23H;

    declare temp byte; /* temporary character storage */
    output(cmd$port)=1; /* point to RRI */
    temp=input(data$port); /* discard character */
    output(cmd$port)=error$reset; /* send error reset */

    /* *** other application dependent
       error processing should be placed here. *** */

end RECEIVE$ERROR$INT;

EXTERNAL$STATUS$CHANGE$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$STATUS$CHANGE$INT;
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, TX_EMPTY_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

**ISIS-II IICS-11 MICRO ASSEMBLER V2.1 ASSEMBLY OF MODULE ASYNCH**

**OBJECT MODULE PLACED IN 'FL ASYNCH OBJ'**

**ASSEMBLER INVOKED BY: ASM66 'FL ASYNCH SRC'**

<table>
<thead>
<tr>
<th>LOC</th>
<th>GET</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 VECTORED INTERRUPT-DRIVEN DATA TRANSFERS</td>
<td>*</td>
</tr>
<tr>
<td>9</td>
<td>;*</td>
<td>THE Initialization Routine ALSO LOADS THE 8266'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>SDK-66 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 PROCEED 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 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 UP 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></td>
<td>*</td>
</tr>
<tr>
<td>30</td>
<td>;****</td>
<td></td>
<td>*</td>
</tr>
</tbody>
</table>

---

2-377
PUBLIC DECLARATIONS FOR COMMAND ROUTINES

PUBLIC INITI .OPTA : initialization routine
PUBLIC TX.COMMAND.CH: .tx buffer command channel B
PUBLIC SX.COMMAND.CH: .tx buffer command channel A
PUBLIC RX.COMMAND.CH: .rx buffer command channel B
PUBLIC RX.COMMAND.CH: .rx buffer command channel A

PUBLIC DECLARATIONS FOR STATUS VARIABLES

PUBLIC RXREADY.CH: .rx ready flag char
PUBLIC RXREADY.CH: .rx ready flag char
PUBLIC TXEMPTY.CH: .tx empty flag char
PUBLIC TXEMPTY.CH: .tx empty flag char
PUBLIC RX.COUNT.CH: .rx buffer counter char
PUBLIC RX.COUNT.CH: .rx buffer counter char
PUBLIC RX.ERR.MSG.CH: .error flag char
PUBLIC RX.ERR.MSG.CH: .error flag char
PUBLIC RX.STATUS.MSG.CH: .status flag char
PUBLIC RX.STATUS.MSG.CH: .status flag char

PUBLIC DECLARATIONS FOR VARIABLES PASSED TO THE TRANSMIT AND RECEIVE COMMANDS.

PUBLIC TX.POINTER.CH: .tx buffer pointer for CHB
PUBLIC TX.LENGTH.CH: .tx length of buffer for CHB
PUBLIC TX.POINTER.CH: .tx buffer pointer for CHA
PUBLIC RX.POINTER.CH: .rx buffer pointer for CHB
PUBLIC RX.POINTER.CH: .rx buffer pointer for CHB

I/O 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 A 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 EQUATES

ASCII CR: .equ 0X: .asci of character code
INT .TABLE .BASE: .equ 2000H .int vector base address
CODE .START: .equ 5000H .start location for code
REJECT

DATA SEGMENT

DATA SEGMENT
MCS-65 MACRO ASSEMBLER

LOC OBJ LINE SOURCE
91 .VECTOR INTERRUPT TABLE - Assume initial 0374 INTERRUPT
92 .VECTOR IS NUMBER 88, 89, 90, 91 FOR EACH VECTOR, THE TABLE
93 .CONTAINS START LOCATION AND CODE SEGMENT REGISTER VALUE
94 .THE TABLE IS LOADED FROM ROM
95
96 ORG INT_TABLEPAGE
97
98 .X VECTOR,CHB DW 0 ; TI INTERRUPT VECTOR FOR CHB
99 .X CS,CHB DW 0
100
101 .X STATUS,VECTOR,CHB DW 0 ; STATUS INTERRUPT VECTOR FOR CHB
102 .X CS,CHB DW 0
103
104 .X RX,VECTOR,CHB DW 0 ; RX INTERRUPT VECTOR FOR CHB
105 .X CS,CHB DW 0
106
107 .X ERR,VECTOR,CHB DW 0 ; ERROR INTERRUPT VECTOR FOR CHB
108 .X CS,CHB DW 0
109
110 .X TX,VECTOR,CHA DW 0 ; TX INTERRUPT VECTOR FOR CHB
111 .X CS,CHA DW 0
112
113 .X STS,VECTOR,CHA DW 0 ; STATUS INTERRUPT VECTOR FOR CHA
114 .X CS,CHA DW 0
115
116 .X RX,VECTOR,CHA DW 0 ; RX INTERRUPT VECTOR FOR CHA
117 .X CS,CHA DW 0
118
119 .X ERR,VECTOR,CHA DW 0 ; ERROR INTERRUPT VECTOR FOR CHA
120 .X CS,CHA DW 0
121
122 .X TX BUFFER POINTER FOR CHB
123 .X LENGTH FOR CHB
124 .X RX BUFFER POINTER FOR CHB
125
126 .X TX BUFFER POINTER FOR CHA
127 .X LENGTH FOR CHA
128 .X RX BUFFER POINTER FOR CHA
129
130 .X TX EMPTY,CHA DB 0 ; TX DONE FLAG
131 .X RX READY,CHA DB 0 ; READY FLAG (1 IF CR,CHA RECEIVED; ELSE 0)
132 .X STATUS,MSG,CHA DB 0 ; STATUS MESSAGE
133 .X RX,READY,CHA DB 0 ; RX READY MESSAGE
134
135 .X TX BUFFER POINTER FOR CHB
136 .X LENGTH FOR CHB
137 .X RX BUFFER POINTER FOR CHB
138
139 .X TX BUFFER POINTER FOR CHA
140 .X LENGTH FOR CHA
141 .X RX BUFFER POINTER FOR CHA
142
143 .X READY FLAG (1 IF CR,CHA RECEIVED; ELSE 0)
144 .X STATUS MESSAGE
145 .X RX READY MESSAGE
146
147 DATA ENDS
148 +1 REJECT

210311-26
LOC OBJ   LINE SOURCE

149
150  
151  
152  
153  
154  
155  
156  
157  
158  
159  
160  
161  
162  
163  
164  
165  
166  
167  
168  
169  
170  
171  
172  
173  
174  
175  
176  
177  
178  
179  
180  
181  
182  
183  
184  
185  
186  
187  
188  
189  

/channel b parameters

/channel a parameters

/source
<table>
<thead>
<tr>
<th>LOC</th>
<th>GBE</th>
<th>LINE</th>
<th>SOURCE</th>
</tr>
</thead>
<tbody>
<tr>
<td>0545 EE</td>
<td>251</td>
<td>OUT</td>
<td>D1 AL</td>
</tr>
<tr>
<td>0546 47</td>
<td>252</td>
<td>INC</td>
<td>D1</td>
</tr>
<tr>
<td>0547 EBF6</td>
<td>253</td>
<td>JMP</td>
<td>SETUP</td>
</tr>
<tr>
<td>0549 C3</td>
<td>254</td>
<td>DONE</td>
<td>RET</td>
</tr>
<tr>
<td>255 REJECT</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>256 TX CHANNEL B COMMAND ROUTINE - ROUTINE IS CALLED TO</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>257 TRANSMIT A BUFFER - THE BUFFER STARTING ADDRESS.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>258 TX_POINTER,CHR. AND THE BUFFER LENGTH. TXLENGTH,CHR.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>259 MUST BE INITIALIZED BY THE CALLING PROGRAM</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>260 BOTH ITEMS ARE WORD VARIABLES</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>261</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>059A 58</td>
<td>260 TX_COMMAND,CHR:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>059B S7</td>
<td>260 PUSH</td>
<td>RX</td>
<td>.SAVE REGISTERS</td>
</tr>
<tr>
<td>059C 52</td>
<td>268 PUSH</td>
<td>D1</td>
<td></td>
</tr>
<tr>
<td>059E 20000000</td>
<td>269 MOV</td>
<td>TXEMPTY,CHR. 0 .CLEAR EMPTY FLAG</td>
<td></td>
</tr>
<tr>
<td>059E B0000000</td>
<td>270 MOV</td>
<td>D1, DARTPORT,CHR .SETUP PORT POINTER</td>
<td></td>
</tr>
<tr>
<td>059F BE2E2002</td>
<td>271 MOV</td>
<td>D1, TX_POINTER,CHR. GET TX BUFFER POINTER CHR</td>
<td></td>
</tr>
<tr>
<td>05A0 0055</td>
<td>272 MOV</td>
<td>AL, [DI] .GET FIRST CHARACTER TO TX</td>
<td></td>
</tr>
<tr>
<td>05A1 EE</td>
<td>273 MOV</td>
<td>D1, AL OUTPUT IT TO 0224 TO GET IT STARTED</td>
<td></td>
</tr>
<tr>
<td>05A2 5A</td>
<td>274 POP</td>
<td>D1</td>
<td></td>
</tr>
<tr>
<td>05A3 5F</td>
<td>275 POP</td>
<td>RX</td>
<td></td>
</tr>
<tr>
<td>05A5 C3</td>
<td>276 RET</td>
<td>RX</td>
<td>.RETURN</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>LOC</th>
<th>CEB</th>
<th>LINE</th>
<th>SOURCE</th>
</tr>
</thead>
<tbody>
<tr>
<td>05C0 58</td>
<td>285 TX_COMMAND,CHR:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>05C1 S7</td>
<td>293 PUSH</td>
<td>RX</td>
<td>.SAVE REGISTERS</td>
</tr>
<tr>
<td>05C2 52</td>
<td>294 PUSH</td>
<td>D1</td>
<td></td>
</tr>
<tr>
<td>05C3 C8040000</td>
<td>295 MOV</td>
<td>TXEMPTY,CHR. 0 .CLEAR EMPTY FLAG</td>
<td></td>
</tr>
<tr>
<td>05C5 00000000</td>
<td>296 MOV</td>
<td>D1, DARTPORT,CHR .SETUP PORT POINTER</td>
<td></td>
</tr>
<tr>
<td>05C6 BE2E2002</td>
<td>297 MOV</td>
<td>D1, TX_POINTER,CHR. GET TX BUFFER POINTER CHR</td>
<td></td>
</tr>
<tr>
<td>05C7 0055</td>
<td>298 MOV</td>
<td>AL, [DI] .GET FIRST CHARACTER TO TX</td>
<td></td>
</tr>
<tr>
<td>05C8 EE</td>
<td>299 MOV</td>
<td>D1, AL OUTPUT IT TO 0224 TO GET IT STARTED</td>
<td></td>
</tr>
<tr>
<td>05C9 5A</td>
<td>300 POP</td>
<td>D1</td>
<td></td>
</tr>
<tr>
<td>05CA 5F</td>
<td>301 POP</td>
<td>RX</td>
<td></td>
</tr>
<tr>
<td>05CB 58</td>
<td>302 POP</td>
<td>RX</td>
<td></td>
</tr>
<tr>
<td>05CC C3</td>
<td>303 RET</td>
<td>RX</td>
<td>.RETURN</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>LOC</th>
<th>CEB</th>
<th>LINE</th>
<th>SOURCE</th>
</tr>
</thead>
<tbody>
<tr>
<td>05E0 58</td>
<td>310 RX COMMAND FOR CHANNEL B - THE CALLING ROUTINE MUST</td>
<td></td>
<td></td>
</tr>
<tr>
<td>05E1 S7</td>
<td>311 INITIALIZE TL_POINTER,CHR TO POINT AT THE RECEIVE</td>
<td></td>
<td></td>
</tr>
<tr>
<td>05E2 52</td>
<td>312 BUFFER BEFORE CALLING THIS ROUTINE</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

210311-20
MK$-86 MACRO ASSEMBLER ASINANCE

LOC OBJ
LINE SOURCE

311 .
312 .**************************************************
313
314 RL.COMMAND.CHR
315 316 PUSH AX . SAVE REGISTERS
317 318 PUSH DX
319 320 MOV RL.RDY.CHR, 0 . CLEAR RX READY FLAG
321 322 MOV RL.COUNT.CHR, 0 . CLEAR RX COUNTER
323 324 MOV RL.COMMAND.PORT.CHR . POINT AT COMMAND PORT
325 326 MOV AX, 3 . SET UP FOR MPS
327 328 OUT DX, AL
329 330 MOV AL, 0EH . WR3 = 8 Bits/CH. ENABLE RX
331 332 POP AX
333 334 RET . RETURN

335
336 ***********************************************
337 RL.COMMAND.CHR
338 339 PUSH AX . SAVE REGISTERS
340 341 PUSH DX
342 343 MOV RL.RDY.CHR, 0 . CLEAR RX READY FLAG
344 345 MOV RL.COUNT.CHR, 0 . CLEAR RX COUNTER
346 347 MOV RL.COMMAND.PORT.CHR . POINT AT COMMAND PORT
348 349 MOV AX, 3 . SET UP FOR MPS
350 351 OUT DX, AL
352 353 MOV AL, 0EH . WR3 = 8 Bits/CH. ENABLE RX
354 355 POP AX
356 357 RET . RETURN

358 + 1 REJECT

360 **********************************************
361 CHANNEL B TRANSMIT DATA SERVICE ROUTINE
362
363 XTIME: PUSH DX . SAVE REGISTERS
364 365 PUSH DI
366 367 PUSH RX
368 369 EOI
370 371 CALL EOI . SEND EOI COMMAND TO 8274
372 373 INC RL.POINTER.CHR . POINT TO NEXT CHARACTER
374 375 DEC RL.LENGTH.CHR . DEC LENGTH COUNTER
376 377 JE XR1 . TEST IF DONE
378 379 MOV AX, EXP.CONTROL.CHR . NOT DONE - GET NEXT CHARACTER
380 381 MOV AX, 10H . PUT CHARACTER IN AL
382 383 OUT DX, AL . OUTPUT IT TO 8274
LOC OBJ LINE SOURCE
0622 50 371 POP AX :RESTORE REGISTERS
0623 5F 372 POP DI
0624 5A 373 POP DX
0625 0F 374 IRET :RETURN TO FOREGROUND
0626 BA00 375 XIB: MOV DX, COMMAND_PORT.CHB : ALL CHARACTERS HAVE BEEN SENT
0628 BB00 376 MOV AL, 00H :RESET TRANSMITTER INTERRUPT PENDING
0628 EE 377 OUT DX, AL
062C C9000000 378 MOV DX:EMPTY.CHB: 1: DONE - SO SET TX EMPTY FLAG CHB
0631 50 379 POP AX :RESTORE REGISTERS
0632 5F 380 POP DI
0633 5A 381 POP DX
0634 0F 382 IRET :RETURN TO FOREGROUND
0635 00 383 .CHANNEL B STATUS CHANGE SERVICE ROUTINE
0635 50 384 DBIB: PUSH DX :SAVE REGISTERS
0636 57 385 PUSH DI
0637 58 386 PUSH RX
0638 E9000000 387 CALL EDI :SEND EDI COMMAND TO 8274
063B BA00 388 MOV DX, COMMAND_PORT.CHB
063E EC 389 IN AL, DX: READ RXB
0640 F202 390 MOV STATUS.MSG.CHB, AL :PUT RXB IN STATUS MESSAGE
0642 BB00 391 MOV AL, 00H :SEND RESET STATUS INT COMMAND TO 8274
0644 EE 392 OUT DX, AL
0645 50 393 POP AX :RESTORE REGISTERS
0646 5F 394 POP DI
0647 5A 395 POP DX
0648 0F 396 IRET
0649 50 397 MOV AX, 00H
064A 57 398 MOV DI, 00H
064B 58 399 MOV DX, 00H
064C 0C100 400 MOV EDI: SEND EDI COMMAND TO 8274
064F B202 401 MOV DX:COMMAND_PORT.CHB :SET RX CHB BUFFER POINTER
0655 BA00 402 MOV DX, DATA_PORT.CHB
0656 EC 403 IN AL, DX: READ CHARACTER
0658 BB00 404 MOV EDI, AL :STORE IN BUFFER
0659 FF020202 405 INC RL_POINTER.CHB :BUMP THE BUFFER POINTER
065C FF020202 406 INC RL_COUNT.CHB :BUMP THE COUNTER
065E 3000 407 CMP AL, 00H :TEST IF LAST CHARACTER TO BE RECEIVED
0660 7300 408 XOR AL
0661 C9000000 409 MOV RL:READY.CHB, 1 :VES. SET READY FLAG
0664 BA00 410 MOV DX:COMMAND_PORT.CHB :POINT AT COMMAND PORT
0666 B003 411 MOV AL, 03H :POINT AT MSR
0668 EE 412 OUT DX, AL
066B BB00 413 MOV AL, MOW :DISABLE RX
066E EE 414 OUT DX, AL
0670 50 415 RSB: POP AX :EITHER WAY: RESTORE REGISTERS
0670 5F 416 POP DI
0671 5A 417 POP DX
0672 0F 418 IRET :RETURN TO FOREGROUND
0673 50 419 MOV AX, 00H
0674 5F 420 POP DI
0675 5A 421 POP DX
0676 0F 422 IRET
0677 50 423 MOV AX, 00H
0677 5F 424 MOV DX, 00H
0678 E9000000 425 CALL EDI :SEND EDI COMMAND TO 5274
067B BA00 426 MOV DX, COMMAND_PORT.CHB
067C 0F 427 IRET

210311-31
REFERENCES


2-386
Synchronous Communication with the 8274 Multiple Protocol Serial Controller

SIKANDAR NAQVI
APPLICATION ENGINEER
SYNCHRONOUS COMMUNICATION WITH THE 8274 MULTIPLE PROTOCOL SERIAL CONTROLLER

CONTENTS

INTRODUCTION ........................................ 2-389
SYNCHRONOUS PROTOCOL OVERVIEW ...................... 2-389
BLOCK DIAGRAM ........................................ 2-391
MPSC INTERRUPT STRUCTURE .............................. 2-394
MPSC MODES OF OPERATION ............................... 2-398
APPLICATION EXAMPLE .................................. 2-403
SPECIAL APPLICATIONS .................................. 2-408
PROGRAMMING HINTS ..................................... 2-410
APPENDIX A: APPLICATION EXAMPLE SOFTWARE LISTINGS ........ 2-411
APPENDIX B: MPSC READ/WRITE REGISTER DESCRIPTIONS .......... 2-422
REFERENCES ............................................. 2-425

2-388
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 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 acknowledgement, 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>*Extendable to 2 or More Bytes.</td>
<td>**Extendable to 2 Bytes.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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 Mono-sync. 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.

---

**Figure 2. Bisync Message Format**

```
| SYNC | SYNC | SOH | HEADER | STX | ETX | ETB | CRC1 | CRC2 |
```

**Figure 3. Bisync Transparent Format**

```
| DLE | STX | TRANSPARENT TRANSMISSION | DLE | ETX | BCC |
```

Enter transparent mode

return to normal mode

2-390
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: TxDQA, TxDQB, RxDQA, RxDQB. Note that TxDQB and RxDQB become I½P and I½P respectively in non-DMA mode. I½P is the Interrupt Priority Input and I½P 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 TxDRA and RxDRA 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 pin may be used for hardware reset while the clock is required to click the internal logic on the MPSC.

Figure 4. 8274 Block Diagram
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 RTS_B and SYNDET_B.

At a given time, this pin is either RTS_B or SYNDET_B. 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
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)

**Figure 7. MPSC Interrupt Structure**
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).

<table>
<thead>
<tr>
<th>PRIORITY</th>
<th>WR2A:D2</th>
<th>Highest</th>
<th>Lowest</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>RXA</td>
<td>TXA</td>
<td>RXB</td>
</tr>
<tr>
<td>1</td>
<td>RXA</td>
<td>RXB</td>
<td>TXA</td>
</tr>
</tbody>
</table>

*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

(j) 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 RR1.

(iii) Framing error: In asynchronous mode, a framing error will generate a special receive interrupt and sets bit D6 in RR1. 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 RR1.

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 RR1. 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 (RR1: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 pin 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 be 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 RRO,D7. A Reset External/Status command will clear this interrupt. However, a termination of the Abort sequence will generate another interrupt and set RRO,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 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.

<table>
<thead>
<tr>
<th>WR2A (D1 D0)</th>
<th>Mode</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 0</td>
<td>CH A and CH B in Interrupt Mode</td>
</tr>
<tr>
<td>0 1</td>
<td>CH A in DMA and CH B in Interrupt Mode</td>
</tr>
<tr>
<td>1 0</td>
<td>CH A and CH B in DMA Mode</td>
</tr>
<tr>
<td>1 1</td>
<td>Illegal</td>
</tr>
</tbody>
</table>

Figure 11. MPSC Mode Selection for Channel A and Channel B
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 = 11) will ensure that at the end of the frame when the transmitter underruns, 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.

### Programmed Interrupts

The receiver may be programmed into any one of the four modes. See Figure 13 for details.

<table>
<thead>
<tr>
<th>WR1, CHA</th>
<th>Rx Interrupt Mode</th>
</tr>
</thead>
<tbody>
<tr>
<td>D4</td>
<td>D3</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></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**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 anyone of the four modes. See Figure 13 for details.

<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**

**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 I 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 are 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 RDY output from the MPSC can be used to extend the CPU read and write cycle by inserting WAIT states. RDY_A or RDY_B are in high impedance state when the corresponding channel is not selected. This makes it possible to connect RDY_A and RDY_B outputs in wired OR configuration. Caution must be exercised here in using the RDY 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 RDY_A is active, forcing the CPU into a wait state. If the CTS goes inactive during this period, the RDY_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>WR2A</th>
<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>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>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>0</td>
<td>8085-1</td>
<td>1100 1101</td>
<td>HI-Z</td>
<td>0000 0000</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>8085-2</td>
<td>HI-Z</td>
<td>V7 V6 V5 V4 V3 V2 V1 V0</td>
<td>HI-Z</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</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>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>0</td>
<td>8086</td>
<td>HI-Z</td>
<td>HI-Z</td>
<td>—</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>(8086)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<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>0</td>
<td></td>
<td>EXT/STAT Change</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td></td>
<td>RX CHAR Available</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td></td>
<td>Special Rx Condition</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>1</td>
<td>0</td>
<td>1</td>
<td></td>
<td>EXT/STAT Change</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td></td>
<td>RX CHAR Available</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td></td>
<td>Special Rx Condition</td>
</tr>
</tbody>
</table>

Rx Special Condition: Parity Error, Framing Error, Rx Over-run Error, EOF (SDLC).
EXT/STAT Change: Change in Modern 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

```c
DECLARE C BYTE;
/*/ TABLE TO INITIALIZE THE 8274 CHANNEL A AND B */
/*/ INITIALIZE CHANNEL A ONLY */
DECLARE TABLE_74_A(*) BYTE DATA
  (00H.1BH. /* 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.05H. /* RX DMA ENABLE */
  05H.05H. /* DTR, RTS, B TX BITS, TX ENABLE */
  07H.05H. /* SDLC CRC, TX CRC ENABLE */
  06H.05H. /* DEFAULT ADDRESS */
  03H.OD9H. /* B RX BITS, AUTO ENABLES, HUNT MODE */
  OFFH); /* END OF INITIALIZATION TABLE */
DECLARE TABLE_74_B(*) BYTE DATA
  (02H.00H. /* INTERRUPT VECTOR */
  01H.1CH. /* STATUS AFFECTS VECTOR */
  OFFH); /* END */
/*/ INITIALIZE THE 8274 */
C=0;
DO WHILE TABLE_74_B(C) <> OFFH;
  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;
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 8273A. 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 to 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.

![Figure 19. Interrupt Response Flow Diagram](image-url)
Figure 20. Typical Main Interrupt Routine

```c
/* MAIN INTERRUPT ROUTINE */
OUTPUT(COMMAND_B_74) = 2;
TEMP = INPUT((STATUS_B_74) AND 07H);
/* FOR THIS APPLICATION CH B INTERRUPTS ARE IGNORED */
DO CASE TEMP:
    CALL IGNORE_INT; /* V2VIVO = 000 */
    CALL CHB_RX_CHAR; /* V2VIVO = 010 */
    CALL IGNORE_INT; /* V2VIVO = 011 */
    CALL IGNORE_INT; /* V2VIVO = 100 */
    CALL CHA_EXTERNAL_CHANGE; /* V2VIVO = 101 */
    CALL CHA_RX_CHAR; /* V2VIVO = 110 */
    CALL CHA_RX_SPECIAL; /* V2VIVO = 111 */
END;
OUTPUT(COMMAND_A_74) = 38H; /* END OF INTERRUPT FOR 8274 */
RETURN;
END OF INTERRUPT 8274;
```

Figure 21. 8274 Typical Interrupt Handling Routines

```c
/* CHANNEL A EXTERNAL/STATUS CHANGE INTERRUPT HANDLER */
CH_A_EXTERNAL_CHANGE: PROCEDURE;
TEMP = INPUT((STATUS_A_74)); /* STATUS REG */
IF (TEMP AND END_OF_TX_MESSAGE) = END_OF_TX_MESSAGE THEN
    TXDONE_S = DONE;
ELSE DO;
    RXDONE_S = DONE;
    RESULTS_S = FAIL /* RX ERROR */
    RXDONE_S = DONE;
    OUTPUT(COMMAND_A_74) = 30H; /* ERROR RESET */
END;
END CHA_EXTERNAL_CHANGE;
/* CHANNEL A SPECIAL RECEIVE CONDITIONS INTERRUPT HANDLER */
CH_A_RX_SPECIAL: PROCEDURE;
OUTPUT(COMMAND_A_74) = 1;
TEMP = INPUT((STATUS_A_74));
IF (TEMP AND END_OF_FRAME) = END_OF_FRAME THEN DO:
    IF (TEMP AND 040H) = 040H THEN
        RESULTS_S = FAIL /* CRC ERROR */
        RXDONE_S = DONE;
        OUTPUT(COMMAND_A_74) = 30H; /* ERROR RESET */
    ELSE DO;
        IF (TEMP AND 20H) = 20H THEN DO:
            RESULTS_S = FAIL /* RX OVERRUN ERROR */
            RXDONE_S = DONE;
            OUTPUT(COMMAND_A_74) = 30H; /* ERROR RESET */
        END;
    END;
RETURN;
END CHA_RX_SPECIAL;
/* CHANNEL A RECEIVE CHARACTER AVAILABLE */
CH_A_RX_CHAR: PROCEDURE;
OUTPUT(SINGLE_MASK) = CHO_SEL; /* ENABLE RX DMA CHANNEL */
RETURN;
END CHA_RX_CHAR;
```
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 disabilding the 8259A.

```
CHA_SDLUC_TEST: PROCEDURE BYTE PUBLIC
CALL ENABLE_INTERRUPTS_3_S;
CALL INIT_8274_SDLUC_S;
ENABLE;
OUTPUT(COMMAND_A_74I) = 26H; /* RESET TX INT/DMA */
OUTPUT(COMMAND_B_74I) = 26H; /* BEFORE INITIALIZING 8237 */
CALL INIT_8237_S;
OUTPUT(DATA_A_74I) = 55H; /* LOAD FIRST CHARACTER FROM */
/* CPU */
/* TO ENSURE CRC TRANSMISSION. RESET TX UNDERRUN LATCH */
OUTPUT(COMMAND_A_74I) = 00H;
RXDONE_S=TIDONE_S=NOT_DONE; /* CLEAR ALL FLAGS */
RESULTS_S=PASS; /* FLAG SET FOR MONITOR */
DO WHILE TIDONE_S=NOT_DONE; /* DO UNTIL TERMINAL COUNT */
END;
DO WHILE (INPUT (STATUS_A_74I) AND O4H) = O4H;
/* 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_SDLUC_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 (CO 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/turn 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 RRI.

**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 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 the 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 INTR 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 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 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.
APPLICATION EXAMPLE: SOFTWARE LISTINGS

---

APENDIX A

PL/M-86 COMPILER ISBC B8/45 B274 CHANNEL A SDLT TEST

SERIES-III PL/M-86 V2.0 COMPIALTION OF MODULE INIT_8274_S
OBJECT MODULE PLACED IN: Fl: SINI74. OBJ
COMPILER INVOLED: PLM86.B6 : Fl: SINI74.PLM TITLE(ISBC B8/45 B274 CHANNEL A SDLT TEST) COMPACT N0INTVECTOR ROM

/**********************************************************************************************/
/* 1. INITIALIZE THE 8274 FOR SDLT MODE */
/* 2. EXTERNAL INTERRUPTS ENABLED */
/* 3. NO WAIT */
/* 4. PIN 10 = RTS */
/* 5. NON-VECTORED INTERRUPT-B09b MODE */
/* 6. CHANNEL A DMA, CH B INT */
/* 7. TX AND RX = 8 BITS/CHAR */
/* 8. ADDRESS SEARCH MODE */
/* 9. CD AND CTS AUTO ENABLE */
/* 10. X1 CLOCK */
/* 11. NON PARITY */
/* 12. SDLC/HSDL MODE */
/* 13. RTS AND DTR */
/* 14. CCITT ~ CRC */
/* 15. SDLC/HDLC MODE */
/* 16. TRANSMITTER AND RECEIVER ENABLED */
/* 17. 7EH = FLAG */
/**********************************************************************************************/

1 INIT_8274_S: DO;
*include (: Fl: PORTS. PLMI
DECLARE LI8 LITERALLY 'LITERALLY';
DECLARE CHO ADDR LIT '0B0H','
DECLARE CHO_CNT LIT '0B1H','
DECLARE CH1 ADDR LIT '0B2H','
DECLARE CH1_CNT LIT '0B3H','
DECLARE CH2 ADDR LIT '0B4H','
DECLARE CH2_CNT LIT '0B5H','
DECLARE CH3 ADDR LIT '0B6H','
DECLARE CH3_CNT LIT '0B7H','
DECLARE STATUS_37 LIT '0B8H','
DECLARE COMMAND_37 LIT '0B9H','
DECLARE REQUEST_REG_37 LIT '0BAH','
DECLARE SINGLE_MASK LIT '0BBH','
DECLARE MODE_REG_37 LIT '0BCH','

PL/M-86 COMPILER ISBC B8/45 B274 CHANNEL A SDLT TEST

DECLAR CTR_00 LIT '090H','
DECLAR CTR_01 LIT '091H','
DECLAR CTR_02 LIT '092H','

210403-15
/* B255 PORTS */
5 1 DECLARE PORTA_55 LIT '0A0H';
   PORTB_55 LIT '0A1H';
   PORTC_55 LIT '0A2H';
   CONTROL_55 LIT '0A3H';
/* B274 PORTS */
6 1 DECLARE DATA_A_74 LIT '0D0H';
   DATA_B_74 LIT '0D1H';
   STATUS_A_74 LIT '0D2H';
   COMMAND_A_74 LIT '0D3H';
   STATUS_B_74 LIT '0D3H';
/* B259A PORTS */
7 1 DECLARE STATUS_POLL_59 LIT '0E0H';
   ICW1_59 LIT '0E0H';
   OCW2_59 LIT '0E0H';
   OCW3_59 LIT '0E0H';
   OCW4_59 LIT '0E1H';
   IGW2_59 LIT '0E1H';
   IGW3_59 LIT '0E1H';
   IGW4_59 LIT '0E1H';
/* B274 REGISTER BIT ASSIGNMENTS */
/* READ REGISTER 0 */
8 1 DECLARE RX_AVAIL LIT '01H';
   INT_PENDING LIT '02H';
   TX_EMPTY LIT '04H';
   CARRIER_DETECT LIT '08H';
   SYNC_HUNT LIT '10H';
   CLEAR_TO_SEND LIT '20H';
PL/M-86 COMPILER 16bc 8/45 B274 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';
= /* 8237 BIT ASSIGNMENTS */

11 1 = DECLARE CH0_SEL LIT '00H';

11 2 = CH1_SEL LIT '01H';

11 3 = CH2_SEL LIT '02H';

11 4 = CH3_SEL LIT '03H';

11 5 = WRITE_XFER LIT '04H';

11 6 = READ_XFER LIT '05H';

11 7 = DEMAND_MODE LIT '06H';

11 8 = SINGLE_MODE LIT '07H';

11 9 = BLOCK_MODE LIT '08H';

11 10 = SET_MASK LIT '09H';

12 1 = DELAY_S: PROCEDURE PUBLIC;

13 2 = DECLARE D WORD;

14 3 = DO WHILE D<BOOH;

15 4 = D=D+1;

16 5 = END;

17 6 = END DELAY_S;

19 1 = INIT_8274_SDLC_S: PROCEDURE PUBLIC;

20 2 = DECLARE C BYTE;

PL/M-86 COMPILER ISBC 88/45 8274 CHANNEL A SDLC TEST

/* TABLE TO INITIALIZE THE 8274 CHANNEL A AND B */
/* FORMAT IS: WRITE REGISTER, REGISTER DATA */
/* INITIALIZE CHANNEL ONLY */

21 2 = DECLARE TABLE_74_A(1) BYTE DATA

197 1 = (00H.1BH. /* CHANNEL RESET */

198 00H.00H. /* RESET TX CRC */

199 02H.11H. /* PIN 10=RTS, A DMA, B INT */

200 04H.20H. /* SDLC/HDLCC MODE, NO PARITY */

201 07H.07EH. /* SDLC FLAG */

202 01H.05H. /* RX DMA ENABLE */

203 05H.0EBH. /* DTR, RTS, B TX BITS, TX ENABLE, TX CRC ENABLE */

204 06H.55H. /* DEFAULT ADDRESS */

205 03H.09H. /* B RX BITS, AUTO ENABLES, HUNT MODE */

206 OFFH; /* END OF INITIALIZATION TABLE */

22 2 = DECLARE TABLE_74_B(1) BYTE DATA

227 1 = (02H.00H. /* INTERRUPT VECTOR */

228 01H.1CH. /* STATUS AFFECTS VECTOR */

229 OFFH; /* END */

/* INITIALIZE THE 8234 */

23 2 = OUTPUT(CONTR0L0_54) = 36H;

24 2 = OUTPUT(CTR_00) = LOW(20); /* BAUD RATE = 400K_BAUD*/

25 2 = OUTPUT(CTR_00) = HIGH(20); /* BAUD RATE = 400K_BAUD*/

/* INITIALIZE THE 8274 */

26 2 = C=O;

27 2 = DO WHILE TABLE_74_B(C) <> OFFH;

28 3 = OUTPUT(COMMAND_B_74) = TABLE_74_B(C);

29 3 = C=C+1;

30 3 = OUTPUT(COMMAND_B_74) = TABLE_74_B(C);

31 3 = C=C+1;

32 3 = END;

210403-17
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;
CALL DELAY_S;
RETURN;
END INIT_B274_SDLC_S;
END INITS274_SDLC_S.

PL/H-B6 COMPILER ISBC 88/43 B274 CHANNEL A SDLC TEST

MODULE INFORMATION:

<table>
<thead>
<tr>
<th>CODE AREA SIZE</th>
<th>00A8H</th>
<th>1680</th>
</tr>
</thead>
<tbody>
<tr>
<td>CONSTANT AREA SIZE</td>
<td>0000H</td>
<td>0D</td>
</tr>
<tr>
<td>VARIABLE AREA SIZE</td>
<td>0003H</td>
<td>3D</td>
</tr>
<tr>
<td>MAXIMUM STACK SIZE</td>
<td>0006H</td>
<td>6D</td>
</tr>
<tr>
<td>213 LINES READ</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0 PROGRAM WARNINGS</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0 PROGRAM ERRORS</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

END OF PL/M-B6 COMPILATION

PL/M-B6 COMPILER ISBC 88/43 B274 CHANNEL A SDLC TEST

SERIES-III PL/M-B6 V2.0 COMPILATION OF MODULE INIT_B237_CHA
OBJECT MODULE PLACED IN :F1:SINI37.OBJ
COMPILER INVOKED BY: PLM86.B6 :F1:SINI37.PLM TITLE(ISBC 88/43 B274 CHANNEL A SDLC TEST) COMPACT NOINTVECTOR ROM

/*****************************************************************************/
/*
 *  8237  INITIALIZATION ROUTINE FOR DMA TRANSFER
 */
/*****************************************************************************/
1 INIT_B237_CHA: DO;
*NOLIST
12 1 INIT_B237_S: PROCEDURE PUBLIC;

13 2 OUTPUT(MASTER_CLEAR_37)=0;
14 2 OUTPUT(COMMAND_37) = 20H; /* EXTENDED WRITE */
15 2 OUTPUT(ALL_MASK_37) = 0FH; /* MASK ALL REQUESTS */
16 2 OUTPUT(MODE_REG_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(CH0_ADDR) = 0D; /* RECEIVE BUFF AT 900H */
20 2 OUTPUT(CH0_ADDR) = 09H;
21 2 OUTPUT(CH0_COUNT) = 01;
22 2 OUTPUT(CH0_COUNT) = 01;
23 2 OUTPUT(CH1_ADDR) = 0D; /* TRANSMIT BUFF AT 800H */
24 2 OUTPUT(CH1_ADDR) = 08H;
25 2 OUTPUT(CH1_COUNT) = 010H;
26 2 OUTPUT(CH1_COUNT) = 00H;
27 2 /* ENABLE TRANSFER */
28 2 OUTPUT(SINGLE_MASK) = CH1_SEL; /* ENABLE TX DMA */
29 2 RETURN;
30 1 /* TURN OFF THE 8237 CHANNELS 0 AND 1 */
31 2 STOP_8237_S: PROCEDURE PUBLIC;
32 2 OUTPUT(SINGLE_MASK) = CH1_SEL OR SET_MASK;
33 2 OUTPUT(SINGLE_MASK) = CH0_SEL OR SET_MASK;
34 2 END STOP_8237_S;
35 1 END INIT_8237_CHA;

MODULE INFORMATION:

CODE AREA SIZE = 004CH 76D
CONSTANT AREA SIZE = 0000H 00
VARIABLE AREA SIZE = 0004H 00

PL/M-B6 COMPILER 1SBC 88/45 8274 CHANNEL A SDLC TEST

MAXIMUM STACK SIZE = 0002H 2D
163 LINES READ 0 PROGRAM WARNINGS
0 PROGRAM ERRORS

END OF PL/M-B6 COMPILATION

SERIES-11 PL/M-B6 V2 0 COMPILATION OF MODULE INTR_8274_S
OBJECT MODULE PLACED IN: F1 SINTR.OBJ
COMPILER INVOKED BY: PLM66 B6. F1 SINTR PLM TITLE(1SBC 88/45 8274 CHANNEL
A SDLC TEST) COMPACT NOINTVECTOR ROM:

/*ERRUPT ROUTINE */
/*

1 INTR_8274_S DO.
\$NOLIST
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 MASK 59 BYTE.
17 1 DECLARE DONE LIT 'OFFH', NOT_DONE LIT '00H', PASS LIT 'OFFH',
FAIL LIT '00H'.

***************
/* IGNORE INTERRUPT HANDLER */
***************

18 1 IGNORE_INT PROCEDURE;
19 2 RESULTS_S = FAIL;
20 2 RETURN;
21 2 END IGNORE_INT.
/***********************************************************/
/* CHANNEL A EXTERNAL/STATUS CHANGE INTERRUPT HANDLER */
/***********************************************************/
22 1 CHA_EXTERNAL_CHANGE: PROCEDURE;
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 TIDONE_S = DONE;
26 2 ELSE DO;
27 3 TIDONE_S = DONE;
28 3 RESULTS_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;
EJECT

PL/M-86 COMPILER 18BC 88/45 B274 CHANNEL A SDLC TEST

/*********************************************************/
/* CHANNEL A SPECIAL RECEIVE CONDITIONS INTERRUPT HANDLER */
/*********************************************************/
33 1 CHA_RX_SPECIAL: PROCEDURE;
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 3 RESULTS_S = FAIL; /* RX OVERRUN ERROR */
46 3 RXDONE_S = DONE;
47 3 OUTPUT (COMMAND_A_74) = 30H; /* ERROR RESET */
48 3 END;
49 2 END;
50 2 RETURN;
51 2 END CHA_RX_SPECIAL;
EJECT

PL/M-86 COMPILER 18BC 88/45 B274 CHANNEL A SDLC TEST

/*********************/
/* CHANNEL A RECEIVE CHARACTER AVAILABLE */
/*********************/
53 1 CHA_RX_CHAR: PROCEDURE;
54 2 OUTPUT (SINGLE_MASK) = CHO_SEL; /* ENABLE RX DMA CHANNEL */
55 2 RETURN;
56 2 END CHA_RX_CHAR;
EJECT

PL/M-86 COMPILER 18BC 88/45 B274 CHANNEL A SDLC TEST

/* ENABLE B274 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);

210403-20
INT_VEC_STORE = INT_VEC;
INT_VEC = INTERRUPTPTR(INT_B274_5);
MASK_59 = INPUT(OCW1_59);
OUTPUT(OCW1_59) = MASK_59 AND CHA_INT_ON;
RETURN:
END ENABLE_INTERRUPTS_S:

/* DISABLE B274 INTERRUPTS - SET UP THE 8259A */
DISABLE_INTERRUPTS_S: PROCEDURE PUBLIC;
DISABLE:
INT_VEC = INT_VEC_STORE;
OUTPUT(OCW1_59) = MASK_59;
ENABLE:
RETURN:
END DISABLE_INTERRUPTS_S:

/* CHANNEL B RECEIVE CHARACTER AVAILABLE */
CHB_RX_CHAR: PROCEDURE;
TEMP = INPUT(DATA_B_74);
OUTPUT(COMMAND_B_74) = 38H;
RETURN:
END CHB_RX_CHAR.

#EJECT

PL/M-86 COMPILER ISBC 88/45 B274 CHANNEL A SDLC TEST

/*******************************************************************/
/* MAIN INTERRUPT ROUTINE */
/*******************************************************************/
INT_B274_S: PROCEDURE INTERRUPT 35 PUBLIC;
OUTPUT(COMMAND_B_74) = 2; /* SET POINTER TO 2 */
TEMP = INPUT(DATA_B_74) AND 07H; /* READ INTERRUPT VECTOR */
DO CASE TEMP,
CALL IGNORE_INT; /* VZVIVO = 000*/
CALL CHA_RX_Char; /* VZVIVO = 010*/
CALL CHA_EXTERNAL_CHANGE; /* VZVIVO = 100*/
CALL CHA_RX_SPECIAL; /* VZVIVO = 111*/
END;
OUTPUT(COMMAND_A_74) = 38H; /* END OF INTERRUPT FOR B274 */
OUTPUT(OCW2_59) = 63H; /* 8259 EOI */
OUTPUT(OCW1_59) = INPUT(OCW1_59) AND 0F7H;
RETURN:
END INT_B274_S.

/*******************************************************************/
/* DEFAULT INTERRUPT ROUTINE - 8259A INTERRUPT 7 */
/* REQUIRED ONLY WHEN DMA CONTROLLER IS ENABLED */
/* BEFORE RECEIVING FIRST CHARACTER WHICH IS */
/* AT HIGH BAUD RATES LIKE 800 BAUD. READ APP */
/* NOTE SECTION 6 FOR DETAILS */
AP-145

INTERRUPT 39

PROCEDURE INTERRUPT 39;

OUTPUT(OCW2_59) = 20H; /* NON-SPECIFIC EOI */
OUTPUT(OCW1_59) = INPUT(OCW1_59) AND 0F7H;
RESULTS_S = FAIL;
END INT_39;

END INTR_8274_S;

MODULE INFORMATION:
CODE AREA SIZE = 01BFH  4470
CONSTANT AREA SIZE = 0000H  00
VARIABLE AREA SIZE = 0020H  60
MAXIMUM STACK SIZE = 0022H  34D
256 LINES READ
0 PROGRAM WARNINGS
0 PROGRAM ERRORS
END OF PL/M-86 COMPILATION

PL/M-86 COMPILER  ISBC B8/45 8274 CHANNEL A SDLC TEST

SERIES-III PL/M-86 V2.0 COMPILED OF MODULE TEST
OBJECT MODULE PlACED IN: F1:STEST.OBJ
COMPILER INVOKED BY: PLMBB.B6+F:1:STEST.PLM TITLE(ISBC B8/45 8274 CHANNEL A SDLC TEST)
COMPACT NOINTVECTOR ROM

/*****************************************************************************/
/* ISBC 8274 PORT A (8274) SDLC TEST */
/*****************************************************************************/
1 TEST: DO;

2 1  DECLARE (RESULTS_S, TXDONE_S, RXDONE_S) BYTE PUBLIC;
2 1  DECLARE DONE LIT 'OFFH',
2 1  NOT_DONE LIT '00H',
2 1  PASS LIT 'OFFH',
2 1  FAIL LIT '00H';

210403-22

2-418
*EXCEPT*

**PL/M-B6 COMPILER 15BC 88/45 B274 CHANNEL A SDLC TEST**

30 1 CALL ENABLE_INTERRUPTS_S;
32 2 CALL INIT_B274_SDLC_S;
33 2 ENABLE;
34 2 CALL OUTPUT(COMMAND_A_74) = 20H; /* RESET TX INT/DMA */
35 2 OUTPUT(COMMAND_B_74) = 28H; /* BEFORE INITIALIZING B237*/
36 2 CALL INIT_B237_S;
37 2 CALL OUTPUT(DATA_A_74) = 35H; /* LOAD FIRST CHARACTER FROM CPU*/
38 2 /* TO ENSURE CRC TRANSMISSION RESET TX UNDERRUN LATCH*/
39 2 CALL RXDONE_S.TXDONE_S.NOT_DONE; /* CLEAR ALL FLAGS */
40 2 CALL RESULTS_S.PASS; /* FLAG SET FOR MONITOR*/
41 2 CALL DO WHILE TXDONE_S.NOT_DONE; /* DO UNTIL TERMINAL COUNT*/
42 3 CALL OUTPUT(C0D0M_A_74) = OC0H; /* DO UNTIL TERMINAL COUNT*/
43 2 CALL RXDONE_S.NT_DONE; /* DO UNTIL TERMINAL COUNT*/
44 3 CALL RXDONE_S.NT_DONE; /* DO UNTIL TERMINAL COUNT*/
46 3 CALL RXDONE_S.NT_DONE; /* DO UNTIL TERMINAL COUNT*/
47 2 CALL STOP_B237_S;
49 2 CALL DISABLE_INTERRUPTS_S;
50 2 RETURN RESULTS_S;
51 2 END CHA_SDLC_TEST;
52 1 END STEST;

**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/M-B6 COMPILATION

**PL/M-B6 COMPILER 15BC 88/45 B274 CHANNEL A SDLC TEST**

SERIES-III PL/M-B6 V2.0 COMPIILATION OF MODULE VECTOR_MODE
OBJECT MODULE PLACED IN :FI:VECTOR OBJ
COMPILER INVOKED BY: PLMB6.86 :FI:VECTOR.PLM TITLE(ISBC 88/45 B274 CHANNEL A SDLC TEST)

```plaintext
/*****************************/
/ 8274 INTERRUPT HANDLING ROUTINE FOR  
/ B274 VECTOR MODE  
/ STATUS AFFECTS VECTOR  
/*****************************/
```

END OF PL/M-B6 COMPILATION
VECTOR_MODE: DO;

DECLARE TEMP BYTE;

DECLARE (RESULTS_S, TXDONE, RXDONE) BYTE, EXTERNAL;

DECLARE RE DONE LITERALLY, "OFFH", NOT_DONE LITERALLY, "OHH", PASS LITERALLY, "OFFH", FAIL LITERALLY, "00H";

TX_INTERRUPT_CHAN: PROCEDURE INTERRUPT 84;

OUTPUT (COMMAND_A_74) = 00101000B; /*RESET TXINT PENDING*/

OUTPUT (COMMAND_A_74) = 00111000B; /*EDI*/

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/STATUS CONDITIONS */

ALSO IN RRI AND THEN TAKE APPROPRIATE ACTION BASED ON THE */

APPLICATION. */

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/STATUS CONDITIONS */

ALSO IN RRI AND THEN TAKE APPROPRIATE ACTION BASED ON THE */

APPLICATION. */

RX_CHAR_AVAILABLE_CHAN: PROCEDURE INTERRUPT 86;

OUTPUT (COMMAND_A_74) = 00111000B; /*EDI*/

END RX_CHAR_AVAILABLE_CHAN;

PL/M-86 COMPILER 185B/45 B274 CHANNEL A SDLC TEST
PL/M-86 COMPILER : SDC 88/45 0274 CHANNEL A SDLC TEST

/**************************************************************************
/* SPECIAL RECEIVE CONDITION INTERRUPT SERVICE ROUTINE CHECKS FOR */
/* END OF FRAME BIT ONLY. SEE SPECIAL SERVICE ROUTINE FOR NON- */
/* VECTORED MODE FOR CRC CHECK AND OVERRUN ERROR CHECK. */
/**************************************************************************

35 1 SPECIAL_RX_CONDITION_CHA: PROCEDURE INTERRUPT 87;
36 2 OUTPUT(COMMAND_A_74) = 1;  /* POINTER 1 */
37 2 TEMP = INPUT(Status_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) = 00110000B;  /* ERROR RESET */
45 2 OUTPUT(COMMAND_A_74) = 00111000B;  /* EDI */
46 2 RETURN;
47 2 END SPECIAL_RX_CONDITION_CHA;

48 1 ENABLE_INTERRUPTS: PROCEDURE PUBLIC;
49 2 DISABLE;
50 2 CALL SET_INTERRUPT(04. TX_INTERRUPT_CHA);
51 2 CALL SET_INTERRUPT(05. EXT_STAT_CHANGE_CHA);
52 2 CALL SET_INTERRUPT(06. RX_CHAR_AVAILABLE_CHA);
53 2 CALL SET_INTERRUPT(07. SPECIAL_RX_CONDITION_CHA);
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>
<th>Command Status Pointer</th>
<th>Register Pointer</th>
</tr>
</thead>
<tbody>
<tr>
<td>D7</td>
<td>D6</td>
<td>D5</td>
<td>D4</td>
</tr>
<tr>
<td>D3</td>
<td>D2</td>
<td>D1</td>
<td>D0</td>
</tr>
</tbody>
</table>

- 0 0 0: NULL CODE
- 0 0 1: SEND ABORT (SDLC)
- 0 1 0: RESET EXT STATUS INTERRUPTS
- 0 1 1: CHANNEL RESET
- 1 0 0: ENABLE INTERRUPT ON NEXT RX CHARACTER
- 1 0 1: RESET TX DMA PENDING
- 1 1 0: ERROR RESET
- 1 1 1: END OF INTERRUPT (Ch. A only)

| Write Register 0 (WR0) | 210403–26 |

WRITE REGISTER 1 (WR1)

<table>
<thead>
<tr>
<th>MSB</th>
<th>LSB</th>
<th>Ext Interrupt, Enable</th>
<th>Tx Interrupt DMA Enable</th>
<th>Status Affects Vector</th>
</tr>
</thead>
<tbody>
<tr>
<td>D7</td>
<td>D6</td>
<td>D5</td>
<td>D4</td>
<td>D3</td>
</tr>
<tr>
<td>D2</td>
<td>D1</td>
<td>D0</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- 0 0 0 0: RxINT/DMA DISABLE
- 0 0 1 0: RxINT ON FIRST CHAR OR SPECIAL CONDITION
- 0 1 0 1: INT ON ALL Rx CHAR (PARITY AFFECTS VECTOR) OR SPECIAL CONDITION
- 1 1 1 1: INT ON ALL Rx CHAR (PARITY DOES NOT AFFECT VECTOR) OR SPECIAL CONDITION

- 1 WAIT ON Rx, 0 WAIT ON Tx
- MUST BE ZERO

| Write Register 1 (WR1) | 210403–27 |

---

2-422
WRITE REGISTER 2 (WR2): CHANNEL A

<table>
<thead>
<tr>
<th>MSB</th>
<th>LSB</th>
</tr>
</thead>
<tbody>
<tr>
<td>D7</td>
<td>D0</td>
</tr>
</tbody>
</table>

- 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 8086/88 MODE
1 1 ILLEGAL

MUST BE ZERO
1 PIN 10 SYNDETs
0 PIN 10 RTSB

*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>MSB</th>
<th>LSB</th>
</tr>
</thead>
<tbody>
<tr>
<td>D7</td>
<td>D0</td>
</tr>
</tbody>
</table>

Rx ENABLE
SYNC CHAR LOAD INHIBIT
ADDR SRCH MODE (SDLC)
Rx CRC ENABLE
ENTER HUNT MODE
AUTO ENABLES

0 0 Rx5 BITS/CHAR
0 1 Rx7 BITS/CHAR
1 0 Rx6 BITS/CHAR
1 1 Rx8 BITS/CHAR
### WRITE REGISTER 4 (WR4)

**MSB**
- D7
- D6
- D5
- D4
- D3
- D2
- D1
- D0

**LSB**
- 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(G1111110)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)

**MSB**
- D7
- D6
- D5
- D4
- D3
- D2
- D1
- D0

**LSB**
- 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
- DTR

### WRITE REGISTER 6 (WR6)

**MSB**
- D7
- D6
- D5
- D4
- D3
- D2
- D1
- D0

**LSB**
- LEAST SIGNIFICANT Sync BYTE (ADDRESS IN SDLC/HDLC MODE)

### WRITE REGISTER (WR7)

**MSB**
- D7
- D6
- D5
- D4
- D3
- D2
- D1
- D0

**LSB**
- MOST SIGNIFICANT Sync BYTE (7EH IN SDLC/HDLC MODE)

### READ REGISTER 0 (RR0)

**MSB**
- D7
- D6
- D5
- D4
- D3
- D2
- D1
- D0

**LSB**
- Rx CHAR AVAILABLE
- INT PENDING (CHA ONLY)
- Tx BUFFER EMPTY
- CARRIER DETECT
- SYNC/HUNT
- CTS
- Tx UNDERRUN/EOM
- BREAK/ABORT
- EXTERNAL STATUS INTERRUPT MODE
READ REGISTER 1 (RR1): (SPECIAL RECEIVE CONDITION MODE)

![Diagram of READ REGISTER 1 (RR1)]

**Read Register 2 (RR2) Channel B Only**

![Diagram of READ REGISTER 2 (RR2)]

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

DFG TECHNICAL MARKETING
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 SDLC communication. This application note shows how to write I/O drivers for the 82530 to do initialization and data links using asynchronous (ASYNC) and SDLC 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.

1. 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>
</tbody>
</table>

80186 pins  | A2    | A/B  | 82530 pins |
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>RD</td>
<td>RD</td>
<td></td>
<td></td>
</tr>
<tr>
<td>WR</td>
<td>WR</td>
<td></td>
<td></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) = reg_no and 0fh'. 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.

```
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
/*-----------------------------------------------*/
/* read selected scc register */
rra: procedure (reg_no) byte;
    declare reg_no byte;
    if (reg_no and 0fh) <> 0
        then output(ch_a_command) = reg_no and 0fh;
        return input(ch_a_command);
    end rra;
rrb: procedure (reg_no) byte;
    declare reg_no byte;
    if (reg_no and 0fh) <> 0
        then output(ch_b_command) = reg_no and 0fh;
        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 0fh) <> 0
        then output(ch_a_command) = reg_no and 0fh;
        output(ch_a_command) = value;
    end wra;
wrb: procedure (reg_no, value);
    declare reg_no byte;
    declare value byte;
    if (reg_no and 0fh) <> 0
        then output(ch_b_command) = reg_no and 0fh;
        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.

```c
/*---------------------------------------------*/
/*                  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. 00000000b); /* 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 @ 6 mhz */
call wrb (13. 00000000b);
call wrb (14. 00000011b); /* BRG source = SYS CLK, enable BRG */
call wrb (15. 00000000b); /* all ext status interrupts off */
/* enables */
call wrb (03. 11000001b); /* 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. 00001001b); /* master IE, vector includes status */
call wrb (01. 00010011b); /* 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-
/*--------------------------------------------------------------------------*/
/* scc data character input from channel B */
scc_in: procedure byte;
    declare char byte;
    do while (input(ch_b_command) and 1h) = 0; end;
    char = input(ch_b_data); /* if rx data character is available */
    return char; /* then input it to buffer */
end scc_in;

/* scc data character output to channel B */
scc_out: procedure (char);
    declare char byte;
    do while (input(ch_b_command) and 4h) = 0; end;
    output(ch_b_data) = char; /* if tx buff empty then transfer the */
    /* data character to tx buff */
end scc_out;

Figure 4. ASYNC Communication in Polling Mode

ter 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);

#include 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>txintr_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 */

txintr_b: procedure interrupt 20h;
output (ch_b_data) = char_out_530;
call wra(00,38h);  /* reset highest IUS */
output (eoir_186) = 8000h;  /* non specific EOI */
return;
end txintr_b;

esi_b: procedure interrupt 22h;
call wrb(00,10h);  /* reset ESI */
call wra(00,38h);  /* reset highest IUS */
output (eoir_186) = 8000h;  /* non specific EOI */
return;
end esi_b;

rxintr_b: procedure interrupt 24h;
char_in_530 = input (ch_b_data);
call wra(00,38h);  /* reset highest IUS */
output (eoir_186) = 8000h;  /* non specific EOI */
return;
end rxintr_b;

src_b: procedure interrupt 26h;
call wrb(00,30h);  /* error reset */
call wra(00,38h);  /* reset highest IUS */
output (eoir_186) = 8000h;  /* non specific EOI */
return;
end src_b;

/*---------------------------------------------*/
```

Figure 5. ASYNC Communication in Interrupt Mode
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 (1)) - 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

```assembly
/*-----------------------------------------------*/

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, 0101010b);   /* node address */
call wra(07, 01111110b); /* SDLC flag */
call wra(10, 10000000b); /* preset CRC, NRZ encoding */
call wra(11, 01010110b); /* rxc = txc = BRG, trxc = BRG out */
call wra(12, 00001110b); /* to generate 125 Kbaud, xi @ 4 mhz */
call wra(13, 00000000b); /* BRG source = SYS CLK, DMA for Tx */
call wra(14, 00000110b); /* all ext status interrupts off */

/* enables */
call wra(14, 0000111b);  /* enable : BRG */
call wra(01, 11100000b); /* enable : dreq */
call wra(09, 00001001b); /* master IE, vector includes status */

end scc_init_a;

/*-----------------------------------------------*/
```

Figure 6. Initialization for SDLC Communication
/*-------------------------------------------*/

rx_init: procedure;

    declare dma_0_mode literally '1010010010000000b';
    /* 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_spl) = 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, 00000000b);  /* error reset */
    call wra(01, 11110001b);  /* 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 '00010111010000000b';
    /* 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, 01010101b);  /* 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
/* channel A interrupt procedures */

txintr_a: procedure interrupt 28h;
    call wra(00.38h);  // reset highest IUS */
    output (eoir_186) = 8000h;  // non specific EOI */
    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 EOI */
    return;
end esi_a;

rxintr_a: procedure interrupt 2ch;
    call wra(00.38h);  // reset highest IUS */
    output (eoir_186) = 8000h;  // non specific EOI */
    return;
end rxintr_a;

src_a:  procedure interrupt 2eh;
    rx_stat = rra(1);  // error reset */
    call wra(00.30h);  // disable rx */
    call wra(03.11000000b);  // set frame received flag */
    frame_recd = 0ffh;  // set frame received flag */
    call wra(00.38h);  // reset highest IUS */
    output (eoir_186) = 8000h;  // non specific EOI */
    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

\[
BRTC = \frac{4000000}{2 \times (9600 \times 16)} - 2
\]

\[
= \frac{4000000}{2 \times 153600} - 2
\]

\[
= 13.021 - 2 = 11.021
\]

![Diagram of Write Register 11](image-url)
Table 1. BRTC - Baud Rate Time Constant

<table>
<thead>
<tr>
<th>Baud Rate</th>
<th>1</th>
<th>16</th>
<th>32</th>
<th>64</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rate</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<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 x 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
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.

Figure 1. 82530–80186 Interface
**Figure 2. 80186–82530 Interface Read Cycle**

**Figure 3. 80186–82530 Interface Write Cycle**

**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.

* $\text{Iol} (2.0 \text{ mA}) > \text{lil} (-0.4 \text{ mA} + -0.5 \text{ mA})$
* $\text{loh} (-400 \mu\text{A}) > \text{lih} (20 \mu\text{A} + 20 \mu\text{A})$

**PCS5**: The PCS5 signal drives U2 and U4.

* $\text{Iol} (2.0 \text{ mA}) > \text{lil} (-0.5 \text{ mA} + -0.5 \text{ mA})$
* $\text{loh} (-400 \mu\text{A}) > \text{lih} (20 \mu\text{A} + 20 \mu\text{A})$

**INTA**: The INTA signal drives 2(U1) and U5.

* $\text{Iol} (2.0 \text{ mA}) > \text{lil} (-0.4 \text{ mA} + -0.8 \text{ mA} + -0.4 \text{ mA})$
* $\text{loh} (-400 \mu\text{A}) > \text{lih} (20 \mu\text{A} + 40 \mu\text{A} + 20 \mu\text{A})$

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., tRR).
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., Tclcl or '373 Tpd).
3. In the write cycle analysis, the timing variable TpdWR186-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. $t\text{AR}$: 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.

* $t\text{AR} = T\text{clcl} - T\text{clav(max)} - 245 \text{Tpd(max)} + T\text{clrl(min)} + 2(U2) T\text{pd(min)} - t\text{AR(min)}$

$$= 125 - 55 - 20.8 + 10 + 2(2) - 0 = 63.2 \text{ ns margin}$$
2. **tRA**: Address to \( \overline{RD} \) inactive hold time. The ALE delay is the worst case path and the 82530 requires 0 ns minimum.

\[
\begin{align*}
&\text{Tclcl} - \text{Tclrh} (\text{max}) + \text{Tcllh}(\text{min}) + '373 \text{ LE} \\
&= 55 - 55 + 5 + 8 - 2(5.5) = 2 \text{ ns margin}
\end{align*}
\]

3. **tCLR**: \( \overline{CS} \) active low to \( \overline{RD} \) active low set up time. The 82530 spec value is 0 ns minimum.

\[
\begin{align*}
&\text{Tclcl} - \text{Tclsv}(\text{max}) - \text{Tclr}(\text{min}) - U2 \\
&= 125 - 66 - 10 - 1 + 2 = 50 \text{ ns margin}
\end{align*}
\]

4. **tRCS**: \( \overline{RD} \) inactive to \( \overline{CS} \) inactive hold time. The 82530 spec calls for 0 ns minimum.

\[
\begin{align*}
&\text{Tcscsx}(\text{min}) - U2 \text{ skew}(\overline{RD} - \overline{CS}) - U2 \text{Tpd}(\text{max}) \\
&= 35 - 1 - 5.5 = 28.5 \text{ ns margin}
\end{align*}
\]

5. **tCHR**: \( \overline{CS} \) inactive to \( \overline{RD} \) active set up time. The 82530 requires 5 ns minimum.

\[
\begin{align*}
&\text{Tclcl} + 1(\text{Tclclwait state}) - 2(U2 \text{ skew}) - \text{tCHR} \\
&= (250 - 50) + 1(125) - 2(1) - 150 = 173 \text{ ns margin}
\end{align*}
\]

6. **tRR**: \( \overline{RD} \) pulse active low time. One 80186 wait state is included to meet the 150 ns minimum timing requirements of the 82530.

\[
\begin{align*}
&\text{Tclr}(\text{min}) + 1(\text{Tclclwait state}) - 2(U2 \text{ skew}) - \text{tRR} \\
&= (250 - 50) + 1(125) - 2(1) - 150 = 173 \text{ ns margin}
\end{align*}
\]

7. **tRDV**: \( \overline{RD} \) active low to valid data maximum delay for 80186 read data set up time \( (Tdvcl = 20 \text{ ns}) \). The margin is calculated on the Propagation delay path (worst case).

\[
\begin{align*}
&\text{Tclcl} + 1(\text{Tclclwait state}) - \text{Tclrh}(\text{max}) - Tdvcl(\text{min}) \\
&= 2(125) + 1(125) - 70 - 20 - 14.2 - 105 - 2(5.5) = 154 \text{ ns margin}
\end{align*}
\]

8. **tDF**: \( \overline{RD} \) inactive to data output float delay. The margin is calculated on \( \overline{DEN} \) active low of next cycle.

\[
\begin{align*}
&\text{Tclcl} + \text{Tclch}(\text{min}) - \text{Tclrh}(\text{max}) + \text{Tchctv}(\text{min}) - \\
&2(U2) \text{Tpd}(\text{max}) = 82530 \text{Tdf}(\text{max}) \\
&= 250 + 55 - 55 + 10 - 11 - 70 = 179 \text{ ns margin}
\end{align*}
\]

9. **tAD**: Address required valid to read data valid maximum delay. The 82530 spec value is 325 ns maximum.

\[
\begin{align*}
&3 \text{Tclcl} + 1(\text{Tclclwait state}) - \text{Tclav}(\text{max}) - '373 \\
&\text{Tpd}(\text{max}) - '245 \text{Tpd} - Tdvcl(\text{min}) - \text{tAD} \\
&= 375 + 125 - 55 - 20.8 - 14.2 - 20 - 325 = 65 \text{ ns margin}
\end{align*}
\]

**Write Cycle**

1. **tAW**: Address required valid to \( \overline{WR} \) active low set up time. The 82530 spec is 0 ns minimum.

\[
\begin{align*}
&\text{Tclcl} - \text{Tclav}(\text{max}) + \text{Tcvctv}(\text{min}) + '373 \text{ LE} \\
&= 125 - 66 + 5 - 20.8 + [125 - 5 + 1 + 4.4] - 0 = 170.6 \text{ ns margin}
\end{align*}
\]

2. **tWA**: \( \overline{WR} \) inactive to address invalid hold time. The 82530 spec is 0 ns.

\[
\begin{align*}
&\text{Tclcl} - \text{Tclcsx}(\text{min}) - \text{Tclr}(\text{min}) - U2 \text{Tpd}(\text{max}) \\
&= 55 - 55 + 5 + 8 - [5.5 + 3 + 7.1] = -2.6 \text{ ns margin}
\end{align*}
\]

3. **tCLW**: Chip select active low to \( \overline{WR} \) active low hold time. The 82530 spec is 0 ns.

\[
\begin{align*}
&\text{Tclcl} + \text{Tclcsx}(\text{min}) + \text{Tcvctv}(\text{min}) - '373 \text{ LE} \\
&\text{Tpd}(\text{min}) - \text{TpdWR186} - \text{WR530(LOW)} [\text{Tclcl} - \text{Tcvctv}(\text{min}) + \text{U3 Tpd}(\text{max}) + \text{U4 Tpd}(\text{min})] - \text{tAW} \\
&= 125 - 66 + 5.5 + [125 - 5 + 1 + 4.4] = 183.9 \text{ ns margin}
\end{align*}
\]

4. **tWCS**: \( \overline{WR} \) invalid to Chip Select invalid hold time. 82530 spec is 0 ns.

\[
\begin{align*}
&\text{Tclcl} - \text{Tclcsx}(\text{min}) - \text{U2 Tpd}(\text{max}) \\
&\text{TpdWR186} - \text{WR530(HIGH)} [\text{U2 Tpd}(\text{max}) + \text{U3 Tpd}(\text{max}) + \text{U4 Tpd}(\text{max})] \\
&= 35 + 1.5 - [5.5 + 3 + 7.1] = 20.9 \text{ ns margin}
\end{align*}
\]

5. **tCHW**: Chip Select inactive high to \( \overline{WR} \) active low set up time. The 82530 spec is 5 ns.

\[
\begin{align*}
&\text{Tclcl} + \text{Tclch}(\text{min}) + \text{Tcvctv}(\text{min}) - \text{Tchcsx}(\text{max}) \\
&\text{U2 Tpd}(\text{max}) + \text{TpdWR186} - \text{WR530(LOW)} [\text{Tclcl} - \text{Tcvctv}(\text{min}) + \text{U3 Tpd}(\text{min}) + \text{U4 Tpd}(\text{min})] - \text{tCHW} \\
&= 125 + 55 + 5 - 35 - 5.5 + [125 - 5 + 1 + 4.4] - 5 = 264 \text{ ns margin}
\end{align*}
\]

6. **tWW**: \( \overline{WR} \) active low pulse. 82530 requires a minimum of 60 ns from the falling to the rising edge of \( \overline{WR} \). This includes one wait state.
* Tdwh = 2(Tclcl – 40) + 1 (Tclclwait state) – TpdWR/186 – WR530(LOW) [Tclcl – Tcvctv(min) + U3 Tpd(max) + U4 Tpd(max)] + TpdWR/186 = WR530(HIGH) [U2 Tpd(min) + U3 Tpd(min) + U4 Tpd(min)] – tWW

= 210 + 1(125) – [125 – 5 + 4.5 + 9.2] – [1.5 + 1 + 3.2] – 60 = 135.6 ns margin

7. tDW: Data valid to WR active low setup time. The 82530 spec requires 0 ns.

* Tcvctv(min) – Tclcl – TpdWR186 – WR530(LOW) [Tclcl – Tcvctv(min) + U3 Tpd(max) + U4 Tpd(max)]

= 5 – 44 – 14.2 + 125 – 5 + 1.0 + 4.4 = 72.2 ns margin

8. tWD: Data valid to WR inactive high hold time. The 82530 requires a hold time of 0 ns.

* Tclch – skew [Tcvctv(max) + Tcvctv(min)] + 2'(45 Tpd(max) + TpdWR186 – WR530(HIGH) [U2 Tpd(max) + U3 Tpd(max) + U4 Tpd(max)]

= 55 – 5 + 11.25 – [5.5 + 3.0 + 7.1] = –50.6 ns margin

**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

= 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. 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. tW: WR inactive high to INTA active low minimum hold time. The spec is 0 ns and the margin assumes CLK coincident with INTA.

* Tclcl – Tcvctv(max) – TpdWR186 – WR530(HIGH) [U3 Tpd(max) + U4 Tpd(max)] + Tcvctv(min) + U1 Tpd(min)

= 125 – 55 – [5.5 + 3 + 7.1] + 5 + 10 = 69.4 ns margin

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.

* Tclcl – Tcrlh(max) – 2 U2 Tpd(max) + Tcvctv(min) + U1 Tpd(min)

= 125 – 55 – 2(5.5) + 5 + 10 = 74 ns margin

7. tIID: INTA active low to RD active low minimum setup time. This 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 this particular SCC, and tEIEO for each device separating them in the daisy chain. The typical system with only 1 SCC requires tIID 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 tIID > tCEQ = 250 ns.

* 4 Tclcl + 2 Tidle states – Tcvctv(max) – tIC [U1 Tpd(max) + 82530 CLK period] + Tcvctv(min) + U5 Tpd(min) + U2 Tpd(min) – tIID

= 500 + 250 – 70 – [45 + 250] + 5 + 6 + 2 – 250

= 148 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). tIDV spec is 100 ns maximum.

* 3 Tclcl – Tcvctv(max) – U5 Tpd(max) – U2 Tpd(max) – tIDV(max) – ‘245 Tpd(max) – Tdvcl(min)

= 375 – 70 – 25 – 5.5 – 100 – 14.2 – 20 = 140.3 ns margin

2-446
9. **tII**: RD pulse low time. The 82530 requires a minimum of 125 ns.

\* 3 Tcicl - Tcvctv(max) - U5 Tpd(max) - U2
\* Tpd(max) + Tcvctv(min) + U5 Tpd(min) + U2 Tpd(min)
\* Tcvctv(max) - U5 Tpd(max) - Tdrqcl - tll(min)

\[\begin{align*}
&= 375 - 70 - 25 - 5.5 + 5 + 6 + 1.5 - 125 = \\
&= 162 \text{ ns margin}
\end{align*}\]

**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 equation. 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 Tcicl - Tciclsv(max) - U4Tpd(max) - Tdrqcl(min)

\[\begin{align*}
&= 500 - 66 - 10.5 - 25 = 398.5 \text{ ns margin}
\end{align*}\]

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 Tcicl + 2(Tciclwait state) - Tcirl(max) - 2(U2)
\* Tpd(max) - Tdrqcl - tRRI

\[\begin{align*}
&= 2(125) + 2(125) - 70 - 2(5.5) - 200 = 219 \text{ ns margin}
\end{align*}\]

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 Tcicl + n(wait states) minus WR active delay for the 82530 to inactivate its REQ signal.

\* Tcicl + 2(Tciclwait state) - Tcvctv(min) - TpdWR186-WR530(LOW) [Tcicl - Tcvctv(min) + U3 Tpd(max) + U4 Tpd(max)] - Tdrqcl - tWRI

\[\begin{align*}
&= 375 - 5 - [125 - 5 + 4.5 + 9.21 - 25 - 200] = \\
&= 11.3 \text{ ns margin}
\end{align*}\]

**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 Tcicl - U3 Tpd(max) - 2(U2) Tpd(max) + U4 Tpd(min) - tREC

\[\begin{align*}
&= 1000 - 17.5 - 2(5.5) + 3.5 - 250 = 725 \text{ ns margin}
\end{align*}\]

**82530 VALID ACCESS LOGIC**

Due to the unique internal data path of the 82530, an intra-access recovery time must be provided to settle any internal metastable conditions. This internal metastable condition gives rise to the Clock Recovery (tREC) specification required by the 82530. This tREC is measured from the rising edge of a RD or WR to the falling edge of the next RD or WR intended for the 82530, and equates to 6 CLK's + 130 ns. Effectively, this specification implies that the system must provide 1130 ns (6 MHz 82530) between every CPU or other DMA access to the 82530. (Figure 1.)

Systems that only allow CPU access to the 82530 are not significantly impacted by this clock recovery time. In CPU access only designs, the software designer can insert NOP's to guarantee the tREC idle time in between successive CPU RD or WR cycles to the 82530. Unfortunately, systems that contain more than one direct memory access device, interfacing with the 82530, will require external hardware to arbitrate 82530 accesses and thereby guaranteeing the tREC restriction.
EXTERNAL VALID ACCESS HARDWARE

To accommodate this clock recovery specification, external hardware has been designed for the 82530 systems containing several DMA devices accessing the 82530 (i.e., a CPU and a DMA controller). This logic has been tailored for an 80186 environment but can easily be modified to fit 8086 or 80286 systems.

LOGIC STATE MACHINE

There are two basic functions that need to be performed by the external logic. The first is to mask the CS signal from reaching the 82530 until the tREC intra-access idle time has elapsed. The second task is to generate a not ready condition to the CPU or DMA device until the tREC period has expired and the minimum wait state requirement for the particular access has been satisfied. The simple state machine, Figure 7, illustrates the required operation.

The TTL logic pictured in Figure 7 implements the state machine with some assorted gates, a flip-flop, and a shift register. PCS from the 80186 should be qualified with RD + WR to eliminate switching glitches during T1. The LS74 and LS00 perform rising edge detection to reset the shift register. The shift register clocks out the tREC period to enable CS and the additional 2 CLK’s (82530) to satisfy the 82530 3 wait state requirement. The 80186 should be programmed to use the internal wait state generator (3 wait states for the 82530 and an 8 MHz 80186) and the external READY signal.

Note of caution: This hardware logic has not been verified on a bread board in an actual system. The hardware designer should verify that this logic fulfills his particular system timing requirements.
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 Tranceivers 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.

Figure 1. Block Diagram

Figure 2. Pin Configuration

November 1986
Order Number: 205248-002
8291A FEATURES AND IMPROVEMENTS

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 SFC (Serial Poll Complete) interrupt on the 8291A. SFC 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 SFC 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>D0–D7</td>
<td>12–19</td>
<td>I/O</td>
<td>DATA BUS PORT: To be connected to microprocessor data bus.</td>
</tr>
<tr>
<td>RS0–RS2</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 RS0–RS2.</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 “idle” (initialization) mode. The device will remain at “idle” until released by the microprocessor, with the “Immediate Execute pon” local message.</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 DIO1–DIO8 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 DIO&lt;sub&gt;1&lt;/sub&gt;–DIO&lt;sub&gt;8&lt;/sub&gt; and DAV lines and input signals on the NRFD and NDAC lines (active source handshake). Set low to indicate input data/signals on the DIO&lt;sub&gt;1&lt;/sub&gt;–DIO&lt;sub&gt;8&lt;/sub&gt; 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 D<sub>0</sub>–D<sub>7</sub> to DIO<sub>0</sub>–DIO<sub>8</sub> 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 state 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 can 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</td>
</tr>
<tr>
<td>PPAS</td>
<td>Parallel Poll Active State</td>
</tr>
<tr>
<td>PPIS</td>
<td>Parallel Poll Idle State</td>
</tr>
<tr>
<td>PPSS</td>
<td>Parallel Poll Standby State</td>
</tr>
<tr>
<td>PUCS</td>
<td>Parallel Poll Unaddressed to Configure</td>
</tr>
<tr>
<td>REMS</td>
<td>Remote State</td>
</tr>
<tr>
<td>RWLS</td>
<td>Remote With Lockout State</td>
</tr>
<tr>
<td>SACS</td>
<td>System Control Active State</td>
</tr>
<tr>
<td>SDYS</td>
<td>Source Delay State</td>
</tr>
<tr>
<td>SGNS</td>
<td>Source Generate State</td>
</tr>
<tr>
<td>SIAS</td>
<td>System Control Interface Clear Active</td>
</tr>
<tr>
<td>SIDS</td>
<td>Source Idle State</td>
</tr>
<tr>
<td>SIIS</td>
<td>System Control Interface Clear Idle</td>
</tr>
<tr>
<td>SINS</td>
<td>System Control Interface Clear Not Active</td>
</tr>
<tr>
<td>SIWS</td>
<td>Source Idle Wait State</td>
</tr>
<tr>
<td>SNAS</td>
<td>System Control Not Active State</td>
</tr>
<tr>
<td>SPAS</td>
<td>Serial Poll Active State</td>
</tr>
<tr>
<td>SPIS</td>
<td>Serial Poll Idle State</td>
</tr>
<tr>
<td>SPMS</td>
<td>Serial Poll Mode State</td>
</tr>
<tr>
<td>SRAS</td>
<td>System Control Remote Enable Active</td>
</tr>
<tr>
<td>SRIS</td>
<td>System Control Remote Enable Idle</td>
</tr>
<tr>
<td>SRNS</td>
<td>System Control Remote Enable Not Active</td>
</tr>
<tr>
<td>SRQS</td>
<td>Service Request State</td>
</tr>
<tr>
<td>STRS</td>
<td>Source Transfer State</td>
</tr>
<tr>
<td>SWNS</td>
<td>Source Wait for New Cycle State</td>
</tr>
<tr>
<td>TACS</td>
<td>Talker Active State</td>
</tr>
<tr>
<td>TADS</td>
<td>Talker Addressed State</td>
</tr>
<tr>
<td>TIDS</td>
<td>Talker Idle State</td>
</tr>
<tr>
<td>TPIS</td>
<td>Talker Primary 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</td>
<td>go to standby</td>
<td>C</td>
</tr>
<tr>
<td>ist</td>
<td>individual status</td>
<td>PP, L, LE</td>
</tr>
<tr>
<td>ion</td>
<td>listen only</td>
<td>SH</td>
</tr>
<tr>
<td>lpe</td>
<td>local poll enable</td>
<td>SH, AH, T, TE, L, LE, SR, RL, PP, C</td>
</tr>
<tr>
<td>nba</td>
<td>new byte available</td>
<td>RH</td>
</tr>
<tr>
<td>pon</td>
<td>power on</td>
<td>C</td>
</tr>
<tr>
<td>rdy</td>
<td>ready</td>
<td>C</td>
</tr>
<tr>
<td>rpp</td>
<td>request parallel poll</td>
<td>C</td>
</tr>
<tr>
<td>rsc</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>C</td>
</tr>
<tr>
<td>sic</td>
<td>send interface clear</td>
<td>C</td>
</tr>
<tr>
<td>sref</td>
<td>send remote enable</td>
<td>C</td>
</tr>
<tr>
<td>tca</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></td>
<td><strong>REMOTE MESSAGES RECEIVED</strong></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</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></td>
<td><strong>REMOTE MESSAGES SENT</strong></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>
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>[PPE]</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>Ready 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 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>0</td>
<td>1</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

D17 D16 D15 D14 D13 D12 D11 D10
DATA-IN REGISTER (0R)

DO7 DO6 DO5 DO4 DO3 DO2 DO1 DO0
DATA-OUT REGISTER (0W)

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.

Interrupt Registers

CPT APT GET END DEC ERR BO BI
INTERRUPT STATUS 1 (1R)

INT SPAS LLO REM SPC LLOC REMC ADSC
INTERRUPT STATUS 2 (2R)

CPT APT GET END DEC ERR BO BI
INTERRUPT ENABLE 1 (1W)

0 0 DMA0 DMA1 SPC LLOC REMC ADSC
INTERRUPT ENABLE 2 (2W)

INT DT0 DL0 AD5-0 AD4-0 AD3-0 AD2-0 AD1-0
ADDRESS 0 REGISTER
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.

**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.

### Figure 5. 8291A Registers

<table>
<thead>
<tr>
<th>READ REGISTERS</th>
<th>REGISTER SELECT CODE</th>
<th>WRITE REGISTERS</th>
</tr>
</thead>
<tbody>
<tr>
<td>DATA IN</td>
<td>RS2 RS1 RS0</td>
<td>DATA OUT</td>
</tr>
<tr>
<td>D17 D16 D15 D14 D13 D12 D11 D10</td>
<td>0 0 0</td>
<td>D07 D06 D05 D04 D03 D02 D01 D00</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>INTERRUPT ENABLE 1</td>
</tr>
<tr>
<td>INT SPAS LLO REM SPC LLOC REMC ADSC</td>
<td>0 1 0</td>
<td>0 0 DMAO DMAI SPC LLOC REMC ADSC</td>
</tr>
<tr>
<td>INTERRUPT STATUS 2</td>
<td></td>
<td>INTERRUPT ENABLE 2</td>
</tr>
<tr>
<td>S8 SEQS S6 S5 S4 S3 S2 S1</td>
<td>0 1 1</td>
<td>S8 rsv S6 S5 S4 S3 S2 S1</td>
</tr>
<tr>
<td>SERIAL POLL STATUS</td>
<td></td>
<td>SERIAL POLL MODE</td>
</tr>
<tr>
<td>lrt ion E0I LPAS TPAS LA TA MJMN</td>
<td>1 0 0</td>
<td>TO LO 0 0 0 0 ADM1 ADM0</td>
</tr>
<tr>
<td>ADDRESS STATUS</td>
<td></td>
<td>ADDRESS MODE</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>AUX MODE</td>
</tr>
<tr>
<td>INT DT0 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>ADDRESS 0/1</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>EOS</td>
</tr>
</tbody>
</table>

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>Indicates Undefined Commands</th>
<th>CPT</th>
<th>APT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Set by (TPAS + LPAS)•SCG•ACDS•MODE 3</td>
<td>An undefined command has been received.</td>
<td>A secondary address must be passed through to the microprocessor for recognition.</td>
</tr>
<tr>
<td>Set by DTAS</td>
<td>GET</td>
<td></td>
</tr>
<tr>
<td>Set by (EOS + EOI)•LACS</td>
<td>END</td>
<td></td>
</tr>
<tr>
<td>Set by DCAS</td>
<td>DEC</td>
<td></td>
</tr>
<tr>
<td>Set by TACS•nba•DAC•RFD</td>
<td>ERR</td>
<td></td>
</tr>
<tr>
<td>TACS•(SWNS + SGNS)</td>
<td>BO</td>
<td></td>
</tr>
<tr>
<td>Set by LACS•ACDS</td>
<td>BI</td>
<td></td>
</tr>
<tr>
<td>Shows status of the INT pin</td>
<td>INT</td>
<td></td>
</tr>
<tr>
<td>The device has been enabled for a serial poll</td>
<td>SPAS</td>
<td></td>
</tr>
<tr>
<td>The device is in local lock out state.</td>
<td>LLO</td>
<td></td>
</tr>
<tr>
<td>(LWLS•RWLS)</td>
<td>REM</td>
<td></td>
</tr>
<tr>
<td>The device is in a remote state.</td>
<td>REMC</td>
<td></td>
</tr>
<tr>
<td>(REMS•RWLS)</td>
<td>ADSC</td>
<td></td>
</tr>
</tbody>
</table>

### SPAS → SPAS if APRS:STRS:SPAS was true

- Serial Poll Complete interrupt. (SPC)
- Local lock out change interrupt. (LLC)
- Remote/Local change interrupt. (REMC)
- Address status change interrupt. (ADSC)

### NOTE:
1. In ton (talk-only) and Ion (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 DMAO, 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.

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>ADSC change in LIDS or TIDS or MJMN</td>
</tr>
<tr>
<td>1</td>
<td>REMC change in LOCS or REMS</td>
</tr>
<tr>
<td>2</td>
<td>LLOC change in LWLS or RWLS</td>
</tr>
</tbody>
</table>

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></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**SERIAL POLL STATUS (3R)**

<table>
<thead>
<tr>
<th>S8</th>
<th>rsv</th>
<th>S6</th>
<th>S5</th>
<th>S4</th>
<th>S3</th>
<th>S2</th>
<th>S1</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**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 SPE 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-
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 SRQ line is cleared. The SRQ line and the rsv bit are tied together.

Address Registers

<table>
<thead>
<tr>
<th>ton</th>
<th>ion</th>
<th>EO</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>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 Contents</th>
<th>Mode</th>
</tr>
</thead>
<tbody>
<tr>
<td>100000000</td>
<td>Mode 1 (Primary-Primary)</td>
</tr>
<tr>
<td>010000000</td>
<td>Mode 2 (Primary-Secondary)</td>
</tr>
<tr>
<td>110000000</td>
<td>Mode 3 (Primary/APT-Primary/AFT)</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 EIO bit which, when set, signifies that the EIO 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 8219A 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>110BBBBBB</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>COMMAND PASS THROUGH (5R)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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>AUX MODE (5W)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CNT0–2:CONTROL BITS</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>COM0–4:COMMAND BITS</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

3-13
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>Bits</td>
<td>Bits</td>
<td></td>
</tr>
<tr>
<td>000</td>
<td>0CCCCC</td>
<td>Execute auxiliary command CCCC</td>
</tr>
<tr>
<td>001</td>
<td>ODDDD</td>
<td>Preset internal counter to match external clock frequency of DDDD MHz (DDDD binary representation of 1 to 8 MHz)</td>
</tr>
<tr>
<td>100</td>
<td>DDDDD</td>
<td>Write DDDDD into auxiliary register A</td>
</tr>
<tr>
<td>101</td>
<td>DDDDD</td>
<td>Write DDDDD into auxiliary register B</td>
</tr>
<tr>
<td>011</td>
<td>USP3P2P1</td>
<td>Enable/disable parallel poll either in response to remote messages (PPC followed by PPE or PPD) or as a local lpe message. (Enable if U = 0, 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 line 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 T₁ in IEEE-488 and appears in the Source Handshake state diagram between the SDYS and STRS. As such, DAV is asserted T₁ after the DIO lines are driven. Consequently, T₁ is a major factor in determining the data transfer rate of the 8291A over the GPIB (T₁ = TWRDV2-TWRD15).

When open-collector transceivers are used for connection to the GPIB, T₁ is defined by IEEE-488 to be 2 μs. 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 ≤ N_F ≤ 8, N_F = (DDDD)_2). When N_F = f_c, a 2 μs T₁ delay will be generated before each DAV asserted.

\[ T₁(μs) = \frac{2N_F}{f_c} + t_{SYNC}, 1 ≤ N_F ≤ 8 \]

T₁ 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, tSYNC is less than half the clock cycle).

If it is necessary that T₁ be different from 2 μs, 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 T₁.

When tri-state transceivers are used, IEEE-488 allows a higher transfer rate (lower T₁). Use of the 8291A with such transceivers is enabled by setting B₂ in Auxiliary Register B. In this case, setting N_F = f_c causes a T₁ delay of 2μs to be generated for the first byte transmitted—all subsequent bytes will have a delay of 500 ns.

\[ T₁ (High Speed) μs = \frac{N_F}{2f_c} + t_{SYNC} \]

Thus, the shortest T₁ is achieved by setting N_F = 1 using an 8 MHz clock with a 50% duty cycle clock (tSYNC < 63 ns):

\[ T₁(HS) = \frac{1}{2×8} + 0.063 = 125 \text{ ns max.} \]

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 A₄A₃A₂A₁A₀ byte is written into the Auxiliary Register, it is loaded with the data A₄A₃A₂A₁A₀. Setting the respective bits to “1” enables the following features.

A₀—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.

A₁—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.

A₂—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.

A₃—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.

A₄—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 A₀ = A₁ = 1, a special “continuous Acceptor Handshake” 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 ready local message is automatically generated when in ANRS. As such, the 8291A Acceptor 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 "tcs 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 B_4B_3B_2B_1B_0 is written into the Auxiliary Mode Register, it is loaded with the data B_4B_3B_2B_1B_0. Setting the respective bits to "1" enables the following features:

B_0—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.

B_1—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.

B_2—Enable High Speed Data Transfer: This feature may be enabled when tri-state external transceivers are used. The data transfer rate is limited by T_1 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, T_1 = 2 microseconds is generated for the first byte transmitted after each true to false transition of ATN. For all subsequent bytes, T_1 = 500 ns. Refer to the Internal Counter section for an explanation of T_1 duration as a function of B_2 and of clock frequency.

B_3—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.

B_4—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 011USP_3P_2P_1 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, PPR_N, be sent true (Response = S + ist). The bits P_3P_2P_1 specify which of the eight data lines PPR_N 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 PPR_N true or false according to the comparison.

If a PP_2^ 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 PP_1^ implementation is desired, the undefined command features of the 8291A must be used. In PP_1^, 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>

EOS REGISTER

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. $N_{{\text{f}}}$ in the Internal Counter is set to 8 MHz. This setting causes the longest possible $T_{{\text{f}}}$ delay to be generated in the Source Handshake (16 $\mu$s 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 "lpe" 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 DMA0 and DMA1 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 + ION), 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: A0, 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 GPIB 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.

![Diagram of 8291A and 8293 System Configuration](image)

* = GPIB Bus Transceiver

Figure 6. 8291A and 8293 System Configuration
Figure 7. 8291A, 8292, and 8293 System Configuration
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 (0DH) 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 pong” 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 (0DH) 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, OBI, IBI 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 GPIOB. 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:** This is a production data sheet. The specifications are subject to change without notice.

*WARNING:* Stressing the device beyond the “Absolute Maximum Ratings” may cause permanent damage. These are stress ratings only. Operation beyond the “Operating Conditions” is not recommended and extended exposure beyond the “Operating Conditions” may affect device reliability.

### D.C. CHARACTERISTICS  
\( V_{CC} = 5V \pm 10\%, T_A = 0°C \text{ to } 70°C \) (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>V</td>
<td>IOL = 2 mA (4 mA for TR1 pin)</td>
<td></td>
</tr>
<tr>
<td>( V_{OH} )</td>
<td>Output High Voltage</td>
<td>2.4</td>
<td>V</td>
<td>IOH = -400 ( \mu A ) ( -150 ( \mu A ) for SRQ pin)</td>
<td></td>
</tr>
<tr>
<td>( V_{OH-INT} )</td>
<td>Interrupt Output High Voltage</td>
<td>2.4</td>
<td>3.5</td>
<td>V</td>
<td>IOH = -400 ( \mu A )</td>
</tr>
<tr>
<td>( l_{IL} )</td>
<td>Input Leakage</td>
<td>10</td>
<td>( \mu A )</td>
<td>VIN = 0V to ( V_{CC} )</td>
<td></td>
</tr>
<tr>
<td>( I_{OFL} )</td>
<td>Output Leakage Current</td>
<td>±10</td>
<td>( \mu A )</td>
<td>( V_{OUT} = 0.45V, ( V_{CC} )</td>
<td></td>
</tr>
<tr>
<td>( I_{CC} )</td>
<td>( V_{CC} ) Supply Current</td>
<td>120</td>
<td>mA</td>
<td>( T_A = 0°C )</td>
<td></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>TEO12</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>TNRDR1</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>TRDNR3</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>TWREO5</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, NF = IC, tSYNC = ½IC</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>Symbol</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>
APPENDIX A

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>Convention</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>IEEE-488</td>
</tr>
<tr>
<td>0</td>
<td>T</td>
<td>DAV</td>
</tr>
<tr>
<td>1</td>
<td>F</td>
<td>DAV</td>
</tr>
<tr>
<td>0</td>
<td>T</td>
<td>NDAC</td>
</tr>
<tr>
<td>1</td>
<td>F</td>
<td>NDAC</td>
</tr>
<tr>
<td>0</td>
<td>T</td>
<td>NRFD</td>
</tr>
<tr>
<td>1</td>
<td>F</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\text{OUT} \leq V\text{OL} for AH; V\text{IN} \leq V\text{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
This transition will never occur under normal operation. $T_{\text{DELAY}}$ is about 300 ns for debouncing DAV.

Figure A-1. 8291A State Diagrams (Continued)
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, T, 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</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 DI0, 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

<table>
<thead>
<tr>
<th>DIO1-DIO8</th>
<th>VALID</th>
<th>NOT VALID</th>
<th>VALID</th>
</tr>
</thead>
<tbody>
<tr>
<td>T1</td>
<td></td>
<td>TNDDV1</td>
<td></td>
</tr>
<tr>
<td></td>
<td>TDVNR1</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>TRDNR3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TDVND3</td>
<td></td>
<td>TDVND2</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>TRDNR3</td>
<td></td>
</tr>
<tr>
<td>DREQ(SH)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>TDVDR3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DREQ(AH)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>WR</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RD</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure C-1. 3-Wire Handshake Timing at 8291A
8292
GPIB CONTROLLER

- Complete IEEE Standard 488 Controller Function
- Interface Clear (IFC) Sending Capability Allows Seizure of Bus Control and/or Initialization of the Bus
- Responds to Service Requests (SRQ)
- Sends Remote Enable (REN), Allowing Instruments to Switch to Remote Control
- Complete Implementation of Transfer Control Protocol
- Synchronous Control Seizure Prevents the Destruction of Any Data Transmission in Progress
- Connects with the 8291 to Form a Complete IEEE Standard 488 Interface Talker/Listener/Controller

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 ÷ 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>ATN</td>
<td>22</td>
<td>I</td>
<td>ATTENTION IN: Used by the 8292 to monitor the GPIBATN 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>ATN</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 ±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 μsec 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>EO12</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
I SYC I IFC I ANTI I SRO I 10'

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>IFCR</th>
<th>IBF</th>
<th>OBF</th>
</tr>
</thead>
<tbody>
<tr>
<td>D7</td>
<td>D0</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 (OBF1 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 buffer 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 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>SPI</th>
<th>TCI</th>
<th>SYC</th>
<th>OBF1</th>
<th>IBFI</th>
<th>0</th>
<th>SRQ</th>
</tr>
</thead>
<tbody>
<tr>
<td>D7</td>
<td>D0</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 D2 are undefined. An interrupt is enabled by setting the corresponding register bit.

**SRQ** Enable interrupts on SRQ received.

**IBFI** Enable interrupts on output buffer empty.

**OBF1** 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>A0</td>
<td>1</td>
</tr>
<tr>
<td>D7</td>
<td>D0</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><strong>GPIB (BUS) STATUS</strong></td>
<td><strong>EVENT COUNTER</strong></td>
</tr>
<tr>
<td>REN</td>
<td>DAV</td>
</tr>
<tr>
<td><strong>EVENT COUNTER STATUS</strong></td>
<td><strong>TIME OUT</strong></td>
</tr>
<tr>
<td>D</td>
<td>D</td>
</tr>
<tr>
<td><strong>TIME OUT STATUS</strong></td>
<td><strong>NOTE:</strong> These registers are accessed by a special utility command, see page 7.</td>
</tr>
</tbody>
</table>

D | D | D | D | D | D | D | D | 0*
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></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></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></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></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**

<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>

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**

<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 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 WTOUT command.

**Time Out Status 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>

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**

<table>
<thead>
<tr>
<th>X</th>
<th>X</th>
<th>USER</th>
<th>X</th>
<th>X</th>
<th>TOUT3</th>
<th>TOUT2</th>
<th>TOUT1</th>
</tr>
</thead>
</table>

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-
ing ATN or a new 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 tCµ. 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 tCµ. 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 Interrupts

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 counter will be enabled. If the 8292 was not the active controller, this command will exit immediately. If the 8292 is the active controller, the counter 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 ACS 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—GSBS—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 TCNTR 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 ATNI 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 checking 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—STCNI—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 Ao = 0) following this command will be loaded into the Event Counter Register and the Event Counter Status for byte counting of EOI counting. Only IBFI 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>D7</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Each named bit in an Interrupt Acknowledge (IACK) 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, OBFI, IBFI, DAV, REV).
2) The four interrupt outputs (TCI, SPI, OBFI, IBFI) 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 RATINGS*
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: This is a production data sheet. The specifications are subject to change without notice.
*WARNING: Stressing the device beyond the “Absolute Maximum Ratings” may cause permanent damage. These are stress ratings only. Operation beyond the “Operating Conditions” is not recommended and extended exposure beyond the “Operating Conditions” may affect device reliability.

D.C. CHARACTERISTICS  TA = 0°C to 70°C, VSS = 0V; 8292, VCC = ±5V ±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>VIL1</td>
<td>Input Low Voltage (All Except X1, X2, RESET)</td>
<td>-0.5</td>
<td>0.8</td>
<td>V</td>
<td></td>
</tr>
<tr>
<td>VIL2</td>
<td>Input Low Voltage (X1, X2, RESET)</td>
<td>-0.5</td>
<td>06</td>
<td>V</td>
<td></td>
</tr>
<tr>
<td>VIH1</td>
<td>Input High Voltage (All Except X1, X2, RESET)</td>
<td>2.2</td>
<td>VCC</td>
<td>V</td>
<td></td>
</tr>
<tr>
<td>VIH2</td>
<td>Input High Voltage (X1, X2, RESET)</td>
<td>3.8</td>
<td>VCC</td>
<td>V</td>
<td></td>
</tr>
<tr>
<td>VOL1</td>
<td>Output Low Voltage (D0–D7)</td>
<td>0.45</td>
<td>V</td>
<td>IOL = 2.0 mA</td>
<td></td>
</tr>
<tr>
<td>VOL2</td>
<td>Output Low Voltage (All Other Outputs)</td>
<td>0.45</td>
<td>V</td>
<td>IOL = 1.6 mA</td>
<td></td>
</tr>
<tr>
<td>VOH1</td>
<td>Output High Voltage (D0–D7)</td>
<td>2.4</td>
<td>V</td>
<td>IOH = -400 µA</td>
<td></td>
</tr>
<tr>
<td>VOH2</td>
<td>Output High Voltage (All Other Outputs)</td>
<td>2.4</td>
<td>V</td>
<td>IOH = -50 µA</td>
<td></td>
</tr>
<tr>
<td>IIL</td>
<td>Input Leakage Current (COUNT, IFCL, RD, WR, CS, A0)</td>
<td>±10</td>
<td>µA</td>
<td>VSS ≤ VIN ≤ VCC</td>
<td></td>
</tr>
<tr>
<td>IOZ</td>
<td>Output Leakage Current (D0–D7, High Z State)</td>
<td>±10</td>
<td>µA</td>
<td>VSS + 0.45 ≤ VIN ≤ VCC</td>
<td></td>
</tr>
<tr>
<td>IIL1</td>
<td>Low Input Load Current (Pins 21–24, 27–38)</td>
<td>0.5</td>
<td>mA</td>
<td>VIN = 0.8V</td>
<td></td>
</tr>
<tr>
<td>IIL2</td>
<td>Low Input Load Current (RESET)</td>
<td>0.2</td>
<td>mA</td>
<td>VIN = 0.8V</td>
<td></td>
</tr>
<tr>
<td>ICC</td>
<td>Total Supply Current</td>
<td>125</td>
<td>mA</td>
<td>Typical = 65 mA</td>
<td></td>
</tr>
<tr>
<td>IIH</td>
<td>Input High Leakage Current (Pins 21–24, 27–38)</td>
<td>100</td>
<td>µA</td>
<td>VIN = VCC</td>
<td></td>
</tr>
<tr>
<td>CIN</td>
<td>Input Capacitance</td>
<td>10</td>
<td>pF</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CI/O</td>
<td>I/O Capacitance</td>
<td>20</td>
<td>pF</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

A.C. CHARACTERISTICS  TA = 0°C to 70°C, VSS = 0V; 8292, VCC = ±5V ±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>tAR</td>
<td>CS, A0 Setup to RD ↓</td>
<td>0</td>
<td></td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>tRA</td>
<td>CS, A0 Hold to RD ↑</td>
<td>0</td>
<td></td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>tRR</td>
<td>RD Pulse Width</td>
<td>250</td>
<td></td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>tAD</td>
<td>CS, A0 to Data Out Delay</td>
<td>225</td>
<td>ns</td>
<td>CL = 150 pF</td>
<td></td>
</tr>
<tr>
<td>tRD</td>
<td>RD ↓ to Data Out Delay</td>
<td>225</td>
<td>ns</td>
<td>CL = 150 pF</td>
<td></td>
</tr>
<tr>
<td>tDF</td>
<td>RD ↑ to Data Float Delay</td>
<td>100</td>
<td>ns</td>
<td></td>
<td></td>
</tr>
<tr>
<td>tCY</td>
<td>Cycle Time</td>
<td>2.5</td>
<td>15</td>
<td>µs</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_9 ) Setup to WR</td>
<td>0</td>
<td>ns</td>
<td></td>
<td></td>
</tr>
<tr>
<td>( t_{WA} )</td>
<td>CS, ( A_0 ) Hold after WR</td>
<td>0</td>
<td>ns</td>
<td></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>ns</td>
<td></td>
<td></td>
</tr>
<tr>
<td>( t_{WD} )</td>
<td>Data Hold after WR</td>
<td>0</td>
<td>ns</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### COMMAND TIMINGS\(^{1, 3}\)

<table>
<thead>
<tr>
<th>Code</th>
<th>Name</th>
<th>Execution Time</th>
<th>iBFI ↑</th>
<th>TCI(2)</th>
<th>SPI</th>
<th>ATNO</th>
<th>CIC</th>
<th>IFC</th>
<th>REN</th>
<th>EOI</th>
<th>DAV</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>E1</td>
<td>WTOUT</td>
<td>63 24</td>
<td></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 24</td>
<td></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 24 51</td>
<td></td>
<td></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 24 47</td>
<td></td>
<td></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 24 49</td>
<td></td>
<td></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 24 77</td>
<td></td>
<td></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 24 72</td>
<td></td>
<td></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 24 49</td>
<td></td>
<td></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 24 49</td>
<td></td>
<td></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 24</td>
<td></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 24 70</td>
<td>↑61</td>
<td>↑61</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F2</td>
<td>RST</td>
<td>94 24</td>
<td>↓52</td>
<td></td>
<td></td>
<td></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>214 24 192</td>
<td>↓52</td>
<td>↓179</td>
<td>↓174</td>
<td>↓101</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>System Controller</td>
</tr>
<tr>
<td>F3</td>
<td>RSTi</td>
<td>61 24</td>
<td></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 24 107</td>
<td>↑98</td>
<td></td>
<td></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 24</td>
<td></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 24 100</td>
<td>↑91</td>
<td></td>
<td></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 24 55</td>
<td>↑46</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F8</td>
<td>SREM</td>
<td>91 24 73</td>
<td>↓64</td>
<td></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 24 133</td>
<td>↓120</td>
<td>↓115</td>
<td>↓42</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FA</td>
<td>TCNTR</td>
<td>108 24 86</td>
<td>↓71</td>
<td>↓68</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FC</td>
<td>TCAS</td>
<td>92 24 67</td>
<td>↓55</td>
<td></td>
<td></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 24 91</td>
<td>↓80</td>
<td></td>
<td></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 24</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PIN</td>
<td>RESET</td>
<td>29 ↓7 ↓7</td>
<td></td>
<td></td>
<td></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 ↓73 ↑98</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

Crystal series resistance should be <75Ω at 6 MHz; <180Ω at 3.6 MHz.

LC OSCILLATOR MODE

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 synchronously</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>(ACDS) = accept data state (AH function)</td>
<td></td>
</tr>
<tr>
<td>(ANRS) = acceptor not ready state (AH function)</td>
<td></td>
</tr>
<tr>
<td>(SDYS) = source delay state (SH function)</td>
<td></td>
</tr>
<tr>
<td>(STRS) = source transfer state (SH function)</td>
<td></td>
</tr>
<tr>
<td>(TADS) = talker addressed state (T function)</td>
<td></td>
</tr>
</tbody>
</table>
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

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>D</th>
<th>N</th>
<th>N</th>
<th>A</th>
<th>S</th>
<th>I</th>
<th>R</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACG</td>
<td>Addressed Command Group</td>
<td>M</td>
<td>A</td>
<td>C</td>
<td>Y</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>ATN</td>
<td>Attention</td>
<td>U</td>
<td>U</td>
<td>C</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>DAB</td>
<td>Data Byte</td>
<td>(Notes 1, 9)</td>
<td>M</td>
<td>D</td>
<td>D</td>
<td>D</td>
<td>D</td>
<td>D</td>
<td>D</td>
<td>D</td>
<td>D</td>
<td>X</td>
</tr>
<tr>
<td>DAC</td>
<td>Data Accepted</td>
<td>U</td>
<td>H</td>
<td>S</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>DAV</td>
<td>Data Valid</td>
<td>U</td>
<td>H</td>
<td>S</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>DCL</td>
<td>Device Clear</td>
<td>M</td>
<td>U</td>
<td>C</td>
<td>Y</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>END</td>
<td>End</td>
<td>U</td>
<td>S</td>
<td>T</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>EOS</td>
<td>End of String</td>
<td>(Notes 2, 9)</td>
<td>M</td>
<td>D</td>
<td>D</td>
<td>E</td>
<td>E</td>
<td>E</td>
<td>E</td>
<td>E</td>
<td>E</td>
<td>X</td>
</tr>
<tr>
<td>GET</td>
<td>Group Execute Trigger</td>
<td>M</td>
<td>A</td>
<td>C</td>
<td>Y</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>GTL</td>
<td>Go to Local</td>
<td>M</td>
<td>A</td>
<td>C</td>
<td>Y</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>IDY</td>
<td>Identify</td>
<td>U</td>
<td>U</td>
<td>C</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>IFC</td>
<td>Interface Clear</td>
<td>U</td>
<td>U</td>
<td>C</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>LAG</td>
<td>Listen Address Group</td>
<td>M</td>
<td>A</td>
<td>D</td>
<td>Y</td>
<td>0</td>
<td>1</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>LLO</td>
<td>Local Lock Out</td>
<td>M</td>
<td>U</td>
<td>C</td>
<td>Y</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>MLA</td>
<td>My Listen Address</td>
<td>(Note 3)</td>
<td>M</td>
<td>A</td>
<td>D</td>
<td>Y</td>
<td>0</td>
<td>1</td>
<td>L</td>
<td>L</td>
<td>L</td>
<td>L</td>
</tr>
<tr>
<td>MTA</td>
<td>My Talk Address</td>
<td>(Note 4)</td>
<td>M</td>
<td>A</td>
<td>D</td>
<td>Y</td>
<td>1</td>
<td>0</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>MSA</td>
<td>My Secondary Address</td>
<td>(Note 5)</td>
<td>M</td>
<td>S</td>
<td>E</td>
<td>Y</td>
<td>1</td>
<td>S</td>
<td>S</td>
<td>S</td>
<td>S</td>
<td>S</td>
</tr>
<tr>
<td>NUL</td>
<td>Null Byte</td>
<td>M</td>
<td>D</td>
<td>D</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>
<tr>
<td>OSA</td>
<td>Other Secondary Address</td>
<td>M</td>
<td>S</td>
<td>E</td>
<td>(OSA = SCG \ MSA)</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>A</td>
<td>D</td>
<td>(OTA = TAG \ MTA)</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>—</td>
<td>(PCG = ACG \ UCG \ LAG \ TAG)</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>A</td>
<td>C</td>
<td>Y</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>PPE</td>
<td>Parallel Poll Enable</td>
<td>(Note 6)</td>
<td>M</td>
<td>S</td>
<td>E</td>
<td>Y</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>S</td>
<td>P</td>
<td>P</td>
</tr>
<tr>
<td>PPD</td>
<td>Parallel Poll Disable</td>
<td>(Note 7)</td>
<td>M</td>
<td>S</td>
<td>E</td>
<td>Y</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>D</td>
<td>D</td>
<td>D</td>
</tr>
<tr>
<td>PPR1</td>
<td>Parallel Poll Response 1</td>
<td>U</td>
<td>S</td>
<td>T</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>PPR2</td>
<td>Parallel Poll Response 2</td>
<td>U</td>
<td>S</td>
<td>T</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>PPR3</td>
<td>Parallel Poll Response 3</td>
<td>U</td>
<td>S</td>
<td>T</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>PPR4</td>
<td>Parallel Poll Response 4</td>
<td>U</td>
<td>S</td>
<td>T</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>PPR5</td>
<td>Parallel Poll Response 5</td>
<td>U</td>
<td>S</td>
<td>T</td>
<td>X</td>
<td>X</td>
<td>1</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>PPR6</td>
<td>Parallel Poll Response 6</td>
<td>U</td>
<td>S</td>
<td>T</td>
<td>X</td>
<td>1</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>PPR7</td>
<td>Parallel Poll Response 7</td>
<td>U</td>
<td>S</td>
<td>T</td>
<td>1</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>PPR8</td>
<td>Parallel Poll Response 8</td>
<td>U</td>
<td>S</td>
<td>T</td>
<td>1</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>PPU</td>
<td>Parallel Poll Unconfigure</td>
<td>M</td>
<td>U</td>
<td>C</td>
<td>Y</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>REN</td>
<td>Remote Enable</td>
<td>U</td>
<td>U</td>
<td>C</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>RFD</td>
<td>Ready for Data</td>
<td>U</td>
<td>H</td>
<td>S</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>RQS</td>
<td>Request Service</td>
<td>(Note 9)</td>
<td>U</td>
<td>S</td>
<td>T</td>
<td>1</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>SCG</td>
<td>Secondary Command Group</td>
<td>M</td>
<td>S</td>
<td>E</td>
<td>Y</td>
<td>1</td>
<td>1</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>SDC</td>
<td>Selected Device Clear</td>
<td>M</td>
<td>A</td>
<td>C</td>
<td>Y</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>SPD</td>
<td>Serial Poll Disable</td>
<td>M</td>
<td>U</td>
<td>C</td>
<td>Y</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

Notes:
1. X: Don't Care Bit
2. E: End of String
3. T: Tag Bit
4. S: Secondary Talk Address
5. D: Data Accepted
6. D: Data Valid
7. C: Command Group
8. A: Addressed Command Group
9. E: End of String
10. T: Tag Bit

References:
REMOTE MESSAGE CODING (Continued)

| Mnemonic   | Message Name                       | C   | T | L | D    | D   | N | N | E | S | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | V | D | C | N | I | Q | C | N |
| SPE        | Serial Poll Enable                 |     |   |   |      |     |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| SRQ        | Service Request                    |     |   |   |      |     |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| STB        | Status Byte                        |     |   |   |      |     |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| TCT        | Take Control                       |     |   |   |      |     |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| TAG        | Talk Address Group                 |     |   |   |      |     |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| UCG        | Universal Command Group           |     |   |   |      |     |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| UNL        | Unlisten                           |     |   |   |      |     |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| UNT        | Untalk                             |     |   |   |      |     |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |

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.
   Response = S↑≠↓
   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></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></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.
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 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>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></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></td>
<td>GROUND: This pin must be tied to ground.</td>
</tr>
<tr>
<td>VCC</td>
<td>40</td>
<td></td>
<td>POWER: +5V power input: +5V ± 10%.</td>
</tr>
<tr>
<td>NC</td>
<td>39</td>
<td></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></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>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 asset 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>

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:

<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>X</td>
<td>KPE</td>
<td>CF</td>
<td>DEC</td>
<td>IBF</td>
<td>OBF</td>
<td></td>
</tr>
</tbody>
</table>

FUNCTION:

<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>

3-54
OBF  Output Buffer Full; OBF = 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 Decryp5; indicates whether the DEU is in an encrypt or a decrypt mode. DEC = 1 implies the encrypt mode. DEC = 0 implies the decrypt 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
MSB LSB

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
MSB LSB

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 4. Flowchart for Entering a New Key

**Encrypting or Decrypting Data**

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 OBQ 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.

Figure 5. Encrypting/Decrypting Data
USING SOFTWARE COUNTER

START

1 = 0

NO
IBF = 0?
YES
DATA REGISTER ← 1 DATA BYTE

i = i + 1

NO
i = 8?
YES

READ 1 CODED DATA BYTE

i = i - 1

NO
i = 0?
YES

END

210465-6

USING CF FLAG

START

NO
IBF = 0?
YES

DATA REGISTER ← 1 DATA BYTE

YES

CF = 1?
YES

CF = 0?
END

NO

OBF = 1?
YES

READ 1 CODED DATA BYTE

210465-7

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

Figure 6. Data Conversion Flowcharts

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.
Figure 11. Pacify Protocol

Figure 12. Polling Interface

Figure 13. Single Interrupt Interface

Figure 14. Dual Interrupt Interface

Figure 15. DMA Interface

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:** This is a production data sheet. The specifications are subject to change without notice.

*WARNING: Stressing the device beyond the “Absolute Maximum Ratings” may cause permanent damage. These are stress ratings only. Operation beyond the “Operating Conditions” is not recommended and extended exposure beyond the “Operating Conditions” 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 X1, X2, RESET)</td>
<td>−0.5</td>
<td>0.8</td>
<td>V</td>
</tr>
<tr>
<td>$V_{IL1}$</td>
<td>Input Low Voltage (X1, X2, 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 X1, RESET)</td>
<td>2.0</td>
<td>$V_{CC}$</td>
<td>V</td>
</tr>
<tr>
<td>$V_{IH1}$</td>
<td>Input High Voltage (X1, RESET)</td>
<td>3.5</td>
<td>$V_{CC}$</td>
<td>V</td>
</tr>
<tr>
<td>$V_{IH2}$</td>
<td>Input High Voltage (X2)</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>$V_{OL} = 2.0 mA$</td>
<td></td>
</tr>
<tr>
<td>$V_{OL1}$</td>
<td>Output Low Voltage (All Other Outputs)</td>
<td>0.45</td>
<td>$V_{OL} = 1.6 mA$</td>
<td></td>
</tr>
<tr>
<td>$V_{OH}$</td>
<td>Output High Voltage ($D_0$–$D_7$)</td>
<td>2.4</td>
<td>$V_{OH} = −400 \mu A$</td>
<td></td>
</tr>
<tr>
<td>$V_{OH1}$</td>
<td>Output High Voltage (All Other Outputs)</td>
<td>2.4</td>
<td>$V_{OH} = −50 \mu A$</td>
<td></td>
</tr>
<tr>
<td>$I_{IL}$</td>
<td>Input Leakage Current (RD, WR, CS, A0)</td>
<td>$±10 \mu A$</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>$±10 \mu A$</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>$I_{LI}$</td>
<td>Low Input Load Current (Pins 24, 27–38)</td>
<td>0.3</td>
<td>$V_{IL} = 0.8V$</td>
<td></td>
</tr>
<tr>
<td>$I_{L1}$</td>
<td>Low Input Load Current (RESET)</td>
<td>0.2</td>
<td>$V_{IL} = 0.8V$</td>
<td></td>
</tr>
<tr>
<td>$I_{IH}$</td>
<td>Input High Leakage Current (Pins 24, 27–38)</td>
<td>100</td>
<td>$V_{IN} = V_{CC}$</td>
<td></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>

3-61
**A.C. CHARACTERISTICS** \( T_A = 0^\circ C \text{ to } +70^\circ C, V_{CC} = V_{DD} = +5V \pm 10\%, V_{SS} = 0V \)

### 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 \ 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 \ 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 \ 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></td>
<td>10</td>
<td>ns</td>
</tr>
<tr>
<td>( t_{F} )</td>
<td>Clock Fall Time</td>
<td></td>
<td>10</td>
<td>ns</td>
</tr>
</tbody>
</table>

**NOTE:**
1. \( t_{CY} = 15/l(\text{XTAL}) \)
A.C. TESTING INPUT, OUTPUT WAVEFORM

![Waveform Diagram](image)

WAVEFORMS

READ OPERATION—OUTPUT BUFFER REGISTER

![Read Diagram](image)

WRITE OPERATION—INPUT BUFFER REGISTER

![Write Diagram](image)
DMA AND INTERRUPT TIMING

CLOCK TIMING
Using the 8291A GPIB Talker/Listener
USING THE 8291A GPIB TALKER/LISTENER

CONTENTS

INTRODUCTION .................................. 3-67
OVERVIEW OF IEEE 488/GPIB .............. 3-68
Interface Functions .......................... 3-68
Electrical Signal Lines ....................... 3-68
Transfer Control Lines ...................... 3-70
Bus Commands ............................... 3-70
Addressing Techniques ....................... 3-70
INTEL'S ® GPIB COMPONENTS ............... 3-70
Overview of the 8291A GPIB Talker/Listener ................................................. 3-71
Address Mode ................................ 3-72
APPLICATION OF THE 8291A .......... 3-73
Talker Functions ............................. 3-73
Addressed Talker (via MTA Message) ... 3-73
LISTENER FUNCTIONS ............... 3-73
Addressed Listening (via the MLA Message) ....................................................... 3-73
Remote/Local and Lockout .................... 3-73
Polling ....................................... 3-74
Serial Poll .................................. 3-74
Parallel Poll ................................ 3-76
APPLICATION EXAMPLES ........ 3-77
Two Software Drivers ......................... 3-77
8291A with HP 9835A ......................... 3-77
CONCLUSION .................................. 3-78
REFERENCES .................................. 3-78
APPENDIX A: SYSTEM BLOCK DIAGRAM WITH 8088 .............. 3-79
APPENDIX B: SOFTWARE DRIVERS FOR BLOCK DATA TRANSFER .............. 3-80
APPENDIX C: SOFTWARE FOR HP 9835A ......................... 3-88
APPENDIX D: SOFTWARE FOR HP 8088/HP 9835A VIA GPIB ......................... 3-89
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.
NOTE:
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 (note 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/ion), primary, and secondary.
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.

![Figure 3. 8291A Registers](image-url)
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 I 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 an interrupt will be generated if enabled and 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 (Ion). 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) Ion 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 Ion 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 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 2891A 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 REN (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-
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 finished 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.

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 0FDH 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 OOH 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 8219A 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

---

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.
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 SP2P1 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).

3-76
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/ion 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 timeout 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, EOI is sent true by the talker (BOARD 1) with the last byte of data. Upon detection of EOI, 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. EOI again is sent with the last byte and BOARD 2 switches back to listen-only. BOARD 1 upon detecting EOI 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 μs 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. EOI is then sent true with a false byte of OOH. This false byte is due to the 1975 standard which allows asynchronous sending and reception of EOI. (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 and 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 Change (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 transfer is determined by whether the 8291A is 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 VM 1 Compilat10n of Module Board 1
Object Module Placed In: F1: BD1 OBJ
Compiler Invoked By: . PLM86, F1: BD1. SRC Symbols Medium
/* BOARD 1 TPT PROGRAM */
/* THIS BOARD TALKS TO THE OTHER BOARD BY */
/* TRANSFfERING 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/D 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:
/* PROCEII'URES */
2
3 2 CO: PROCEDURE (XXX):
4
5 DECLARE XXX BYTE.
6 SER$STAT LITERALLY 'OFFF2H',
7 SER$DATA LITERALLY 'OFFFOH',
8 TXRDY LITERALLY 'OIH'.
9
10 UG WHILE (INPUT (SER$STAT) AND TXRDY) <> TXRDY:
11 END:
12 OUTPUT (SER$DATA) = XXX;
13
14 END CO:

/* SETUP BUFFERS */

15 DECLARE BUFF2 (100) BYTE; /* RAM STORAGE AREA */
16 DECLARE BUFF1 (100) BYTE DATA


230832-5
<table>
<thead>
<tr>
<th>Line</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>DECLARE BUFF3(17) BYTE DATA (ODH, OAH, 'COMPARE ERROR', ODH, OAH); /* ROM STORAGE AREA */</td>
</tr>
<tr>
<td>11</td>
<td>DECLARE CLEAR$FF LITERALLY 'OFFDDH'; /* MASTER CLEAR */</td>
</tr>
<tr>
<td></td>
<td>START$0$LO LITERALLY 'OFFDDH',</td>
</tr>
<tr>
<td></td>
<td>START$0$HI LITERALLY 'OFFDDH',</td>
</tr>
<tr>
<td></td>
<td>DSCOUNT$LO LITERALLY 'OFFDD1H',</td>
</tr>
<tr>
<td></td>
<td>DSCOUNT$HI LITERALLY 'OFFDD1H',</td>
</tr>
<tr>
<td></td>
<td>SET$MODE LITERALLY 'OFFDDBH',</td>
</tr>
<tr>
<td></td>
<td>CMD$37 LITERALLY 'OFFDDBH',</td>
</tr>
<tr>
<td></td>
<td>SET$MASK LITERALLY 'OFFDDFH',</td>
</tr>
<tr>
<td>12</td>
<td>DECLARE DMA$ADR$TALK POINTER;</td>
</tr>
<tr>
<td></td>
<td>DECLARE DMA$ADR$LSTN POINTER;</td>
</tr>
<tr>
<td>13</td>
<td>DECLARE DMA$ADR$TALK POINTER;</td>
</tr>
<tr>
<td>14</td>
<td>DECLARE RD$TRANSFER LITERALLY '48H',</td>
</tr>
<tr>
<td></td>
<td>WR$TRANSFER LITERALLY '44H',</td>
</tr>
<tr>
<td></td>
<td>NORM$TIME LITERALLY '20H',</td>
</tr>
<tr>
<td></td>
<td>TC$LO LITERALLY 'OFFH',</td>
</tr>
<tr>
<td></td>
<td>TC$HI LITERALLY 'O0H',</td>
</tr>
<tr>
<td></td>
<td>TC$LO2 LITERALLY '990', /* 100 Xfers */</td>
</tr>
<tr>
<td></td>
<td>I BYTE;</td>
</tr>
<tr>
<td>15</td>
<td>DECLARE DMA$WORD$TALK (2) WORD AT (@DMA$ADR$TALK),</td>
</tr>
<tr>
<td></td>
<td>DMA$WORD$LSTN(2) WORD AT (@DMA$ADR$LSTN);</td>
</tr>
<tr>
<td>16</td>
<td>DECLARE PORT$OUT LITERALLY 'OFFCOH'; /* DATA OUT */</td>
</tr>
<tr>
<td></td>
<td>PORT$IN LITERALLY 'OFFCOH';</td>
</tr>
<tr>
<td></td>
<td>STATUS$1 LITERALLY 'OFFC1H'; /* INTR STAT 2 */</td>
</tr>
<tr>
<td></td>
<td>STATUS$2 LITERALLY 'OFFC2H'; /* INTR STAT 2 */</td>
</tr>
<tr>
<td></td>
<td>ADDR$STATUS LITERALLY 'OFFC4H',</td>
</tr>
<tr>
<td></td>
<td>COMMAND$MOD LITERALLY 'OFFC5H'; /* CMD PASS THRU */</td>
</tr>
<tr>
<td></td>
<td>ADDR$0 LITERALLY 'OFFC6H',</td>
</tr>
<tr>
<td></td>
<td>EOS$REG LITERALLY 'OFFC7H'; /* EOS REGISTER */</td>
</tr>
</tbody>
</table>
DECLARE
END$EDI LITERALLY '88H',
DNE LITERALLY '10H',
PON LITERALLY '00H',
RESET LITERALLY '02H',
CLEAR LITERALLY '00H',
DMA$REG$L LITERALLY '10H',
DMA$REG$T LITERALLY '20H',
MOD$I$TO LITERALLY '80H',
MOD$I$LO LITERALLY '40H',
EOS LITERALLY '0DH',
PRESCALER LITERALLY '23H',
HIGH$PEED LITERALLY '04H',
OKAY LITERALLY '0FFFH',
XYZ BYTE.
MATCH WORD.
BO LITERALLY '02H',
BI LITERALLY '01H',
ERR LITERALLY '04H'.
/* CODE BEGINS */
18 1 START91:
OUTPUT (STATUS$2) = CLEAR; /* SHUT-OFF DMA REG BITS TO */
/* PREVENT EXTRA DMA REGS */
/* FROM 8291A */
/* MANIPULATE DMA ADDRESS VARIABLES */
19 1 DMA$ARDS$TALK = (@BUFF1);
20 1 DMA$ARDS$LSTN = (@BUFF2);
21 1 DMA$WRD$STALK(1) = SHL (DMA$WRD$STALK(1), 4);
22 1 DMA$WRD$STALK(0) = DMA$WRD$STALK (0) + DMA$WRD$STALK (1);
23 1 DMA$WRS$LSTN(1) = SHL (DMA$WRS$LSTN (1), 4);
24 1 DMA$WRS$LSTN(0) = DMA$WRS$LSTN (0) + DMA$WRS$LSTN (1);
25 1 INIT371:
/* INIT 8237 FOR TALKER FUNCTIONS */
26 1 OUTPUT (CLEAR$FF) = CLEAR; /* TOGGLE MASTER CLEAR */
27 1 OUTPUT (CMD$37) = NORM$TIME;
28 1 OUTPUT (SET$MODE) = RD$TRANSFER;
29 1 OUTPUT (START$&LO) = DMA$WRS$TALK (0);
30 1 DMA$WRS$TALK (0) = SHR (DMA$WRS$TALK (0), 8);
31 1 OUTPUT (START$&HI) = DMA$WRS$TALK (0);
32 1 OUTPUT (COUNT$&LO) = TC$LO2;
33 1 OUTPUT (COUNT$&HI) = TC$HI2;
/* INIT 8291A FOR TALKER FUNCTIONS */
34 1  OUTPUT (EOS$REG) =EOS;
35 1  OUTPUT (COMMAND$MOD) = END$EOI; /* EO1 ON EOS SENT */
36 1  OUTPUT (ADDR$STATUS) = MOD1$TG; /* TALK ONLY */
37 1  OUTPUT (COMMAND$MOD) = PRESCALER;
38 1  OUTPUT (COMMAND$MOD) = HIGH$SPEED;
39 1  OUTPUT (COMMAND$MOD) = PON;
40 1  DO WHILE (INPUT (STATUS$1) AND B0) = 0;
41 2  END; /* WAIT FOR B0 INTR */
42 1  OUTPUT (PORT$OUT) = GAATH;
43 1  DO WHILE (INPUT (STATUS$1) AND ERR) = ERR;
44 2  DO WHILE (INPUT (STATUS$1) AND B0) = 0;
46 2  END; /* WAIT FOR B0 INTR */
47 2  END;
48 1  OUTPUT (STATUS$2) = DMA$REG$T; /* ENABLE DMA REGS */
49 1  DO WHILE (INPUT (CMD$J7) AND TC) = TC;
50 2  END;
51 1  INIT$37L;
52 1  OUTPUT (STATUS$2) = CLEAR; /* DISABLE DMA REGS */
53 2  /* INIT $237 FOR LISTENER FUNCTIONS */
54 2  OUTPUT (CLEAR$FF) = CLEAR; /* TOGGLE MASTER RESET */
55 1  OUTPUT (CMD$J7) = NORM$TIME;
56 1  OUTPUT (SET$MODE) = WR$TRANSFER;
57 1  OUTPUT (CMD$J7) = CLEAR;
58 1  OUTPUT (START$SOLG) = DMA$WRDLSTN (O);
59 1  OUTPUT (START$SOLG) = DMA$WRDLSTN (O), B;
50 1  OUTPUT (START$SOLH) = DMA$WRDLSTN (O);  
59 1  OUTPUT (START$SOLH) = DMA$WRDLSTN (O);  
60 1  OUTPUT (START$SOLH) = DMA$WRDLSTN (O);  
61 1  OUTPUT (START$SOLH) = DMA$WRDLSTN (O);  
62 1  OUTPUT (START$SOLH) = DMA$WRDLSTN (O);  
63 1  OUTPUT (START$SOLH) = DMA$WRDLSTN (O);  
64 1  DO WHILE (INPUT (STATUS$1) AND BI) = 0;
65 2  END; /* WAIT FOR BI INTR */
66 1  XYZ = INPUT (PORT$IN);
67 1  OUTPUT (STATUS$2) = DMA$REG$T; /* ENABLE DMA REGS */
68 1  DO WHILE (INPUT (STATUS$1) AND DNE) = DNE;
69 2  /* WAIT FOR EOI RECEIVED */
PL/M-66 COMPILER  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 (S)

END OF PL/M-66 COMPILATION  230832-9
AP-166

PL/M-86 COMPILER    BOARD2
ISIS-II PL/M-86 V1.1 COMPILED OF MODULE BOARD2
OBJECT MODULE PLACED IN   FL: BRD2. OBJ
COMPILER INVOKED BY:   PLM86  FL: BRD2. SRC

/* BOARD 2 TPT PROGRAM */
/* */
/* THIS BOARD LISTENS TO THE OTHER 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 GPID */
/* WHEN THE BLOCK IS FINISHED THE 8291A IS */
/* PROGRAMMED BACK INTO THE LISTENER MODE */

1 BOARD2.

DO:
/* 237 PORT ADDRESSES */
2 1 DECLARE
CLEAR$FF      LITERALLY 'OFFDDH',    /*MASTER CLEAR */
START$0$LO    LITERALLY 'OFFDOH',
START$0$HI    LITERALLY 'OFFDOH',
O$COUNT$LO    LITERALLY 'OFFD1H',
O$COUNT$HI    LITERALLY 'OFFD1H',
SET$MODE      LITERALLY 'OFFD8H',
CMD$37        LITERALLY 'OFFD8H',
SET$MASK      LITERALLY 'OFFD8H',
/* 237 COMMAND - DATA BYTES */

3 1 DECLARE
RD$TRANSFER   LITERALLY '48H',
WR$TRANSFER   LITERALLY '44H',
ADDR$1$A      LITERALLY '00H',
ADDR$1$0      LITERALLY '01H',
NORM$TIME     LITERALLY '20H',
TC$LO$1       LITERALLY '06H',
TC$HI$1       LITERALLY '00H',
TC$LO$2       LITERALLY '99H',
TC$HI$2       LITERALLY '00H',
TC            LITERALLY '01H',
/* 8291A PORT ADDRESSES */

4 1 DECLARE
PORT$OUT      LITERALLY 'OFFCOH',
PORT$IN       LITERALLY 'OFFCOH',    /* DATA IN */
STATUS$1      LITERALLY 'OFFC1H',    /* INTR STAT 1 */
STATUS$2      LITERALLY 'OFFC2H',    /* INTR STAT 2 */
ADDR$STATUS   LITERALLY 'OFFC4H',    /* ADDR STAT */
COMMAND$MOD   LITERALLY 'OFFCSH',    /* CMD PASS THRU */
PL/M-86 COMPILER ROAD02

ADDR$0  LITERALLY 'OFFC6H'
EOS$REG LITERALLY 'OFFC7H'. /* EOS REGISTER */
/* 8291A COMMAND - DATA BYTES */

5 1
DECLARE
END$EOI LITERALLY '88H'
DNE LITERALLY '10H'
FON LITERALLY '00H'
RESET LITERALLY '02H'
CLEAR LITERALLY '00H'
DMA$REG$L LITERALLY '10H'
DMA$REG$T LITERALLY '20H'
MODI$TO LITERALLY '80H'
MODI$LO LITERALLY '40H'
EOS LITERALLY '0D' H
PRESCALER LITERALLY '23H'
HIGH$SPEED LITERALLY 'A4H'
XYZ BYTE
B0 LITERALLY '02H'
B1 LITERALLY '01H'
ERR LITERALLY '04H'

6 1
START91.

OUTPUT (STATUS$2) =CLEAR; /* END INITIALIZATION STATE */

/* INIT 8237 FOR LISTENER FUNCTION */

7 1
INIT37L:

OUTPUT (CLEAR$FF) =CLEAR; /* TOGGLE MASTER RESET */
OUTPUT (CHD$FF) =NORM$TIME
OUTPUT (SET$MODE) =WR$TRANSFER, /* BLOCK XFER MODE */
OUTPUT (SET$MASK) =CLEAR,
OUTPUT (START$O$LO) =ADDR$IA;
OUTPUT (START$O$HI) =ADDR$IB;
OUTPUT (C$COUNT$LO) =TC$LO;
OUTPUT (C$COUNT$HI) =TC$HI;

/* INIT 8291A FOR LISTENER FUNCTIONS */

8 1
OUTPUT (COMMAND$MOD) =RESET;
9 1
OUTPUT (ADDR$STATUS) =MODI$LO;
10 1
OUTPUT (COMMAND$MOD) =FON;
11 1
DO WHILE (INPUT (STATUS$1) AND B1) =0;
12 2
END; /* WAIT FOR BI INTR */
13 1
XYZ= INPUT (PORT$IN);
14 1
OUTPUT (STATUS$2) =DMA$REG$L;

/* WAIT UNTIL EDI RCV'D AND END INTR-BIT SET */

DO WHILE (INPUT (STATUS$1) AND DNE ) <= DNE;

22 1

3-86
FL/M-8e COMPILER BOARD2

23 1 END;

24 1 INIT37:
    /* INIT 8237 FOR TALKER FUNCTION */
    OUTPUT (STATUS$2) =CLEAR; /* CLEAR 8291A DRQ */
25 1 OUTPUT (CLEAR$FF) =CLEAR;
26 1 OUTPUT (CMD$37) =NORM$TIME;
27 1 OUTPUT (SET$MODE) =RD$TRANSFER; /* BLOCK XFER MODE */
28 1 OUTPUT (SET$MASK) =CLEAR;
29 1 OUTPUT (START$S%LO) =ADDR$IA;
30 1 OUTPUT (START$S%HI) =ADDR$IB;
31 1 OUTPUT (O$COUNT$LO) =TC$L02;
32 1 OUTPUT (O$COUNT$HI) =TC$HI2;
    /* INIT 8291A FOR TALKER FUNCTION */
33 1 OUTPUT (EOS$REG) =EOS;
34 1 OUTPUT (COMMAND$MOD) =END$EOI; /* EOI ON EOS SENT */
35 1 OUTPUT (ADDR$STATUS) =MOD1$TO; /* TALK ONLY */
36 1 OUTPUT (COMMAND$MOD) =PRESCALER;
37 1 OUTPUT (COMMAND$MOD) =HIGH$SPEED;
38 1 OUTPUT (COMMAND$MOD) =PON;
39 1 DO WHILE (INPUT (STATUS$1) AND BO) =0;
40 2 END; /* WAIT FOR BO INTR */
41 1 OUTPUT (PORT$OUT) =0AAH;
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) =0AAH;
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

<table>
<thead>
<tr>
<th>Code Area Size</th>
<th>Code Area Size</th>
<th>Constant Area Size</th>
<th>Constant Area Size</th>
<th>Variable Area Size</th>
<th>Variable Area Size</th>
<th>Maximum Stack Size</th>
<th>Maximum Stack Size</th>
<th>152 Lines Read</th>
<th>0 Program Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>0122H</td>
<td>2900</td>
<td>0000H</td>
<td>0000H</td>
<td>0001H</td>
<td>1D</td>
<td>0000H</td>
<td>0000H</td>
<td>152</td>
<td>0</td>
</tr>
</tbody>
</table>

230832-12
APPENDIX C
SOFTWARE FOR HP 9835A

10 REM SEND IN
20 REM BEGIN EXECUTION
30 REM FOR LOADING
40 REM UNTIL END
50 REM FOR DEBUG
60 REM FOR TESTING
70 REM FOR DEBUGGING
80 REM FOR TESTING
90 REM FOR TESTING
100 REM FOR TESTING
110 REM FOR TESTING
120 REM FOR TESTING
130 REM FOR TESTING
140 REM FOR TESTING
150 REM FOR TESTING
160 REM FOR TESTING
170 REM FOR TESTING
180 REM FOR TESTING
190 REM FOR TESTING
200 REM FOR TESTING
210 REM FOR TESTING
220 REM FOR TESTING
230 REM FOR TESTING
240 REM FOR TESTING
250 REM FOR TESTING
260 REM FOR TESTING
270 REM FOR TESTING
280 REM FOR TESTING
290 REM FOR TESTING
300 REM FOR TESTING
310 REM FOR TESTING
320 REM FOR TESTING
330 REM FOR TESTING
340 REM FOR TESTING
350 REM FOR TESTING
360 REM FOR TESTING
370 REM FOR TESTING
380 REM FOR TESTING
390 REM FOR TESTING
400 REM FOR TESTING
410 REM FOR TESTING
420 REM FOR TESTING
430 REM FOR TESTING
440 REM FOR TESTING
450 REM FOR TESTING
460 REM FOR TESTING
470 REM FOR TESTING
480 REM FOR TESTING
490 REM FOR TESTING
500 REM FOR TESTING
510 REM FOR TESTING
520 REM FOR TESTING
530 REM FOR TESTING
540 REM FOR TESTING
550 REM FOR TESTING
560 REM FOR TESTING
570 REM FOR TESTING
580 REM FOR TESTING
590 REM FOR TESTING
600 REM FOR TESTING
610 REM FOR TESTING
620 REM FOR TESTING
630 REM FOR TESTING
640 REM FOR TESTING
650 REM FOR TESTING
660 REM FOR TESTING
670 REM FOR TESTING
680 REM FOR TESTING
690 REM FOR TESTING
700 REM FOR TESTING
710 REM FOR TESTING
720 REM FOR TESTING
730 REM FOR TESTING
740 REM FOR TESTING
750 REM FOR TESTING
760 REM FOR TESTING
770 REM FOR TESTING
780 REM FOR TESTING
790 REM FOR TESTING
800 REM FOR TESTING
810 REM FOR TESTING
820 REM FOR TESTING
830 REM FOR TESTING
840 REM FOR TESTING
850 REM FOR TESTING
860 REM FOR TESTING
870 REM FOR TESTING
880 REM FOR TESTING
890 REM FOR TESTING
900 REM FOR TESTING
910 REM FOR TESTING
920 REM FOR TESTING
930 REM FOR TESTING
940 REM FOR TESTING
950 REM FOR TESTING
960 REM FOR TESTING
970 REM FOR TESTING
980 REM FOR TESTING
990 REM FOR TESTING

3-88
APPENDIX D
SOFTWARE FOR HP 8088/HP 9835A VIA GPIB

PL/M-B6 COMPILER HPIB

ISIS-II PL/M-B6 VI.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   1 DECs
    ADDRSHI LITERALLY '01H',
    ADDRLO LITERALLY '00H',
    ADSC LITERALLY '01H',
    BI LITERALLY '01H',
    BO LITERALLY '02H',
    CHARCOUNT BYTE,
    CHAR BYTE,
    CHAR(BO) BYTE,
    CLEAR LITERALLY '00H',
    CPT LITERALLY '00H',
    CRLF LITERALLY '0AH',
    DEC LITERALLY '08H',
    DMAADR@LSTN POINTER,
    DMAADR@TALK POINTER,
    DMA@RDSLSN(2) WORD AT (EDA@RDSLSN),
    DMA@RDSLTALK(2) WORD AT (EDA@RDSLTALK),
    DMA@REG# LITERALLY '10H',
    DMA@REG#T LITERALLY '20H',
    DNE LITERALLY '10H',
    END@EOI LITERALLY 'BBH',
    EOG LITERALLY '0DH',
    ERR LITERALLY '04H',
    GET LITERALLY '20H',
    I BYTE,
    LISTEN LITERALLY '04H',
    MLA LITERALLY '04H',
    MODE# LITERALLY '01H',
    NGDMA LITERALLY '00H',
    NGRSV LITERALLY '02H',
    NGRTIME LITERALLY '20H',
    PON LITERALLY '00H',
    PPC LITERALLY '05H',
    PPE@MASL LITERALLY '60H',
    PPOLL@CNF#FLAG LITERALLY '01H',
    PPOLL@EN#BYTE BYTE,
    PRSBUF(BO) BYTE AT (@CHARS),
    RD@XFER LITERALLY '4BH',
    RESET LITERALLY '02H',
    REMC LITERALLY '02H',
    RSV LITERALLY '04H',
    RXRDY LITERALLY '02H',
SROS  LITERALLY '40H',
STAT1  BYTE,
STAT2  BYTE,
TALK   LITERALLY '02H',
TABORLA BYTE,
TRG    LITERALLY '41H',
TC     LITERALLY '01H',
TC#HI  LITERALLY '00H',
TC#LO  LITERALLY '0FFH',
TXRDY  LITERALLY '01H',
UDC    BYTE,
WR#XFER LITERALLY '44H',
XYZ    BYTE;
/*/ 
PORT DECLARATIONS
*/

3 1 DECLARE
ADDR#0  LITERALLY '0FFC6H',
ADDR#STATUS LITERALLY '0FFC4H',
CLEAR#FF LITERALLY '0FFDDH',
CMD#37   LITERALLY '0FFC3H',
COMMAND#MOD LITERALLY '0FFC3H',
COUNT#HI LITERALLY '0FF11H',
COUNT#LO LITERALLY '0FFD1H',
CPTS#REG LITERALLY '0FFC5H',
EOSS#REG LITERALLY '0FFC7H',
PORT#IN  LITERALLY '0FFC0H',
PORT#OUT LITERALLY '0FFC0H',
SER#DATA LITERALLY '0FFF0H',
SER#STAT LITERALLY '0FFF2H',
SET#MASK LITERALLY '0FFDFH',
SET#MODE LITERALLY '0FFDFH',
SPOLL#STAT LITERALLY '0FFC3H',
START#HI LITERALLY '0FFD0H',
START#LO LITERALLY '0FFD0H',
STATUS#1 LITERALLY '0FFC1H',
STATUS#2 LITERALLY '0FFC2H';

/* crt messages list */

4 1 DECLARE GET#MSG(11) BYTE DATA (ODH, OAH, 'TRIGGER', OAH, ODH);
5 1 DECLARE DECS#MSG(16) BYTE DATA (ODH, OAH, 'DEVICE CLEAR', OAH, ODH);
6 1 DECLARE REMCS#SQ(10) BYTE DATA (ODH, OAH, 'REMOTE', OAH, ODH);
7 1 DECLARE CPTS#MSG(22) BYTE DATA (ODH, OAH, 'UNDEF CMD RECEIVED', OAH, ODH);
8 1 DECLARE HUH#MSG(11) BYTE DATA (ODH, OAH, 'HUH ???', OAH, ODH);

/* called procedures */

9 1 REOSER:    PROCEDURE;

230832-17
PL/M-86 Compiler

HPIB

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 2
CO: PROCEDURE(XXX);

16 2
DECLARE
XXX BYTE;

17 2
DO WHILE (INPUT (SER*STAT) AND TXRDY)<TXRDY;

18 2
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:

32 3
CHAR=(INPUT (SER*DATA) AND 7FH);

33 3
CALL CO (CHAR);

34 3
CHARS(I)=CHAR;

35 3
I=I+1;

36 3
IF CHAR <> CRLF THEN

37 3
DO;

38 4
DO WHILE (INPUT (SER*STAT) AND RXRDY)<RXRDY;

39 5
END;

40 4
GOTO STORE*CHAR;

41 4
END;

42 3
CALL REGSER;

43 3
END;

44 2
END CI;

45 1
TALK$EXEC: PROCEDURE;

46 2
OUTPUT (STATUS$2)=CLEAR;

/*
 manipulate address bits for DMA controller
 */

47 2
DMA$ADR*TALK=(CHARS);

48 2
DMA$WRD*TALK(1)=SHL(DMA$WRD*TALK(1),4);

49 2
DMA$WRD*TALK(0)=DMA$WRD*TALK(0)+DMA$WRD*TALK(1);

50 2
OUTPUT (CLEAR$FF)=CLEAR;
PL/M-86 COMPILER

51  2  OUTPUT (CMD37)=NORM$TIME;
52  2  OUTPUT (SET$MODE)=RD$XFER;
53  2  OUTPUT (SET$MASK)=CLEAR;
54  2  OUTPUT (START$LO)=DMA$WRD$TALK(0);
55  2  DMA$WRD$TALK(0)=SHR(DMA$WRD$TALK(0),8);
56  2  OUTPUT (START$HI)=DMA$WRD$TALK(0);
57  2  OUTPUT (COUNT$LO)=CHAR$COUNT;
58  2  OUTPUT (COUNT$HI)=0;

59  2  OUTPUT (EDS$REG)=EDS;
60  2  OUTPUT (COMMAND$MOD)=END$EOI;
61  3  DO WHILE (INPUT (STATUS$1) AND BO)=0;
62  3  END;
63  2  OUTPUT (PORT$OUT)=OAAH;

64  2  DO WHILE (INPUT (STATUS$1) AND ERR)=ERR;
65  3  DO WHILE (INPUT (STATUS$1) AND BO)=0;
66  4  END;
67  3  OUTPUT (PORT$OUT)=OAAH;
68  3  END;
69  2  OUTPUT (STATUS$2)=DMA$REG$T;
70  2  END TALK$EXEC;

LISTEN$EXEC:  PROCEDURE;

71  2  OUTPUT (STATUS$2)=CLEAR;
72  2  OUTPUT (CLEAR$FF)=CLEAR;
73  2  OUTPUT (CMD$37)=NORM$TIME;
74  2  OUTPUT (SET$MODE)=WR$XFER;
75  2  OUTPUT (SET$MASK)=CLEAR;
76  2  DMA$ADR$LN$1=8CHARS;
77  2  DMA$WRD$LN$1=SHL(DMA$WRD$LN$1,4);
78  2  DMA$WRD$LN$0=DMA$WRD$LN$0+DMA$WRD$LN$1;
79  2  DMA$WRD$LN(0)=DMA$WRD$LN(0)+DMA$WRD$LN(1);
80  2  DMA$WRD$LN(0)=SHR(DMA$WRD$LN(0),8);
81  2  DMA$WRD$LN(0)=SHR(DMA$WRD$LN(0),8);
82  2  DMA$WRD$LN(0)=DMA$WRD$LN(0);
83  2  DMA$WRD$LN(0)=DMA$WRD$LN(0);
84  2  DMA$WRD$LN(0)=DMA$WRD$LN(0);
85  2  DMA$WRD$LN(0)=DMA$WRD$LN(0);
86  2  END LISTEN$EXEC;

PRINT$EXEC:  PROCEDURE;

87  2  I=0;
88  2  DO PRI$BUF(I)<CRLF;
89  3  CALL CO (PRI$BUF(I));
90  3  I=I+1;
91  3  END;
92  3  CALL CO (PRI$BUF(I));
93  2  END PRINT$EXEC;
94  2  END PRINT$EXEC;

230832-19
ADSC*EXEC: PROCEDURE;

TA*OR*LA=INPUT (ADDR*STATUS);

IF (TA*OR*LA AND TALK) = TALK THEN
CALL TALK*EXEC;

IF (TA*OR*LA AND LISTEN) = LISTEN THEN
CALL LISTEN*EXEC;

END ADSC*EXEC;

GET*EXEC: PROCEDURE;
DO I=0 TO 10;
CALL CO (GET*MSG(I));
END;
END GET*EXEC;

DEC*EXEC: PROCEDURE;
DO I=0 TO 15;
CALL CO (DEC*MSG(I));
END;
END DEC*EXEC;

REMC*EXEC: PROCEDURE;
DO I=0 TO 9;
CALL CO (REMC*MSG(I));
END;
END REMC*EXEC;

PPOLL*CON: PROCEDURE;
OUTPUT (COMMAND*MOD)=PPOLL*CNFG*FLAG;
END PPOLL*CON;

PPOLL*EN: PROCEDURE;
PPOLL*EN*BYTE=(UDC AND 6FHI);
OUTPUT (COMMAND*MOD)=PPOLL*EN*BYTE;
END PPOLL*EN;

CPT*EXEC: PROCEDURE;
DO I=0 TO 21;
CALL CO (CPT*MSG(I));
END;

UDC=INPUT (CPT*REG);
UDC=(UDC AND 7FHI);
IF (UDC AND PPC) = PPC THEN
CALL PPOLL*CON;

IF (UDC AND PPE*MASK) = PPE*MASK THEN
CALL PPOLL*EN;

230832-20
END CPTEXEC;

BEGIN CODE

INIT:
OUTPUT (CLEAR#FF) =CLEAR;
OUTPUT (COMMAND#MOD) =RESET;
OUTPUT (ADDR#STATUS) =MODE#1;
OUTPUT (ADDR#O) =MLA;
OUTPUT (STATUS#2) =NO#DMA;
OUTPUT (COMMAND#MOD) =PON;

LISTENERS:

/* response to listeners check */

DO WHILE (INPUT (STATUS#1) AND BI)=O;
END;

XYZ=INPUT (PORT#IN);
XYZ=INPUT (STATUS#2);

RDSTAT:

/* read status registers and interpret command */

STAT1=INPUT (STATUS#1);
STAT2=INPUT (STATUS#2);

IF (STAT1 AND DNE)=DNE THEN
CALL PRINTER;

IF (STAT1 AND CPT)=CPT THEN
DO:
CALL CPTEXEC;
STAT2=(STAT2 AND OFEH);
END;

IF (STAT1 AND GET)=GET THEN
DO:
CALL GETEXEC;
STAT2=(STAT2 AND OFEH);
END;

IF (STAT1 AND DEC)=DEC THEN
DO:
CALL DECEEXEC;
STAT2=(STAT2 AND OFEH);
END;

IF (STAT2 AND REMC)=REMC THEN
DO:
CALL REMCEEXEC;
STAT2=(STAT2 AND OFEH);
END;

IF (STAT2 AND ADSC)=ADSC THEN
PL/M-B6 COMPILER

170 1 DD;
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
<table>
<thead>
<tr>
<th>CONTENTS</th>
<th>PAGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>INTRODUCTION</td>
<td>3-98</td>
</tr>
<tr>
<td>GPIB/IEEE 488 OVERVIEW</td>
<td>3-98</td>
</tr>
<tr>
<td>HARDWARE ASPECTS OF THE SYSTEM</td>
<td>3-104</td>
</tr>
<tr>
<td>8291 Talker/Listener</td>
<td>3-104</td>
</tr>
<tr>
<td>8292 Controller</td>
<td>3-105</td>
</tr>
<tr>
<td>8293 Bus Transceivers</td>
<td>3-105</td>
</tr>
<tr>
<td>ZT7488/18 GPIB Controller</td>
<td>3-108</td>
</tr>
<tr>
<td>8292 COMMAND DESCRIPTION</td>
<td>3-110</td>
</tr>
<tr>
<td>SOFTWARE DRIVER OUTLINE</td>
<td>3-114</td>
</tr>
<tr>
<td>Initialization</td>
<td>3-115</td>
</tr>
<tr>
<td>Talker/Listener</td>
<td>3-115</td>
</tr>
<tr>
<td>Send Data</td>
<td>3-115</td>
</tr>
<tr>
<td>Receive Data</td>
<td>3-117</td>
</tr>
<tr>
<td>Transfer Data</td>
<td>3-119</td>
</tr>
<tr>
<td>Controller</td>
<td>3-119</td>
</tr>
<tr>
<td>Trigger</td>
<td>3-119</td>
</tr>
<tr>
<td>Device Clear</td>
<td>3-120</td>
</tr>
<tr>
<td>Serial Poll</td>
<td>3-120</td>
</tr>
<tr>
<td>Parallel Poll</td>
<td>3-121</td>
</tr>
<tr>
<td>Pass Control</td>
<td>3-123</td>
</tr>
<tr>
<td>Receive Control</td>
<td>3-124</td>
</tr>
<tr>
<td>Service Request</td>
<td>3-126</td>
</tr>
<tr>
<td>System Controller</td>
<td>3-126</td>
</tr>
<tr>
<td>Remote</td>
<td>3-126</td>
</tr>
<tr>
<td>Local</td>
<td>3-126</td>
</tr>
<tr>
<td>Interface Clear/Abort</td>
<td>3-127</td>
</tr>
<tr>
<td>INTERRUPT AND DMA CONSIDERATIONS</td>
<td>3-127</td>
</tr>
<tr>
<td>APPLICATION EXAMPLE</td>
<td>3-128</td>
</tr>
<tr>
<td>CONCLUSION</td>
<td>3-129</td>
</tr>
<tr>
<td>APPENDIX A</td>
<td>3-130</td>
</tr>
<tr>
<td>Source Listings</td>
<td>3-130–3-147</td>
</tr>
<tr>
<td>APPENDIX B</td>
<td>3-148</td>
</tr>
<tr>
<td>Test Cases for the Software Drivers</td>
<td>3-148</td>
</tr>
<tr>
<td>APPENDIX C</td>
<td>3-154</td>
</tr>
<tr>
<td>Remote Message Coding</td>
<td>3-154</td>
</tr>
</tbody>
</table>
INTRODUCTION

The Intel® 8292 is a preprogrammed UPITM-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 1 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. D101 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 is not yet ready for the next data or control byte. 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—Acceptor 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) **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) **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) **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) **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) **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) **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) **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 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™ 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
2) Device A Secondary Address (if any)
3) Universal Unlisten
4) Device B Primary (Listen) Address
   Device B Secondary Address (if any)
   Device C Primary (Listen) Address
   etc.
5) First Data Byte
   Second Data Byte
   ...
   Last Data Byte (EOI)

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 A Primary (Listen) Address
   Device A 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
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.
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 Talk-only, Listen-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 Acceptor 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.
Figure 10. 8291 Registers

Figure 11. DMA/Interrupt GPIB Controller Block Diagram
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</th>
<th>PORT #</th>
<th>WRITE TO 8292</th>
</tr>
</thead>
<tbody>
<tr>
<td>INTERRUPT STATUS</td>
<td>69H</td>
<td>1 1 1 OP C C C C</td>
</tr>
<tr>
<td>SYC ERR SRQ EV X IFCR IBF OBF D7 D0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ERROR FLAG*</td>
<td>66H</td>
<td>1</td>
</tr>
<tr>
<td>X X USER X X TOUT3 TOUT2 TOUT1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CONTROLLER STATUS*</td>
<td>66H</td>
<td>1</td>
</tr>
<tr>
<td>CSBS CA X X SYCS IFC REN SRQ D7 D0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>GP1B (BUS) STATUS*</td>
<td>66H</td>
<td>0 0 0 USER 0 0 TOUT4 TOUT3 TOUT1</td>
</tr>
<tr>
<td>REN DAV EOI X SYC IFC ANTI SRQ</td>
<td></td>
<td></td>
</tr>
<tr>
<td>EVENT COUNTER STATUS*</td>
<td>66H</td>
<td>D D D D D D D D</td>
</tr>
<tr>
<td>D D D D D D D D</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TIME OUT STATUS*</td>
<td>66H</td>
<td>D D D D D D D D</td>
</tr>
<tr>
<td>66H *Note: These registers are accessed by a special utility command.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure 12. 8292 Registers
LOAD ERROR MASK

This command loads the Error Mask with D7–0. 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., TNDDV1 + 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 = 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.

RCST—Read Controller Status Register
(Command = 0E6H)
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.

**RTOUT**—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, DAV 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 μs (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 μs 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.

**STCNI—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.

**SPCNI—Stop Counter Interrupts**  
(Command = 0F0H)

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 arbitrarily sets ATN TRUE and ECI TRUE. Note that this action may cause devices on the bus to lose a data byte or cause them to interpret a data byte as a command byte. Both Actions can result in anomalous behavior. TCAS 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 INITIALIZATION</th>
</tr>
</thead>
<tbody>
<tr>
<td>Talker/Listener</td>
</tr>
<tr>
<td>SEND SEND DATA</td>
</tr>
<tr>
<td>RECV RECEIVE DATA</td>
</tr>
<tr>
<td>XFER TRANSFER DATA</td>
</tr>
<tr>
<td>Controller</td>
</tr>
<tr>
<td>TRIG GROUP EXECUTE TRIGGER</td>
</tr>
<tr>
<td>DCLR DEVICE CLEAR</td>
</tr>
<tr>
<td>SPOL SERIAL POLL</td>
</tr>
<tr>
<td>PPEN PARALLEL POLL ENABLE</td>
</tr>
<tr>
<td>PPDS PARALLEL POLL DISABLE</td>
</tr>
<tr>
<td>PPUN PARALLEL POLL UNCONFIGURE</td>
</tr>
<tr>
<td>PPOL PARALLEL POLL</td>
</tr>
<tr>
<td>PCTL PASS CONTROL</td>
</tr>
<tr>
<td>RCTL RECEIVE CONTROL</td>
</tr>
<tr>
<td>SRQD SERVICE REQUESTED</td>
</tr>
<tr>
<td>System Controller</td>
</tr>
<tr>
<td>REME REMOTE ENABLE</td>
</tr>
<tr>
<td>LOCL LOCAL</td>
</tr>
<tr>
<td>IFCL 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 |
| Enable-8291 | ;In controller usage, the 8291 |
| Disable major address | ;Is set to talk only and/or listen only |
| Disable minor address | ;Talk only is our rest state |
| ton | ;3 MHz in this ap note example |
| Clock frequency | ;Releases 8291 from init. state |
| All interrupts off | |
| 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).

This routine assumes a non-null listener list in that it always sends Universal 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.
SEND:
Output-to-8291 MTA, UNL
Put EOS into 8291
While 20H ≤ 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)
(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 flowchart 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 ≤ talker ≤ 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 lon
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

3-118
TRANSFER DATA

\textit{XFER} \textless Talker \textgreater \textless Listener list \textgreater \textless EOS \textgreater

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.

\begin{verbatim}
XFER:
Output-to-8291: Talker, UNL
While 20H \leq \text{listen} \leq 3EH
Output-to-8291: Listener
Increment listen list pointer
Enable-8291
\textit{ion}, no \textit{ton}
Continuous AH mode
End on EOS received
Immediate execute \textit{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
\textit{ton}
Immediate execute \textit{pon}
Return
\end{verbatim}

Controller

GROUP EXECUTE TRIGGER

\textit{TRIG} \textless Listener list \textgreater

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.

\begin{verbatim}
TRIG:
Output-to-8291 UNL
While 20H \leq \text{listener} \leq 3EH
Output-to-8291 Listener
Increment listen list pointer
Output-to-8291 GET
Return
\end{verbatim}

;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

;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.

<table>
<thead>
<tr>
<th>DCLR:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Output-to-8291 UNL</td>
</tr>
<tr>
<td>While 20H ≤ listener ≤ 3EH</td>
</tr>
<tr>
<td>Output-to-8291 Listener</td>
</tr>
<tr>
<td>Increment listen list pointer</td>
</tr>
<tr>
<td>Output-to-8291 SDC</td>
</tr>
<tr>
<td>Return</td>
</tr>
</tbody>
</table>

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.
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 XXXXIP3P2Pl as defined by the IEEE Std. P3P2Pl 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:**
Output-to-829l UNL ;Universal unlisten
While 20H ≤ Listener ≤ 3EH ;Check for valid listener
Output-to-829l listener ;Stop old listener, address new
Output-to-829l PPC, (PPE or data) ;Send parallel poll info
Increment listener list pointer ;Point to next listener
Increment buffer pointer ;One configuration byte per listener
Return

**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:**
Output-to-829l UNL ;Universal Unlisten
While 20H ≤ Listener ≤ 3EH ;Check for valid listener
Output-to-829l listener ;Address listener
Increment listener list pointer
Output-to-829l PPC, PPD ;Disable PP on all listeners
Return

---

**Figure 23. PPDS “1”, “+”, “>”**

**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.

**PPUN:**
Output-to-8291 PPU
Return ;Unconfigure all parallel poll

CONDUCT A PARALLEL POLL

PPOL

This system command causes the controller to conduct a Parallel Poll on the GPIB for approximately 12.5 \( \mu \)sec (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.

**PPOL:**
Enable-8291
lon ;Listen only
Immediate execute pon ;This resets ton
Output-to-8292 EXFP ;Execute parallel poll
Upon BI ;When byte is input
Input-from-8291 data ;Read it
Enable-8291
ton ;Talk only
Immediate execute pon ;This resets lon
Return Data (status byte)

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.

**PCTL:**
If 40H \( \leq \) talker \( \leq \) 5EH then
if talker \( < \) MTA then
output-to-8291 talker, TCT ;Cannot pass control to myself
Enable-8291 ;Take control message to talker
not ton, not lon ;Set up 8291 as normal device
Immediate execute pon
My device address, mode 1
Undefined command pass through (Parallel Poll Configuration)
Output-to-8292 GIDL ;Optional use of PP
Return ;Put controller in idle

3-123
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
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 28. REME

Figure 27. RCTL
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.

SRQD:
If SRQ then ;Test 92 status bit
Output-to-8292 IACK.SRQ ;Acknowledge it
Return SRQ
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.

REME:
Output-to-8292 SREM . ;8292 asserts remote enable line
Return

LOCAL

LOCL

This system command deasserts the REN line on the GPIB. The devices will go local immediately.

LOCL:
Output-to-8292 SLOC ;8292 stops asserting remote enable
Return
INTERFACE 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 ;8292 asserts Interface Clear Return ;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 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.
### APPENDIX A

**ISIS-II 8088/8085 MACHO ASSEMBLER, V3.0**  
**GPIB CONTROLLER SUBROUTINES**

**LOC ONJ**  
**LINE**  
**SOURCE STATEMENT**

1  
**STITLE('GPIB CONTROLLER SUBROUTINES')**  
2  
3  
**GPIB CONTROLLER SUBROUTINES**  
4  
5  
6  
**for Intel 8291, 8292 on ZT 7488/18**  
7  
8  
**H Bert Forbes, Ziatech Corporation**  
9  
10  
11  
**San Luis Obispo, CA, USA 93401**

#### 1000

15  
**ORG 1000H ; For ZT7488/18 w/8085**

**0669**  
17 **PRT91 EQU 60H ;8291 base port #1**

18  
19  
**Reg 0 Data in & Data out**

20  
21  
**DIN EQU PRT91+0 ;91 Data in reg**

22  
23  
**DOUT EQU PRT91+0 ;91 Data out reg**

**066A**  
24  
25  
**Reg 1 Interrupt 1 Constants**

26  
27  
**INT1 EQU PRT91+1 ;INT Reg 1**

28  
29  
**INTM1 EQU PRT91+1 ;INT Mask Reg, 1**

30  
31  
**INM EQU 02 ;91 80 INTRP Mask**

32  
33  
**INM EQU 01 ;91 81 INTRP Mask**

**066B**  
34  
35  
**Reg 2 Interrupt 2**

36  
37  
**INT2 EQU PRT91+2**

38  
39  
**Reg 3 Interrupt 3**

40  
41  
**ADRST EQU PRT91+4 ;reg #4**

42  
43  
**ADRM EQU PRT91+4 ;91 address mode register #4**

44  
45  
**ADR EQU PRT91+4 ;address mode register #4**

**066C**  
46  
47  
**TON EQU 80H ;91 talk only mode & not listen only**

48  
49  
**TON EQU 80H ;91 listen only & not ton**

**066D**  
50  
51  
**TLDN EQU 0CBH ;91 talk & listen only**

52  
53  
**MODE1 EQU 01 ;mode 1 addressing for device**

54  
55  
**MODEL EQU 01 ;mode 1 addressing for device**

**066E**  
56  
57  
**Reg 4 (Read) Address Status Register**

58  
59  
**ADRS EQU PRT91+4 ;reg #4**

60  
61  
**ADRS EQU PRT91+4 ;reg #4**

**066F**  
62  
63  
**REG 5 (Write) Auxiliary Mode Register**

64  
65  
**TMD EQU PRT91+4 ;91 auxiliary mode register #5**

**0670**  
66  
67  
**TMD EQU PRT91+4 ;91 auxiliary mode register #5**

**0671**  
68  
69  
**CLKRT EQU 23H ;91 3 Mhz clock input**

---

231324-30

---

3-130
<table>
<thead>
<tr>
<th>Address</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PRT91+5</td>
<td>CPTNG</td>
<td>Command Pass Through Enable</td>
</tr>
<tr>
<td>PRT91+6</td>
<td>8292</td>
<td>Base Port 1 (CS7)</td>
</tr>
<tr>
<td>PRT92+0</td>
<td>INTRP</td>
<td>INTRP Mask Reg</td>
</tr>
<tr>
<td>PRT92+2</td>
<td>INTM</td>
<td>INTM Reg</td>
</tr>
<tr>
<td>PRT92+4</td>
<td>ERRR4</td>
<td>Error Mask Reg</td>
</tr>
<tr>
<td>PRT92+6</td>
<td>TOUF1</td>
<td>Time Out for Pass Control</td>
</tr>
<tr>
<td>PRT92+8</td>
<td>TOUFE2</td>
<td>Time Out for Standby</td>
</tr>
<tr>
<td>PRT92+10</td>
<td>EVREG</td>
<td>Event Counter Pseudo Reg</td>
</tr>
<tr>
<td>PRT92+12</td>
<td>TOREG</td>
<td>Time Out Pseudo Reg</td>
</tr>
<tr>
<td>91</td>
<td>CMD92</td>
<td>Command Register 92</td>
</tr>
<tr>
<td>93</td>
<td>INTST</td>
<td>Interrupt Status Reg</td>
</tr>
<tr>
<td>94</td>
<td>EVBIT</td>
<td>Event Counter Bit</td>
</tr>
<tr>
<td>95</td>
<td>IBFRT</td>
<td>Input Buffer Full Bit</td>
</tr>
<tr>
<td>96</td>
<td>SMOFT</td>
<td>Seg bit</td>
</tr>
<tr>
<td>98</td>
<td>8292</td>
<td>CONTROL VALUES</td>
</tr>
<tr>
<td>100</td>
<td>EAPLG</td>
<td>Error Flag Pseudo Reg</td>
</tr>
<tr>
<td>102</td>
<td>BUSST</td>
<td>Status Pseudo Reg</td>
</tr>
<tr>
<td>104</td>
<td>EVCST</td>
<td>Event Counter Status Pseudo Reg</td>
</tr>
<tr>
<td>106</td>
<td>TOGST</td>
<td>Time Out Status Pseudo Reg</td>
</tr>
<tr>
<td>108</td>
<td>8292</td>
<td>OPERATION COMMANDS</td>
</tr>
<tr>
<td>110</td>
<td>SPCHI</td>
<td>Stop Counter Interrupts</td>
</tr>
<tr>
<td>112</td>
<td>GIDL</td>
<td>Go to Idle</td>
</tr>
<tr>
<td>114</td>
<td>RSET</td>
<td>Reset</td>
</tr>
<tr>
<td>116</td>
<td>RSTI</td>
<td>Reset Interrupts</td>
</tr>
<tr>
<td>118</td>
<td>SEC</td>
<td>Goto standby, enable counting</td>
</tr>
<tr>
<td>120</td>
<td>EPPX</td>
<td>Execute parallel poll</td>
</tr>
<tr>
<td>122</td>
<td>GTSH</td>
<td>Goto standby</td>
</tr>
<tr>
<td>124</td>
<td>SLOE</td>
<td>Set local mode</td>
</tr>
<tr>
<td>126</td>
<td>SREM</td>
<td>Set interface to remote</td>
</tr>
<tr>
<td>128</td>
<td>ABORT</td>
<td>Abort all operation, clear interface</td>
</tr>
<tr>
<td>130</td>
<td>TCNTR</td>
<td>Take control(R2ceive control)</td>
</tr>
<tr>
<td>132</td>
<td>TCASY</td>
<td>Take control asynchronously</td>
</tr>
<tr>
<td>134</td>
<td>TCSY</td>
<td>Take control synchronously</td>
</tr>
<tr>
<td>136</td>
<td>STCH</td>
<td>Start counter interrupts</td>
</tr>
<tr>
<td>Line</td>
<td>Code</td>
<td>Description</td>
</tr>
<tr>
<td>------</td>
<td>------</td>
<td>-------------</td>
</tr>
<tr>
<td>123</td>
<td>8292</td>
<td>UTILITY COMMANDS</td>
</tr>
<tr>
<td>124</td>
<td></td>
<td></td>
</tr>
<tr>
<td>125</td>
<td></td>
<td></td>
</tr>
<tr>
<td>126</td>
<td>WOUT</td>
<td>EQU $0E1H; Write to timeout reg</td>
</tr>
<tr>
<td>127</td>
<td>WEVC</td>
<td>EQU $0E2H; Write to event counter</td>
</tr>
<tr>
<td>128</td>
<td>REVC</td>
<td>EQU $0E3H; Read event counter status</td>
</tr>
<tr>
<td>129</td>
<td>RERF</td>
<td>EQU $0E4H; Read error flag reg</td>
</tr>
<tr>
<td>130</td>
<td>RMHM</td>
<td>EQU $0E5H; Read interrupt mask reg</td>
</tr>
<tr>
<td>131</td>
<td>RCST</td>
<td>EQU $0E6H; Read controller status reg</td>
</tr>
<tr>
<td>132</td>
<td>RRST</td>
<td>EQU $0E7H; Read GPIB Bus status reg</td>
</tr>
<tr>
<td>133</td>
<td>RTOUT</td>
<td>EQU $0E8H; Read timeout status reg</td>
</tr>
<tr>
<td>134</td>
<td>RERH</td>
<td>EQU $0E9H; Read error mask reg</td>
</tr>
<tr>
<td>135</td>
<td>IACK</td>
<td>EQU $0AH; Interrupt Acknowledge</td>
</tr>
<tr>
<td>136</td>
<td></td>
<td></td>
</tr>
<tr>
<td>137</td>
<td></td>
<td></td>
</tr>
<tr>
<td>138</td>
<td></td>
<td></td>
</tr>
<tr>
<td>139</td>
<td></td>
<td></td>
</tr>
<tr>
<td>140</td>
<td></td>
<td></td>
</tr>
<tr>
<td>141</td>
<td></td>
<td></td>
</tr>
<tr>
<td>142</td>
<td>PRTF</td>
<td>EQU PRT91+5FH; 2749A port 6F for interrupts</td>
</tr>
<tr>
<td>143</td>
<td>TCIF</td>
<td>EQU $02H; Task complete interrupt</td>
</tr>
<tr>
<td>144</td>
<td>SPIF</td>
<td>EQU $04H; Special interrupt</td>
</tr>
<tr>
<td>145</td>
<td>OBFF</td>
<td>EQU $08H; 92 Output (to CPU) Buffer full</td>
</tr>
<tr>
<td>146</td>
<td>IBFF</td>
<td>EQU $10H; 92 Input (from CPU) Buffer empty</td>
</tr>
<tr>
<td>147</td>
<td>80F</td>
<td>EQU $11H; 91 Int line (8D in this case)</td>
</tr>
<tr>
<td>148</td>
<td></td>
<td></td>
</tr>
<tr>
<td>149</td>
<td></td>
<td></td>
</tr>
<tr>
<td>150</td>
<td></td>
<td></td>
</tr>
<tr>
<td>151</td>
<td>MDA</td>
<td>EQU 1; My device address is 1</td>
</tr>
<tr>
<td>152</td>
<td>MTA</td>
<td>EQU MDA+40H; My talk address is 1 (&quot;A&quot;)</td>
</tr>
<tr>
<td>153</td>
<td>MLA</td>
<td>EQU MDA+20H; My listen address is 1 (&quot;B&quot;)</td>
</tr>
<tr>
<td>154</td>
<td>UML</td>
<td>EQU 3FH; Universal unlisten</td>
</tr>
<tr>
<td>155</td>
<td>GET</td>
<td>EQU $06; Group Execute Trigger</td>
</tr>
<tr>
<td>156</td>
<td>SDC</td>
<td>EQU $04H; Device Clear</td>
</tr>
<tr>
<td>157</td>
<td>SPE</td>
<td>EQU $18H; Serial poll enable</td>
</tr>
<tr>
<td>158</td>
<td>SPD</td>
<td>EQU $19H; Serial poll disable</td>
</tr>
<tr>
<td>159</td>
<td>PPC</td>
<td>EQU $05; Parallel poll configure</td>
</tr>
<tr>
<td>160</td>
<td>PPD</td>
<td>EQU $70H; Parallel poll disable</td>
</tr>
<tr>
<td>161</td>
<td>PPE</td>
<td>EQU $60H; Parallel poll disable</td>
</tr>
<tr>
<td>162</td>
<td>PPU</td>
<td>EQU $15H; Parallel poll unconfigured</td>
</tr>
<tr>
<td>163</td>
<td>TCT</td>
<td>EQU $09; Take control (pass control)</td>
</tr>
<tr>
<td>164</td>
<td></td>
<td></td>
</tr>
<tr>
<td>165</td>
<td></td>
<td></td>
</tr>
<tr>
<td>166</td>
<td></td>
<td></td>
</tr>
<tr>
<td>167</td>
<td></td>
<td></td>
</tr>
<tr>
<td>168</td>
<td>SETF</td>
<td>MACRO; Sets flags on A register</td>
</tr>
<tr>
<td>169</td>
<td>ORA</td>
<td>A</td>
</tr>
<tr>
<td>170</td>
<td>ENDM</td>
<td></td>
</tr>
<tr>
<td>171</td>
<td></td>
<td></td>
</tr>
<tr>
<td>172</td>
<td>WAITO</td>
<td>MACRO; Wait for last 91 byte to be done</td>
</tr>
<tr>
<td>173</td>
<td>WALT</td>
<td>LOCAL WAILT; Get INTI status</td>
</tr>
<tr>
<td>174</td>
<td>ANI</td>
<td>BOM; Check for byte out</td>
</tr>
<tr>
<td>175</td>
<td>JZ</td>
<td>WAILT; If not, try again</td>
</tr>
<tr>
<td>176</td>
<td>ENDM</td>
<td></td>
</tr>
<tr>
<td>177</td>
<td></td>
<td></td>
</tr>
<tr>
<td>178</td>
<td></td>
<td></td>
</tr>
<tr>
<td>179</td>
<td></td>
<td></td>
</tr>
<tr>
<td>180</td>
<td></td>
<td></td>
</tr>
<tr>
<td>181</td>
<td>WAITI</td>
<td>MACRO; Wait for 91 byte to be input</td>
</tr>
<tr>
<td>182</td>
<td>LOCAL</td>
<td>WAILT</td>
</tr>
<tr>
<td>183</td>
<td>WAILT</td>
<td>IN INT1; Get INTI status</td>
</tr>
<tr>
<td>184</td>
<td>MOV</td>
<td>B,A; Save status in B</td>
</tr>
<tr>
<td>185</td>
<td>ANI</td>
<td>BIM; Check for byte in</td>
</tr>
<tr>
<td>186</td>
<td>JZ</td>
<td>WAILT; If not, just try again</td>
</tr>
<tr>
<td>187</td>
<td>ENDM</td>
<td></td>
</tr>
<tr>
<td>188</td>
<td></td>
<td></td>
</tr>
<tr>
<td>189</td>
<td>WAITX</td>
<td>MACRO; Wait for 92's TCI to go false</td>
</tr>
<tr>
<td>190</td>
<td>LOCAL</td>
<td>WAILT</td>
</tr>
<tr>
<td>191</td>
<td>WAITI</td>
<td>IN PRTF</td>
</tr>
<tr>
<td>192</td>
<td>ANI</td>
<td>TCIF</td>
</tr>
<tr>
<td>193</td>
<td>JZ</td>
<td>WAILT</td>
</tr>
<tr>
<td>194</td>
<td>ENDM</td>
<td></td>
</tr>
<tr>
<td>195</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

231324-32
AP-66

196 WAITT MACRO
197 LOCAL WAITL
198 WAITL: IN PORTF ;Get task complete int, etc.
199 ANI TCIF ;Mask it
200 JZ WAITL ;Wait for task to be complete
201 ENDM

202 RANGE MACRO LOWE, UPPER, LABEL
203 ;Checks for value in range
204 ;Branches to label if not
205 ;in range. Falls through if
206 ;lower <= (H)(L) <= upper.
207 ;Get next byte.
208 MOV A,M
209 CPI LOWER
210 JM LABEL
211 CPI UPPER+1
212 JP LABEL
213 ENDM

214

215 ;
216 CLRA MACRO A
217 XRA A ;A XOR A = 0
218 ENDM

219 ;
220 ;
221 ; All of the following routines have these common
222 ; assumptions about the state of the 8291 & 9292 upon entry
223 ; to the routine and will exit the routine in an identical state.
224 ;
225 ; 8291: B0 is or has been set,
226 ; All interrupts are masked off
227 ; TON mode, not LA
228 ; No holdoffs in effect or enabled
229 ; No holdoffs waiting for finish command
230 ;
231 ; 8292: ATN asserted (active controller)
232 ; note: RCTL is an exception--- it expects
233 ; to not be active controller
234 ; Any previous task is complete & 92 is
235 ; ready to receive next command.
236 ; 8855: Pointer registers (DF, HL) end one
237 ; beyond last legal entry
238 ;*********************************************************************
239 ;
240 ; INITIALIZATION ROUTINE
241 ;
242 ; INPUTS: None
243 ; OUTPUTS: None
244 ; CALLS: None
245 ; DESTROYS: A, F

1087 3EAA
1088 D348
1089 3E60
1090 D364
1091 3E68
1092 D364
1093 3E89
1094 D348
1095 3E23
1096 D355
1097 3EA0
1098 D348
1099 3E60
1100 D364
1101 3E68
1102 D348
1103 3E23
1104 D355
1105 AF
1106 D341
1107 D342
1108 D355
1109 C9

246 ;*********************************************************************
247 ; SEND ROUTINE

231324-33

3-133
270 ;
271 ;
272 ; 
273 ; INPUTS:  
274 ; b EOS character-- software detected  
275 ;
276 ; OUTPUTS: none  
277 ;
278 ; DESTROYS: A, C, DE, HL, F  
279 ;
280 ;
281 ;
181C 3E41  
181E D360  
1820 D611  
1822 EF92  
1824 CA2B18  
1827 3E3F  
1829 3D68  
1828 7B  
182C D57  
182D 7E  
182F FE28  
1831 PA4710  
1834 FE3F  
1836 F24710  
1839 DB61  
183E EB92  
183D CA9110  
1840 7E  
1841 3D5B  
1843 23  
1844 3C2E10  
1855 DB6F  
185E EB92  
185A C25510  
185D 6B5F  
185F EB92  
1861 CA5D10  
1864 79  
1865 B7  
1865 CA8B10  
1869 1A  
186A 3D5B  
186C 88  
1872 SEND: MOV A,M ;Send MTA to turn off any  
1873 OUT DOUT ;previous talker  
1878 WAITO  
1879 IN INT1 ;Get Int1 status  
187A JZ ??9991 ;If not, try again  
187B MVI A,UNL ;Send universal unlisten  
187D OUT DOUT ;to stop previous listeners  
1882 MOV A,B ;Get EOS character  
1883 OUT EOS ;Output it to $291  
188A ;while listener....  
188C RANGE 20H,3EH ;SEND2 ;Check next listen address  
189C ;Checks for value in range  
189D ;branches to label if not  
189E ;in range. Falls through if  
189F ;lower <= (H)(L) <= upper.  
18A0 ;Get next byte.  
18A3 MOV A,M  
18A5 CPI 28H  
18A8 JM SEND2  
18A9 CPI 3EH+1  
18AC JP SEND2  
18AD WAITO ;Wait for previous listener sent  
18AF IN INT1 ;Get Int1 status  
18B1 ANI BOM ;Check for byte out  
18B3 JZ ??9992 ;If not, try again  
18B4 MOV A,M ;Get this listener  
18B6 OUT DOUT ;Output to GPIB  
18B8 INX H ;Increment listener list pointer  
18BA JMP SEND1 ;Loop till non-valid listener  
18BB ;Enable 91 ending conditions  
18BC WAITO ;Wait for last addr accepted  
18BD IN INT1 ;Get Int1 status  
18BE ANI BOM ;Check for byte out  
18BF JZ ??9993 ;If not, try again  
18C0 MOV A,M ;Get this listener  
18C2 OUT EOS ;Send EOS with EOS character  
18C4 AUXD  
18C5 WAITX ;Wait for TCI to go false  
18C6 IN PRTF  
18C8 JNZ ??9994  
18C9 WAITT ;Wait for TCI on GTSb  
18CA IN PRTF ;Get task complete int,etc.  
18CD JNZ ??9995 ;Wait for task to be complete  
18CE JM SENDD  
18D0 MOV A,C ;Get count  
18D1 SETF A  
18D2 MOV A,A  
18D3 MOV A,AX #EOS ;Test EOS...this is faster  
18D4 MOV A,AX #EOS  
18D5 MOV A,AX #EOS  
18D6 MOV A,AX #EOS  
18D7 MOV A,AX #EOS  
18D8 MOV A,AX #EOS  
18D9 MOV A,AX #EOS  
18DA MOV A,AX #EOS  
18DB MOV A,AX #EOS  
18DC MOV A,AX #EOS  
18DD MOV A,AX #EOS  
18DE MOV A,AX #EOS  
18DF MOV A,AX #EOS  
18E0 MOV A,AX #EOS  
18E1 MOV A,AX #EOS  
18E2 MOV A,AX #EOS  
18E3 MOV A,AX #EOS  
18E4 MOV A,AX #EOS  
18E5 MOV A,AX #EOS  
18E6 MOV A,AX #EOS  
18E7 MOV A,AX #EOS  
18E8 MOV A,AX #EOS  
18E9 MOV A,AX #EOS  
18EA MOV A,AX #EOS  
18EB MOV A,AX #EOS  
18EC MOV A,AX #EOS  
18ED MOV A,AX #EOS  
18EE MOV A,AX #EOS  
18EF MOV A,AX #EOS  
18F0 MOV A,AX #EOS  
18F1 MOV A,AX #EOS  
18F2 MOV A,AX #EOS  
18F3 MOV A,AX #EOS  
18F4 MOV A,AX #EOS  
18F5 MOV A,AX #EOS  
18F6 MOV A,AX #EOS  
18F7 MOV A,AX #EOS  
18F8 MOV A,AX #EOS  
18F9 MOV A,AX #EOS  
18FA MOV A,AX #EOS  
18FB MOV A,AX #EOS  
18FC MOV A,AX #EOS  
18FD MOV A,AX #EOS  
18FE MOV A,AX #EOS  
18FF MOV A,AX #EOS  

231324-34
If char = EOS, go finish

Get IntI status

Check for byte out

if not, try again

Increment buffer pointer

Decrement count

if count < 6, go send

Else go finish

for consistency

This ensures that the standard entry

Get IntI status

Check for byte out

if not, try again

A,BC;'DE,HL'

A=0 normal termination--EOS detected

A=40 Error--- count overrun

A<>4A or A>8EH Error--- bad talk address

;Get next byte.

;Checks for value in range

;branches to label if not

;in range. Falls through if

;lower <= (H) (L) <= upper.

;Get next byte.
10C2 3E21 416  
10C4 D348 417  
10C6 3E86 418  
10C8 D355 419  
10CA 3661 420  
10CC D361 421  
10CE CACA10 423  
10D1 3E48 424  
10D3 D364 425  
10D5 AF 427  
10D6 D365 428  
10D8 3EF6 429  
10DA D349 430  
10DC D36F 432  
10DE 3E82 433  
10EE 20C10 434  
10E3 D36F 435  
10E5 E582 437  
10E6 D851 439  
10EC 47 440  
10ED E610 441  
10EF C28511 442  
10F2 78 443  
10F3 E681 444  
10F5 CAE10 445  
10F8 DB68 446  
10FA 12 447  
10FB 13 448  
10FD C2EA10 450  
1100 6640 451  
1102 C31711 452  
1105 7A 454  
1106 E581 455  
1108 C28110 456  
1108 D861 457  
110D C38110 458  
1110 D868 459  
1112 12 460  
1113 13 461  
1114 8D 462  
1115 8600 463  
1117 3EFD 465  
1119 D369 466  
111A D66F 468  
111D E582 469  
111F C28111 470  
1122 D86F 472  
1124 E582 474  
1126 CA2211 475  
1129 3E88 480  
112A D345 481  
112C 3E80 482  
112D 3E88 483  
1131 3E63 484  
1133 D345 485  
1135 AF 487  
1136 D365 488  
1138 78 489  
1139 C9 490  

; For completeness
; Get Int status
; Check for byte out
; if not, try again
; Listen only
; Immediate XEQ PON
; A XOR A =
; Goto standby
; Wait for TCI=
; Wait for TCI=1
; Get task complete int, etc.
; Mask it
; Get 91 int status (END &/or BI)
; Check for EOS or EO1
; go wait for BI
; NO, retrieve status &
; check for BI
; NO, go wait for either END or BI
; YES, BI --- get data
; Store it in buffer
; Increment buffer pointer
; Decrement counter
; If count < > 0 go back & wait
; Else set error indicator
; And go take control
; Retrieve status
; Check for BI
; If BI then go input data
; Else wait for last BI
; In loop
; Get data byte
; Store it in buffer
; Incr data pointer
; Decrement count, but ignore it
; Set normal completion indicators
; Take control synchronously
; Wait for TCI=0 (7 tcy)
; Wait for TCI=1
; Get task complete int, etc.
; Mask it
; Wait for task to be complete
; if timeout 3 is to be checked, the above WAITT should be omitted and the appropriate code to look for TCI or TOUT3 inserted here.
; Pattern to clear 91 END conditions
; This bit pattern already in "A"
; Output TON
; Finish handshake
; Immediate execute PON-Reset LON
; Get completion character

; For completeness
; Get Int status
; Check for byte out
; if not, try again
; Listen only
; Immediate XEQ PON
; A XOR A =
; Goto standby
; Wait for TCI=
; Wait for TCI=1
; Get task complete int, etc.
; Mask it
; Get 91 int status (END &/or BI)
; Check for EOS or EO1
; go wait for BI
; NO, retrieve status &
; check for BI
; NO, go wait for either END or BI
; YES, BI --- get data
; Store it in buffer
; Increment buffer pointer
; Decrement counter
; If count < > 0 go back & wait
; Else set error indicator
; And go take control
; Retrieve status
; Check for BI
; If BI then go input data
; Else wait for last BI
; In loop
; Get data byte
; Store it in buffer
; Incr data pointer
; Decrement count, but ignore it
; Set normal completion indicators
; Take control synchronously
; Wait for TCI=0 (7 tcy)
; Wait for TCI=1
; Get task complete int, etc.
; Mask it
; Wait for task to be complete
; if timeout 3 is to be checked, the above WAITT should be omitted and the appropriate code to look for TCI or TOUT3 inserted here.
; Pattern to clear 91 END conditions
; This bit pattern already in "A"
; Output TON
; Finish handshake
; Immediate execute PON-Reset LON
; Get completion character
491 ;
492 ;*******************************************************************************
493 ;
494 ;
495 ; INPUTS: HL device list pointer
496 ; B EOS character
497 ;
498 ; OUTPUTS: None
499 ;
500 ; CALLS: None
501 ;
502 ;
503 ;
504 ; DESTROYS: A, HL, F
505 ;
506 ;
507 ;
508 ;
509 ;
510 ;
511 ; XFER: RANGE 4AH,5EH,XFER4 ;Check for valid talker
512 ;
513 ;
514 ;
515 ;
516 ;
517 ; 113A 7E 517+ MOV A,M
518 ; 113B PE40 518+ CPI 40H
519 ; 113D FABB1 519+ JM XFER4
520 ; 1140 FE5F 520+ CPI 5EH+1
521 ; 1142 FBO11 521+ JP XFER4
522 ; 1145 D360 522+ OUT DOUT ;Send it to GPIB
523 ; 1147 23 523+ INX H ;incr pointer
524 ; 1148 DB61 524+ WAITO
525 ; 114A E602 525+790017: IN INT1 ;Get Int interrupt status
526 ; 114C CA011 526+ ANI BOM ;Check for byte out
527 ; 114F 3E01 527+ JZ 790017 ;If not, try again
528 ; 1151 D360 528+ MV1 A,UNL ;Universal unlisten
529 ; 1152 XFER1: RANGE 2AH,3EH,XFER2 ;Check for valid listener
530 ;
531 ;
532 ;
533 ;
534 ;
535 ;
536 ; 1153 7E 536+ MOV A,M
537 ; 1154 PE20 537+ CPI 20H
538 ; 1156 FAGC11 538+ JM XFER2
539 ; 1159 FE3F 539+ CPI 3EH+1
540 ; 115B F2611 540+ JP XFER2
541 ; 115E DB61 541+ WAITO
542 ; 1160 E602 542+79B018: IN INT1 ;Get Int interrupt status
543 ; 1162 CB611 543+ ANI BOM ;Check for byte out
544 ; 1165 7E 544+ CPI A,M ;Get listener
545 ; 1166 D360 545+ OUT DOUT
546 ; 1168 23 546+ INX H ;incr pointer
547 ; 1169 C5311 547+ JMP XFER1 ;Loop until non-valid listener
548 ; 116C DB61 548+ WAITO
549 ; 1170 E602 549+79B019: IN INT1 ;Get Int interrupt status
550 ; 1172 CB611 550+ ANI BOM ;Check for byte out
551 ; 1175 D365 551+ JJ JZ 790019 ;If not, try again
552 ; 1177 3E01 552+ OUT A,UNL ;Listen only
553 ; 1179 D364 553+ OUT ADMD
554 ; 117B AP 554+ CLRA A ;A XOR A = 0
555 ; 117C D365 555+ OUT AUXMD ;Issued, XEQ PON
556 ; 117E 78 556+ MOV A,B ;Get EOS
557 ; 117F D367 557+ OUT EOSR ;Output it to 91
558 ; 1181 3EF6 558+ MV1 A,ADMD ;Go to standby
559 ; 1183 D369 559+ OUT CMD92

231324-37

3-137
<table>
<thead>
<tr>
<th>Line</th>
<th>Instruction</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>564</td>
<td>WAITX</td>
<td></td>
</tr>
<tr>
<td>565+</td>
<td></td>
<td>;Wait for UCS</td>
</tr>
<tr>
<td>566</td>
<td></td>
<td>Get task complete, etc.</td>
</tr>
<tr>
<td>567+</td>
<td></td>
<td>;Mask it</td>
</tr>
<tr>
<td>568</td>
<td></td>
<td>;Wait for task to be complete</td>
</tr>
<tr>
<td>569+</td>
<td></td>
<td>;Get task complete, etc.</td>
</tr>
<tr>
<td>570+</td>
<td></td>
<td>;Mask it</td>
</tr>
<tr>
<td>571+</td>
<td></td>
<td>;Wait for task to be complete</td>
</tr>
<tr>
<td>572</td>
<td></td>
<td>;Get task complete, etc.</td>
</tr>
<tr>
<td>573+</td>
<td></td>
<td>;Mask it</td>
</tr>
<tr>
<td>574+</td>
<td></td>
<td>;Wait for task to be complete</td>
</tr>
<tr>
<td>575+</td>
<td></td>
<td>;Take control synchronously</td>
</tr>
<tr>
<td>576</td>
<td>D569</td>
<td>OUT CM982</td>
</tr>
<tr>
<td>577</td>
<td></td>
<td>;Wait</td>
</tr>
<tr>
<td>578+</td>
<td>D869</td>
<td>IN PRTF</td>
</tr>
<tr>
<td>579+</td>
<td></td>
<td>;Get task complete, etc.</td>
</tr>
<tr>
<td>580+</td>
<td></td>
<td>;Mask it</td>
</tr>
<tr>
<td>581+</td>
<td></td>
<td>;Wait for TCI</td>
</tr>
<tr>
<td>582+</td>
<td></td>
<td>;Get task complete, etc.</td>
</tr>
<tr>
<td>583+</td>
<td></td>
<td>;Mask it</td>
</tr>
<tr>
<td>584+</td>
<td></td>
<td>;Wait for task to be complete</td>
</tr>
<tr>
<td>585+</td>
<td></td>
<td>;Wait for task to be complete</td>
</tr>
<tr>
<td>586+</td>
<td></td>
<td>;Mask it</td>
</tr>
<tr>
<td>587+</td>
<td></td>
<td>;Finish handshake</td>
</tr>
<tr>
<td>588+</td>
<td></td>
<td>;Send universal unlisten</td>
</tr>
<tr>
<td>589+</td>
<td></td>
<td>;Send universal unlisten</td>
</tr>
<tr>
<td>590+</td>
<td></td>
<td>;Send universal unlisten</td>
</tr>
<tr>
<td>591+</td>
<td></td>
<td>;Send universal unlisten</td>
</tr>
<tr>
<td>592+</td>
<td></td>
<td>;Send universal unlisten</td>
</tr>
<tr>
<td>593+</td>
<td></td>
<td>;Send universal unlisten</td>
</tr>
<tr>
<td>594</td>
<td></td>
<td>;Send universal unlisten</td>
</tr>
<tr>
<td>595</td>
<td></td>
<td>;Send universal unlisten</td>
</tr>
<tr>
<td>596</td>
<td></td>
<td>;Send universal unlisten</td>
</tr>
<tr>
<td>597+</td>
<td></td>
<td>;Send universal unlisten</td>
</tr>
<tr>
<td>598+</td>
<td></td>
<td>;Send universal unlisten</td>
</tr>
<tr>
<td>599+</td>
<td></td>
<td>;Send universal unlisten</td>
</tr>
<tr>
<td>600+</td>
<td></td>
<td>;Send universal unlisten</td>
</tr>
<tr>
<td>601+</td>
<td></td>
<td>;Send universal unlisten</td>
</tr>
<tr>
<td>602+</td>
<td></td>
<td>;Send universal unlisten</td>
</tr>
<tr>
<td>603+</td>
<td></td>
<td>;Send universal unlisten</td>
</tr>
<tr>
<td>604+</td>
<td></td>
<td>;Send universal unlisten</td>
</tr>
<tr>
<td>605+</td>
<td></td>
<td>;Send universal unlisten</td>
</tr>
<tr>
<td>606+</td>
<td></td>
<td>;Send universal unlisten</td>
</tr>
<tr>
<td>607+</td>
<td></td>
<td>;Send universal unlisten</td>
</tr>
<tr>
<td>608+</td>
<td></td>
<td>;Send universal unlisten</td>
</tr>
<tr>
<td>609+</td>
<td></td>
<td>;Send universal unlisten</td>
</tr>
<tr>
<td>610+</td>
<td></td>
<td>;Send universal unlisten</td>
</tr>
<tr>
<td>611+</td>
<td></td>
<td>;Send universal unlisten</td>
</tr>
<tr>
<td>612+</td>
<td></td>
<td>;Send universal unlisten</td>
</tr>
<tr>
<td>613+</td>
<td></td>
<td>;Send universal unlisten</td>
</tr>
<tr>
<td>614+</td>
<td></td>
<td>;Send universal unlisten</td>
</tr>
<tr>
<td>615+</td>
<td></td>
<td>;Send universal unlisten</td>
</tr>
<tr>
<td>616+</td>
<td></td>
<td>;Send universal unlisten</td>
</tr>
<tr>
<td>617+</td>
<td></td>
<td>;Send universal unlisten</td>
</tr>
<tr>
<td>618+</td>
<td></td>
<td>;Send universal unlisten</td>
</tr>
<tr>
<td>619+</td>
<td></td>
<td>;Send universal unlisten</td>
</tr>
<tr>
<td>620+</td>
<td></td>
<td>;Send universal unlisten</td>
</tr>
<tr>
<td>621+</td>
<td></td>
<td>;Send universal unlisten</td>
</tr>
<tr>
<td>622+</td>
<td></td>
<td>;Send universal unlisten</td>
</tr>
<tr>
<td>623+</td>
<td></td>
<td>;Send universal unlisten</td>
</tr>
<tr>
<td>624+</td>
<td></td>
<td>;Send universal unlisten</td>
</tr>
<tr>
<td>625+</td>
<td></td>
<td>;Send universal unlisten</td>
</tr>
<tr>
<td>626+</td>
<td></td>
<td>;Send universal unlisten</td>
</tr>
<tr>
<td>627+</td>
<td></td>
<td>;Send universal unlisten</td>
</tr>
<tr>
<td>628+</td>
<td></td>
<td>;Send universal unlisten</td>
</tr>
<tr>
<td>629+</td>
<td></td>
<td>;Send universal unlisten</td>
</tr>
<tr>
<td>630+</td>
<td></td>
<td>;Send universal unlisten</td>
</tr>
<tr>
<td>631+</td>
<td></td>
<td>;Send universal unlisten</td>
</tr>
<tr>
<td>632+</td>
<td></td>
<td>;Send universal unlisten</td>
</tr>
<tr>
<td>633+</td>
<td></td>
<td>;Send universal unlisten</td>
</tr>
<tr>
<td>634+</td>
<td></td>
<td>;Send universal unlisten</td>
</tr>
<tr>
<td>635+</td>
<td></td>
<td>;Send universal unlisten</td>
</tr>
<tr>
<td>636+</td>
<td></td>
<td>;Send universal unlisten</td>
</tr>
<tr>
<td>637+</td>
<td></td>
<td>;Send universal unlisten</td>
</tr>
</tbody>
</table>
AP-66

3-139

11EB CAE411 638+ JI 770826 ;If not, try again
11EB C9 639 RET
640 ;
641 ;*******************************************************************
642 ;
643 ;DEVICE CLEAR ROUTINE
644 ;
645 ;
646 ;
647 ;INPUTS: HL listener pointer
648 ;OUTPUT: None
649 ;CALLS: None
650 ;DESTROYS: A, HL, F
651 ;
652 DCLR: MVI A, UNL
653 OUT DOUT
654 DCLR1: RANGE 28H, 3EH, DCLR2
655 ;Checks for value in range
656 ;branches to label if not
657 ;in range. Falls through if
658 ;lower <= (H)(L) <= upper.
659 ;Get next byte.

11FF 7E 660+ MOV A, M
11FF FA9912 661+ CPI 28H
11FF FE3F 662+ JP DCLR2
11FF F29912 663+ CPI 3EH+1
11FF DCLR1 664+ OUT DOUT
665 WAITO

11FB DB61 666+ 770827: IN INT1 ;Get Intel status
11FD E682 667+ ANI BOM ;Check for byte out
11FF CA9011 668+ JZ 770827 ;If not, try again
1203 7E 669 MOV A, M
1203 D360 670+ OUT DOUT ;Send listener to GPIB
1205 23 671 INX H
1206 CA9111 672 JMP DCLR1
1209 D641 673 DCLR2: WAITO
120B E682 674+ 770828: IN INT1 ;Get Intel status
120D CA9912 675+ ANI BOM ;Check for byte out
1210 3E84 676+ JZ 770828 ;If not, try again
1212 D360 677 MVI A, BDC ;Send device clear
1212 D360 678 OUT DOUT ;To all addressed listeners
1214 DB61 679 WAITO
1216 E682 680+ 770829: IN INT1 ;Get Intel status
1218 CA1412 681+ ANI BOM ;Check for byte out
121A C9 682+ JZ 770829 ;If not, try again
121B C9 683 RET
684 ;
685 ;*******************************************************************
686 ;
687 ;** SERIAL POLL ROUTINE **
688 ;
689 ;INPUTS: HL talker list pointer
690 ;OUTPUTS: DE status buffer pointer
691 ;CALLS: Fills buffer pointed to by DE
692 ;DESTROYS: A, BC, DE, HL, F
693 ;
694 ;
695 SPOL: MVI A, UNL ;Universal unlisten
696 D360 697 OUT DOUT
698 WAITO
1220 E682 699+ 770830: IN INT1 ;Get Intel status
1222 CA9912 700+ ANI BOM ;Check for byte out
1224 CA2912 701+ JZ 770830 ;If not, try again
1227 3E21 702 MVI A, MLA ;My listen address
1229 D360 703 OUT DOUT
122B DB61 704+ 770831: IN INT1 ;Get Intel status
122D E682 705+ ANI BOM ;Check for byte out
122F CA2912 706+ JZ 770831 ;If not, try again
1232 3E18 707 MVI A, SPE ;Serial poll enable
1234 D360 708 OUT DOUT ;To be formal about it
1236 DB61 709 WAITO
1238 DB61 710+ 770832: IN INT1 ;Get Intel status

231324-39
1238 E682 711+ ANI 80H ;Check for byte out
1239 CA3612 712+ JZ ??#032 ;If not, try again
1239 SPOL: RANGE 48H,58H,SPOL2 ;Check for valid talker
1239 714+ ;Checks for value in range
1239 715+ ;Branches to label if not
1239 716+ ;In range. Falls through if
1239 717+ ;lower <= (H)(L) <= upper.
1239 718+ ;Get next byte.
1239 719+ MOV A,M
1239 720+ CPI #88H
1239 721+ JM SPOL2
1239 722+ CPI #89H+1
1239 723+ JP SPOL2
1239 724+ MOV A,M
1239 725 OUT DOUT ;Send to GPIB
1239 726 INX H ;Inc talker label pointer
1239 727 MVI A,LOV ;Listen only
1239 728 OUT ADRMD
1239 729 WATIP ;Wait for talk address to complete
1239 730+ ??#033; IN INT1 ;Get INT1 status
1239 731+ ANI 80H ;Check for byte out
1239 732+ JZ ??#033 ;If not, try again
1239 733 CLRA
1239 734+ XRA A ;Pattern for immediate XEQ PON
1239 735 OUT AUXMD ;A XOR A = 0
1239 736 MVI A,GTB ;Goto standby
1239 737 OUT CMOD2
1239 738 WATIX ;Wait for TCI false
1239 739+ ??#034; IN PRTF
1239 740+ ANI TCIF
1239 741+ JNZ ??#034 ;Wait for TCI
1239 742 WATIP ;Wait for TCI complete, int, etc.
1239 743+ ??#035; IN PRTF ;Get task complete int, etc.
1239 744+ ANI TCIF ;Mask it
1239 745+ JZ ??#035 ;Wait for task to be complete
1239 746+ WATIP ;Wait for status byte input
1239 747+ ??#036; IN INT1 ;Get INT1 status
1239 748+ MOV B,A ;Save status in B
1239 749+ ANI LIM ;Check for byte in
1239 750+ JZ ??#036 ;If not, just try again
1239 751 MVI A,TCSY ;Take control sync
1239 752 OUT CMOD2
1239 753 WATIX ;Wait for TCI false
1239 754+ ??#037; IN PRTF
1239 755+ ANI TCIF
1239 756+ JNZ ??#037 ;Wait for TCI
1239 757 WATIP ;Get task complete int, etc.
1239 758+ ??#038; IN PRTF ;Get task complete int, etc.
1239 759+ ANI TCIF ;Mask it
1239 760+ JZ ??#038 ;Wait for task to be complete
1239 761 IN DIN ;Get serial poll status byte
1239 762 OUT CMOD2
1239 763 OUT ADRMD
1239 764 MVI A,TCSY ;Take control sync
1239 765 INX H ;Inc talker
1239 766 MVI A,TON ;Talk only for controller
1239 767 OUT CMOD2
1239 768+ ??#039; IN INT1 ;Get INT1 status
1239 769+ ANI 80H ;Check for byte out
1239 770+ JZ ??#039 ;If not, try again
1239 771+ JMP SPOL1 ;Go on to next device on list
1239 772 SPOL2: MVI A,SPD ;Serial poll disable
1239 773 OUT DOUT ;We know 80 was set (WATIP above)
1239 774 WATIP ;Wait
1239 775+ ??#039; IN INT1 ;Get INT1 status
1239 776+ ANI 80H ;Check for byte out
1239 777+ JZ ??#039 ;If not, try again
1239 778 CLRA
1239 779+ XRA A ;A XOR A = 0
1239 780+ OUT AUXMD ;Immediate XEQ PON to clear LA
1239 781 RET
1239 782 ;
1239 783 ;**************************************************************************
1239 784 ;
PARALLEL POLL ENABLE ROUTINE

; INPUTS: HL listener list pointer
; OUTPUTS: None
; CALLS: None
; DESTROYS: A, DE, HL, F

; BEGIN

PPEN: MOV A, UNL ; Universal unlisten
OUT

RANGE 2OH, 3EH, PPEN2 ; Check for valid listener

GET next byte.

; END

PARALLEL POLL DISABLE ROUTINE

; INPUTS: HL listener list pointer
; OUTPUTS: None
; CALLS: None
; DESTROYS: A, HL, F

; BEGIN

PPDS: MOV A, UNL ; Universal unlisten
OUT

RANGE 2OH, 3EH, PPDS2 ; Check for valid listener

GET next byte.

; END
12F6 7E 859  MOV  A, M  ; Get listener
12F7 D348 858  OUT  DOUT
12F9 23 861  INX  H  ; Incr pointer
12FA C3E412 852  JMP  PPOS2  ; Loop until invalid listener
12FD D861 863  PPOS2:  WAITO
12FF E582 864  ??045:  IN  INT1  ; Get INTI status
1301 CAF012 855  ??045:  ANI  BOM  ; Check for byte out
1304 3E85 856  ??045:  JZ  ??044:  ; If not, try again
1305 D358 857  ??045:  MVI  A, PPC  ; Parallel poll configure
1308 D861 858  ??045:  OUT  DOUT
130B E582 859  ??045:  WAITO
130C CA013 860  ??045:  IN  INT1  ; Get INTI status
130F 3E78 861  ??045:  ANI  BOM  ; Check for byte out
1311 D358 862  ??045:  OUT  DOUT
1313 D861 863  ??045:  WAITO
1315 E582 864  ??045:  IN  INT1  ; Get INTI status
1317 C413 865  ??045:  ANI  BOM  ; Check for byte out
131A C9 866  ??045:  JZ  ??044:  ; If not, try again
131B 3E15 867  ??045:  RET
131C D358 868  ??045:  PARALLEL POLL UNCONFIGURE ALL ROUTINE
131D D861 869  ??045:  OUT  DOUT
131F D861 870  ??045:  WAITO
1321 E582 871  ??045:  IN  INT1  ; Get INTI status
1323 CA13 872  ??045:  ANI  BOM  ; Check for byte out
1326 C9 873  ??045:  JZ  ??044:  ; If not, try again
1327 3E48 874  ??045:  RET
1329 D364 875  ??045:  PPON:  MVI  A, PPON  ; Parallel poll unconfigure
132B AF 876  ??045:  CLRA
132C D365 877  ??045:  OUT  AUXMD  ; Reset TON
132D 3E5F 878  ??045:  MVI  A, EXPP  ; Execute parallel poll
1330 D369 879  ??045:  OUT  CMD92
1332 D861 880  ??045:  WAITI  ; Wait for completion= RI on 91
1334 47 881  ??045:  MOV  B, A  ; Save status in B
1335 E681 882  ??045:  ANI  BIM  ; Check for byte in
1337 CA3213 883  ??045:  JZ  ??044:  ; If not, just try again
133A 3E68 884  ??045:  MVI  A, TON  ; Talk only
133C D364 885  ??045:  OUT  AUXMD
133D 234 886  ??045:  CLRA
133E AF 887  ??045:  XRA  A  ; A XOR A = 8
133F D365 888  ??045:  OUT  AUXMD  ; Reset LOW
1341 D868 889  ??045:  IN  DIN  ; Get PP byte
1343 C9 890  ??045:  RET
1347 231324-42

3-142
1344 7E 941+ MOV A,M
1345 FE40 942+ CPI 40H
1347 FABA13 943+ JM PCTL1
134A FE5F 944+ CPI 5EH
134C F2A13 945+ JP PCTL1
134F FE41 946 CPI MTA
1351 CABA13 947 JI PCTL1
1354 D560 948 OUT DOUT
1356 DB61 949 WAITO
1358 E071 950+ ANI BOM
135A CA513 952+ JZ 770859
135D 3E9F 953+ MVI A,TCT
135F D560 954 OUT DOUT
1361 DB61 955+ WAITO
1363 D662 956+ ANI BOM
1365 CA5113 958+ JZ 770851
1368 3E81 959 MVI A,MOD1
136A D564 960 OUT ADRMD
136C AF 961 CLR A
136D D565 962 OUT AUXMD
136F 3E01 964 MVI A,MDA
1371 D566 965 OUT ADRBD
1373 3EA1 966 MVI A,AUX8+CPTEN
1375 D545 967 OUT AUXMD
1377 3E81 968+ /*optional PP configuration goes here*********/
1379 D569 969 MVI A,UDIL
137B 3F01 970 GO IDLE COMMAND
137D 3F69 972 OUT CMD92
137F DB65 974 WAITX
1380 3E69 976+ IN PRTF
1383 C27013 977+ ANI TCIF
1385 DB6F 978+ JZ 770852
1388 D665 979+ OUT PCTR
138D E692 980+ ANI TCIF
138F CACF13 981+ JZ PCTL2
1392 DB65 983+ IN CPTRD
1394 FE09 1005+ CPI TCT

933 *CALLS: None
934 *DESTROYS: A, HL, F
935 PCTL1 RANGE 40H,5EH,PCTL1
936+ *Checks for value in range
937+ *Branches to label if not
938+ *in range. Falls through if
939+ *lower <= (LH) <= upper.
940+ *Get next byte.

994 ; RECEIVE CONTROL ROUTINE
995 ; CALLS: None
996 ; DESTROYS: A, F
997 ; RETURNS: 8H = invalid (not take control to us or CPT bit not on)
998 ; < 8H = valid (take control to us or CPT bit on)
999 ; NOTE: THIS CODE MUST BE TIGHTLY INTEGRATED INTO ANY USER
1000 ; SOFTWARE THAT FUNCTIONS WITH THE #291 AS A DEVICE.
1001 ; NORMALY SOME ADVANCE WARNING OF IMPENDING PASS
1002 ; CONTROL SHOULD BE GIVEN TO US BY THE CONTROLLER
1003 ; WITH OTHER USEFUL INFO, THIS PROTOCOL IS SITUATION
1004 ; SPECIFIC AND WILL NOT BE COVERED HERE.

1088 D651 1801 HCTL: IN PCL
108D E680 1802 ANI CPT
108F CACF13 1803 JZ PCTL2
1092 DB65 1804 IN CPTRD
1094 FE09 1805+ CPI TCT
1396 C2CA13 1085  JNZ  RCTL1  ;No, go return invalid
1399 DBK4  1087  IN  ADDR $ ;Get address status
1399 E682  1088  ANI  TA  ;Is TA on ?
139D CACA13 1089  JZ  RCTL1  ;No -- go return invalid
13A0 3E68  1090  MVI  A,DTDL1 ;Disable talker listener
13A2 D355  1091  OUT  ADDR$1
13A4 3E8F  1092  MVI  A,TON  ;Talk only
13A6 D344  1093  OUT  ADDR$2
13A8 AF  1095  XRA  A  ;A XOR A = #
13A9 D361  1096  OUT  INT $1  ;Mask off INT hits
13AB D352  1097  OUT  INT2
13AD D355  1098  OUT  AUXMD
13AF 3EFA  1099  MVI  A,TCNT $ ;Take (receive) control 92 command
13B1 D369  109A  OUT  CMD92
13B3 3E8F  109B  MVI  A,VS CMD $ ;Valid command pattern for 91
13B5 D365  109C  OUT  AUXMD
109D 1023  ;******* optional TOUT1 check could be put here *******
1024  WAITX
13B7 DB6F  1025+07054:  IN  PRTF
13B9 3E02  1026+  ANI  TCIF
13BB C2B713  1027+  JNZ  79054
13BE DB6F  1028  WAITT  ;Wait for TCI
13C0 E682  1029+  ANI  TCIF  ;Mask it
13C2 CABA13  1030+  JZ  79055  ;Wait for task to be complete
13C2 3E8F  1031  MVI  A,TC $ ;Valid return pattern
13C7 C3C1F3  1032  JMP  RCTL2  ;Only one return per routine
13CA 3E8F  1033  RCT $1:  MVI  A,VS CMD $ ;Acknowledge CPT
13CC D365  1034  OUT  AUXMD
1336 CLRA  1035  ;Error return pattern
13CE AF  1037+  XRA  A  ;A XOR A = #
13CF C9  1038  RCTL2:  RET
1039 ;
1040 ;**********************************************************************
1041 ;
1042 ; SRO ROUTINE
1043 ;
1044 ; INPUTS: None
1045 ; OUTPUTS: None
1046 ; CALLS: None
1047 ; RETURNS: A = 0 no SRQ
1048 ; A < > 0 SRQ occurred
1049 ;
1050 ;
13DB DB69  1051  SROD:  IN  INTST  ;Get 92's INTRO status
13DF E682  1052  ANI  SRODT  ;Mask off SRQ
13D4 C2A13  1053  JZ  SROD2  ;Not set--- go return
13D7 F69B  1054  ORI  IACK  ;Set--- must clear it with IACK
13DE D365  1055  OUT  CMD92
13DF DB69  1056  IN  INTST  ;Get IAP
13DD E682  1057  ANI  IAPTA  ;Mask it
13DF CADD13  1058  JZ  SROD1  ;Wait if not set
13D2 C9  1059  SROD2:  RET
1060 ;
1061 ;**********************************************************************
1062 ;
1063 ; REMOTE ENABLE ROUTINE
1064 ;
1065 ; INPUTS: None
1066 ; OUTPUTS: None
1067 ; CALLS: None
1068 ; DESTRYS: A, F
1069 ;
13E3 3E8F  1070  REME:  MVI  A,SRE $2
13E5 D369  1071  OUT  CMD92  ;92 asserts remote enable
1072  WAITX  ;Wait for TCI = #
13E7 DB4F  1073+79055:  IN  PRTF
13E9 E682  1074+  ANI  TCIF
13EE C2713  1075+  JNZ  79055
13E6 DB6F  1076  WAITT  ;Wait for TCI
13F8 E682  1077+79057:  IN  PRTF
13F1 CAAA13  1078+  ANI  TCIF  ;Mask it
13F2 C2A13  1079+  JZ  79057  ;Wait for task to be complete

231324-44

3-144
13F5 C9 1080 RET
1081 ;
1082 ;*****************************
1083 ;
1084 ;LOCAL ROUTINE
1085 ;
1086 ;
1087 ;INPUTS: None
1088 ;OUTPUTS: None
1089 ;CALLS: None
1090 ;DESTROYS: A, F
1091 ;
13F6 3F7 1092 LOCL: MVI A, $0C
13F8 D369 1093 OUT CMD92 ;92 stops asserting remote enable
13FA DB6F 1094 #WAITX ;Wait for TCI = 0
13FC E6#2 1095+ ??0058: IN PRTF
13FE C2F13 1097+ JNZ ??0858
1401 DB6F 1098 WAITT ;Wait for TCI
1403 E6#2 1099+ ??0859: IN PRTF ;Get task complete int, etc.
1405 CA0114 1100+ ANI TCIF ;Mask it
1408 C9 1101+ JZ ??0058 ;Wait for task to be complete
140B C9 1102 RET
1103 ;
1104 ;*****************************
1105 ;
1106 ;INTERFACE CLEAR / ABDON ROUTINE
1107 ;
1108 ;
1109 ;INPUTS: None
1110 ;OUTPUTS: None
1111 ;CALLS: None
1112 ;DESTROYS: A, F
1113 ;
1114 ;
1115 IFCL: MVI A, $0B
1116 OUT CMD92 ;Send IFC
1117 WAITX ;Wait for TCI = 0
1118+ ??0058: IN PRTF
1119+ ANI TCIF
1121+ JNZ ??0058 ;
1414 DB6F 1122+ ??0061: IN PRTF ;Get task complete int, etc.
1416 E6#2 1123+ ANI TCIF ;Mask it
1418 CA0144 1124+ JZ ??0061 ;Wait for task to be complete
1125 ;Delete both WAITX & WAITT if this routine
1126 ;is to be called while the 3292 is
1127 ;Controller-in-Charge. If not C.I.C. then
1128 ;TCI is set, else nothing is set (IFC is sent)
1129 ;and the WAIT'S will hang forever
1418 C9 1130 RET
1132 ;
1133 ;APPLICATION EXAMPLE CODE FOR 8085
1134 ;
0032 1135 FGDNL EQU '2' ;Func gen device num "2" ASCII,latin
0031 1136 FCNLD EQU '1' ;Freq ctr device num "1" ASCII,latin
0051 1137 FCNLD EQU '0' ;Freq ctr talk address
000D 1138 CR EQU 0DH ;ASCII carriage return
000A 1139 LF EQU 0AH ;ASCII line feed
008F 1140 CL EQU $FF ;Set end for Talk/Listen lists
0040 1141 SRQM EQU 48H ;Bit indicating device sent SRQ
1142 ;
141C 46553146 1143 FGDATA: DB 'FUIPR37KH4M2VO',CR ;Data to set up func. gen
1420 52333748
1424 4B414D32
1428 564F
142A 8D
142E 5A633447 1144 LIM1 EQU 15 ;Buffer length
142F 3754 1145 FGDATA: DB 'PPAG7F', ;Data to set up freq ctr
1406 1146 LIM2 EQU 6 ;Buffer length
1431 31 1147 LL1: DB FCNLD,LEND ;Listen list for freq ctr
1432 FF
1433 32 1148 LL2: DB FGDNL,LEND ;Listen list for func. gen
1434 FF
1435 51 1149 TL1: DB FCNLD,LEND ;Talk list for freq ctr
1436 FF
1150 ;
1151 ;SETUP FUNCTION GENERATOR
1437 680D 1152 MVI B,CR ;EOS
1439 88EF 1153 MVI C,LIM1 ;Count
143B 111C14 1154 LXI D,FGDATA ;Data pointer
143C 213314 1155 LXI H,LL1 ;Listen list pointer
1441 CDIC10 1156 CALL SEND
1157 ;
1158 ;SETUP FREQ COUNTER
1159 ;
1444 8554 1160 MVI B,'T' ;EOS
1446 8566 1161 MVI C,LIM2 ;Count
1448 112014 1162 LXI D,FCNLD ;Data pointer
1449 213114 1163 LXI H,LL2 ;Listen list pointer
144E CDIC10 1164 CALL SEND
1165 ;
1166 ;WAIT FOR SRQ FROM FREQ CTR
1167 ;
1451 CD0813 1168 LOOP: CALL SRQD ;Has SRQ occurred ?
1454 CA5114 1169 JZ LOOP ;No, wait for it
1178 ;
1171 ;SERIAL POLL TO CLEAR SRQ
1172 ;
1457 11003C 1173 LXI D,SBYTE ;Buffer pointer
145A 213514 1174 LXI H,TL1 ;Talk list pointer
145D CDIC12 1175 CALL SPOL
1460 1B 1176 DCX D ;Backup buffer pointer to ctr byte
1461 1A 1177 LDAX D ;Get status byte
1462 E640 1178 ANI SRQM ;Did ctr assert SRQ ?
1464 CA7714 1179 JZ ERROR ;Ctr should have said yes
1180 ;
1181 ;RECEIVE READING FROM COUNTER
1182 ;
1467 680A 1183 MVI B,LF ;EOS
1469 8E11 1184 MVI C,LI3 ;Count
146B 213514 1185 LXI H,TL1 ;Talk list pointer
146E 1103C 1186 LXI D,FCNLD ;Data in buffer pointer
1471 CD9F10 1187 CALL REC
1474 C27714 1188 JNZ ERROR
1189 ;
1190 ;******* rest of user processing goes here *****
1191 ;
1192 ;
1477 88 1193 ERROR: NOP ;User dependant error handling
1194 ; ETC.
3C00 1195 ORG 3C00
3C00 1196 SBYTE: DS 1 ;Location for serial poll byte
3111 1197 LIM3 EQU 17 ;Max freq counter input

231324-46

3-146
PUBLIC SYMBOLS

EXTERNAL SYMBOLS

ASSEMBLY COMPLETE, NO ERRORS
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>C</th>
<th>DE</th>
<th>HL</th>
<th>3E70:</th>
<th>3E80:</th>
<th>GPIB output:</th>
<th>Ending B=</th>
<th>Ending C=</th>
<th>Ending DE=</th>
<th>Ending HL=</th>
</tr>
</thead>
<tbody>
<tr>
<td>44</td>
<td>30</td>
<td>3E80</td>
<td>3E70</td>
<td>20 30</td>
<td>3E 3F</td>
<td>41 ATN</td>
<td>44</td>
<td>2E</td>
<td>3E82</td>
<td>3E73</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>3E70:</td>
<td></td>
<td>3F ATN</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>20 ATN</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>30 ATN</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>3E ATN</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>11</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>44 EOI</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Receive Test Cases

<table>
<thead>
<tr>
<th>B</th>
<th>C</th>
<th>DE</th>
<th>HL</th>
<th>3E70:</th>
<th>GPIB output:</th>
<th>ZT488 Data</th>
<th>Ending A =</th>
<th>Ending B =</th>
<th>Ending C =</th>
<th>Ending DE=</th>
<th>Ending HL=</th>
</tr>
</thead>
<tbody>
<tr>
<td>44</td>
<td>30</td>
<td>3E80</td>
<td>3E70</td>
<td>40</td>
<td>40 ATN</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>3E71</td>
<td>3E71</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>50</td>
<td>50 ATN</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>3E71</td>
<td>3E71</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>5E</td>
<td>5E ATN</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>3E71</td>
<td>3E71</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>5F</td>
<td>5F ATN</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>3E71</td>
<td>3E71</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>40</td>
<td>40 ATN</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>3E71</td>
<td>3E71</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>40</td>
<td>40 ATN</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>3E71</td>
<td>3E71</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>40</td>
<td>40 ATN</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>3E71</td>
<td>3E71</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>40</td>
<td>40 ATN</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>3E71</td>
<td>3E71</td>
</tr>
</tbody>
</table>

3-148
Serial Poll Test Cases

- C = 30
- DE = 3E80
- HL = 3E70
- 3E70: 40
  - 50
  - 5E
  - 5F
  - GPIB output: 3F ATN
- Ending C = 30
- Ending DE = 3E80
- Ending HL = 3E70
- GPIB output: 21 ATN
- output: 18 ATN
- output: 40 ATN
- input*: 00
- input*: 41
- input*: 5E ATN
- output: 7F
- output: 19 ATN

*NOTE: leave ZT488 in single step mode even on input

Pass Control Test Cases

- HL = 3E70
- 3E70: 40
  - 5F
  - GPIB output: 40 ATN
- Ending HL = 3E71
- Ending A = 02

Receive Control Test Cases

- GPIB input
- Run Receive Control
- GPIB Input
- Ending A =

3-149
Parallel Poll Enable Test Cases

- **DE**: 3E80
- **HL**: 3E70
- **3E70**: 20 30 3E 3F
- **3E80**: 01 02 03

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

- **HL**: 3E70
- **3E70**: 20 30 3E 3F

GPIB output:
- 3F ATN
- 20 ATN
- 05 ATN
- 61 ATN
- 30 ATN
- 05 ATN
- 62 ATN
- 3E ATN
- 05 ATN
- 63 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
- Reset SRQ

- Ending A: 02
- Reset SRQ: 00
Trigger Test

HL = 3E70
DE = 3E80
BC = 4430

3E70: 20 30 3E 3F

GPIB output:
3F ATN
20 ATN
30 ATN
3E ATN
08 ATN

Ending HL = 3E73
DE = 3E80
BC = 4430

Device Clear Test

HL = 3E70
DE = 3E80
BC = 4430

3E70: 20 30 3E 3F

GPIB output:
3F ATN
20 ATN
30 ATN
3E ATN
14 ATN

Ending HL = 3E73
DE = 3E80
RC = 4430

XFER Test

B = 44
HL = 3E70:

3E70: 40 20 30 3E 3F

GPIB output:
40 ATN
3F ATN
20 ATN
30 ATN
3E ATN

GPIB input:
0
1
2
3
44

Ending A = 0
B = 44
HL = 3E74
Application Example
GPIB Output/Input

GPIB output:

GPIB input:

GPIB output:

GPIB 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
3F ATN
21 ATN
18 ATN
51 ATN
40 SRQ
19 ATN
51 ATN
3F ATN
21 ATN
### GPIB input:

- 20
- 2B
- 20
- 20
- 33
- 37
- 30
- 30
- 30
- 2E
- 30
- 45
- 2B
- 30
- 0D
- 0A

### GPIB output:

XX ATN
# APPENDIX C

## REMOTE MESSAGE CODING

<table>
<thead>
<tr>
<th>Mnemonic</th>
<th>Message Name</th>
<th>C T I D y a l p s O e s 8 7 6 5 4 3 2 1 VDC N I Q C N</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACG</td>
<td>addressed command group</td>
<td>M AC Y 0 0 0 X X X X XXX 1 X X X X</td>
</tr>
<tr>
<td>ATN</td>
<td>attention</td>
<td>U UC X X X X X X X X XXX 1 X X X X</td>
</tr>
<tr>
<td>DAB</td>
<td>data byte</td>
<td>U DD D D D D D D D D XXX 0 X X X X</td>
</tr>
<tr>
<td>DAC</td>
<td>data accepted</td>
<td>U HS X X X X X X X X X XXX 1 X X X X</td>
</tr>
<tr>
<td>DAV</td>
<td>data valid</td>
<td>U HS X X X X X X X X X XXX 1 X X X X</td>
</tr>
<tr>
<td>DCL</td>
<td>device clear</td>
<td>M UC Y 0 0 1 0 1 0 0 XXX 1 X X X X</td>
</tr>
<tr>
<td>END</td>
<td>end</td>
<td>U ST X X X X X X XXX 0 1 X X X X</td>
</tr>
<tr>
<td>EOS</td>
<td>end of string</td>
<td>M DD E E E E E E E XXX 0 X X X X</td>
</tr>
<tr>
<td>GET</td>
<td>group execute trigger</td>
<td>M AC Y 0 0 0 1 0 0 0 XXX 1 X X X X</td>
</tr>
<tr>
<td>GTL</td>
<td>go to local</td>
<td>M AC Y 0 0 0 0 0 0 1 XXX 1 X X X X</td>
</tr>
<tr>
<td>IDY</td>
<td>identify</td>
<td>U UC X X X X X X X XXX 1 X X X X</td>
</tr>
<tr>
<td>IFC</td>
<td>interface clear</td>
<td>U UC X X X X X X X XXX 1 X X X X</td>
</tr>
<tr>
<td>LAG</td>
<td>listen address group</td>
<td>M AD Y 0 1 X X X X XXX 1 X X X X</td>
</tr>
<tr>
<td>LLO</td>
<td>local lock out</td>
<td>M UC Y 0 0 1 0 0 0 1 XXX 1 X X X X</td>
</tr>
<tr>
<td>MLA</td>
<td>my listen address</td>
<td>(Note 3) M AD Y 0 1 1 L L L L L XXX 1 X X X X</td>
</tr>
<tr>
<td>MTA</td>
<td>my talk address</td>
<td>(Note 4) M AD Y 1 0 T T T T XXX 1 X X X X</td>
</tr>
<tr>
<td>MSA</td>
<td>my secondary address</td>
<td>(Note 5) M SE Y 1 1 S S S S S XXX 1 X X X X</td>
</tr>
<tr>
<td>NUL</td>
<td>null byte</td>
<td>M DD 0 0 0 0 0 0 0 XXX 1 X X X X</td>
</tr>
<tr>
<td>OSA</td>
<td>other secondary address</td>
<td>M SE (OSA = SCG ∧ MSA)</td>
</tr>
<tr>
<td>OTA</td>
<td>other talk address</td>
<td>M AD (OTA = TAG ∧ MTA)</td>
</tr>
<tr>
<td>PCG</td>
<td>primary command group</td>
<td>M — (PCG = ACG ∨ UCG ∨ LAG ∨ TAG)</td>
</tr>
<tr>
<td>PPC</td>
<td>parallel poll configure</td>
<td>M AC Y 0 0 0 1 0 1 0 1 XXX 1 X X X X</td>
</tr>
<tr>
<td>PPE</td>
<td>parallel poll enable</td>
<td>(Note 6) M SE Y 1 1 0 S P P P XXX 1 X X X X</td>
</tr>
<tr>
<td>PPD</td>
<td>parallel poll disable</td>
<td>(Note 7) M SE Y 1 1 1 D D D D XXX 1 X X X X</td>
</tr>
<tr>
<td>PPR1</td>
<td>parallel poll response 1</td>
<td>U ST X X X X X X X X 1 XXX 1 1 X X X X</td>
</tr>
<tr>
<td>PPR2</td>
<td>parallel poll response 2</td>
<td>(Note 10) U ST X X X X X X 1 XXX 1 1 X X X X</td>
</tr>
</tbody>
</table>

Notes:
1. [Notes 1, 9]
2. [Notes 2, 9]
3. [Note 3]
4. [Note 4]
5. [Note 5]
6. [Note 6]
7. [Note 7]
8. [Note 10]

Bus Signal Line(s) and Coding That Asserts the True Value of the Message
REMOTE MESSAGE CODING (Continued)

Bus Signal Line(s) and Coding That Asserts the True Value of the Message

| Mnemonic       | Message Name              | C | T | I | D | y | a | l | O | s | P | R | E | N | C | N |
|----------------|---------------------------|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| PPR3           | parallel poll response 3  | 8 | 6 | 5 | 4 | 3 | 2 | 1 | VDC | N | I | O | Q | C | N |
| PPR4           | parallel poll response 4  | 8 | 6 | 5 | 4 | 3 | 2 | 1 | VDC | N | I | O | Q | C | N |
| PPR5           | parallel poll response 5  | 8 | 6 | 5 | 4 | 3 | 2 | 1 | VDC | N | I | O | Q | C | N |
| PPR6           | parallel poll response 6  | 8 | 6 | 5 | 4 | 3 | 2 | 1 | VDC | N | I | O | Q | C | N |
| PPR7           | parallel poll response 7  | 8 | 6 | 5 | 4 | 3 | 2 | 1 | VDC | N | I | O | Q | C | N |
| PPR8           | parallel poll response 8  | 8 | 6 | 5 | 4 | 3 | 2 | 1 | VDC | N | I | O | Q | C | N |
| PPU            | parallel poll unconfigure | M | UC | Y | 0 | 0 | 1 | 0 | 1 | 0 | 1 | XXX | 1 | X | X | X |
| REN            | remote enable             | U | UC | X | X | X | X | X | X | XXX | 1 | X | X | X | X | X |
| RFD            | ready for data            | U | HS | X | X | X | X | X | X | XXX | 0 | X | X | X | X | X |
| RQS            | request service           | U | ST | X | 1 | X | X | X | X | XXX | 1 | X | X | X | X | X |
| SCG            | secondary command group   | M | SE | Y | 1 | 1 | X | X | X | X | XXX | 1 | X | X | X | X | X |
| SDC            | selected device clear     | M | AC | Y | 0 | 0 | 0 | 1 | 0 | 0 | 0 | XXX | 1 | X | X | X | X | X |
| SPD            | serial poll disable       | M | UC | Y | 0 | 0 | 1 | 1 | 0 | 0 | 1 | XXX | 1 | X | X | X | X | X |
| SPE            | serial poll enable        | M | UC | Y | 0 | 0 | 1 | 1 | 0 | 0 | 0 | XXX | 1 | X | X | X | X | X |
| SRQ            | service request           | U | ST | X | X | X | X | X | X | XXX | 1 | X | X | X | X | X |
| STB            | status byte               | M | ST | S | X | S | S | S | S | XXX | 0 | X | X | X | X | X |
| TCT            | take control              | M | AC | Y | 0 | 0 | 0 | 1 | 0 | 0 | 1 | XXX | 1 | X | X | X | X | X |
| TAG            | talk address group        | M | AD | Y | 1 | 0 | X | X | X | X | XXX | 1 | X | X | X | X | X |
| UCG            | universal command group   | M | UC | Y | 0 | 0 | 1 | X | X | X | XXX | 1 | X | X | X | X | X |
| UNL            | unlisten                 | M | AD | Y | 0 | 1 | 1 | 1 | 1 | 1 | 1 | XXX | 1 | X | X | X | X | X |
| UNT            | untalk                   | M | AD | Y | 1 | 0 | 1 | 1 | 1 | 1 | 1 | XXX | 1 | X | X | X | X | X |

The 1/0 coding on ATN when sent concurrent with multline 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 Response</th>
<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>0</td>
<td>PPR1</td>
</tr>
<tr>
<td>1</td>
<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. (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 DI0 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 DI0 lines is always the PP function.
11. This code is provided for system use, see 6.3.
Modem Products
89024
2400 BPS INTELLIGENT MODEM CHIP SET

- For Public Switched Telephone Network and Unconditioned Leased Line 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
- 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
- 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

*Hayes, Smartmodem 2400 and Smartcom II are registered trademarks of Hayes Microcomputer Products, Inc.

September 1990
Order Number: 270242-005
GENERAL DESCRIPTION

The Intel 89024 chip set is a highly integrated, high performance, intelligent modem, providing a complete system in two chips. The system is compatible with 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
- 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/PSK/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 supports two versions of firmware. These are internal, which is asynchronous only and external, which is asynchronous and synchronous. Any differences in operation are highlighted with footnotes.

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.
PACKAGING

The 89027 is available in PLCC and standard plastic DIP packages. The 89026 is available in a PLCC package.

Figure 2. Device Packages
CALL ESTABLISHMENT, TERMINATION AND RETRAIN

The 89024 modem system incorporates all protocols and functions required for automatic or manual call establishment. The modem system also incorporates all protocols and functions required for progress and termination of a data call.

The modem chip-set has a built-in auto-dialer, both DTMF and Pulse type. 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 re-dialing 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 is compatible with 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 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 300</th>
<th>Bell 1200</th>
<th>CCITT 300</th>
<th>CCITT 1200</th>
<th>CCITT 2400</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bell 300</td>
<td>300</td>
<td>300</td>
<td>—</td>
<td>300*</td>
<td>300*</td>
</tr>
<tr>
<td>1200</td>
<td>1200*</td>
<td>1200</td>
<td>—</td>
<td>1200</td>
<td>1200</td>
</tr>
<tr>
<td>CCITT 300</td>
<td>—</td>
<td>—</td>
<td>300</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>1200</td>
<td>1200*</td>
<td>1200</td>
<td>—</td>
<td>1200</td>
<td>1200</td>
</tr>
<tr>
<td>2400</td>
<td>1200*</td>
<td>1200</td>
<td>—</td>
<td>1200</td>
<td>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(1)</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></td>
<td>If &amp;J1 option is selected, H1 will also switch the auxiliary relay</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 & Command Set

<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(1)</td>
<td>Async/Sync Mode Selection</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(1)</td>
<td>Sync Clock Source</td>
</tr>
<tr>
<td>&amp;Z</td>
<td>Store Telephone Number</td>
</tr>
</tbody>
</table>

NOTE:
1. Available in external code only.
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>Register</th>
<th>Description</th>
<th>Bit Mapped Options Register</th>
</tr>
</thead>
<tbody>
<tr>
<td>S0*</td>
<td>Ring to Answer</td>
<td>21</td>
</tr>
<tr>
<td>S1</td>
<td>Ring Count. (Read Only)</td>
<td>22</td>
</tr>
<tr>
<td>S2</td>
<td>Escape Code Character</td>
<td>23</td>
</tr>
<tr>
<td>S3</td>
<td>Carriage Return Character</td>
<td>24</td>
</tr>
<tr>
<td>S4</td>
<td>Line Feed Character</td>
<td>25</td>
</tr>
<tr>
<td>S5</td>
<td>Back Space Character</td>
<td>26</td>
</tr>
<tr>
<td>S6</td>
<td>Wait for Dial Tone</td>
<td>27</td>
</tr>
<tr>
<td>S7</td>
<td>Wait for Data Carrier</td>
<td>28</td>
</tr>
<tr>
<td>S8</td>
<td>Pause Time for the Comma Dial Modifier</td>
<td>29</td>
</tr>
<tr>
<td>S9</td>
<td>Carrier Detect Response Time</td>
<td>30</td>
</tr>
<tr>
<td>S10</td>
<td>Lost Carrier to Hang Up Delay</td>
<td>31</td>
</tr>
<tr>
<td>S11**</td>
<td>DTMF Tone Duration</td>
<td>32</td>
</tr>
<tr>
<td>S12</td>
<td>Escape Code Guard Time</td>
<td>33</td>
</tr>
<tr>
<td>S13</td>
<td>Not Used</td>
<td>34</td>
</tr>
<tr>
<td>S14*</td>
<td>Bit Mapped Option Register</td>
<td>35</td>
</tr>
<tr>
<td>S15</td>
<td>Not Used</td>
<td>36</td>
</tr>
<tr>
<td>S16</td>
<td>Modem Test Options</td>
<td>37</td>
</tr>
<tr>
<td>S17</td>
<td>Not Used</td>
<td>38</td>
</tr>
<tr>
<td>S18*</td>
<td>Test Timer</td>
<td>39</td>
</tr>
<tr>
<td>S19</td>
<td>Not Used</td>
<td>40</td>
</tr>
<tr>
<td>S20</td>
<td>Not Used</td>
<td>41</td>
</tr>
<tr>
<td>S21*</td>
<td>Bit Mapped Options Register</td>
<td>42</td>
</tr>
<tr>
<td>S22*</td>
<td>Bit Mapped Options Register</td>
<td>43</td>
</tr>
<tr>
<td>S23*</td>
<td>Bit Mapped Options Register</td>
<td>44</td>
</tr>
<tr>
<td>S24</td>
<td>Not Used</td>
<td>45</td>
</tr>
<tr>
<td>S25*</td>
<td>Delay to DTR (Sync Only)</td>
<td>46</td>
</tr>
<tr>
<td>S26*</td>
<td>RTS to CTS Delay (Half Dup.)</td>
<td>47</td>
</tr>
<tr>
<td>S27*</td>
<td>Bit Mapped Options Register</td>
<td>48</td>
</tr>
</tbody>
</table>

NOTE:
* These S registers can be stored in the NVRAM.
**Available in internal code only.

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 with External Hybrid

*Internal code supports 9346
External code supports X2444

Figure 4. Typical Telephone Line Interface Using Internal 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><strong>Asynchronous</strong></td>
<td>2400, 1200 bps, character asynchronous. 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>10 bits, including start, stop, parity. 8, 9, 11 bits optional via S/W customization.</td>
</tr>
<tr>
<td><strong>Synchronous Timing Source</strong></td>
<td>a) Internal, derived from the local oscillator.</td>
</tr>
<tr>
<td></td>
<td>b) External, provided by DTE through XTCLK.</td>
</tr>
<tr>
<td></td>
<td>c) 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.</td>
</tr>
<tr>
<td></td>
<td>On-chip hybrid and billing delay timers.</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 PSK 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>Typical 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>Energy Detect Hysteresis</strong></td>
<td>A minimum Hysteresis of 2 dB for QAM scrambled mark.</td>
</tr>
<tr>
<td><strong>Line Equalization</strong></td>
<td>Fixed compromise equalization, transmit.</td>
</tr>
<tr>
<td></td>
<td>Adaptive equalizer for PSK/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><strong>Self Test Pattern Generator</strong></td>
<td>Alternate 'ones' and 'zeros' and error detector, to be used along with most loopbacks.</td>
</tr>
<tr>
<td></td>
<td>A number indicating the bit errors detected is sent to DTE.</td>
</tr>
</tbody>
</table>

*External code only.
## RECEIVER PERFORMANCE

<table>
<thead>
<tr>
<th>Test Cases</th>
<th>Typical SNR for 10&lt;sup&gt;−5&lt;/sup&gt; BER Performance</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Rx Level (dBm)</td>
</tr>
<tr>
<td>V.22 bis</td>
<td>−30</td>
</tr>
<tr>
<td>Synchronous</td>
<td>−40</td>
</tr>
<tr>
<td>V.22/Bell 212A</td>
<td>−30</td>
</tr>
<tr>
<td>Synchronous</td>
<td>−40</td>
</tr>
<tr>
<td>V.21</td>
<td>−30</td>
</tr>
<tr>
<td>Asynchronous</td>
<td>−40</td>
</tr>
<tr>
<td>Bell 103</td>
<td>−30</td>
</tr>
<tr>
<td>Asynchronous</td>
<td>−40</td>
</tr>
</tbody>
</table>

### Test Conditions:

- Receive signal (Rx) measured at A02 (transmit level set at −9 dBm)
- Unconditioned 3002 line
- 3 kHz Flat-band 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></td>
<td>4.0</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></td>
<td>3</td>
<td></td>
<td>dB</td>
<td></td>
</tr>
<tr>
<td>DTMF Duration</td>
<td></td>
<td>100</td>
<td></td>
<td>ms</td>
<td>Software Controlled</td>
</tr>
<tr>
<td>Pulse Dialing Rate</td>
<td></td>
<td>10</td>
<td></td>
<td>pps</td>
<td></td>
</tr>
<tr>
<td>Pulse Dialing Make/Break</td>
<td>39/61</td>
<td>33/67</td>
<td></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>2.1</td>
<td></td>
<td>sec</td>
<td></td>
</tr>
<tr>
<td>Guard Tone Frequency Amplitude</td>
<td>540</td>
<td>−3</td>
<td></td>
<td>Hz</td>
<td>referenced to High Channel transmit.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>−6</td>
<td>dB</td>
<td>QAM/PSK Modes Only</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 Duration</td>
<td>0.75</td>
<td></td>
<td>1.5</td>
<td>sec</td>
<td>Off/On Ratio</td>
</tr>
<tr>
<td>Cadence</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Busy Tone Detect Duration</td>
<td>0.67</td>
<td>0.2</td>
<td>1.5</td>
<td>sec</td>
<td>Off/On Ratio</td>
</tr>
<tr>
<td>Cadence</td>
<td></td>
<td></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, PSK and QAM is used to send 2 or 4 bits of information respectively at 600 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>CLKIN</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>TCOL</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/Å</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>Reserved for future use (VCC)^{(3)}</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>SS</td>
<td>(Note 4)</td>
<td>In</td>
<td>5</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>5</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 (VCC)^{(3)}</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>VPP</td>
<td>(NC)^{(2)}</td>
<td>In</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^{(5)}</td>
<td>Out</td>
<td>60</td>
</tr>
<tr>
<td>JS</td>
<td>Jack Select^{(5)}</td>
<td>Out</td>
<td>59</td>
</tr>
</tbody>
</table>

*Available in external code only.
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>BUSWIDTH</td>
<td>Bus Width (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 \(\overline{AD}\) and \(\overline{JS}\).
6. Pins with direction "In" must not be left floating.

### 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.

**Vpp**
This function is not used and should not be connected.

\(^{*}\)External code only.

**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/CSI
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 FSK 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 transferred 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 sets an off hook condition, high sets an on hook. When dialing, this signal 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/A
The function of this pin is re-defined as external NVRAM CE.

CONFIG
Reserved for future use. This signal should be pulled high.

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.

*External code only

89026 ABSOLUTE MAXIMUM RATINGS**
Temperature Under Bias ............0°C to + 70°C
Storage Temperature ............−40°C 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

*WARNING: Stressing the device beyond the "Absolute Maximum Ratings" may cause permanent damage. These are stress ratings only. Operation beyond the "Operating Conditions" is not recommended and extended exposure beyond the "Operating Conditions" may affect device reliability.
## 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>FREQ</td>
<td>CLKIN Frequency 12.96 MHz</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:**
The AGND and VSS on both the 89026 and the 89027 must be nominally at the same potential.

## D.C. CHARACTERISTICS

Test Conditions: \( V_{CC}, V_{REF}, V_{PD}, V_{PP}, V_{EA} = 5.0V \pm 0.25V; \)
\( F_{OSC} = 12.96 \text{ MHz}; T_A = 0°C \text{ to } 70°C, V_{SS}, AGND = 0V \)

<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>( I_{CC} )</td>
<td>( V_{CC} ) Supply Current (( 0°C \leq T_A \leq 70°C ))</td>
<td>240</td>
<td>mA</td>
<td>All Outputs Disconnected</td>
<td></td>
</tr>
<tr>
<td>( I_{CC1} )</td>
<td>( V_{CC} ) Supply Current (( T_A = 70°C ))</td>
<td>185</td>
<td>mA</td>
<td>Normal Operation and Power-Down</td>
<td></td>
</tr>
<tr>
<td>( I_{PD} )</td>
<td>( V_{PD} ) Supply Current</td>
<td>1</td>
<td>mA</td>
<td></td>
<td></td>
</tr>
<tr>
<td>( I_{REF} )</td>
<td>( V_{REF} ) Supply Current</td>
<td>8</td>
<td>mA</td>
<td></td>
<td></td>
</tr>
<tr>
<td>( V_{IL} )</td>
<td>Input Low Voltage</td>
<td>-0.3</td>
<td>V</td>
<td></td>
<td></td>
</tr>
<tr>
<td>( V_{IH} )</td>
<td>Input High Voltage (Except RESET, NMI, CLKIN)</td>
<td>2.0 ( V_{CC} + 0.5 )</td>
<td>V</td>
<td></td>
<td></td>
</tr>
<tr>
<td>( V_{IH1} )</td>
<td>Input High Voltage, RESET Rising</td>
<td>2.4 ( V_{CC} + 0.5 )</td>
<td>V</td>
<td></td>
<td></td>
</tr>
<tr>
<td>( V_{IH2} )</td>
<td>Input High Voltage, RESET Falling Hysteresis</td>
<td>2.1 ( V_{CC} + 0.5 )</td>
<td>V</td>
<td></td>
<td></td>
</tr>
<tr>
<td>( V_{IH3} )</td>
<td>Input High Voltage, NMI, CLKIN</td>
<td>2.2 ( V_{CC} + 0.5 )</td>
<td>V</td>
<td></td>
<td></td>
</tr>
<tr>
<td>( I_{IL} )</td>
<td>Input Leakage Current(1)</td>
<td>±10</td>
<td>μA</td>
<td>( V_{IN} = 0 ) to ( V_{CC} )</td>
<td></td>
</tr>
<tr>
<td>( I_{IL1} )</td>
<td>D.C. Input Leakage Current(2)</td>
<td>+3</td>
<td>μA</td>
<td>( V_{IN} = 0 ) to ( V_{CC} )</td>
<td></td>
</tr>
<tr>
<td>( I_{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>( I_{IL} )</td>
<td>Input Low Current(3)</td>
<td>-125</td>
<td>μA</td>
<td>( V_{IL} = 0.45V )</td>
<td></td>
</tr>
<tr>
<td>( I_{IL1} )</td>
<td>Input Low Current to RESET</td>
<td>-0.25</td>
<td>mA</td>
<td>( V_{IL} = 0.45V )</td>
<td></td>
</tr>
<tr>
<td>( I_{IL2} )</td>
<td>Input Low Current, READY, BUSWIDTH(4)</td>
<td>-50</td>
<td>μA</td>
<td>( V_{IL} = 0.45V )</td>
<td></td>
</tr>
<tr>
<td>( V_{OL} )</td>
<td>Output Low Voltage(5)</td>
<td>0.45</td>
<td>V</td>
<td>( I_{OL} = 0.8 \text{ mA} )</td>
<td></td>
</tr>
<tr>
<td>( V_{OL1} )</td>
<td>Output Low Voltage(6)</td>
<td>0.75</td>
<td>V</td>
<td>( I_{OL} = 2.0 \text{ mA} )</td>
<td></td>
</tr>
<tr>
<td>( V_{OL2} )</td>
<td>Output Low Voltage(6)</td>
<td>0.45</td>
<td>V</td>
<td>( I_{OL} = 2.0 \text{ mA} )</td>
<td></td>
</tr>
<tr>
<td>( V_{OH} )</td>
<td>Output High Voltage(5)</td>
<td>2.4</td>
<td>V</td>
<td>( I_{OH} = -20 \mu A )</td>
<td></td>
</tr>
<tr>
<td>( V_{OH1} )</td>
<td>Output High Voltage on Bus Control Pins(7)</td>
<td>2.4</td>
<td>V</td>
<td>( I_{OH} = -200 \mu A )</td>
<td></td>
</tr>
<tr>
<td>( I_{OH3} )</td>
<td>Output High Current on RESET</td>
<td>-50</td>
<td>μA</td>
<td>( V_{OH} = 2.4V )</td>
<td></td>
</tr>
<tr>
<td>( C_S )</td>
<td>Pin Capacitance (Any Pin to VSS)</td>
<td>10</td>
<td>pF</td>
<td>( f_{TEST} = 1.0 \text{ MHz} )</td>
<td></td>
</tr>
</tbody>
</table>

**NOTES:**
1. STR, DTR, XTCLK, TXD
2. S/D, SS, REMLB, LCLKB, I, Q, CONFIG, ED
3. TCLK, RTS
4. Also, B/C, SH, RI
5. TCLK0, S/A, CTS, DSR, DCD, SI, OH, AR
6. SCLK, SDATA, TM, TCLK, RXD, RCLK, TSYNC
7. Bus/Control pins include CLKOUT, ALE, BHE, RD, WR, INST and AD0–15.
**AC CHARACTERISTICS** *(V_{CC}, V_{PD} = 4.75 to 5.25 Volts; T_{A} = 0°C to 70°C; CLKIN = 12.96 MHz)*

Test Conditions: Load capacitance on output pins = 80 pF

TOSC = 1/12.96 MHz

**TIMING REQUIREMENTS** *(Other system components must meet these specs.)*

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Parameter</th>
<th>Min</th>
<th>Max</th>
<th>Units</th>
</tr>
</thead>
<tbody>
<tr>
<td>T_{CLYX}</td>
<td>READY Hold after CLKOUT Edge</td>
<td>0</td>
<td></td>
<td>ns</td>
</tr>
<tr>
<td>T_{LLYV}</td>
<td>End of ALE/ADV to READY Valid</td>
<td></td>
<td>2Tosc - 70</td>
<td>ns</td>
</tr>
<tr>
<td>T_{LLYH}</td>
<td>End of ALE/ADV to READY High</td>
<td>2Tosc + 40</td>
<td>4Tosc - 80</td>
<td>ns</td>
</tr>
<tr>
<td>T_{LYLYH}</td>
<td>Non-Ready Time</td>
<td></td>
<td>1000</td>
<td>ns</td>
</tr>
<tr>
<td>T_{AVDV}</td>
<td>Address Valid to Input Data Valid</td>
<td></td>
<td>5Tosc - 120</td>
<td>ns</td>
</tr>
<tr>
<td>T_{RLDV}</td>
<td>RD Active to Input Data Valid</td>
<td></td>
<td>3Tosc - 100</td>
<td>ns</td>
</tr>
<tr>
<td>T_{RHDX}</td>
<td>Data Hold after RD Inactive</td>
<td>0</td>
<td></td>
<td>ns</td>
</tr>
<tr>
<td>T_{RHDZ}</td>
<td>RD Inactive to Input Data Float</td>
<td>0</td>
<td>Tosc - 25</td>
<td>ns</td>
</tr>
<tr>
<td>T_{AVGV}</td>
<td>Address Valid to BUSWIDTH Valid</td>
<td></td>
<td>2Tosc - 125</td>
<td>ns</td>
</tr>
<tr>
<td>T_{LLGX}</td>
<td>BUSWIDTH Hold after ALE/ADV Low</td>
<td>Tosc + 40</td>
<td></td>
<td>ns</td>
</tr>
<tr>
<td>T_{LLGV}</td>
<td>ALE/ADV Low to BUSWIDTH Valid</td>
<td></td>
<td>Tosc - 75</td>
<td>ns</td>
</tr>
</tbody>
</table>

**NOTE:**

1. The term "Address Valid" applies to AD0–15, BHE and INST.

**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>F_{CLKIN}</td>
<td>Oscillator Frequency</td>
<td>12.95870</td>
<td>12.96129</td>
<td>MHz</td>
</tr>
<tr>
<td>T_{OSC}</td>
<td>Oscillator Period</td>
<td>1/F_{CLKIN}(MAX)</td>
<td>1/F_{CLKIN}(MIN)</td>
<td>ns</td>
</tr>
<tr>
<td>T_{OHCH}</td>
<td>Rising Edge to Clock Rising Edge</td>
<td>0</td>
<td>120</td>
<td>ns</td>
</tr>
<tr>
<td>T_{CHCH}</td>
<td>CLKOUT Period(2)</td>
<td>3Tosc(2)</td>
<td>3Tosc(2)</td>
<td>ns</td>
</tr>
<tr>
<td>T_{CHCL}</td>
<td>CLKOUT High Time</td>
<td>Tosc - 35</td>
<td>Tosc + 10</td>
<td>ns</td>
</tr>
<tr>
<td>T_{CLLH}</td>
<td>CLKOUT Low to ALE High</td>
<td>-20</td>
<td>+25</td>
<td>ns</td>
</tr>
<tr>
<td>T_{LLCH}</td>
<td>ALE/ADV Low to CLKOUT High</td>
<td>Tosc - 25</td>
<td>Tosc + 45</td>
<td>ns</td>
</tr>
<tr>
<td>T_{LHLL}</td>
<td>ALE/ADV High Time</td>
<td>Tosc - 30</td>
<td>Tosc + 35(3)</td>
<td>ns</td>
</tr>
<tr>
<td>T_{AVLL}</td>
<td>Address Setup to End of ALE/ADV</td>
<td>Tosc - 50</td>
<td></td>
<td>ns</td>
</tr>
<tr>
<td>T_{RLAZ}</td>
<td>RD or WR Low to Address Float</td>
<td>Typ. = 0</td>
<td>10</td>
<td>ns</td>
</tr>
<tr>
<td>T_{LLRL}</td>
<td>End of ALE/ADV to RD or WR Active</td>
<td>Tosc - 40</td>
<td></td>
<td>ns</td>
</tr>
<tr>
<td>T_{LLAX}</td>
<td>Address Hold after End of ALE/ADV</td>
<td>Tosc - 40</td>
<td></td>
<td>ns</td>
</tr>
<tr>
<td>T_{WLWH}</td>
<td>WR Pulse Width</td>
<td>3Tosc - 35</td>
<td></td>
<td>ns</td>
</tr>
<tr>
<td>T_{QVWH}</td>
<td>Output Data Valid to End of WR/WRL/WRH</td>
<td>3Tosc - 60</td>
<td></td>
<td>ns</td>
</tr>
<tr>
<td>T_{WHQX}</td>
<td>Output Data Hold after WR/WRL/WRH</td>
<td>Tosc - 50</td>
<td></td>
<td>ns</td>
</tr>
<tr>
<td>T_{WHLL}</td>
<td>End of WR/WRL/WRH to ALE/ADV High</td>
<td>Tosc - 75</td>
<td></td>
<td>ns</td>
</tr>
<tr>
<td>T_{RLRH}</td>
<td>RD Pulse Width</td>
<td>3Tosc - 30</td>
<td></td>
<td>ns</td>
</tr>
<tr>
<td>T_{RHLH}</td>
<td>End of RD to ALE/ADV High</td>
<td>Tosc - 45</td>
<td></td>
<td>ns</td>
</tr>
</tbody>
</table>
TIMING RESPONSES (Continued)

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Parameter</th>
<th>Min</th>
<th>Max</th>
<th>Units</th>
</tr>
</thead>
<tbody>
<tr>
<td>T\textsubscript{CLLL}</td>
<td>CLKOUT to Low ALE/ADV Low</td>
<td>Tosc – 40</td>
<td>Tosc + 35</td>
<td>ns</td>
</tr>
<tr>
<td>T\textsubscript{RRBX}</td>
<td>RD High to INST, BHE, AD8–15 Inactive</td>
<td>Tosc – 25</td>
<td>Tosc + 30</td>
<td>ns</td>
</tr>
<tr>
<td>T\textsubscript{WHBX}</td>
<td>WR High to INST, BHE, AD8–15 Inactive</td>
<td>Tosc – 50</td>
<td>Tosc + 100</td>
<td>ns</td>
</tr>
<tr>
<td>T\textsubscript{HLHH}</td>
<td>WRL, WRH Low to WRL, WRH High</td>
<td>2Tosc – 35</td>
<td>2Tosc + 40</td>
<td>ns</td>
</tr>
<tr>
<td>T\textsubscript{LHH}</td>
<td>ALE/ADV Low to WRL, WRH Low</td>
<td>2Tosc – 30</td>
<td>2Tosc + 55</td>
<td>ns</td>
</tr>
<tr>
<td>T\textsubscript{QVHL}</td>
<td>Output Data Valid to WRL, WRH Low</td>
<td>Tosc – 60</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

NOTES:
1. If more than one wait state is desired, add 3Tosc for each additional wait state.
2. CLKOUT is directly generated as a divide by 3 of the oscillator. The period will be 3Tosc ± 10 ns if Tosc is constant and the rise and fall times are less than 10 ns.
3. Max spec applies only to ALE. Min spec applies to both ALE and ADV.
4. The term “Address Valid” applies to AD0–15, BHE and INST.
5. The term “Address” in this definition applies to AD0–7 for 8-bit cycles, and AD0–15 for 16-bit cycles.

WAVEFORM

![Waveform Diagram]

NOTES:
1. 8-bit bus only
2. 8-bit or 16-bit bus and write strobe mode selected.
3. When ADV selected.
4. 8- or 16-bit bus and no write strobe mode selected.

Figure 6. Bus Signal Timings
89027 OVERVIEW

The 89027 is a 28 pin CHMOS 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 CHMOS 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.
### 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 89026</td>
<td>Out</td>
<td>26</td>
</tr>
<tr>
<td>RST</td>
<td>Chip reset (active low)(1)</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>
<tr>
<td>NC</td>
<td>(Note 2)</td>
<td>Out</td>
<td>19</td>
</tr>
<tr>
<td>NC</td>
<td>(Note 3)</td>
<td>In</td>
<td>22</td>
</tr>
</tbody>
</table>

**NOTES:**
1. When held high, these pins must be connected through 10K resistors to VCC.
2. Must be left No Connect. Will affect operation of 89027
3. Reserved Pin. Must be left No Connect.

### 89027 Pinout Description

**TX0-3**

These four pins control the transmitted signal level. Refer to Transmit Level Table.

**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: -0°C to 70°C
- Storage Temperature: -40°C to 125°C
- All Input and Output Voltages with Respect to $V_{BB}$: $-0.3V$ to $+13.0V$
- All Input and Output Voltages with Respect to $V_{CC} & AV_{CC}$: $-13.0V$ to $0.3V$
- Power Dissipation: 1.35W
- Voltage with Respect to $V_{SS}$: $-0.3V$ 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, $V_{CC} = AV_{CC} = 5V \pm 5\%$, $V_{SS} = AGND = 0V$.

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Parameter Description</th>
<th>Min</th>
<th>Typ</th>
<th>Max</th>
<th>Units</th>
</tr>
</thead>
<tbody>
<tr>
<td>AlCc1</td>
<td>$AV_{CC}$ Operating Current</td>
<td>15</td>
<td>21</td>
<td>mA</td>
<td></td>
</tr>
<tr>
<td>Icc1</td>
<td>$V_{CC}$ Operating Current</td>
<td>5</td>
<td>6</td>
<td>mA</td>
<td></td>
</tr>
<tr>
<td>Ibb1</td>
<td>$V_{BB}$ Operating Current</td>
<td>-15</td>
<td>-21</td>
<td>mA</td>
<td></td>
</tr>
<tr>
<td>Alccs</td>
<td>$AV_{CC}$ Standby Current</td>
<td>0.2</td>
<td>1</td>
<td>mA</td>
<td></td>
</tr>
<tr>
<td>Iccs</td>
<td>$V_{CC}$ Standby Current</td>
<td>5</td>
<td>6</td>
<td>mA</td>
<td></td>
</tr>
<tr>
<td>Ibbss</td>
<td>$V_{BB}$ 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>175</td>
<td>250</td>
<td>mW</td>
<td></td>
</tr>
<tr>
<td>Pds</td>
<td>Standby Power Dissipation</td>
<td>30</td>
<td>50</td>
<td>mW</td>
<td></td>
</tr>
</tbody>
</table>

**DC CHARACTERISTICS** $(Ta = 0^\circ C$ to $70^\circ C$, $AV_{CC} = V_{CC} = 5V \pm 5\%$, $V_{BB} = 5V \pm 5\%$, $AGND = V_{SS} = 0V$), supply voltage must be at the same potential as the 89026 power supply. Typical Values are for $Ta = 25^\circ C$ and nominal power supply values. $V_{CC}$, $AV_{CC}$ and 89026 $V_{REF}$ must be nominally at the same potential.

**Inputs:** TX0, TX1, TX2, TX3, HYB, RST
**Outputs:** CLKOUT

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Parameter Description</th>
<th>Min</th>
<th>Max</th>
<th>Units</th>
<th>Test Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>ill</td>
<td>Input Leakage Current</td>
<td>-10</td>
<td>+10</td>
<td>$\mu A$</td>
<td>$V_{SS} \leq Vin \leq V_{CC}$</td>
</tr>
<tr>
<td>vil</td>
<td>Input Low Voltage</td>
<td>$V_{SS}$</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>$V_{CC}$</td>
<td>V</td>
<td></td>
</tr>
<tr>
<td>vol</td>
<td>Output Low Voltage</td>
<td>0.4</td>
<td>V</td>
<td>$ioh \geq -1.6mA, 1$ TTL load</td>
<td></td>
</tr>
<tr>
<td>voh</td>
<td>Output High Voltage</td>
<td>2.4</td>
<td>V</td>
<td>$ioh \leq 50\mu A, 1$ TTL load</td>
<td></td>
</tr>
<tr>
<td>vcol</td>
<td>CLKOUT Low Voltage</td>
<td>0.4</td>
<td>V</td>
<td>Load Capacitance = 60 pF</td>
<td></td>
</tr>
<tr>
<td>vcoh</td>
<td>CLKOUT High Voltage</td>
<td>0.7 $V_{CC}$</td>
<td>V</td>
<td>Load Capacitance = 60 pF</td>
<td></td>
</tr>
</tbody>
</table>
**AC CHARACTERISTICS** (Ta = 25°C, VCC = AVCC = 5V, VSS = AGND = 0V, VBB = −5V)

**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 Receive Signal</td>
<td>-9</td>
<td></td>
<td></td>
<td>dBm</td>
<td>Hybrid Enabled</td>
</tr>
<tr>
<td>AO2 Input Resistance</td>
<td>10</td>
<td></td>
<td></td>
<td>MOhms</td>
<td>-2.5V &lt; Vin &lt; +2.5V</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 = ±20%
Voltage Rating = 10V
Type = Non-Electrolytic, low leakage.

![Figure 8. Crystal Equivalent Circuit](image)

**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>12.96</td>
<td>+0.0035%</td>
<td>MHz</td>
<td>Refer to Figure 8</td>
</tr>
<tr>
<td>Rx</td>
<td></td>
<td>10</td>
<td>16</td>
<td>Ohms</td>
<td></td>
</tr>
<tr>
<td>Cx</td>
<td></td>
<td>0.024</td>
<td></td>
<td>pF</td>
<td></td>
</tr>
<tr>
<td>Co</td>
<td>5.1</td>
<td>5.6</td>
<td>6.1</td>
<td>pF</td>
<td>2 Load Capacitors</td>
</tr>
<tr>
<td>CL</td>
<td>-5%</td>
<td>33</td>
<td>+5%</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Crystal Type: Parallel 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 AO1 to Amp</td>
<td>-9</td>
<td></td>
<td></td>
<td>dB</td>
<td>Max</td>
</tr>
<tr>
<td></td>
<td>-18</td>
<td></td>
<td></td>
<td>dB</td>
<td>Mid</td>
</tr>
<tr>
<td></td>
<td>-26</td>
<td></td>
<td></td>
<td>dB</td>
<td>Min</td>
</tr>
<tr>
<td></td>
<td>-70</td>
<td></td>
<td></td>
<td>dB</td>
<td>Off</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Software Selectable</td>
</tr>
<tr>
<td>Audio Amp Gain (1) AO2 to Amp</td>
<td>+12</td>
<td></td>
<td></td>
<td>dB</td>
<td>Max</td>
</tr>
<tr>
<td></td>
<td>+3</td>
<td></td>
<td></td>
<td>dB</td>
<td>Mid</td>
</tr>
<tr>
<td></td>
<td>-4</td>
<td></td>
<td></td>
<td>dB</td>
<td>Min</td>
</tr>
<tr>
<td></td>
<td>-60</td>
<td></td>
<td></td>
<td>dB</td>
<td>Off</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Software 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.
**TRANSMIT LEVEL**

<table>
<thead>
<tr>
<th>TX 3,2,1,0</th>
<th>Typ</th>
<th>Units</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 0 0 0</td>
<td>+5</td>
<td>dBm</td>
</tr>
<tr>
<td>0 0 0 1</td>
<td>+4</td>
<td>dBm</td>
</tr>
<tr>
<td>•</td>
<td>•</td>
<td>•</td>
</tr>
<tr>
<td>•</td>
<td>•</td>
<td>•</td>
</tr>
<tr>
<td>1 1 1 0</td>
<td>−9</td>
<td>dBm</td>
</tr>
<tr>
<td>1 1 1 1</td>
<td>−10</td>
<td>dBm</td>
</tr>
</tbody>
</table>

**NOTE:**
1. For PSK and QAM transmit signal. For FSK, signal levels are typically 1 dB lower. All signals are measured at AO1.
2. The tolerance for the above transmit levels are ±1 dBm.

**MODEM REFERENCE MANUAL**

The Modem 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 Modem Chip Set.
89024 REVISION -005 HISTORY

The following differences exist between Rev. -004 and this version of the data sheet:

1. The internal code has been revised from 1.6 to 2.0. The following are the major changes and improvements:
   i) Expanded NVRAM (9346) support which allows storage and retrieval of four phone numbers and two user profiles. Commands supported are &n, DS, &Wh and Zn.
   ii) S11 DTMF duration and spacing register support.
   iii) The B command selects between V.21 and Bell 103 modes for 300 bps connections. The B/C pin on 89026 is ignored.
   iv) Factory defaults have been changed from &CO &DO to &C1, &D2 respectively.
   v) Expanded functionality of ATO command. ATO from the call program mode causes the modem to go into originate mode instead of answer mode. When off line, ATO gives "NO CARRIER" response, instead of going on line.

   vi) Changed response to test modes.
   vii) DTR transition received during idle state will not generate any result codes, and will not have any effect on the modem.
   viii) 600 bps operation is not supported.
   ix) Switch Hook (SH) option is not supported.
   x) LCLLB and REMLB loopback pins are not supported.

2. Receiver BER performance table has been expanded.

3. Frequency and Phase jitter specs have been removed.

4. 89024 Reference Manual has been superseded by the modem Reference Manual. This is a single manual for Intel's modem product line.
89C024LT
ERROR CORRECTING LAPTOP
MODEM CHIP SET

- CHMOS for Low Operating Power
- Low Standby Power Requirements
- Minimum Chip Count for Small size
- MNP* Operation through Class 4 for Error Correction
- MNP Class 5 Data Compression and DTE Interface Rates of 9600 bps Can Provide Increased Throughput up to 4:1
- Serial Command Set Compatible with Hayes** Smartmodem 2400**
- V.22 bis, V.22 A/B, V.21, Bell 212A, and Bell 103 Compatible
- For Public Switched Telephone Network and Unconditioned Leased Line Applications
- Automatically Detects Remote Modem Type and Data Rate
- On-Chip Hybrid
- DTMF and Pulse Dialing
- On-Chip Serial Port and Handshake Signals for RS-232/V.24 Interface
- Simple Serial Interface to External NVRAM
- Automatic Speed Matching in MNP and Normal Modes
- Hardware and Software Flow Control
- Analog/Digital Loopback Diagnostics with Mark/Space Pattern Generation and Error Detection
- Telephone Line Audio Monitor Output
- Full Set of Control Signals for DAA Interface
- International Call Progress Tone Detection Capabilities
- Automatic Adaptive Equalization
- Synchronous Modes
- Easily Customized Command Set and Features
- Intel's MNP Software Co-Developed with R. Scott Associates***

Figure 1. 89C024LT System Block Diagram

*MNP is a registered trademark of Microcom, Inc.
**Hayes, Smartmodem 2400, and Smartcom II are registered trademarks of Hayes Microcomputer Products, Inc.
***R. Scott Associates, Inc., 5711 Six Forks Road, Suite 301, Raleigh, North Carolina 27609, (919) 846-7171
GENERAL DESCRIPTION

Intel 89C024LT is a highly integrated, low power, error correcting laptop modem chip set. This two chip solution is composed of the 89027 Analog Front End and 89C026LT microcontroller. At 12.96 MHz the microcontroller is capable of executing error correction and data compression routines. The system is compatible with the following CCITT and BELL standards.

- CCITT V.22 bis
  2400 bps sync and async
  1200 bps sync and async
- CCITT V.22 A & B
  1200 bps sync and async
- 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 89C024LT system consists of a 16 bit application specific processor (89C026LT) and an analog front end device (89027). The 89C026LT 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/PSK/FSK modems. Refer to Figure 1 for a simplified block diagram of the system.

In laptop modem applications, the 89C024LT chip set along with a Data Access Arrangement (DAA), a single 8-bit EPROM, and an 8K x 8 static RAM represent the circuitry required for implementing an auto-dial, auto-answer, 300 to 2400 bps, MNP class 5 full duplex Hayes compatible intelligent modem. Refer to Figure 2 for a block diagram of this application.

A complete set of Industry Standard AT commands is provided for modem configuration and user interface. Additional commands have been implemented for power down modes and MNP feature control. 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 89C024LT command module with custom proprietary software.

The 89C024LT 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 optional external NVRAM with the &W command.

The 89C024LT 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.

![Figure 2. 89C024LT Laptop Modem Application](image-url)
PACKAGING

89027 is available in PLCC and standard plastic DIP packages. The 89C026LT is available in a PLCC package. Packages are shown from top view, looking down on component side of PC board.
CALL ESTABLISHMENT, TERMINATION AND RETRAIN

The 89C024LT modem system incorporates all protocols and functions required for automatic or manual call establishment. The modem system also incorporates all protocols and functions required for progress and termination of a data call.

The modem chip set has a built-in auto-dialer, both DTMF and Pulse type. It can detect dial, busy, and ringback signals from the remote end, and will provide call progress messages to the user. The modem is also capable of re-dialing the last number dialed.

The modem, when configured for auto-answer, will answer an incoming call, remain silent for the two second billing delay interval, and then transmit 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. A transition from voice (i.e., for the purpose of manual dialing) to data mode can be done by the use of a mechanical switch (exclusion key) on 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 is compatible with 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.

Because the CCITT and Bell modem connection protocols 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 89C024LT 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 89C024LT 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 separately or in string fashion. 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 89C024LT Modem</th>
<th>Answering Modem</th>
<th>Bell 300</th>
<th>Bell 1200</th>
<th>CCITT 300</th>
<th>CCITT 1200</th>
<th>CCITT 2400</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bell</td>
<td>Bell</td>
<td>300</td>
<td>300</td>
<td>—</td>
<td>300*</td>
<td>300*</td>
</tr>
<tr>
<td>1200</td>
<td>1200*</td>
<td>1200</td>
<td>1200</td>
<td>—</td>
<td>1200</td>
<td>1200</td>
</tr>
<tr>
<td>CCITT</td>
<td>CCITT</td>
<td>—</td>
<td>—</td>
<td>300</td>
<td>—</td>
<td>300</td>
</tr>
<tr>
<td>300</td>
<td>1200</td>
<td>1200</td>
<td>1200</td>
<td>—</td>
<td>1200</td>
<td>1200</td>
</tr>
<tr>
<td>1200</td>
<td>1200*</td>
<td>1200*</td>
<td>1200*</td>
<td>—</td>
<td>1200</td>
<td>1200</td>
</tr>
<tr>
<td>2400</td>
<td>1200*</td>
<td>1200*</td>
<td>1200*</td>
<td>—</td>
<td>1200</td>
<td>2400</td>
</tr>
</tbody>
</table>

### 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 300 and 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 If &amp;J1 option is selected, H1 will also switch the auxiliary relay</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>

### MNP Feature Control Command Set

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>\An</td>
<td>Maximum MNP Block Size</td>
</tr>
<tr>
<td>%An</td>
<td>Set Auto-Reliable Fallback Character</td>
</tr>
<tr>
<td>\Bn</td>
<td>Transmit Break</td>
</tr>
<tr>
<td>\Cn</td>
<td>Set Auto-Reliable Buffer</td>
</tr>
<tr>
<td>\Gn</td>
<td>Set Modem Port Flow Control</td>
</tr>
<tr>
<td>\Jn</td>
<td>Bits per Second Rate Adjust</td>
</tr>
<tr>
<td>\Kn</td>
<td>Set Break Control</td>
</tr>
<tr>
<td>\Nn</td>
<td>Set Operating Mode</td>
</tr>
<tr>
<td>\O</td>
<td>Originate Reliable Link</td>
</tr>
<tr>
<td>\Qn</td>
<td>Set Serial Port Flow Control</td>
</tr>
<tr>
<td>\Tn</td>
<td>Set Inactivity Timer</td>
</tr>
<tr>
<td>\U</td>
<td>Accept Reliable Link</td>
</tr>
<tr>
<td>\Vn</td>
<td>Modify Result Code Form</td>
</tr>
<tr>
<td>\Xn</td>
<td>Set XON/XOFF Pass-Through</td>
</tr>
<tr>
<td>\Y</td>
<td>Switch to Reliable Mode</td>
</tr>
<tr>
<td>\Z</td>
<td>Switch to Normal Mode</td>
</tr>
</tbody>
</table>

### & Command Set

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&amp;Cn</td>
<td>DCD Options</td>
</tr>
<tr>
<td>&amp;Dn</td>
<td>DTR Options</td>
</tr>
<tr>
<td>&amp;En</td>
<td>Fetch Factory Configuration Profile</td>
</tr>
<tr>
<td>&amp;Gn</td>
<td>Guard Tone</td>
</tr>
<tr>
<td>&amp;Jn</td>
<td>Telephone Jack Selection</td>
</tr>
<tr>
<td>&amp;Ln</td>
<td>Leased/Dial-up Line Selection</td>
</tr>
<tr>
<td>&amp;Mn</td>
<td>Async/Sync Mode Selection</td>
</tr>
<tr>
<td>&amp;Pn</td>
<td>Make/Break Pulse Ratio</td>
</tr>
<tr>
<td>&amp;Rn</td>
<td>RTS/CTS Options</td>
</tr>
<tr>
<td>&amp;Sn</td>
<td>DSR Options</td>
</tr>
<tr>
<td>&amp;Tn</td>
<td>Test Commands</td>
</tr>
<tr>
<td>&amp;Wn</td>
<td>Write Configuration to Non Volatile Memory</td>
</tr>
<tr>
<td>&amp;Xn</td>
<td>Sync Clock Source</td>
</tr>
<tr>
<td>&amp;Yn</td>
<td>Default NVRAM Profile Select</td>
</tr>
<tr>
<td>&amp;Zn</td>
<td>Store Telephone Number</td>
</tr>
</tbody>
</table>

### + Command Set

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>+En</td>
<td>Disable/Enable Power Down</td>
</tr>
<tr>
<td>+Tn</td>
<td>Time to Power Down</td>
</tr>
</tbody>
</table>
CONFIGURATION REGISTERS

The modem stores all the configuration information in a set of registers. Some registers are dedicated to a 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>Register</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>S0*</td>
<td>Ring to Answer</td>
</tr>
<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>DTMF Tone Duration</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>Modem 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>
<tr>
<td>S31*</td>
<td>Bit Mapped Options Register</td>
</tr>
</tbody>
</table>

NOTE:
* These S registers can be stored in the NVRAM.

DIALING

Dial modifiers are available for adding conditions to dialed phone numbers.

<table>
<thead>
<tr>
<th>Dial Modifiers</th>
</tr>
</thead>
<tbody>
<tr>
<td>P</td>
</tr>
<tr>
<td>R</td>
</tr>
<tr>
<td>T</td>
</tr>
<tr>
<td>S</td>
</tr>
<tr>
<td>W</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>@</td>
</tr>
</tbody>
</table>

Example:
Terminal: AT &Z0 = T 1 (602) 555-1212
Modem: OK
Result: Modern stores the Tone Dial (T) modifier and phone number T16025551212 in the external NVRAM.

The number can be dialed from asynchronous mode by issuing the following command:
Terminal: AT DS0
Modem: T16025551212
Result: Modern dials phone number and attempts to establish a connection.

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.

POWER MANAGEMENT

The flexible power management controls allow for a variety of command and hardware driver options. The power down sequence is initiated by placing a logic "low" on pin 15 (PD) of the 89C026LT. The laptop can control the PD signal directly. If such a signal is unavailable, PD can be controlled by communications software via DTR. Lack of data activity or an incoming ring signal can also be used to control PD.

Placing the crystal on the 89C026LT (Figure 10) allows it to reduce power consumption by turning off the oscillator. When online and connected to a remote modem, the power consumption for the 89C024LT is typically 400 mW. Additionally, when the 89027 is not needed (on-hook, not connected to a remote modem) the 89C026LT places it in standby. In standby the chip set power consumption is typically 255 mW. When powered down via the PD pin on the 89C026LT, the chip set typically consumes 5 mW. Minimum memory-system power-consumption can be achieved by chip selecting memory only when addressed.

APPLICATIONS OVERVIEW

The block diagram of a stand-alone 300 to 2400 bps Hayes compatible modem is depicted in Figure 4. The DAA section shown in this diagram may be implemented using the suggested diagram in Figure 5. Figure 6 shows the use of the power-down feature.
Figure 4. Typical Laptop Modem

Figure 5. Typical Telephone Line Interface Using Internal Hybrid

Figure 6. Power-Down Control
## 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><strong>Asynchronous</strong></td>
<td>2400, 1200 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>10 bits, including start, stop, parity. (8, 9, 11 bits optional via S/W customization.)</td>
</tr>
<tr>
<td><strong>Synchronous Timing Source</strong></td>
<td>a) Internal, derived from the local oscillator.</td>
</tr>
<tr>
<td></td>
<td>b) External, provided by DTE through XTCLK.</td>
</tr>
<tr>
<td></td>
<td>c) 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.</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 PSK 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>V.22 bis, V.22, 212A</td>
</tr>
<tr>
<td></td>
<td>Originate 1200 Hz ± 0.2%</td>
</tr>
<tr>
<td></td>
<td>Answer 2400 Hz ± 0.2%</td>
</tr>
<tr>
<td></td>
<td>V.21</td>
</tr>
<tr>
<td></td>
<td>Originate 'space' 1180 Hz ± 0.2%</td>
</tr>
<tr>
<td></td>
<td>Originate 'mark' 980 Hz ± 0.2%</td>
</tr>
<tr>
<td></td>
<td>Answer 'space' 1850 Hz ± 0.2%</td>
</tr>
<tr>
<td></td>
<td>Answer 'mark' 1650 Hz ± 0.2%</td>
</tr>
<tr>
<td></td>
<td>Bell 103 mode</td>
</tr>
<tr>
<td></td>
<td>Originate 'space' 1070 Hz ± 0.2%</td>
</tr>
<tr>
<td></td>
<td>Originate 'mark' 1270 Hz ± 0.2%</td>
</tr>
<tr>
<td></td>
<td>Answer 'space' 2020 Hz ± 0.2%</td>
</tr>
<tr>
<td></td>
<td>Answer 'mark' 2225 Hz ± 0.2%</td>
</tr>
<tr>
<td><strong>Received Signal Frequency Tolerance</strong></td>
<td>V.22 bis, V.22, 212A</td>
</tr>
<tr>
<td></td>
<td>Originate 2400 Hz ± 7 Hz</td>
</tr>
<tr>
<td></td>
<td>Answer 1200 Hz ± 7 Hz</td>
</tr>
<tr>
<td></td>
<td>V.21</td>
</tr>
<tr>
<td></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></td>
<td>Bell 103</td>
</tr>
<tr>
<td></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>Typical Energy Detect Sensitivity</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Greater than −43 dBm ED is ON. Less than −48 dBm ED is OFF. Signal in dBm measured at AO2.</td>
</tr>
<tr>
<td><strong>Energy Detect Hysteresis</strong></td>
<td>A minimum Hysteresis of 2 dB for QAM scrambled mark.</td>
</tr>
<tr>
<td><strong>Line Equalization</strong></td>
<td>Fixed compromise equalization, transmit. Adaptive equalizer for PSK/QAM, receive.</td>
</tr>
<tr>
<td><strong>Diagnostics Available</strong></td>
<td>Local analog loopback. Local digital loopback. Remote digital 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>
## RECEIVER PERFORMANCE SPECIFICATIONS

<table>
<thead>
<tr>
<th>Data Mode</th>
<th>Rx Level (dBm)</th>
<th>Answer (dB)</th>
<th>Originate (dB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>V.22 bis</td>
<td>−30</td>
<td>16</td>
<td>16.5</td>
</tr>
<tr>
<td>Synchronous</td>
<td>−40</td>
<td>16.5</td>
<td>18</td>
</tr>
<tr>
<td>V.22/Bell 212A</td>
<td>−30</td>
<td>6.5</td>
<td>6.5</td>
</tr>
<tr>
<td>Synchronous</td>
<td>−40</td>
<td>6.5</td>
<td>6.5</td>
</tr>
<tr>
<td>V.21</td>
<td>−30</td>
<td>9</td>
<td>7.5</td>
</tr>
<tr>
<td>Asynchronous</td>
<td>−40</td>
<td>9</td>
<td>8</td>
</tr>
<tr>
<td>Bell 103</td>
<td>−30</td>
<td>10</td>
<td>11.5</td>
</tr>
<tr>
<td>Asynchronous</td>
<td>−40</td>
<td>10</td>
<td>11.5</td>
</tr>
</tbody>
</table>

**Test Conditions:**
- Receive Signal (Rx) measured at A02 (transmit level set at −9 dBm)
- Unconditioned 3002 Line
- 3 KHz Flat-Band 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>4.0</td>
<td>dBm</td>
<td></td>
<td></td>
<td>at A01</td>
</tr>
<tr>
<td>DTMF Second Harmonic</td>
<td></td>
<td>−35</td>
<td>dB</td>
<td></td>
<td>HYB enabled into 600Ω</td>
</tr>
<tr>
<td>DTMF Twist (Balance)</td>
<td>3</td>
<td>dB</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Default DTMF Duration</td>
<td>100</td>
<td>ms</td>
<td></td>
<td></td>
<td>Software Controlled</td>
</tr>
<tr>
<td>Pulse Dialing Rate</td>
<td>10/20</td>
<td>pps</td>
<td></td>
<td></td>
<td>Software Controlled</td>
</tr>
<tr>
<td>Pulse Dialing Make/Break</td>
<td>39/61</td>
<td>%</td>
<td></td>
<td></td>
<td>US</td>
</tr>
<tr>
<td></td>
<td>33/67</td>
<td>%</td>
<td></td>
<td></td>
<td>UK, Hong Kong</td>
</tr>
<tr>
<td>Pulse Interdigit Interval</td>
<td>785</td>
<td>ms</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Billing Delay Interval</td>
<td>2.1</td>
<td>sec</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Guard Tone Frequency Amplitude</td>
<td>540</td>
<td>Hz</td>
<td>−3</td>
<td>dB</td>
<td>referenced to High Channel transmit.</td>
</tr>
<tr>
<td></td>
<td>1800</td>
<td>Hz</td>
<td>−6</td>
<td>dB</td>
<td>QAM/PSK Modes Only</td>
</tr>
<tr>
<td>Dial Tone Detect Duration</td>
<td>3.0</td>
<td>sec</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Ringback Tone Detect Duration</td>
<td>0.75</td>
<td>sec</td>
<td></td>
<td></td>
<td>Off/On Ratio</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>sec</td>
<td>0.2</td>
<td>1.5</td>
<td>Off/On Ratio</td>
</tr>
<tr>
<td>Cadence</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
89C026LT OVERVIEW

The 89C026LT processor performs data manipulation, signal processing and user interface functions. It requires a single ROM and RAM to execute standard, and/or custom code with high level protocol functions. A block diagram of the 89C026LT is provided in Figure 7.

89C026LT contains a TTL compatible serial link to DTE equipment, along with a full complement of V.24/RS-232-C control signals. A UART or USART may be used to transfer data to and from a microcomputer bus. The 89C026LT supports the industry standard AT command set facilitating compatibility with most PC software.

During transmit operation, the 89C026LT 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 89C026LT transmits digitized phase and amplitude samples to 89027 over the high speed serial link.

In the receive operation, the information is received by the 89C026LT 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 89C026LT. By using DSP algorithms, the received signals are processed using adaptive equalization for telephone line delay, amplitude distortion and gain adjustments and the signal demodulated. Following demodulation, the data is unscrambled, and if necessary, returned to asynchronous format.
### 89C026LT PINOUT

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Function (89C024LT)</th>
<th>Direction(4)</th>
<th>Pin No.</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLKIN</td>
<td>12.96 MHz master clock from 89027</td>
<td>In</td>
<td>67</td>
</tr>
<tr>
<td>CLKIN2</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>PD</td>
<td>Power-down control</td>
<td>In</td>
<td>15</td>
</tr>
<tr>
<td>S/Å</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>Reserved for future use (VCC)(2)</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>CDE</td>
<td>Clock detect enable (VSS)</td>
<td>GND</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>VPP</td>
<td>Timing pin for return from power-down</td>
<td>In</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(3)</td>
<td>Out</td>
<td>60</td>
</tr>
<tr>
<td>JS</td>
<td>Jack Select(3)</td>
<td>Out</td>
<td>59</td>
</tr>
<tr>
<td>CD</td>
<td>Carrier Detect Indicator(3)</td>
<td>Out</td>
<td>58</td>
</tr>
<tr>
<td>MR</td>
<td>Modern Ready Indicator(3)</td>
<td>Out</td>
<td>57</td>
</tr>
<tr>
<td>REL</td>
<td>MNP Reliable Link Active(3)</td>
<td>Out</td>
<td>56</td>
</tr>
<tr>
<td>CLASS5</td>
<td>MNP Class 5 Compression Active(3)</td>
<td>Out</td>
<td>55</td>
</tr>
<tr>
<td>ERR</td>
<td>Error detected by MNP(3)</td>
<td>Out</td>
<td>54</td>
</tr>
</tbody>
</table>
89C026LT PINOUT (Continued)

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Function (89C026LT)</th>
<th>Direction(4)</th>
<th>Pin No.</th>
</tr>
</thead>
<tbody>
<tr>
<td>NMI</td>
<td>Non-maskable Interrupt(VSS)(1)</td>
<td>In</td>
<td>3</td>
</tr>
<tr>
<td>X2</td>
<td>Crystal output</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>BUSWIDTH</td>
<td>Bus Width</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</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 (VCC) must be connected to VCC.
3. A0–A3 are used as A A, JS, CD, MR, REL, CLASS 5, and ERR respectively.
4. Pins with direction "IN" must not be left floating.

89C026LT 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, 89C026LT samples this data on the rising edges of TCLK.

TCLK
Clock output from 89C026LT 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. A logic 'high' is mark. 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.

PD
Power-down control. A low on this input pin, in conjunction with the + En and + Tn commands, will cause the modem to go into a power-down mode.

Vpp
Timing pin for return from power-down. Connect a 1.0 µF capacitor between Vpp and Vss if the power-down option is used. This capacitor causes an internal timing circuit to give the oscillator time to stabilize before turning on internal clocks. This pin may be left floating or connected through a 1.0 µF capacitor to VSS if power-down mode is not required.

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
A low indicates modem is off-hook, 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.
**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 as transferred 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 sets an off hook condition, high indicates an on hook. When dialing, this signal 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 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 low on this pin initiates a remote loopback condition.

**CD**
A Low indicates the presence of carrier signal on the line.

**MR**
A low indicates the presence of the DSR signal. Toggling indicates that a test mode is active.

**REL**
A low indicates that an MNP reliable link has been established.

**CLASS5**
A low indicates that MNP Class 5 (data compression) is in operation.

**ERR**
Goes low for 1 second whenever MNP detects an error.

**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.

**CDE**
This pin must be connected to VSS.

**S/A**
The function of this pin is re-defined as external NVRAM CE.

**CONFIG**
Reserved for future use. This signal should be pulled high.

**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. This pin must be tied high.

**JS**
Low is used to pulse A and A1 leads to control a 1A2 Key System jack.

**BUSWIDTH**
When high, external memory accesses are 16 bits wide. When low, external memory accesses are 8 bits wide. This pin must be tied low.

**READY**
When high, no wait states are inserted in external memory accesses. When low, one wait state is inserted in each external memory access.
**89C026LT ABSOLUTE MAXIMUM RATINGS**

- Temperature Under Bias: 0°C to +70°C
- Storage Temperature: -40°C to +125°C
- Voltage from Any Pin to $V_{SS}$ or AGND: -0.5V to +7.0V
- Average Output Current from Any Pin: 10 mA
- Power Dissipation: 1.5 Watts

**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>$T_A$</td>
<td>Ambient Temperature Under Bias</td>
<td>0</td>
<td>+70</td>
<td>°C</td>
</tr>
<tr>
<td>$V_{CC}$</td>
<td>Digital Supply Voltage</td>
<td>4.75</td>
<td>5.25</td>
<td>V</td>
</tr>
<tr>
<td>$V_{REF}$</td>
<td>Analog Supply Voltage</td>
<td>4.75</td>
<td>5.25</td>
<td>V</td>
</tr>
<tr>
<td>$f_{OSC}$</td>
<td>CLKIN Frequency</td>
<td>12.95870</td>
<td>12.96130</td>
<td>MHz</td>
</tr>
</tbody>
</table>

**NOTE:**
The AGND and $V_{SS}$ on both the 89C026LT and the 89027 must be nominally at the same potential.

**D.C. CHARACTERISTICS**

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Parameter</th>
<th>Min</th>
<th>Typ(7)</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>
<td></td>
</tr>
<tr>
<td>$V_{IH}$</td>
<td>Input High Voltage(1)</td>
<td>0.2</td>
<td>$V_{CC}$ + 0.9</td>
<td>$V_{CC}$ + 0.5</td>
<td>V</td>
<td></td>
</tr>
<tr>
<td>$V_{IH1}$</td>
<td>Input High Voltage on CLKIN</td>
<td>0.7</td>
<td>$V_{CC}$</td>
<td>$V_{CC}$ + 0.5</td>
<td>V</td>
<td></td>
</tr>
<tr>
<td>$V_{IH2}$</td>
<td>Input High Voltage on RESET</td>
<td>2.2</td>
<td>$V_{CC}$ + 0.5</td>
<td>V</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$V_{OL}$</td>
<td>Output Low Voltage</td>
<td>0.3</td>
<td>0.45</td>
<td>V</td>
<td></td>
<td>$I_{OL} = 200 \mu A$</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1.5</td>
<td>V</td>
<td></td>
<td>$I_{OL} = 1.5 V$</td>
</tr>
<tr>
<td>$V_{OH}$</td>
<td>Output High Voltage(4)</td>
<td>$V_{CC} - 0.3$</td>
<td>$V_{CC} - 0.7$</td>
<td>$V_{CC} - 1.5$</td>
<td>V</td>
<td></td>
</tr>
<tr>
<td>$V_{OH1}$</td>
<td>Output High Voltage(3)</td>
<td>$V_{CC} - 0.3$</td>
<td>$V_{CC} - 0.7$</td>
<td>$V_{CC} - 1.5$</td>
<td>V</td>
<td></td>
</tr>
<tr>
<td>$I_{IL}$</td>
<td>Input Leakage Current(5)</td>
<td>±10</td>
<td>µA</td>
<td>0 &lt; $V_{IN} &lt; V_{CC} - 0.3$ V</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$I_{IL1}$</td>
<td>Input Leakage Current(6)</td>
<td>±3</td>
<td>µA</td>
<td>0 &lt; $V_{IN} &lt; V_{REF}$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$I_{IL2}$</td>
<td>Logical 0 Input Current(3)</td>
<td>-50</td>
<td>µA</td>
<td>$V_{IN} = 0.45 V$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$I_{IL1}$</td>
<td>Logical 0 Input Current in RESET(2) (ALE, RŘ, WR, BHE, INST, SCLK)</td>
<td>-850</td>
<td>µA</td>
<td>$V_{IN} = 0.45 V$</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
## D.C. CHARACTERISTICS (Continued)

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Parameter Description</th>
<th>Min</th>
<th>Typ(7)</th>
<th>Max</th>
<th>Units</th>
<th>Test Conditions</th>
</tr>
</thead>
<tbody>
<tr>
<td>IREF</td>
<td>A/D Converter Reference Current</td>
<td>2</td>
<td>5</td>
<td>mA</td>
<td></td>
<td>CLKIN = 12.96 MHz, VCC = VPP = VREF = 5.25</td>
</tr>
<tr>
<td>ICC1</td>
<td>Active Mode Current (Typical)</td>
<td>45</td>
<td>60</td>
<td>mA</td>
<td></td>
<td>CLKIN = 12.96 MHz</td>
</tr>
<tr>
<td>RRSST</td>
<td>RESET Pullup Resistor</td>
<td>6K</td>
<td>50K</td>
<td>Ω</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Cs</td>
<td>Pin Capacitance (Any Pin to VSS)</td>
<td>10</td>
<td></td>
<td>pF</td>
<td></td>
<td>fTEST = 1.0 MHz</td>
</tr>
<tr>
<td>IPD</td>
<td>Power-Down Mode Current</td>
<td>5</td>
<td>50</td>
<td>μA</td>
<td></td>
<td>VCC = VPP = VREF = 5.25</td>
</tr>
</tbody>
</table>

**NOTES:**

1. All pins except RESET and CLKin.
2. Holding these pins below VIH in RESET may cause the part to enter test modes.
3. TCK, TCI, S/A, RTS, CTS, DSR, DCD, SI, OH.
4. BHE, INST, CLKOUT, RESET, TCK, RXD, TXCLK, HS, B/C, CLKIN2, and RI.
5. CDE, EA, READY, BUSWIDTH, NMI, STR, DTR, XTCLK, TXD, B/C, CLKIN2, and RI.
6. S/D, SH, REMLB, LCLLB, I, Q, Config, ED.
7. Typicals are based on a limited number of samples and are not guaranteed. The values listed are at room temperature and VREF = VCC = 5V.

## A.C. CHARACTERISTICS (Over specified operating conditions)

Test Conditions: Capacitive load on all pins = 100 pF, Rise and fall times = 10 ns, fOSC 12.96 MHz

This system must meet these specifications to work with 89C026LT:

<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>TAVYV</td>
<td>Address Valid to READY Setup</td>
<td>2TOSC - 85</td>
<td>ns</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TLVYV</td>
<td>ALE Low to READY Setup</td>
<td>TOSC - 72</td>
<td>ns</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TLYLH</td>
<td>Non READY Time</td>
<td>No Upper Limit</td>
<td>ns</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TCKX</td>
<td>READY Hold after CLKOUT Low</td>
<td>0</td>
<td>TOSC - 30</td>
<td>ns</td>
<td>(Note 1)</td>
</tr>
<tr>
<td>TLLX</td>
<td>READY Hold after ALE Low</td>
<td>TOSC - 15</td>
<td>2 TOSC - 40</td>
<td>ns</td>
<td>(Note 1)</td>
</tr>
<tr>
<td>TAVGV</td>
<td>Address Valid to Buswidth Setup</td>
<td>2 TOSC - 85</td>
<td>ns</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TLLGV</td>
<td>ALE Low to Buswidth Setup</td>
<td>TOSC - 70</td>
<td>ns</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TCLGX</td>
<td>Buswidth Hold after CLKOUT Low</td>
<td>0</td>
<td>ns</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TAVDV</td>
<td>Address Valid to Input Data Valid</td>
<td>3 TOSC - 67</td>
<td>ns</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TRLDV</td>
<td>RD Active to Input Data Valid</td>
<td>TOSC - 23</td>
<td>ns</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TLCVD</td>
<td>CLKOUT Low to Input Data Valid</td>
<td>TOSC - 50</td>
<td>ns</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TRHDZ</td>
<td>End of RD to Input Data Float</td>
<td>TOSC - 20</td>
<td>ns</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TRD X</td>
<td>Data Hold after RD Inactive</td>
<td>0</td>
<td>ns</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**NOTE:**

1. If max is exceeded, additional wait states will occur.
A.C. CHARACTERISTICS (Continued)
Test Conditions: Capacitive load on all pins = 100 pF, Rise and fall times = 10 ns, fOSC 12.96 MHz

The 89C026LT 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>Oscillator Frequency</td>
<td>12.95870</td>
<td>12.96130</td>
<td>MHz</td>
<td></td>
</tr>
<tr>
<td>TOSC</td>
<td>Oscillator Period</td>
<td>1/FCLKIN(MAX)</td>
<td>1/FCLKIN(MIN)</td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>TXHCH</td>
<td>FCLKIN High to CLKOUT High or Low</td>
<td>40</td>
<td>110</td>
<td>ns</td>
<td>(Note 1)</td>
</tr>
<tr>
<td>TCLCL</td>
<td>CLKOUT Cycle Time</td>
<td>2 TOSC</td>
<td></td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>TCHCL</td>
<td>CLKOUT High Period</td>
<td>TOSC - 10</td>
<td>TOSC + 10</td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>TCHLH</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>TLHLL</td>
<td>ALE High Period</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 - 20</td>
<td></td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>TLLAX</td>
<td>Address Hold after ALE Falling Edge</td>
<td>TOSC - 40</td>
<td></td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>TLLRL</td>
<td>ALE Falling Edge to RD Falling Edge</td>
<td>TOSC - 40</td>
<td></td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>TRLCL</td>
<td>RD Low to CLKOUT Falling Edge</td>
<td>5</td>
<td>30</td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>TRLRH</td>
<td>RD Low Period</td>
<td>TOSC - 5</td>
<td>TOSC + 25</td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>TRHLL</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>TRLAZ</td>
<td>RD Low to Address Float</td>
<td></td>
<td>10</td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>TLLWL</td>
<td>ALE Falling Edge to WR Falling Edge</td>
<td>TOSC - 10</td>
<td></td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>TCLWL</td>
<td>CLKOUT Low to WR Falling 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 - 23</td>
<td></td>
<td>ns</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>TOSC + 5</td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>TWHOX</td>
<td>Data Hold after WR Rising Edge</td>
<td>TOSC - 10</td>
<td></td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>TWHLL</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></td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>TRHBX</td>
<td>BHE, INST, Hold after RD Rising Edge</td>
<td>TOSC - 10</td>
<td></td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>TWHAX</td>
<td>AD8–15 Hold after WR Rising Edge</td>
<td>TOSC - 50</td>
<td></td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>TRHAX</td>
<td>AD8–15 Hold after RD Rising Edge</td>
<td>TOSC - 25</td>
<td></td>
<td>ns</td>
<td></td>
</tr>
</tbody>
</table>

NOTES:
1. Typical specifications, not guaranteed.
Figure 8. Bus Signal Timings
89027 OVERVIEW

The 89027 is a 28 pin CHMOS 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 9. Most of the analog signal processing functions in this chip are implemented with CHMOS switched capacitor technology. The 89027 functions are controlled by 89C026LT, through a high speed serial data link.

During FSK transmit operation, the 89027 receives digitally synthesized mark and space sinusoid amplitude information from the 89C026LT. 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 89C026LT 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 9. 89027 Block Diagram**

![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>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>+5V</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 89C026LT</td>
<td>Out</td>
<td>26</td>
</tr>
<tr>
<td>CLKOUT2</td>
<td>270 KHz Clock Output to 89C026LT</td>
<td>Out</td>
<td>19</td>
</tr>
<tr>
<td>RST</td>
<td>Chip reset (active low)(3)</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 89C026LT</td>
<td>In</td>
<td>2</td>
</tr>
<tr>
<td>SCLK</td>
<td>Serial clock from 89C026LT</td>
<td>In</td>
<td>1</td>
</tr>
<tr>
<td>TSYNC</td>
<td>Transmitter sync from 89C026LT</td>
<td>In</td>
<td>3</td>
</tr>
<tr>
<td>STR</td>
<td>Symbol timing to 89C026LT</td>
<td>Out</td>
<td>27</td>
</tr>
<tr>
<td>ED</td>
<td>Receiver energy detect to 89C026LT</td>
<td>Out</td>
<td>18</td>
</tr>
<tr>
<td>I</td>
<td>In phase received signal to 89C026LT</td>
<td>Out</td>
<td>13</td>
</tr>
<tr>
<td>Q</td>
<td>Quadrature-phase received signal to 89C026LT</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>
<tr>
<td>NC</td>
<td>(Note 2)</td>
<td>In</td>
<td>22</td>
</tr>
</tbody>
</table>

**NOTE:**
1. When held high, these pins must be connected through 10K resistors to VCC.
2. Reserved Pin. Must be left No Connect.
3. Connect to reset circuitry through a 10K resistor.

### 89027 Pinout Description

#### TX0-3
These four pins control the transmitted signal level. Refer to Transmit Level Table.

#### 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(2)

Temperature Under Bias ............. 0 to +70° 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
  to VSS(1) ............. −0.3V to 6.5V

NOTES:
1. Applies to pins SCLK, SDATA, TSYNC, RST, HYB, TX0–TX3 only.
2. 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.

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>VBB</td>
<td>Analog Supply Voltage</td>
<td>−4.75</td>
<td>−5.25</td>
<td>V</td>
</tr>
</tbody>
</table>

POWER DISSIPATION  
Ambient Temp = 0° to 70° C, VCC = AVCC = 5 ± 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>15</td>
<td>21</td>
<td>mA</td>
<td></td>
</tr>
<tr>
<td>Icc1</td>
<td>VCC Operating Current</td>
<td>5</td>
<td>6</td>
<td>mA</td>
<td></td>
</tr>
<tr>
<td>Ibb1</td>
<td>VBB Operating Current</td>
<td>−15</td>
<td>−21</td>
<td>mA</td>
<td></td>
</tr>
<tr>
<td>Alccs</td>
<td>AVCC 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>5</td>
<td>6</td>
<td>mA</td>
<td></td>
</tr>
<tr>
<td>Ibs</td>
<td>VBB Standby Current</td>
<td>−0.6</td>
<td>−2</td>
<td>mA</td>
<td></td>
</tr>
<tr>
<td>Alccp</td>
<td>AVCC Power-Down Current</td>
<td>100</td>
<td></td>
<td>μA</td>
<td></td>
</tr>
<tr>
<td>Iccp</td>
<td>VCC Power-Down Current</td>
<td>450</td>
<td></td>
<td>μA</td>
<td></td>
</tr>
<tr>
<td>Ibpp</td>
<td>VBB Power-Down Current</td>
<td>450</td>
<td></td>
<td>μA</td>
<td></td>
</tr>
<tr>
<td>Pdo</td>
<td>Operating Power Dissipation</td>
<td>175</td>
<td>250</td>
<td>mW</td>
<td></td>
</tr>
<tr>
<td>Pds</td>
<td>Standby Power Dissipation</td>
<td>30</td>
<td>50</td>
<td>mW</td>
<td></td>
</tr>
<tr>
<td>Pdp</td>
<td>Power Down Power Dissipation</td>
<td>5</td>
<td></td>
<td>mW</td>
<td></td>
</tr>
</tbody>
</table>
D.C. 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 89C026LT power supply. Typical Values are for TA = 25°C and nominal power supply values. VCC, and AVCC. VCC, AVCC and 89C026LT 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 Conditions</th>
</tr>
</thead>
<tbody>
<tr>
<td>ilI</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>2.0</td>
<td>VCC</td>
<td>V</td>
<td></td>
</tr>
<tr>
<td>VOH</td>
<td>Output High Voltage</td>
<td>0.4</td>
<td>V</td>
<td>V</td>
<td>ioH ≤ 50μA, 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>ioL ≥ −1.6mA, 1 TTL load</td>
</tr>
<tr>
<td>VCOL</td>
<td>CLKOUT Low Voltage</td>
<td>0.4</td>
<td>V</td>
<td>V</td>
<td>Load Capacitance = 60 pF</td>
</tr>
<tr>
<td>VCOH</td>
<td>CLKOUT High Voltage</td>
<td>0.7 VCC</td>
<td>V</td>
<td>V</td>
<td>Load Capacitance = 60 pF</td>
</tr>
</tbody>
</table>

A.C. CHARACTERISTICS (TA = 25°C, VCC = AVCC = 5V, VSS = AGND = 0V, VBB = −5V)

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 Receive Signal Level</td>
<td>-9</td>
<td>-9</td>
<td>-9</td>
<td>dBm</td>
<td>Hybrid Enabled</td>
</tr>
<tr>
<td>AO2 Input Resistance</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>MOhms</td>
<td>−2.5V &lt; Vin &lt; +2.5V</td>
</tr>
<tr>
<td>AO2 Allowed DC offset</td>
<td>-30</td>
<td>+30</td>
<td>+30</td>
<td>mV</td>
<td>Relative to AGND</td>
</tr>
</tbody>
</table>

AUTO ZERO CAPACITANCE

Capacitance = 0.015 μF
Tolerance = ±20%
Voltage Rating = 10V
Type = Non-Electrolytic, low leakage.
**CRYSTAL REQUIREMENTS**(1)

<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>Frequency Accuracy (0°C−70°C)</td>
<td>−0.0035%</td>
<td>12.96</td>
<td>+0.0035%</td>
<td>MHz</td>
<td>Refer to Figure 10</td>
</tr>
<tr>
<td>Rx</td>
<td>10</td>
<td>16</td>
<td></td>
<td>Ohms</td>
<td></td>
</tr>
<tr>
<td>Cx</td>
<td>0.024</td>
<td></td>
<td></td>
<td>pF</td>
<td></td>
</tr>
<tr>
<td>Co</td>
<td>5.1</td>
<td>5.6</td>
<td>6.1</td>
<td>pF</td>
<td></td>
</tr>
<tr>
<td>$C_L$(2)</td>
<td>−5%</td>
<td>47</td>
<td>+5%</td>
<td>pF</td>
<td></td>
</tr>
</tbody>
</table>

**NOTES:**

1. Crystal Type: Parallel Resonant
2. Crystal manufacturers usually specify the accuracy of a parallel resonant circuit at a given “load capacitance”. This “load capacitance” is specified to the crystal manufacturer as 33 pF. 33 pF includes the parallel combination of the capacitances seen at the pins of the crystal. These capacitances include $C_L$, IC pin capacitances, and a 3 ± 2 pF trace capacitance.

---

**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>10</td>
<td></td>
<td>Ohms</td>
<td></td>
</tr>
<tr>
<td>AMP</td>
<td></td>
<td></td>
<td>100</td>
<td>kOhms</td>
<td></td>
</tr>
<tr>
<td>Load Capacitance</td>
<td></td>
<td></td>
<td></td>
<td>pF</td>
<td></td>
</tr>
<tr>
<td>AMP</td>
<td></td>
<td></td>
<td>100</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Audio Amp Gain</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>AO1 to Amp</td>
<td>−9</td>
<td>−18</td>
<td>−26</td>
<td>dB</td>
<td>Max</td>
</tr>
<tr>
<td></td>
<td>−70</td>
<td></td>
<td></td>
<td>dB</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>−4</td>
<td>−60</td>
<td>dB</td>
<td></td>
</tr>
<tr>
<td>Audio Amp Gain(1)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>AO2 to Amp</td>
<td>+12</td>
<td>+3</td>
<td>−4</td>
<td>dB</td>
<td>Max</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 Output Level(1)

<table>
<thead>
<tr>
<th>TX 3,2,1,0</th>
<th>Typ</th>
<th>Units</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 0 0 0</td>
<td>+5</td>
<td>dBm</td>
</tr>
<tr>
<td>0 0 0 1</td>
<td>+4</td>
<td>dBm</td>
</tr>
<tr>
<td>•</td>
<td>•</td>
<td>•</td>
</tr>
<tr>
<td>•</td>
<td>•</td>
<td>•</td>
</tr>
<tr>
<td>1 1 1 0</td>
<td>−9</td>
<td>dBm</td>
</tr>
<tr>
<td>1 1 1 1</td>
<td>−10</td>
<td>dBm</td>
</tr>
</tbody>
</table>

**NOTE:**
1. For PSK and QAM transmit signal. For FSK transmit signal levels, they are typically 1 dB lower. All signals are measured at A01. The tolerance for the transmit levels is ±1 dBm.

### 89C024XE CHIP SET USERS:

Note the following when using the 89C024LT Laptop chip set in place of the 89C024XE:

1. PD (B/C on 89C024XE) can be left tied to digital VCC or can be connected to the power-down control. If the power down mode is used, a 1.0 μf capacitor must be connected from VPP on the 89C026LT to digital ground.

2. Buswidth pin is tied directly to ground because 89C024LT uses 8-bit EPROM.

3. No 74HC373 latch is required for AD8–AD15 because 89C026LT latches them internally.

4. A single chip select inverter is required from AD15 to select between EP ROM and RAM memory.

5. Optional inverter from AD14 can be used to power-down RAM.

6. If minimum power-down currents are not required, the 89C024XE crystal configuration and specifications may be used.

---

### REFERENCE MANUALS

The Modem Reference Manual (Order Number 296235-002) contains pin descriptions, AT and MNP command descriptions, schematics, and important design guidelines for the 89C024LT, 89C024XE, and 89024 modem chip sets. The Modem Software Reference Manual (Order Number 296503-001) provides information about the modem software routines. Contact your local Intel sales office for the latest information.
89C024FT
V.42/42bis MODEM CHIP-SET

- CHMOS for Low Operating Power
- Low Standby Power
- Minimum Chip Count for Small size
- V.42 Compliant Error Correction (LAPM and *MNP4)
- V.42bis and MNP5 Data Compression Increase throughput up to 4 times with DTE rates of 9600
- AT Command Set
- Automatically Detects Remote Modem Type and Data Rate
- On-Chip Hybrid
- DTMF and Pulse Dialing
- On-Chip Serial Port and Handshake Signals for RS-232/V.24 Interface
- Serial Interface to External NVRAM
- Automatic Speed Matching in Reliable and Normal Modes
- Hardware and Software Flow Control
- Analog/Digital Loopback Diagnostics
- Synchronous Modes
- Easily Customized Command Set and Features
- Intel's V.42/42bis and MNP Software Co-Developed with R. Scott Associates**

Figure 1. 89C024FT System Block Diagram

*MNP is a registered trademark of Microcom, Inc.
**R. Scott Associates, Inc., 5711 Six Forks Road, Suite 301, Raleigh, North Carolina 27609, (919) 846-7171

Intel Corporation assumes no responsibility for the use of any circuitry other than circuitry embodied in an Intel product. No other circuit patent licenses are implied. Information contained herein supersedes previously published specifications on these devices from Intel. September 1990
© INTEL CORPORATION, 1990
Order Number: 290398-001
4-46
GENERAL DESCRIPTION

The Intel 89C024FT is a highly integrated, low power V.42/42bis compliant modem chip-set. This two chip solution is composed of the 89027 analog front end and the 89C026FT (an 80C196 based microcontroller).

V.42/42bis compliancy assures adherence to an international error correction (V.42: LAPM & MNP class 4) and data compression (V.42bis: BTLZ & MNP class 5) standards. V.42bis uses BTLZ (British Telecom Lempel Ziv) data compression algorithm to achieve throughputs of up to 4 times the transmission rate, effectively providing up to 9600 bps throughput with a 2400 bps modem.

The benefits of V.42/42bis compliancy are, compatibility with the installed base of MNP class 4 modems, and an increased throughput of up to 4:1. The chip-set also provides MNP class 5 operation. This provides 2:1 compression with the large installed base of MNP class 5 modems. These benefits allow the 89C024FT chip-set to provide fast and reliable data transfer with the current and upcoming installed base of modems products.

The 89C024FT chip-set, along with a Data Access Arrangement (DAA), a single 64K x 8 EPROM, a 32K x 8 static RAM, represent the circuitry necessary for implementing an auto dial/answer, 300 to 2400 bps, V.42/42bis compliant modem. Refer to figure 2 for a block diagram of this application. The system is compatible with the following CCITT and BELL transmission standards:

- CCITT V.22bis
  - 2400 bps sync and async
  - 1200 bps sync and async
- CCITT V.22 A & B
  - 1200 bps sync and async
- 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

This chip-set supports power-down modes that are selected via the AT command set, providing flexible power-down management control. The power-down modes make the 89C024FT a good fit for laptop computer applications. Power consumption for the chip-set is typically 400 mW during a connection. When powered-down, the chip-set consumes 5 mW.

A complete set of industry standard AT commands are provided for modem configuration and user interface. Additional commands have been implemented for power down modes and V.42/42bis/MNP feature control. Virtually all PC software written for the AT command set 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 89C024FT command module with custom proprietary software.

---

**Figure 2. 89C024FT Modem Application**

---
PACKAGING

89027 is available in PLCC and standard plastic DIP packages. The 89C026FT is available in a PLCC package. Packages are shown from top view, looking down on component side of PC board.

Figure 3. Device Packages
CALL ESTABLISHMENT, TERMINATION AND RETRAIN

The 89C024FT modem system incorporates all protocols and functions required for automatic or manual call establishment. The modem system also incorporates all protocols and functions required for progress and termination of a data call.

The modem chip-set has a built-in auto-dialer, both DTMF and Pulse type. It can detect dial, busy, and ringback signals from the remote end, and will provide call progress messages to the user. The modem is also capable of re-dialing the last number dialed.

The modem, when configured for auto-answer, will answer an incoming call, remain silent for the two second billing delay interval, and then transmit 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. A transition from voice (i.e., for the purpose of manual dialing) to data mode can be done by the use of a mechanical switch (exclusion key) on 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 is compatible with 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.

Because the CCITT and Bell modem connection protocols 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 an 89C024FT 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 89C024FT 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 separately or in string fashion. 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 89C024FT Modem</th>
<th>Bell 300</th>
<th>Bell 1200</th>
<th>Answering Modem</th>
<th>CCITT 300</th>
<th>CCITT 1200</th>
<th>CCITT 2400</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bell</td>
<td>300</td>
<td>—</td>
<td>Bell</td>
<td>300*</td>
<td>300</td>
<td>—</td>
</tr>
<tr>
<td></td>
<td>1200</td>
<td>—</td>
<td>Bell</td>
<td>1200</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>CCITT</td>
<td>300</td>
<td>—</td>
<td>CCITT</td>
<td>300</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td></td>
<td>1200</td>
<td>—</td>
<td>CCITT</td>
<td>1200</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td></td>
<td>2400</td>
<td>—</td>
<td>CCITT</td>
<td>1200</td>
<td>—</td>
<td>—</td>
</tr>
</tbody>
</table>

* These connection data rates are obtained when connecting 89C024FT based modems end to end. The same results may not be obtained when a 89C024FT 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 300 and 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></td>
<td>If &amp;J1 option is selected, H1 will also switch the auxiliary relay</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>Nn</td>
<td>Maximum Line (DCE) rate</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>
<tr>
<td>&amp;Cn</td>
<td>DDC Options</td>
</tr>
<tr>
<td>&amp;Dn</td>
<td>DTR Options</td>
</tr>
<tr>
<td>&amp;Fn</td>
<td>Fetch Factory Configuration Profile</td>
</tr>
<tr>
<td>&amp;Gn</td>
<td>Guard Tone</td>
</tr>
<tr>
<td>&amp;Jn</td>
<td>Telephone Jack Selection</td>
</tr>
<tr>
<td>&amp;Ln</td>
<td>Leased/Dial-up Line Selection</td>
</tr>
<tr>
<td>&amp;Mn</td>
<td>Async/Async Mode Selection</td>
</tr>
<tr>
<td>&amp;Pn</td>
<td>Make/Break Pulse Ratio</td>
</tr>
<tr>
<td>&amp;Rn</td>
<td>RTS/CTS Options</td>
</tr>
<tr>
<td>&amp;Sn</td>
<td>DSR Options</td>
</tr>
<tr>
<td>&amp;Tn</td>
<td>Test Commands</td>
</tr>
<tr>
<td>&amp;Wn</td>
<td>Write Configuration to Non-Volatile Memory</td>
</tr>
<tr>
<td>&amp;Xn</td>
<td>Sync Clock Source</td>
</tr>
<tr>
<td>&amp;Yn</td>
<td>Default NVRAM Profile Select</td>
</tr>
<tr>
<td>&amp;Zn</td>
<td>Store Telephone Number</td>
</tr>
</tbody>
</table>

### V.42/42bis Feature Control Commands

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>\An</td>
<td>Maximum MNP Block Size</td>
</tr>
<tr>
<td>%An</td>
<td>Set Auto-Reliable Fallback Character</td>
</tr>
<tr>
<td>\Bn</td>
<td>Transmit Break</td>
</tr>
<tr>
<td>\Cn</td>
<td>Set Auto-Reliable Buffer</td>
</tr>
<tr>
<td>\Gn</td>
<td>Set Modem Port Flow Control</td>
</tr>
<tr>
<td>\Jn</td>
<td>Bits per Second Rate Adjust</td>
</tr>
<tr>
<td>\Kn</td>
<td>Set Break Control</td>
</tr>
<tr>
<td>\Nn</td>
<td>Set Operating Mode</td>
</tr>
<tr>
<td>\O</td>
<td>Originate Reliable Link</td>
</tr>
<tr>
<td>\Qn</td>
<td>Set Serial Port Flow Control</td>
</tr>
<tr>
<td>\Tn</td>
<td>Set Inactivity Timer</td>
</tr>
<tr>
<td>\U</td>
<td>Accept Reliable Link</td>
</tr>
<tr>
<td>\Vn</td>
<td>Modify Result Code Form</td>
</tr>
<tr>
<td>\Xn</td>
<td>Set XON/XOFF Pass-Through</td>
</tr>
<tr>
<td>\Y</td>
<td>Switch to Reliable Mode</td>
</tr>
<tr>
<td>\Z</td>
<td>Switch to Normal Mode</td>
</tr>
</tbody>
</table>

### MNP Feature Control Command Set

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>\Jn</td>
<td>V.42 Detection Phase Control</td>
</tr>
<tr>
<td>'Hn</td>
<td>V.42bis Compression Control</td>
</tr>
<tr>
<td>'Nn</td>
<td>V.42bis Dictionary Size</td>
</tr>
<tr>
<td>*On</td>
<td>V.42bis Dictionary String Length</td>
</tr>
</tbody>
</table>

### Power Down Commands

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>+En</td>
<td>Disable/Enable Power Down</td>
</tr>
<tr>
<td>+Tn</td>
<td>Time to Power Down</td>
</tr>
</tbody>
</table>
CONFIGURATION REGISTERS
The modem stores all the configuration information in a set of registers. Some registers are dedicated to a special command and function, and others are bit-mapped, with different commands sharing the register space to store the command status.

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>S0*</td>
<td>Ring to Answer</td>
</tr>
<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>DTMF Tone Duration</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>
<tr>
<td>S31*</td>
<td>Bit Mapped Options Register</td>
</tr>
<tr>
<td>S37</td>
<td>Maximum Line (DCE) Rate</td>
</tr>
<tr>
<td>S100</td>
<td>Mean Error Monitor Register</td>
</tr>
</tbody>
</table>

NOTE: * These S registers can be stored in the NVRAM.

DIALING
Dial modifiers are available for adding conditions to dialed phone numbers.

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 quiet</td>
</tr>
</tbody>
</table>

Example:
Terminal: AT &Z0 = T 1 (602) 555-1212
Modem: OK
Result: Modem stores the Tone Dial (T) modifier and phone number T16025551212 in the external NVRAM.

The number can be dialed from asynchronous mode by issuing the following command:
Terminal: AT DS0
Modem: T16025551212
Result: Modem dials phone number and attempts to establish a connection.

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.

POWER MANAGEMENT
The flexible power management controls allow for a variety of command and hardware driver options. The power down sequence is initiated by placing a logic "low" on pin 15 (PD) of the 89C026FT. The laptop can control the PD signal directly. If such a signal is unavailable, PD can be controlled by communications software via DTR. Lack of data activity or an in-coming ring signal can also be used to control PD.

Placing the crystal on the 89C026FT (Figure 10) allows it to reduce power consumption by turning off the oscillator. When online and connected to a remote modem, the power consumption for the 89C024FT is typically 400 mW. Additionally, when the 89027 is not needed (on-hook, not connected to a remote modem) the 89C026FT places it in stand-by. In stand-by the chip-set power consumption is typically 255 mW. When powered down via the PD pin on the 89C026FT, the chip-set typically consumes 5 mW. Minimum memory-system power-consumption can be achieved by chip selecting memory only when addressed.

APPLICATIONS OVERVIEW
The block diagram of a stand-alone 300 to 2400 bps Hayes compatible modem is depicted in Figure 4. The DAA section shown in this diagram may be implemented using the suggested diagram in Figure 5. Figure 6 shows the use of the power-down feature.
Figure 4. Typical Laptop Modem

Figure 5. Typical Telephone Line Interface Using Internal Hybrid

Figure 6. Power-Down Control
### SYSTEM COMPATIBILITY SPECIFICATIONS

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Specification</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Synchronous</strong></td>
<td></td>
</tr>
<tr>
<td>Parameter</td>
<td>Specification</td>
</tr>
<tr>
<td>Synchronous</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><strong>Asynchronous</strong></td>
<td></td>
</tr>
<tr>
<td>Parameter</td>
<td>Specification</td>
</tr>
<tr>
<td>Asynchronous</td>
<td>2400, 1200 bps, character asynchronous. 0 - 300 bps anisochronous.</td>
</tr>
<tr>
<td>Asynchronous Speed Range</td>
<td>+ 1% − 2.5% default. Extended + 2.3% − 2.5% range of CCITT standards optional via software customization.</td>
</tr>
<tr>
<td>Asynchronous Format</td>
<td>10 bits, including start, stop, parity. (8, 9, 11 bits optional via S/W customization.)</td>
</tr>
<tr>
<td>Synchronous Timing Source</td>
<td>a) Internal, derived from the local oscillator.</td>
</tr>
<tr>
<td></td>
<td>b) External, provided by DTE through XTCLK.</td>
</tr>
<tr>
<td></td>
<td>c) 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>Modulation</td>
<td>V.22 bis, 16 point QAM at 600 baud.</td>
</tr>
<tr>
<td></td>
<td>V.22 and 212A, 4 point PSK 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></td>
</tr>
<tr>
<td>V.22 bis, V.22, 212A</td>
<td>Originate 1200 Hz ± .02%</td>
</tr>
<tr>
<td></td>
<td>Answer 2400 Hz ± .02%</td>
</tr>
<tr>
<td>V.21</td>
<td>Originate 'space' 1180 Hz ± .02%</td>
</tr>
<tr>
<td></td>
<td>Originate 'mark' 980 Hz ± .02%</td>
</tr>
<tr>
<td></td>
<td>Answer 'space' 1850 Hz ± .02%</td>
</tr>
<tr>
<td></td>
<td>Answer 'mark' 1650 Hz ± .02%</td>
</tr>
<tr>
<td>Bell 103 mode</td>
<td>Originate 'space' 1070 Hz ± .02%</td>
</tr>
<tr>
<td></td>
<td>Originate 'mark' 1270 Hz ± .02%</td>
</tr>
<tr>
<td></td>
<td>Answer 'space' 2020 Hz ± .02%</td>
</tr>
<tr>
<td></td>
<td>Answer 'mark' 2225 Hz ± .02%</td>
</tr>
<tr>
<td>Received Signal Frequency Tolerance</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>Typical Energy Detect Sensitivity</td>
<td>Greater than −43 dBm ED is ON. Less than −48 dBm ED is OFF. Signal in dBm measured at AO2.</td>
</tr>
<tr>
<td>Energy Detect Hysteresis</td>
<td>A minimum Hysteresis of 2 dB for QAM scrambled mark.</td>
</tr>
<tr>
<td>Line Equalization</td>
<td>Fixed compromise equalization, transmit.</td>
</tr>
<tr>
<td></td>
<td>Adaptive equalizer for PSK/QAM, receive.</td>
</tr>
<tr>
<td>Diagnostics Available</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>Self Test Pattern Generator</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>
RECEIVER PERFORMANCE SPECIFICATIONS

<table>
<thead>
<tr>
<th>Test Cases</th>
<th>Typical SNR for $10^{-5}$ BER Performance</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Data Mode</td>
</tr>
<tr>
<td>V.22 bis</td>
<td>Synchronous</td>
</tr>
<tr>
<td>V.22/Bell 212A</td>
<td>Synchronous</td>
</tr>
<tr>
<td>V.21</td>
<td>Asynchronous</td>
</tr>
<tr>
<td>Bell 103</td>
<td>Asynchronous</td>
</tr>
</tbody>
</table>

Test Conditions:
- Receive Signal (Rx) measured at A02 (transmit level set at -9 dBm)
- Unconditioned 3002 Line
- 3 KHz Flat-Band 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>4.0</td>
<td>dBm</td>
<td>dBm</td>
<td></td>
<td>at AO1</td>
</tr>
<tr>
<td>DTMF Second Harmonic</td>
<td>-35</td>
<td>dB</td>
<td>dB</td>
<td>HYB enabled into 600Ω</td>
<td></td>
</tr>
<tr>
<td>DTMF Twist (Balance)</td>
<td>3</td>
<td>dB</td>
<td>dB</td>
<td></td>
<td>Software Controlled</td>
</tr>
<tr>
<td>Default DTMF Duration</td>
<td>100</td>
<td>ms</td>
<td>ms</td>
<td></td>
<td>Software Controlled</td>
</tr>
<tr>
<td>Pulse Dialing Rate</td>
<td>10/20</td>
<td>pps</td>
<td>pps</td>
<td></td>
<td>Software Controlled</td>
</tr>
<tr>
<td>Pulse Dialing Make/Break</td>
<td>39/61</td>
<td>%</td>
<td>%</td>
<td>US</td>
<td></td>
</tr>
<tr>
<td></td>
<td>33/67</td>
<td>%</td>
<td>%</td>
<td>UK, Hong Kong</td>
<td></td>
</tr>
<tr>
<td>Pulse Interdigit Interval</td>
<td>785</td>
<td>ms</td>
<td>ms</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Billing Delay Interval</td>
<td>2.1</td>
<td>sec</td>
<td>sec</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Guard Tone Frequency Amplitude</td>
<td>540</td>
<td>Hz</td>
<td>Hz</td>
<td>referenced to High Channel transmit.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>-3</td>
<td>dB</td>
<td>dB</td>
<td>QAM/PSK Modes Only</td>
<td></td>
</tr>
<tr>
<td>Frequency Amplitude</td>
<td>1800</td>
<td>Hz</td>
<td>Hz</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>-6</td>
<td>dB</td>
<td>dB</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Dial Tone Detect Duration</td>
<td>3.0</td>
<td>sec</td>
<td>sec</td>
<td></td>
<td>Off/On Ratio</td>
</tr>
<tr>
<td>Ringback Tone Detect Duration</td>
<td>0.75</td>
<td>sec</td>
<td>sec</td>
<td>Off/On Ratio</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>0.2</td>
<td>sec</td>
<td>sec</td>
<td>Off/On Ratio</td>
<td></td>
</tr>
<tr>
<td>Cadence</td>
<td>0.67</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
**89C026FT OVERVIEW**

The 89C026FT processor performs data manipulation, signal processing and user interface functions. It requires a single 64K x 8 ROM and 32K x 8 RAM to execute standard, and/or custom code to perform the V.42/42bis and MNP4/5 protocol functions. The ROM and RAM addresses overlap in external memory and are decoded using the INST and AD15 signals. A block diagram of the 89C026FT is provided in Figure 7.

89C026FT contains a TTL compatible serial link to DTE equipment, along with a full complement of V.24/RS-232-C control signals. A UART or USART may be used to transfer data to and from a microcomputer bus. The 89C026FT supports the industry standard AT command set facilitating compatibility with most PC software.

During transmit operation, the 89C026FT 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 89C026FT transmits digitized phase and amplitude samples to 89027 over the high speed serial link.

In the receive operation, the information is received by the 89C026FT 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 89C026FT. By using DSP algorithms, the received signals are processed using adaptive equalization for telephone line delay, amplitude distortion and gain adjustments and the signal demodulated. Following demodulation, the data is unscrambled, and if necessary, returned to asynchronous format.

---

**Figure 7. 89C026FT Block Diagram**
### 89C026FT PINOUT

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Function (89C026FT)</th>
<th>Direction(^{(4)})</th>
<th>Pin No.</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLKIN</td>
<td>12.96 MHz master clock from 89027</td>
<td>In</td>
<td>67</td>
</tr>
<tr>
<td>CLKIN2</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 Defect 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>PD</td>
<td>Power-down control</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>Reserved for future use ((V_{cc}))^{(2)}</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>DS0</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>(V_{cc})</td>
<td>Positive power supply ((+5V))</td>
<td>(+5V)</td>
<td>1</td>
</tr>
<tr>
<td>CDE</td>
<td>Clock detect enable ((V_{SS}))^{(1)}</td>
<td>GND</td>
<td>14</td>
</tr>
<tr>
<td>(V_{REF})</td>
<td>A/D converter reference</td>
<td>(+5V)</td>
<td>13</td>
</tr>
<tr>
<td>(V_{SS1})</td>
<td>Digital ground</td>
<td>GND</td>
<td>36</td>
</tr>
<tr>
<td>(V_{SS2})</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>(V_{pp})</td>
<td>Timing pin for return from power-down</td>
<td>In</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/date(^{(3)})</td>
<td>I/O</td>
<td>60-45</td>
</tr>
<tr>
<td>AA</td>
<td>Auto Answer(^{(3)})</td>
<td>Out</td>
<td>60</td>
</tr>
<tr>
<td>JS</td>
<td>Jack Select(^{(3)})</td>
<td>Out</td>
<td>59</td>
</tr>
<tr>
<td>CD</td>
<td>Carrier Detect Indicator(^{(3)})</td>
<td>Out</td>
<td>58</td>
</tr>
<tr>
<td>MR</td>
<td>Modem Ready Indicator(^{(3)})</td>
<td>Out</td>
<td>57</td>
</tr>
<tr>
<td>REL</td>
<td>MNP Reliable Link Active(^{(3)})</td>
<td>Out</td>
<td>56</td>
</tr>
<tr>
<td>COMP</td>
<td>Compression Active V.42bis or MNP 5(^{(3)})</td>
<td>Out</td>
<td>55</td>
</tr>
<tr>
<td>ERR</td>
<td>Error detected by LAPM or MNP(^{(3)})</td>
<td>Out</td>
<td>54</td>
</tr>
<tr>
<td>LAPM</td>
<td>LAPM Reliable Link Active(^{(3)})</td>
<td>Out</td>
<td>53</td>
</tr>
</tbody>
</table>
89C026FT PINOUT (Continued)

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Function (89C026FT)</th>
<th>Direction(4)</th>
<th>Pin No.</th>
</tr>
</thead>
<tbody>
<tr>
<td>NMI</td>
<td>Non-maskable interrupt(V&lt;sub&gt;SS&lt;/sub&gt;)&lt;sup&gt;(1)&lt;/sup&gt;</td>
<td>In</td>
<td>3</td>
</tr>
<tr>
<td>X2</td>
<td>Crystal output</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>BUSWIDTH</td>
<td>Bus Width</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</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 (V<sub>SS</sub>) must be corrected to V<sub>SS</sub>.
2. Pins marked with (V<sub>EE</sub>) must be connected to V<sub>EE</sub>.
3. AD0-AD3 are used as AA, JS, CD, MR, REL, COMP, ERR, and LAPM respectively.
4. Pins with direction "IN" must not be left floating.

89C026FT PIN DESCRIPTION

**XTCLK**
Transmitter timing from DTE, when external clock option is selected.

**RXD**
The serial data from DTE to be transmitted on the line. A logic 'high' is mark. In synchronous mode, 89C026FT samples this data on the rising edges of TCLK.

**TCLK**
Clock output from 89C026FT 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. A logic 'high' is mark. 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.

**PD**
Power-down control. A low on this input pin, in conjunction with the + En and + Tn commands, will cause the modem to go into a power-down mode.

**V<sub>PP</sub>**
Timing pin for return from power-down. Connect a 1.0 µF capacitor between V<sub>PP</sub> and V<sub>SS</sub> if the power-down option is used. This capacitor causes an internal timing circuit to give the oscillator time to stabilize before turning on internal clocks. This pin may be left floating or connected through a 1.0 µF capacitor to V<sub>SS</sub> if power-down mode is not required.

**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**
A low indicates modem is off-hook, 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**
&DO 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.
**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 transferred 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 sets an off hook condition, high indicates an on hook. When dialing, this signal 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 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 low on this pin initiates a remote loopback condition.

**CD**

A low indicates the presence of carrier signal on the line.

**MR**

A low indicates the presence of the DSR signal. Toggling indicates that a test mode is active.

**REL**

A low indicates that an MNP reliable link has been established.

**COMP**

A low indicates that data compression is in operation (V.42bis or MNP Class 5).

**LAPM**

A low indicates that a LAPM reliable link has been established.

**ERR**

Goes low for 1 second whenever a reliable connection detects an error.

**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.

**V REF**

Voltage reference for the analog to digital converter should be connected to the 89027 AVcc.

**CDE**

This pin must be connected to Vss.

**S/Å**

The function of this pin is re-defined as external NVRAM CE.

**CONFIG**

Reserved for future use. This signal should be pulled high.

**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. This pin must be tied high.

**JS**

Low is used to pulse A and A1 leads to control a 1A2 Key System jack.

**BUSWIDTH**

When high, external memory accesses are 16 bits wide. When low, external memory accesses are 8 bits wide. This pin must be tied low.

**READY**

When high, no wait states are inserted in external memory accesses. When low, one wait state is inserted in each external memory access.

**INST**

Output high during an external memory read indicates the read is an instruction fetch. INST is activated only during external memory accesses and output low for data fetch. INST along with AD15 are used to decode the overlapping external ROM and RAM.
89C026FT ABSOLUTE MAXIMUM RATINGS*

Temperature Under Bias .......... 0°C to + 70°C
Storage Temperature .............. -40°C to +125°C
Voltage from Any Pin to
  VSS or AGND ................. -0.5V 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>fOSC</td>
<td>CLKIN Frequency</td>
<td>12.95870</td>
<td>12.96130</td>
<td>MHz</td>
</tr>
</tbody>
</table>

NOTE: The AGND and VSS on both the 89C026FT and the 89027 must be nominally at the same potential.

D.C. CHARACTERISTICS

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Parameter</th>
<th>Min</th>
<th>Typ(7)</th>
<th>Max</th>
<th>Units</th>
<th>Test Conditions</th>
</tr>
</thead>
<tbody>
<tr>
<td>VIL</td>
<td>Input Low Voltage</td>
<td>-0.5</td>
<td></td>
<td>+0.8</td>
<td>V</td>
<td></td>
</tr>
<tr>
<td>VIH</td>
<td>Input High Voltage(1)</td>
<td>0.2 VCC + 0.9</td>
<td>VCC + 0.5</td>
<td>V</td>
<td></td>
<td></td>
</tr>
<tr>
<td>VIH1</td>
<td>Input High Voltage on CLKIN</td>
<td>0.7 VCC</td>
<td>VCC + 0.5</td>
<td>V</td>
<td></td>
<td></td>
</tr>
<tr>
<td>VIH2</td>
<td>Input High Voltage on RESET</td>
<td>2.2</td>
<td></td>
<td>VCC + 0.5</td>
<td>V</td>
<td></td>
</tr>
<tr>
<td>VOL</td>
<td>Output Low Voltage</td>
<td>0.3</td>
<td>0.45</td>
<td>1.5</td>
<td>V</td>
<td>IOL = 200 µA</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>V</td>
<td>IOL = 3.2 mA</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>V</td>
<td>IOL = 7 mA</td>
</tr>
<tr>
<td>VOH</td>
<td>Output High Voltage(4)</td>
<td>VCC - 0.3</td>
<td>VCC - 0.7</td>
<td>VCC - 1.5</td>
<td>V</td>
<td>IOH = -200 µA</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>V</td>
<td>IOH = -3.2 mA</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>V</td>
<td>IOH = -7 mA</td>
</tr>
<tr>
<td>VOH1</td>
<td>Output High Voltage(3)</td>
<td>VCC - 0.3</td>
<td>VCC - 0.7</td>
<td>VCC - 1.5</td>
<td>V</td>
<td>IOH = -10 µA</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>V</td>
<td>IOH = -30 µA</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>V</td>
<td>IOH = -60 µA</td>
</tr>
<tr>
<td>IIL</td>
<td>Input Leakage Current(5)</td>
<td>±10</td>
<td></td>
<td>µA</td>
<td>0 &lt; VIN &lt; VCC - 0.3V</td>
<td></td>
</tr>
<tr>
<td>IIL1</td>
<td>Input Leakage Current(6)</td>
<td>±3</td>
<td></td>
<td>µA</td>
<td>0 &lt; VIN &lt; VREF</td>
<td></td>
</tr>
<tr>
<td>IIL</td>
<td>Logical 0 Input Current(3)</td>
<td>-50</td>
<td></td>
<td>µA</td>
<td>VIN = 0.45V</td>
<td></td>
</tr>
<tr>
<td>IL1</td>
<td>Logical 0 Input Current in RESET(2) (ALE, RD, WR, BHE, INST, SCLK)</td>
<td>-850</td>
<td>µA</td>
<td>VIN = 0.45V</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

4-59
D.C. CHARACTERISTICS (Continued)

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Parameter</th>
<th>Min</th>
<th>Typ(7)</th>
<th>Max</th>
<th>Units</th>
<th>Test Conditions</th>
</tr>
</thead>
<tbody>
<tr>
<td>IREF</td>
<td>A/D Converter Reference Current</td>
<td>2</td>
<td>5</td>
<td>mA</td>
<td></td>
<td>CLKin = 12.96 MHz</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>VCC = VPP = VREF = 5.25</td>
</tr>
<tr>
<td>ICC1</td>
<td>Active Mode Current (Typical)</td>
<td>45</td>
<td>60</td>
<td>mA</td>
<td></td>
<td>CLKin = 12.96 MHz</td>
</tr>
<tr>
<td>RRST</td>
<td>RESET Pullup Resistor</td>
<td>6K</td>
<td>50K</td>
<td>Ω</td>
<td></td>
<td>fTEST = 1.0 MHz</td>
</tr>
<tr>
<td>CS</td>
<td>Pin Capacitance (Any Pin to VSS)</td>
<td>10</td>
<td></td>
<td>pF</td>
<td></td>
<td></td>
</tr>
<tr>
<td>IPD</td>
<td>Power-Down Mode Current</td>
<td>5</td>
<td>50</td>
<td>μA</td>
<td></td>
<td>VCC = VPP = VREF = 5.25</td>
</tr>
</tbody>
</table>

NOTES:
(Notes apply to all specifications)
1. All pins except RESET and CLKin.
2. Holding these pins below V_H in RESET may cause the part to enter test modes.
3. TCL0, TCL1, S/Å, RTS, CTS, DSR, DCD, SI, OH.
4. BHE, INST, CLKOUT, RESET, TCLK, RXD, RCLK, TSYNC, TM, SCLK, SDATA. The VOH specification is not valid for RESET.
5. CDE, EA, READY, BUSWIDTH, NMI, STR, DTR, XTCLK, TXD, B/C, CLKIN2, and RI.
7. Typicals are based on a limited number of samples and are not guaranteed. The values listed are at room temperature and VREF = VCC = 5V.

A.C. CHARACTERISTICS (Over specified operating conditions)
Test Conditions: Capacitive load on all pins = 100 pF, Rise and fall times = 10 ns, fOSC 12.96 MHz

This system must meet these specifications to work with 89C026FT:

<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>TAVVYV</td>
<td>Address Valid to READY Setup</td>
<td></td>
<td>2TOSC - 85</td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>TLLYV</td>
<td>ALE Low to READY Setup</td>
<td></td>
<td>TOSC - 72</td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>TYLYH</td>
<td>Non READY Time</td>
<td></td>
<td>No Upper Limit</td>
<td></td>
<td>ns</td>
</tr>
<tr>
<td>TCLOYX</td>
<td>READY Hold after CLKOUT Low</td>
<td>0</td>
<td>TOSC - 30</td>
<td>ns</td>
<td>(Note 1)</td>
</tr>
<tr>
<td>TLLYX</td>
<td>READY Hold after ALE Low</td>
<td>TOSC - 15</td>
<td>2TOSC - 40</td>
<td>ns</td>
<td>(Note 1)</td>
</tr>
<tr>
<td>TAVGV</td>
<td>Address Valid to Buswidth Setup</td>
<td></td>
<td>2TOSC - 85</td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>TLLGV</td>
<td>ALE Low to Buswidth Setup</td>
<td></td>
<td>TOSC - 70</td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>TCLGX</td>
<td>Buswidth Hold after CLKOUT Low</td>
<td>0</td>
<td>TOSC - 85</td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>TAVDV</td>
<td>Address Valid to Input Data Valid</td>
<td></td>
<td>3TOSC - 67</td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>TRLDV</td>
<td>RD Active to Input Data Valid</td>
<td></td>
<td>TOSC - 23</td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>TCLDV</td>
<td>CLKOUT Low to Input Data Valid</td>
<td></td>
<td>TOSC - 50</td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>TRHDZ</td>
<td>End of RD to Input Data Float</td>
<td></td>
<td>TOSC - 20</td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>TRXDX</td>
<td>Data Hold afterRD Inactive</td>
<td>0</td>
<td></td>
<td>ns</td>
<td></td>
</tr>
</tbody>
</table>

NOTE:
1. If max is exceeded, additional wait states will occur.
The 89C026FT 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>Oscillator Frequency</td>
<td>12.95870</td>
<td>12.96130</td>
<td>MHz</td>
<td></td>
</tr>
<tr>
<td>TOSC</td>
<td>Oscillator Period</td>
<td>1/(F_{\text{CLKIN}})_{\text{MAX}} )</td>
<td>1/(F_{\text{CLKIN}})_{\text{MIN}}</td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>TXHCH</td>
<td>FCLKIN High to CLKOUT High or Low</td>
<td>40</td>
<td>110</td>
<td>ns</td>
<td>(Note 1)</td>
</tr>
<tr>
<td>TCLCL</td>
<td>CLKOUT Cycle Time</td>
<td>2 (T_{\text{OSC}})</td>
<td></td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>TCHCH</td>
<td>CLKOUT High Period</td>
<td>(T_{\text{OSC}} - 10)</td>
<td>(T_{\text{OSC}} + 10)</td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>TCLCH</td>
<td>CLKOUT Low to CLKOUT Falling Edge</td>
<td>5</td>
<td>15</td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>TLHCH</td>
<td>ALE Falling Edge to CLKOUT Rising</td>
<td>-15</td>
<td>15</td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>TLHHL</td>
<td>ALE Cycle Time</td>
<td>4 (T_{\text{OSC}})</td>
<td></td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>TLHLL</td>
<td>ALE High Period</td>
<td>(T_{\text{OSC}} - 10)</td>
<td>(T_{\text{OSC}} + 10)</td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>TAVLL</td>
<td>Address Setup to ALE Falling Edge</td>
<td>(T_{\text{OSC}} - 20)</td>
<td></td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>TLXAX</td>
<td>ALE Falling Edge to RD Falling Edge</td>
<td>(T_{\text{OSC}} - 40)</td>
<td></td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>TLLRL</td>
<td>ALE Falling Edge to RD Rising Edge</td>
<td>(T_{\text{OSC}} - 40)</td>
<td></td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>TRLCL</td>
<td>RD Low to CLKOUT Falling Edge</td>
<td>5</td>
<td>30</td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>TRLRH</td>
<td>RD Low Period</td>
<td>(T_{\text{OSC}} - 5)</td>
<td>(T_{\text{OSC}} + 25)</td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>TRHHL</td>
<td>RD Rising Edge to ALE Rising Edge</td>
<td>(T_{\text{OSC}})</td>
<td>(T_{\text{OSC}} + 25)</td>
<td>ns</td>
<td>(Note 2)</td>
</tr>
<tr>
<td>TRILA</td>
<td>RD Low to Address Float</td>
<td>10</td>
<td></td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>TLLLL</td>
<td>ALE Falling Edge to WR Falling Edge</td>
<td>(T_{\text{OSC}} - 10)</td>
<td></td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>TCLWL</td>
<td>CLKOUT Low to WR Falling 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>(T_{\text{OSC}} - 23)</td>
<td></td>
<td>ns</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>(T_{\text{OSC}} - 30)</td>
<td>(T_{\text{OSC}} + 5)</td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>TWHQX</td>
<td>Data Hold after WR Rising Edge</td>
<td>(T_{\text{OSC}} - 10)</td>
<td></td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>TWHLH</td>
<td>WR Rising Edge to ALE Rising Edge</td>
<td>(T_{\text{OSC}} - 10)</td>
<td>(T_{\text{OSC}} + 15)</td>
<td>ns</td>
<td>(Note 2)</td>
</tr>
<tr>
<td>TWHBX</td>
<td>BHE, INST, Hold after WR Rising Edge</td>
<td>(T_{\text{OSC}} - 10)</td>
<td></td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>TRHBE</td>
<td>BHE, INST, Hold after RD Rising Edge</td>
<td>(T_{\text{OSC}} - 10)</td>
<td></td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>THRAX</td>
<td>AD8–15 Hold after WR Rising Edge</td>
<td>(T_{\text{OSC}} - 50)</td>
<td></td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>THRAX</td>
<td>AD8–15 Hold after RD Rising Edge</td>
<td>(T_{\text{OSC}} - 25)</td>
<td></td>
<td>ns</td>
<td></td>
</tr>
</tbody>
</table>

NOTES:
1. Typical specifications, not guaranteed.
Figure 8. Bus Signal Timings
89027 OVERVIEW

The 89027 is a 28 pin CHMOS 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 9. Most of the analog signal processing functions in this chip are implemented with CHMOS switched capacitor technology. The 89027 functions are controlled by 89C026FT, through a high speed serial data link.

During FSK transmit operation, the 89027 receives digitally synthesized mark and space sinusoid amplitude information from the 89C026FT. The 89027 converts the signal to its analog equivalents, 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 89C026FT 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 9. 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 89C026FT</td>
<td>Out</td>
<td>26</td>
</tr>
<tr>
<td>CLKOUT2</td>
<td>270 KHz Clock Output to 89C026FT</td>
<td>Out</td>
<td>19</td>
</tr>
<tr>
<td>RST</td>
<td>Chip reset (active low)(3)</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 89C026FT</td>
<td>In</td>
<td>2</td>
</tr>
<tr>
<td>SCLK</td>
<td>Serial clock from 89C026FT</td>
<td>In</td>
<td>1</td>
</tr>
<tr>
<td>TSYNC</td>
<td>Transmitter sync from 89C026FT</td>
<td>In</td>
<td>3</td>
</tr>
<tr>
<td>STR</td>
<td>Symbol timing to 89C026FT</td>
<td>Out</td>
<td>27</td>
</tr>
<tr>
<td>ED</td>
<td>Receiver energy detect to 89C026FT</td>
<td>Out</td>
<td>18</td>
</tr>
<tr>
<td>I</td>
<td>In phase received signal to 89C026FT</td>
<td>Out</td>
<td>13</td>
</tr>
<tr>
<td>Q</td>
<td>Quadrature-phase received signal to 89C026FT</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>
<tr>
<td>NC</td>
<td>(Note 2)</td>
<td>In</td>
<td>22</td>
</tr>
</tbody>
</table>

### NOTE:
1. When held high, these pins must be connected through 10K resistors to VCC.
2. Reserved Pin. Must be left No Connect.
3. Connect to reset circuitry through a 10K resistor.

### 89027 Pinout Description

**TX0-3**

These four pins control the transmitted signal level. Refer to Transmit Level Table.

**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**

Temperature Under Bias .......... 0 to +70°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
to VSS(1) .................. -0.3V to 6.5V

**NOTES:**
1. Applies to pins SCLK, SDATA, TSYNC, RST, HYB, TX0–TX3 only.
2. 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.

**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>T&lt;sub&gt;A&lt;/sub&gt;</td>
<td>Ambient Temperature Under Bias</td>
<td>0</td>
<td>+70</td>
<td>°C</td>
</tr>
<tr>
<td>V&lt;sub&gt;CC&lt;/sub&gt;</td>
<td>Digital Supply Voltage</td>
<td>4.75</td>
<td>5.25</td>
<td>V</td>
</tr>
<tr>
<td>V&lt;sub&gt;BB&lt;/sub&gt;</td>
<td>Analog Supply Voltage</td>
<td>-4.75</td>
<td>-5.25</td>
<td>V</td>
</tr>
</tbody>
</table>

**POWER DISSIPATION** Ambient Temp = 0° to 70° C, V<sub>CC</sub> = AV<sub>CC</sub> = 5 ± 5%, V<sub>SS</sub> = AGND = 0V.
Typical values shown are for T<sub>A</sub> = 25°C and nominal power supplies.

<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&lt;sub&gt;C1&lt;/sub&gt;</td>
<td>AVCC Operating Current</td>
<td>15</td>
<td>21</td>
<td>mA</td>
<td></td>
</tr>
<tr>
<td>lcc&lt;sub&gt;C1&lt;/sub&gt;</td>
<td>VCC Operating Current</td>
<td>5</td>
<td>6</td>
<td>mA</td>
<td></td>
</tr>
<tr>
<td>lb&lt;sub&gt;b1&lt;/sub&gt;</td>
<td>VBB Operating Current</td>
<td>-15</td>
<td>-21</td>
<td>mA</td>
<td></td>
</tr>
<tr>
<td>Alc&lt;sub&gt;Cs&lt;/sub&gt;</td>
<td>AVCC Standby Current</td>
<td>0.2</td>
<td>1</td>
<td>mA</td>
<td></td>
</tr>
<tr>
<td>lcc&lt;sub&gt;s&lt;/sub&gt;</td>
<td>VCC Standby Current</td>
<td>5</td>
<td>6</td>
<td>mA</td>
<td></td>
</tr>
<tr>
<td>lb&lt;sub&gt;b&lt;/sub&gt;</td>
<td>VBB Standby Current</td>
<td>-0.6</td>
<td>-2</td>
<td>mA</td>
<td></td>
</tr>
<tr>
<td>Alc&lt;sub&gt;cp&lt;/sub&gt;</td>
<td>AVCC Power-Down Current</td>
<td>100</td>
<td></td>
<td>μA</td>
<td></td>
</tr>
<tr>
<td>lcc&lt;sub&gt;p&lt;/sub&gt;</td>
<td>VCC Power-Down Current</td>
<td>450</td>
<td></td>
<td>μA</td>
<td></td>
</tr>
<tr>
<td>lb&lt;sub&gt;b&lt;/sub&gt;</td>
<td>VBB Power-Down Current</td>
<td>450</td>
<td></td>
<td>μA</td>
<td></td>
</tr>
<tr>
<td>Pdo</td>
<td>Operating Power Dissipation</td>
<td>175</td>
<td>250</td>
<td>mW</td>
<td></td>
</tr>
<tr>
<td>Pds</td>
<td>Standby Power Dissipation</td>
<td>30</td>
<td>50</td>
<td>mW</td>
<td></td>
</tr>
<tr>
<td>Pdp</td>
<td>Power Down Power Dissipation</td>
<td>5</td>
<td></td>
<td>mW</td>
<td></td>
</tr>
</tbody>
</table>
D.C. CHARACTERISTICS \((T_A = 0°C \text{ to } 70°C, AV_{CC} = V_{CC} = 5V \pm 5\%, V_{BB} = 5V \pm 5\%, AGND = V_{SS} = 0V)\).

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 Conditions</th>
</tr>
</thead>
<tbody>
<tr>
<td>lil</td>
<td>Input Leakage Current</td>
<td>-10</td>
<td>+10</td>
<td>(\mu)A</td>
<td>(V_{SS} \leq V_{in} \leq V_{CC})</td>
</tr>
<tr>
<td>Vil</td>
<td>Input Low Voltage</td>
<td>(V_{SS})</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>(V_{CC})</td>
<td>V</td>
<td></td>
</tr>
<tr>
<td>Vol</td>
<td>Output Low Voltage</td>
<td>0.4</td>
<td>V</td>
<td></td>
<td>(I_{OL} \geq -1.6mA, 1 \text{TTL load})</td>
</tr>
<tr>
<td>Voh</td>
<td>Output High Voltage</td>
<td>2.4</td>
<td>V</td>
<td></td>
<td>(I_{OH} \leq 50\mu A, 1 \text{TTL load})</td>
</tr>
<tr>
<td>Vcol</td>
<td>CLKOUT Low Voltage</td>
<td>0.4</td>
<td>V</td>
<td></td>
<td>(\text{Load Capacitance} = 60 \text{pF})</td>
</tr>
<tr>
<td>Vcoh</td>
<td>CLKOUT High Voltage</td>
<td>0.7</td>
<td>(V_{CC})</td>
<td>V</td>
<td>(\text{Load Capacitance} = 60 \text{pF})</td>
</tr>
</tbody>
</table>

A.C. CHARACTERISTICS \((T_A = 25°C, V_{CC} = AV_{CC} = 5V, V_{SS} = AGND = 0V, V_{BB} = -5V)\)

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 Receive Signal Level</td>
<td>-9</td>
<td></td>
<td></td>
<td>dBm</td>
<td>Hybrid Enabled</td>
</tr>
<tr>
<td>AO2 Input Resistance</td>
<td>10</td>
<td></td>
<td></td>
<td>MOhms</td>
<td>(-2.5V &lt; V_{in} &lt; +2.5V)</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 \(\mu F\)
Tolerance = \(\pm 20\%\)
Voltage Rating = 10V
Type = Non-Electrolytic, low leakage.
**CRYSTAL REQUIREMENTS(1)**

<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>Frequency Accuracy (0°C–70°C)</td>
<td>-0.0035%</td>
<td>12.96</td>
<td>+0.0035%</td>
<td>MHz</td>
<td>Refer to Figure 10</td>
</tr>
<tr>
<td>Rx</td>
<td>10</td>
<td>16</td>
<td></td>
<td>Ohms</td>
<td></td>
</tr>
<tr>
<td>Cx</td>
<td>0.024</td>
<td></td>
<td></td>
<td>pF</td>
<td></td>
</tr>
<tr>
<td>Co</td>
<td>5.6</td>
<td>6.1</td>
<td>+5%</td>
<td>pF</td>
<td></td>
</tr>
<tr>
<td>CL(2)</td>
<td>-5%</td>
<td>47</td>
<td></td>
<td>pF</td>
<td>2 Load Capacitors</td>
</tr>
</tbody>
</table>

**NOTES:**
1. Crystal Type: Parallel Resonant
2. Crystal manufacturers usually specify the accuracy of a parallel resonant circuit at a given "load capacitance". This "load capacitance" is specified to the crystal manufacturer as 33 pF. 33 pF includes the parallel combination of the capacitances seen at the pins of the crystal. These capacitances include \( C_L \), IC pin capacitances, and a \( 3 \pm 2 \) pF trace capacitance.

![Figure 10. Crystal Equivalent Circuit](image)

**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></td>
<td>600</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></td>
<td></td>
<td></td>
<td>dB</td>
<td></td>
</tr>
<tr>
<td>AO1 to Amp</td>
<td>-9</td>
<td>-18</td>
<td>-26</td>
<td>dB</td>
<td>Max</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>-70</td>
<td>dB</td>
<td>Mid (Selectable)</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>+12</td>
<td>+3</td>
<td>-4</td>
<td>dB</td>
<td>Max</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>-60</td>
<td>dB</td>
<td>Mid (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.
### Transmit Output Level\(^{(1)}\)

<table>
<thead>
<tr>
<th>TX 3,2,1,0</th>
<th>Typ</th>
<th>Units</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 0 0 0</td>
<td>+ 5</td>
<td>dBm</td>
</tr>
<tr>
<td>0 0 0 1</td>
<td>+ 4</td>
<td>dBm</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1 1 1 0</td>
<td>- 9</td>
<td>dBm</td>
</tr>
<tr>
<td>1 1 1 1</td>
<td>-10</td>
<td>dBm</td>
</tr>
</tbody>
</table>

**NOTE:**
1. For PSK and QAM transmit signal. For FSK transmit signal levels, they are typically 1 dB lower. All signals are measured at A01. The tolerance for the transmit levels is ± 1 dBm.

### 89C024LT CHIP-SET USERS:

An example of how to design a single board for both an 89C024LT and 89C024FT modem product is available from your local Intel sales office. This document is titled “Upgrading From the 89C024LT to 89C024FT.” It provides information for a smooth upgrade from an 89C024LT to an 89C024FT product.

### REFERENCE MANUALS

The Modem Reference Manual (Order Number 296235) contains pin descriptions, AT and MNP command descriptions, schematics, and important design guidelines for the 89C024LT, 89C024XE, and 89024 modem chip-sets. The Modem Software Reference Manual (Order Number 296503-001) provides information about the modem software routines. The 89C024FT Modem Reference Manual Addendum contains design information on the 89C024FT chip-set. Contact your local Intel sales office for the latest information.
The MD024LT is a modem demo board in a pocket modem style. It facilitates simple demonstration of the 89C024LT operation. The 89C024LT is a low power, error correcting Laptop modem chip-set. The 89C024LT is composed of the 89027 Analog Front End and the 89C026LT Microcontroller.

The MD024LT includes a 32K x 8 EPROM and an 8k x 8 SRAM to demonstrate the AT auto-dialer, MNP class 4 error correction and MNP class 5 data compression. Phone numbers and parameters may be stored in the on-board NVRAM. A terminal can be used to configure the modem via the RS232 serial port. A DAA is provided for direct phone hook up via an RJ11 connector.

The MD024LT board is shipped with a 120V power supply, demonstration utility diskette for an IBM Personal computer, a demonstration kit document, and an RJ11 phone cord. The diskette contains a program to demonstrate and measure the 89C024LT throughput, and a copy of Intel’s communication software iTerm.

*MNP is a registered trademark of Microcom Inc.
The MD024FT is a modem demo board in a pocket modem style. It facilitates simple demonstration of the 89C024FT operation. The 89C024FT is a low power V.42/42bis compliant Laptop modem chip-set. The 89C024FT is composed of the 89027 Analog Front End and the 89C026FT Microcontroller.

The MD024FT includes a 64k by 8 EPROM and a 32k by 8 SRAM to demonstrate the AT auto-dialer, V.42 error correction (LAPM and MNP class 4) and V.42bis data compression (BTLZ and MNP class 5). Phone numbers and parameters may be stored in the optional on-board NVRAM. A terminal or PC can be used to configure the modem via the RS232 serial port. A DAA is also provided for direct phone hook up via an RJ11 connector.

The MD024FT board is shipped with a 120V AC power supply, demonstration utility diskette for an IBM Personal computer and an RJ11 phone cord. The utility diskette contains a program to demonstrate and measure the 89C024FT’s throughput, and a copy of Intel’s communication software, iTerm.

*MNP is a registered trademark of Microcom Inc.
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 user developed 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 Modem Evaluation Kit for the 89024, 89C024XE and MNP* Class 5 Firmware
- 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 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.

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.

NOTE:
This product is not FCC part 68 and part 15 approved. It is intended for laboratory evaluation only.

Figure 1. MEK3 Block Diagram

*MNP is a registered trademark of Microcom Incorporated.
SOFTWARE DEVELOPMENT

The MEK3 provides EPROM sockets for evaluating user developed software and provides the necessary clocking provisions to interface to a VLSICE 96 for 89024 and an 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 89024 and 89C024XE MNP chip sets 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
5. 89C024XE-MNP5 Modem design kit

Reference Manual

The Modem Reference Manual is available. It describes the 89024 and the 89C024XE chip-sets. It also provides a full description of all the “AT” commands and S-registers supported by the 89024 and 89C024XE Modem chip-sets.

MEK3 ORDERING INFORMATION

MEK3
89024 Modem Customization for V.23 Data Transmission

BRIAN D. WALSH
APPLICATIONS ENGINEER
INTEL CORPORATION
CONTENTS

INTRODUCTION .................................. 4-77
GENERAL DESCRIPTION .......................... 4-77
HARDWARE DESCRIPTION ......................... 4-77
Schematic Diagram ............................. 4-78
SOFTWARE DESCRIPTION ......................... 4-79
Decoding AT&A1 Command ....................... 4-79
Updating the Output Pins ...................... 4-80
V.23 Call Progress Monitor .................... 4-82
V.23 Handshake ................................ 4-82
V.23 Data Mode ................................ 4-84
Assembling the Source Files ................... 4-85
Linking the Object Files ....................... 4-85
Programming the EPROMs ...................... 4-85
INTRODUCTION

This application brief will illustrate the steps involved in customizing a modem application using the 89024 modem chip set. Specifically, it will show how one may add V.23 capability to an 89024 modem design as embodied in the MEK II (Intel Modem Evaluation Kit) running software version 3.2.

GENERAL DESCRIPTION

This design consists of using the 89026 processor to control a separate V.23 Data Pump IC (Texas Instruments TCM3105) to support V.23 modulation in addition to the currently supported V.22bis/V.22/V.21/Bell212/Bell103.

The modem is placed in V.23 mode using the “AT&Al” command and is returned to normal operation with the “AT&AO” command. The originating modem dials normally using “AT” commands and then 2 seconds after completion of dialing, the modem sends 75 bps V.23 carrier. The answering modem, upon detecting a ring signal, goes off hook and sends 1200 bps V.23 carrier. The originate modem sends data at 75 bps and receives data at 1200 bps, while the answer modem sends at 1200 bps and receives at 75 bps. Both respond to “escape” at 1200 bps and command mode is always at 1200/1200 bps. The V.23 transmit level is fixed. Backward channel CCITT circuits are not supported, data is always transmitted from pin 2 and received at pin 3.

This application brief does not address the issues of V.25 calling tones or V.25 calling station identification.

HARDWARE DESCRIPTION

The MEK II is modified by adding a Texas Instruments TCM3105 FSK Modem IC. This Modem chip does not have an on-chip 4-wire to 2-wire hybrid circuit, so we use a dual op-amp MC1458 for this purpose. In order to control the TCM3105 we use 3 additional outputs of the 74LS373 latch that is already used to latch the JS and AA signals from the microcontroller address/data bus. A 74LS157 2-to-1 line data selector is used to select the source of received data and the source of “energy detect” signal to the microcontroller.

V.23 Modem IC

The TCM3105 (U102) is a CMOS V.23 modem in a 16-pin package that consumes only 40 mW. It requires an external 4.4336 MHz crystal connected between pins 15 and 16 to derive timing. A resistor divider sets the carrier detect threshold by adjusting the voltage at pin 10. Bias distortion may be minimized by adjusting the voltage at pin 7. Pins 5, 13 and 12 together set the various modes of operation. These pins are connected to pins 6, 9 and 12 respectively of 74LS373 (U18) and are controlled through bits 2, 3 and 4 and executing a “STore” instruction to any even address of external memory (since this is the only external memory to be used). The modes of interest to us are:

<table>
<thead>
<tr>
<th>Mode</th>
<th>Bits Setting</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Off</td>
<td>xxx111xx</td>
<td>Transmit Disabled</td>
</tr>
<tr>
<td>Orig</td>
<td>xxx001xx</td>
<td>TX 1200 bps, RX 75 bps (V.23 Answer mode)</td>
</tr>
<tr>
<td>Ans</td>
<td>xxx010xx</td>
<td>TX 75 bps, RX 1200 bps (V.23 Originate mode)</td>
</tr>
</tbody>
</table>

74LS157 Data Selector

This IC is always enabled and the select signal is connected to the 6th output (bit 5) of the 74LS373 latch (U18). “SToring” a “0” to bit 5 of the latch selects “normal” mode of operation, while “SToring” a “1” to bit 5 selects V.23 mode. During “normal” mode, Receive Data (RXD) is routed from the 89026 microcontroller to the DTE and Energy Detect (ED) is routed from the 89027 AFE to the microcontroller. During V.23 mode RXD goes from the TCM3105 to the DTE and ED goes from the TCM3105 to the microcontroller. Transmit Data (TXD) is always connected from the DTE to both the 89026 and the TCM3105.

MC1458 Dual Op-Amp

This IC is configured as an active hybrid circuit, converting the 4-wire transmit and receive signals to 2-wire to drive the line transformer. The transmitted signal is also summed, but since only one of the transmitters will be active at a time, this will not be a problem. The 89027 has pin 10 tied low so as to disable the AFE’s on-chip hybrid.

A schematic diagram of these changes is shown in Figure 1.
Figure 1. Schematic Diagram
SOFTWARE DESCRIPTION

We choose the "&A" command as one that is not currently used by major "AT" compatible modem vendors. We will use S23 bit 3 as the bit to indicate that V.23 mode has been selected, since this bit is unused in "AT" modems. "&A1" will cause S23 bit 3 to be set to a "1" and &A0 or just "&A" will cause it to be cleared. The modem software will examine this bit to determine whether V.23 mode has been selected.

Note that source code will always be written in capital letters and that the assembler ignores the rest of a line after a semi-colon (;). When giving modified source code I will usually "comment out" the original code by adding a semi-colon to the beginning of the line. This is an excellent practice to facilitate the documentation of changes.

By convention we name the source files: nmxxx.SRC (where n.m is the software version and xxx is the generic file name). Since we are using software version 3.2 the files that we will be changing are:

32AAD.SRC register assignment definitions ($INCLUDEd with all source files)
32CMD.SRC Command Decoder
32CPM.SRC Call Progress Monitor routines
32HND.SRC Handshake routines
32DATA.SRC Data Mode routines

Decoding AT&A1 Command and Setting the S23 Bit
All of these changes will be done to the 32CMD.SRC file.

Since many commands simply modify S-register bits, we can take advantage of the "COMMON_REGISTER_OPERATIONS:" code by adding our command to the necessary tables and allowing it to be decoded as a register-modifying command.

Add as the last entry in TABLE_I:

| DCB (3 * 32) + (S23-S0) ; AND_A_CMD |

This will tell the common routine that this command affects bit 3 of S23. The table is set up so that it only occupies one byte per entry, with the bit number in the upper 3 bits and the register number in the lower 5 bits.

Add the command to the command list and the command vector table:

| AND_CMDS:        | DCB "CJIPLRSDG" |
|                 | ;                |
|                 | DCB "MXFWZT", 0 ; was like this |
|                 | DCB "MXAFWZT", 0 ; added &A command betw X and F |

| CMD_LU_TBL:  | .... |
|             | .... |
|             | .... |
|             | DCB AND_G_CMD-G1, AND_M_CMD-G1, AND_X_CMD-G1, AND_F_CMD-G2 |
|             | DCB AND_G_CMD-G1, AND_M_CMD-G1, AND_X_CMD-G1, AND_A_CMD-G1 |
|             | DCB AND_F_CMD-G2 |

The command vector table is the address offset of the command label from that of the first command (G1 EQU A_CMD). In the interests of saving space this offset table is only 1 byte per entry and so it has to be split into 2 groups as the range of addresses of command labels is more than 255 bytes. When modifying command code it is worth checking the list file to make sure that the CMD_LU_TBL: entries do not get bigger than OFFH and wrap around through 0, causing those commands to branch to the wrong address.
Fix the branch vector calculator and the dial command offset calculator because the 1st group of commands are now 33 instead of 32:

```
GENERATE_BRANCH_VECTOR:
  ADD TEMP_CMD_3, #61 ; ADD OFFSET TO 1ST CMD GROUP
  CMPB TEMP_CMD_2, #32 ; FIRST 32 CMDS FIT IN
  CMPB TEMP_CMD_2, #33 ; FIRST 33 CMDS FIT IN
```

```
D_R_CMD:
  SUBB CPM_CONTROL, TEMP_CMD_2, #36
  SUBB CPM_CONTROL, TEMP_CMD_2, #37
```

Add the command label with the rest of the register modifying commands:

```
Y_CMD:
  AND_A_CMD: ; added &A command for V.23 operation
  AND_C_CMD:
```

**Updating the Output Pins to Control the TCM3105 and Data Selector**

The IO_CONTROL: section of code in file 32CMD.SRC runs all the time and could be considered the “background routine”. This is where the RS232 leads are updated, the health of the other routines is checked and the 74LS373 latch (U18) is written and is thus an appropriate place for the TCM3105 chip and the Data Selector (Data Mux) to be updated.
Add the following code after END_JS__UPDATE:

### END_JS__UPDATE:

<table>
<thead>
<tr>
<th>Instruction</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>V_23_UPDATE:</strong></td>
<td></td>
</tr>
<tr>
<td>ANDB TEMP_CMD_1, #11011111B</td>
<td>MUX to NON-V.23 POSN</td>
</tr>
<tr>
<td>ORB TEMP_CMD_1, #0001100B</td>
<td>SET V.23 CHIP OFF</td>
</tr>
<tr>
<td>JBC S23, 3, END_V_23_UPDATE</td>
<td>JMP if NOT IN V.23 MODE</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>JBC CNTRL_C, 1, END_V_23_UPDATE</td>
<td>JMP if NOT in HND or DATA MODE</td>
</tr>
<tr>
<td>ANDB TEMP_CMD_1, #11100111B</td>
<td>SET V.23 CHIP to ANS MODE</td>
</tr>
<tr>
<td>ORB TEMP_CMD_1, #00000100B</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>JBC S14, 7, NOT_ORIG_MODE</td>
<td>JMP if S REG SET to ANS MODE</td>
</tr>
<tr>
<td>ANDB TEMP_CMD_1, #11101011B</td>
<td>SET V.23 CHIP to ORIG MODE</td>
</tr>
<tr>
<td>ORB TEMP_CMD_1, #00001000B</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>NOT_ORIG_MODE:</strong></td>
<td></td>
</tr>
<tr>
<td>JBC CNTRL_C, 0, END_V_23_UPDATE</td>
<td>JMP if NOT in DATA MODE</td>
</tr>
<tr>
<td>JBS CNTRL_C, 2, END_V_23_UPDATE</td>
<td>JMP if CMD FUNCTS ENABLED</td>
</tr>
<tr>
<td>ORB TEMP_CMD_1, #00100000B</td>
<td>DATA MODE, SO MUX to V.23 POSN</td>
</tr>
</tbody>
</table>

The next instruction in the source code STores the contents of TEMP_CMD_1 to PORT3, and so updates the Data Mux.

In order to ensure that the Data Mux gets set before the “OK” message is sent when entering the on-line escape state (response to “+++”), add a line of code after the three “ORB” instructions:

### VALID_ESCAPE_SEQUENCE:

<table>
<thead>
<tr>
<th>Instruction</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ORB CNTRL_F, #00010000B</td>
<td>ENABLE ESCAPE STATE</td>
</tr>
<tr>
<td>ORB CNTRL_C, #00000100B</td>
<td>ENABLE CMD FUNCTIONS</td>
</tr>
<tr>
<td>ORB MSG_RQST, #00100000B</td>
<td>SEND &quot;OK&quot; MESSAGE WITH MSG RQST</td>
</tr>
<tr>
<td>JBS S23, 3, ESCAPE_DETECT_END</td>
<td>TRICK TO FORCE 1 MORE PASS THRU</td>
</tr>
<tr>
<td></td>
<td>IO_CONTROL FOR MUX SETUP BEFORE GOING TO COMMAND DECODER</td>
</tr>
</tbody>
</table>
After a dial command is executed by the Command routine, it will activate the Call Progress routines.

The V.23 Call Progress Monitor Routines
The 32CPM.SRC routines check for call progress signals on the phone line and also for answer tone from the remote answering modem. Since a V.23 modem will answer with a 1300 Hz tone (1200 bps mark frequency), the AFE receive filter must be set to V.22 answer mode so as to pass this frequency to the energy detect circuitry.

Add three lines of code at the label SET_ANSWER_CONT:

```
SET_ANSWER_CONT:
  ANDB CPM_FLAG, #11101111B ; FLAG ANSWER PROCESSING FOR HOUSEKEEPING
  JBC S23, 3, SET_ANSWER_CONT_1 ; IF V23 MODE THEN
  LDB AFE_BYTE3, #01000000B ; SET FILTER TO QAM ANS FOR
  SET_ANSWER_CONT_1: ; 1300Hz CARRIER DETECTION
  SJMP SIGNAL_MONITOR_INIT
```

The CPM routines will hand over control to the Handshake routines which we need to modify for V.23 handshake.

The V.23 Handshake Routines
The Handshake mode 32HND.SRC is entered for the first time after successful completion of the Call Progress routines. The first time that HANDSHAKE_MODE: is called, it goes through the Initialization code before the main routine is executed, thereafter the Initialization is skipped. The Main routine is entered at a rate of 600 times per second or more and consists of checking for Energy Detect and then branching to the routine address saved in TX_RTN_ADDR. The logical flow of the handshaking is controlled by changing the contents of TX_RTN_ADDR to the address of the routine to be executed the next time Handshake is called.

![Flowchart](292058-3)

**Figure 2**

4-82
The Initialization required for V.23 consists of starting the S7 wait-for-carrier timer, starting a 2 second timer and loading a return address for the next time the routine executes. The following lines of source code are added (identified by “V23” at the start of the comment field) to the Handshake Initialization:

```
HANDSHAKE_INIT:
   ANDB    MODE_STATUS, #10111111B  ; CLEAR INIT FLAG
   JBC     S23, 3, NOT_V23_INIT  ; V23
V23_HND_INIT:
   ADDB    S7_TIMER, TIME_BASE_SECOND, S7  ; V23 INIT S7 DCD TIMER
   ADDB    TX_TIMER, TIME_BASE_100MS, #20D  ; V23 INIT 2 SEC TIMER
   LD      TX_RTN_ADDR, #V23_HND_WAIT  ; V23
   SJMP    HND_INIT_END  ; V23
NOT_V23_INIT:
   ; V23
```

After the initialization code is executed once, the software will keep branching to V23_HND_WAIT: until the 2-second timer has expired, then it will initiate a “CONNECT” message. While the Connect message is being sent, the software will branch to V23_HND_MESSAGE:, then it will set up the Data mode and thereafter the Data Mode will be called instead of the Handshake mode.

```
V23_HND_WAIT:
   CMPB    TIME_BASE_100MS, TX_TIMER  ; V23 TIMER EXPIRED YET?
   JNE     V23_HND_END  ; V23
V23_HND_MESSAGE_INIT:
   LDB     MESSAGE_REQUEST, #00100001B  ; V23 START CONNECT MESSAGE
   LD      TX_RTN_ADDR, #V23_HND_MESSAGE  ; V23
   SJMP    V23_HND_END  ; V23
V23_HND_MESSAGE:
   JBS     MESSAGE_REQUEST, 5, V23_HND_END  ; V23 MESSAGE SENT YET?
V23_HND_MESSAGE_END:
   ; V23
   ANDB    COPY_PORT4, #10111111B  ; V23 DCD HIGH AFTER CONNECT
V23_SET_UP_DATA_MODE:
   ; This is where we need to set up for going to data mode
   ORB     CNTRL_C, #00000011B  ; V23 GO TO DATA MODE
   ANDB    AFE_BYTE4, #00111111B  ; V23 TXMITTER OFF, AFE OFF
   CLR     DM_FLAGS  ; V23 CLEAR FLAGS FOR DM
   ORB     MODE_STATUS, #10000000B  ; V23 INIT DATA MODE
V23_HND_END:
   LJMP    HANDSHAKE_MODE_END  ; V23
```

4-83
V.23 Data Mode
The modifications required in the Data Mode consist of checking for V23 mode and skipping past:
- Initialization
- Send space disconnect (twice)
- Receive space disconnect
- Loss of carrier disconnect
- Retrain request
- Test mode

```
DATA_MODE_INIT:
; DM FLAGS ALREADY CLEARED IN HANDSHAKE MODE
    ANDB DM_FLAGS, #7FH ; CLEAR INITIALIZE FLAG
    JBS S23, 3, DATA_MODE_INIT_END ; IF V23 THEN INIT DONE

DISCONNECT_INIT:
ANDB DM_FLAGS, #11111101B ; CLEAR DISCONNECT INIT FLAG
JBS S23, 3, HANG_UP ; V23 FORGET SPACE DISCONNECT

SEND_SPACE:
JBC S21, 7, HANG_UP ; IF BREAK_DISCONNECT DISABLED
JBS S23, 3, HANG_UP ; V23 FORGET BREAK

CHECK_DISCONNECT:
CHECK_BREAK:
JBS S23, 3, SET_BREAK_TIME ; V23 FORGET BREAK

CHECK_CARRIER_LOSS:
JBS PORTO, 7, CHECK_CARRIER_LOSS_END ; SKIP IF ED IS HIGH
ORB DM_FLAGS, #0100000B ; SET CDLOSS FLAG
ADDB EDOFF_TIME, TIME_BASE_100MS, S10 ; CDOFF THRESHOLD IN REGISTER
INCB EDOFF_TIME ; PUT AN OFFSET IN TIME FOR PROPER
; OPERATION DURING TM EXIT
JBS S23, 3, CARRIER_LOSS_END ; ALL DONE IF V23 MODE

QAM_RETRAIN:
JBS S23, 3, SJMP_CHECK_TEST_MODE ; SKIP RETRAIN IF V23 MODE

CHECK_S16_STATUS:
; EXAMINE S16 REGISTER FOR ANY TEST MODES AND SET FLAG
JBS S23, 3, CHECK_S16_STATUS_END ; SKIP RETRAIN IF V23 MODE
```
Assembling the Source Files
The source files can be assembled by issuing the following commands at the DOS prompt:

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ASM96 32CMD.SRC</td>
<td></td>
</tr>
<tr>
<td>ASM96 32CPM.SRC</td>
<td></td>
</tr>
<tr>
<td>ASM96 32HND.SRC</td>
<td></td>
</tr>
<tr>
<td>ASM96 32DATA.SRC</td>
<td></td>
</tr>
</tbody>
</table>

Linking the Object Files
Link the object files by issuing the following command at the DOS prompt:

```
RL96 32HND.OBJ, 32INIT.OBJ, 32CMD.OBJ, 32CPM.OBJ, 32DATA.OBJ,
32S0FT.OBJ, 32HSI.OBJ, 32HSO.OBJ, 32RX.OBJ TO 32ATR
```

Programming the EPROMs
After the code has been linked and located, the code must be split into low and high byte segments for programming into EPROMS. The following IPPS session illustrates that process (IPPS prompts are not shown):

```
IPPS
I 80
FORMAT 32ATR
3 2 1
0 to 32ATR.LO
1 to 32ATR.HI
<enter>
TYPE
27128
COPY 32ATR.LO TO PROM
COPY 32ATR.HI TO PROM
EX
```

Custom routines can now be tested by placing EPROMS into target hardware.

REFERENCES
PCM Codec/Filter and Combo
2910A
PCM CODEC - μ-LAW
8-BIT COMPANDED A/D AND D/A CONVERTER

- Per Channel, Single Chip Codec
- CCITT G711 and G712 Compatible, ATT T1 Compatible with 8th Bit Signaling
- Microcomputer Interface with On-Chip Timeslot Computation
- Simple Direct Mode Interface When Fixed Timeslots are Used
- ±5% Power Supplies: +12V, +5V, -5V

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
<table>
<thead>
<tr>
<th>Pin No.</th>
<th>Symbol</th>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>CAP1&lt;sub&gt;x&lt;/sub&gt;</td>
<td>Hold</td>
<td>Connections for the transmit holding capacitor. Refer to Applications section.</td>
</tr>
<tr>
<td>2</td>
<td>CAP2&lt;sub&gt;x&lt;/sub&gt;</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 FS&lt;sub&gt;x&lt;/sub&gt; and the sample value is held in the external capacitor connected to the CAP1&lt;sub&gt;x&lt;/sub&gt; and CAP2&lt;sub&gt;x&lt;/sub&gt; 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>DR&lt;sub&gt;R&lt;/sub&gt;</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 FS&lt;sub&gt;R&lt;/sub&gt;, CLK&lt;sub&gt;R&lt;/sub&gt;, D&lt;sub&gt;C&lt;/sub&gt;, and CLK&lt;sub&gt;C&lt;/sub&gt;.</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>VFR&lt;sub&gt;R&lt;/sub&gt;</td>
<td>Output</td>
<td>Analog output. The voltage present on VFR&lt;sub&gt;R&lt;/sub&gt; is the decoded value of the PCM word received on lead DR. 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></td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>GRDD</td>
<td>Ground</td>
<td>Ground return common to the logic power supply, V&lt;sub&gt;CC&lt;/sub&gt;.</td>
</tr>
<tr>
<td>14</td>
<td>DX&lt;sub&gt;R&lt;/sub&gt;</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 FS&lt;sub&gt;X&lt;/sub&gt;, CLK&lt;sub&gt;X&lt;/sub&gt;, D&lt;sub&gt;C&lt;/sub&gt;, and CLK&lt;sub&gt;C&lt;/sub&gt;. TTL three-state output.</td>
</tr>
<tr>
<td>15</td>
<td>TSX&lt;sub&gt;X&lt;/sub&gt;</td>
<td>Output</td>
<td>Normally high, this signal goes low while the Codec is transmitting an 8-bit PCM word on the DX&lt;sub&gt;X&lt;/sub&gt; lead. (Timeslot information used for diagnostic purposes and also to gate the data on the DX lead.) Open drain output.</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>CLK&lt;sub&gt;R&lt;/sub&gt;</td>
<td>Input</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>18</td>
<td>FS&lt;sub&gt;R&lt;/sub&gt;</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>$-5V \pm 5%$, 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_R 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.

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.

### Bit Assignments

```
<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 (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 would look something like this:
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). CLKC 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 CLKC 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 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 (VCC or VDD 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 (FSX, FSR) 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. (FSX and FSR may be asynchronous.)

Direct Control Mode

The direct mode of operation will be selected when the CLKC pin is strapped to the +5 volt 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 2910A it will operate properly when plugged into a 2910 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 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 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 theVFX 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 D_x lead at the proper timeslot occurrence of the following frame. The A/D converter saturates at approximately $\pm 2.2$ volts RMS ($\pm 3.1$ volts peak).

### Decoding

The PCM word is fetched by the D_R 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 $\pm 2.2$ volts RMS ($\pm 3.1$ volts peak).

### Signaling

The duration of the F_S_x and F_S_R 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 (CLK_x period for F_S_x, CLK_R period for F_S_R) designates a non-signaling frame.
- A frame synchronization pulse which is two full clock periods in duration (two CLK_x periods for F_S_x, two CLK_R periods for F_S_R) designates a signaling frame.

On the encoding side, when the F_S_x pulse is widened, the 8th bit of the PCM word will be replaced by the value on the S_iG_X input at the time when the 8th bit is output on the D_x lead.

On the decoding side, when the F_S_R pulse is widened, the 8th bit of the PCM word is detected and transmitted on the S_iG_R 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 S_iG_R 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 CLK_X, CLK_R of 1.544 Mb/s). However, the 193rd bit may be blanked (equivalent to CLK_X, CLK_R 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 D_C and CLK_C leads. These allow the Codec to be reactivated. In the microcomputer mode the Codec is placed into standby by loading a control word (D_C) with a “1” in bits 1 and 2 locations. In the direct mode when D_C 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 D_X 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 V_CC or V_DD 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 Codices is plugged into a “hot” system where V_CC or V_DD is the last contact made. It may be necessary to trim back the edge connector pins or fingers on V_CC or V_DD 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 D_X during the application of power supplies.

The device is also tolerant of transients in the negative supply (V_BB) so long as V_BB remains more negative than −3.5 volts. V_BB 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 $x = \text{input signal}$

$$\text{Sgn}(x) = \text{sign of input signal}$$

$$\mu = 255$$ (defined by AT & T)

The 2910A μ = 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:

$$y_n = \frac{x_n + x_{n+1}}{2} \text{ for } 1 \leq n \leq 127$$

$$y_0 = x_0 = 0 \text{ for } n = 0$$

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 $x_n$</th>
<th>PCM Word($^{(3)}$)</th>
<th>Normalized Value at Decoder Output $y_n$</th>
<th>Decoder Output Value Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td>16 x 256</td>
<td>8159($^{(3)}$)</td>
<td>128</td>
<td>(8159)</td>
<td>1 0 0 0 0 0 0 0 0</td>
<td>8031</td>
<td>27</td>
</tr>
<tr>
<td>7</td>
<td>16 x 128</td>
<td>4063</td>
<td>127</td>
<td>7903</td>
<td>(see Note 2)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>16 x 64</td>
<td>2015</td>
<td>96</td>
<td>2015</td>
<td>(see Note 2)</td>
<td></td>
<td></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 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>(see Note 2)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>16 x 8</td>
<td>223</td>
<td>48</td>
<td>223</td>
<td>(see Note 2)</td>
<td></td>
<td></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 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>(see Note 2)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1 x 1</td>
<td>2</td>
<td>3</td>
<td>1 1 1 1 1 1 1 1 0</td>
<td>2</td>
<td>1</td>
<td></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 11. Circuit Interface—without External Auto Zero
**Holding Capacitor**

For an 8 KHz sampling system the transmit holding capacitor $\text{CAP}_x$ should be $2000 \text{ pF} \pm 20\%$. 

**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 ($\text{VF}_x$) 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 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 $\text{VF}_x$ and AUTO leads with the recommended values of $C_1 = 0.3 \mu\text{F}$, $R_1 = 150 \text{ K}\Omega$, $R_2 = 330 \text{ K}\Omega$, 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 $\text{D}_x$ 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 $\text{TS}_x$ 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-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 V_{BB} .................. −0.3V to +20V
V_{CC}, V_{DD}, GRDD, and GRDA with
  Respect to V_{BB} .................. −0.3V to +20V
Power Dissipation .................. 1.35W

**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>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_L</td>
<td>Low Level Input Current</td>
<td>10</td>
<td>μA</td>
<td>V_{IN} &lt; V_{IL}</td>
</tr>
<tr>
<td>Il_H</td>
<td>High Level Input Current</td>
<td>10</td>
<td>μA</td>
<td>V_{IN} &gt; V_{IH}</td>
</tr>
<tr>
<td>V_{IL}</td>
<td>Input Low Voltage</td>
<td>0.6</td>
<td>V</td>
<td></td>
</tr>
<tr>
<td>V_{IH}</td>
<td>Input High Voltage</td>
<td>2.0</td>
<td>V</td>
<td></td>
</tr>
<tr>
<td>V_{OL}</td>
<td>Output Low Voltage</td>
<td>0.4</td>
<td>V</td>
<td>D_X, I_{OL} = 4.0 mA</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>SIGR, I_{OL} = 0.5 mA</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>T_Sx, I_{OL} = 3.2 mA, Open Drain</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>PDN, I_{OL} = 1.6 mA, Open Drain</td>
</tr>
<tr>
<td>V_{OH}</td>
<td>Output High Voltage</td>
<td>2.4</td>
<td>V</td>
<td>D_X, I_{OH} = 15 mA</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>SIGR, I_{OH} = 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>Z_{AI}</td>
<td>Input Impedance when</td>
<td>125</td>
<td>300</td>
<td>500 Ω</td>
</tr>
<tr>
<td></td>
<td>Sampling, V_{FX}</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Z_{AO}</td>
<td>Small Signal Output Impedance</td>
<td>100</td>
<td>180</td>
<td>300 Ω</td>
</tr>
<tr>
<td></td>
<td>V_{FR}</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>V_{OR}</td>
<td>Output Offset Voltage at V_{FR}</td>
<td>±50</td>
<td>mV</td>
<td>all &quot;1s&quot; code sent to D_X</td>
</tr>
<tr>
<td>V_{IX}</td>
<td>Input Offset Voltage at V_{FX}</td>
<td>±5</td>
<td>mV</td>
<td>V_{FX} Voltage Required to Produce all &quot;1s&quot; Code at D_X</td>
</tr>
<tr>
<td>V_{OL}</td>
<td>Output Low Voltage at AUTO</td>
<td>V_{BB} (V_{BB} + 2)</td>
<td>V</td>
<td>400 KΩ to GRDA</td>
</tr>
<tr>
<td>V_{OH}</td>
<td>Output High Voltage at AUTO</td>
<td>(V_{CC} − 2)</td>
<td>V_{CC}</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></td>
<td></td>
<td>Min</td>
<td>Typ(1)</td>
<td>Max</td>
</tr>
<tr>
<td>I_{DDO}</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</td>
<td>7.0</td>
<td>mA</td>
</tr>
<tr>
<td>I_{BBO}</td>
<td>Standby Current</td>
<td>1</td>
<td>2.5</td>
<td>mA</td>
</tr>
<tr>
<td>I_{DDI}</td>
<td>Operating Current</td>
<td>11</td>
<td>16</td>
<td>mA</td>
</tr>
<tr>
<td>I_{CCI}</td>
<td>Operating Current</td>
<td>13</td>
<td>21</td>
<td>mA</td>
</tr>
<tr>
<td>I_{BBI}</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_A = 25°C and nominal power supply values.
A.C. CHARACTERISTICS

$T_A = 0°C$ to $+70°C$, $V_{DD} = +12V \pm 5\%$, $V_{CC} = +5V \pm 5\%$, $V_{BB} = -5V \pm 5\%$, $GRDA = 0V$, $GRDD = 0V$, 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>S/D</td>
<td>Signal/Tone Distortion Ratio,</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>C-Message Weighted</td>
<td>36</td>
<td>dBi</td>
<td>$VF_X = 1.02 \text{ KHz, Sinusoid}$</td>
</tr>
<tr>
<td></td>
<td>Half Channel</td>
<td>30</td>
<td>dBi</td>
<td>$-30 \text{ dBm} \leq VF_X \leq 0 \text{ dBm}$</td>
</tr>
<tr>
<td></td>
<td>(See Figure 1)</td>
<td>27</td>
<td>dBi</td>
<td>$-40 \text{ dBm} \leq VF_X &lt; -30 \text{ dBm}$</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>$-45 \text{ dBm} \leq VF_X &lt; -40 \text{ dBm}$</td>
</tr>
<tr>
<td>$\Delta G$</td>
<td>Gain Tracking Deviation</td>
<td>±0.25 ±0.30</td>
<td>dB/mV</td>
<td>$VF_X = 1.02 \text{ KHz, Sinusoid}$</td>
</tr>
<tr>
<td></td>
<td>Half Channel</td>
<td>±0.60 ±0.70</td>
<td>dB/mV</td>
<td>$-37 \text{ dBm} \leq VF_X \leq 3 \text{ dBm}$</td>
</tr>
<tr>
<td></td>
<td>Reference Level 0 dBm0</td>
<td>±1.5 ±1.8</td>
<td>dB/mV</td>
<td>$-50 \text{ dBm} \leq VF_X &lt; -37 \text{ dBm}$</td>
</tr>
<tr>
<td>$\Delta G_V$</td>
<td>$\Delta G$ Variation with Supplies</td>
<td>±0.0002 ±0.0004</td>
<td>dB/mV</td>
<td>$-37 \text{ dBm} \leq VF_X &lt; 3 \text{ dBm}$</td>
</tr>
<tr>
<td></td>
<td>Half Channel</td>
<td>±0.0004 ±0.0008</td>
<td>dB/mV</td>
<td>$-50 \text{ dBm} \leq VF_X &lt; -37 \text{ dBm}$</td>
</tr>
<tr>
<td>$\Delta G_T$</td>
<td>$\Delta G$ Variation with Temperature</td>
<td>±0.001 ±0.002</td>
<td>dB/mV</td>
<td>$-37 \text{ dBm} \leq VF_X &lt; 3 \text{ dBm}$</td>
</tr>
<tr>
<td></td>
<td>Half Channel</td>
<td>±0.002 ±0.005</td>
<td>dB/mV</td>
<td>$-50 \text{ dBm} \leq VF_X &lt; -37 \text{ dBm}$</td>
</tr>
<tr>
<td>$N_{IC1}$</td>
<td>Idle Channel Noise, C-Message Weighted</td>
<td>2</td>
<td>dBm0</td>
<td>No Signaling(3)</td>
</tr>
<tr>
<td>$N_{IC2}$</td>
<td>Idle Channel Noise, C-Message Weighted</td>
<td>10</td>
<td>dBm0</td>
<td>with 6th and 12th Frame Signaling(3)</td>
</tr>
<tr>
<td>$N_{IC3}$</td>
<td>Idle Channel Noise, C-Message Weighted</td>
<td>14</td>
<td>dBm0</td>
<td>with 1 KHz Sign Bit Toggle</td>
</tr>
<tr>
<td>HD</td>
<td>Harmonic Distortion (2nd or 3rd)</td>
<td>-48</td>
<td>dB</td>
<td>$VF_X = 1.02 \text{ KHz, 0 dBm0}$; Measured at Decoder Output $VF_R$</td>
</tr>
<tr>
<td>IMD</td>
<td>Intermodulation Distortion</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>2nd Order</td>
<td>-45</td>
<td>dB</td>
<td>4-Tone Stimulus in Accordance with $BSTR \text{ PUB 41009}$</td>
</tr>
<tr>
<td></td>
<td>3rd Order</td>
<td>-55</td>
<td>dB</td>
<td></td>
</tr>
</tbody>
</table>

NOTES:

1. Typical values are for $T_A = 25°C$ and nominal supply values.
2. Measured in one direction, either decoder or encoder and an ideal device, at 23°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)

Figure 14. Gain Tracking Deviation ($\Delta G$) (Half-Channel)
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>$D_{mW}$</td>
<td>Digital Milliwatt Response</td>
<td>5.53</td>
<td>5.63</td>
<td>5.73</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>dB</td>
<td>Supplies ±5%(4)</td>
</tr>
<tr>
<td>$A_{IR}$</td>
<td>Input Dynamic Range</td>
<td>2.17</td>
<td>2.20</td>
<td>2.23</td>
</tr>
<tr>
<td>$A_{IRT}$</td>
<td>Input Dynamic Range with Temperature</td>
<td>$-0.5$</td>
<td>$mV_{RMS}/°C$</td>
<td>Relative to 23°C</td>
</tr>
<tr>
<td>$A_{IRS}$</td>
<td>Input Dynamic Range with Supplies</td>
<td>± 18</td>
<td>$mV_{RMS}$</td>
<td>Supplies ±5%</td>
</tr>
<tr>
<td>$A_{OR}$</td>
<td>Output Dynamic Range, $V_{FR}$</td>
<td>2.13</td>
<td>2.16</td>
<td>2.19</td>
</tr>
<tr>
<td>$A_{ORT}$</td>
<td>$A_{OR}$ Variation with Temperature</td>
<td>$-0.5$</td>
<td>$mV_{RMS}/°C$</td>
<td>Relative to 23°C</td>
</tr>
<tr>
<td>$A_{ORS}$</td>
<td>$A_{OR}$ Variation with Supplies</td>
<td>± 18</td>
<td>$mV_{RMS}$</td>
<td>Supplies ±5%</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>$PSRR_1$</td>
<td>$V_{DD}$ Power Supply Rejection Ratio</td>
<td>45</td>
<td></td>
<td>dB</td>
</tr>
<tr>
<td>$PSRR_2$</td>
<td>$V_{BB}$ Power Supply Rejection Ratio</td>
<td>35</td>
<td></td>
<td>dB</td>
</tr>
<tr>
<td>$PSRR_3$</td>
<td>$V_{CC}$ Power Supply Rejection Ratio</td>
<td>50</td>
<td></td>
<td>dB</td>
</tr>
<tr>
<td>$PSRR_4$</td>
<td>$V_{DD}$ Power Supply Rejection Ratio</td>
<td>50</td>
<td></td>
<td>dB</td>
</tr>
<tr>
<td>$PSRR_5$</td>
<td>$V_{BB}$ Power Supply Rejection Ratio</td>
<td>45</td>
<td></td>
<td>dB</td>
</tr>
<tr>
<td>$PSRR_6$</td>
<td>$V_{CC}$ Power Supply Rejection Ratio</td>
<td>50</td>
<td></td>
<td>dB</td>
</tr>
<tr>
<td>$CT_R$</td>
<td>Crosstalk Isolation, Receive Side</td>
<td>75</td>
<td>80</td>
<td>dB</td>
</tr>
<tr>
<td>$CT_T$</td>
<td>Crosstalk Isolation, Transmit Side</td>
<td>75</td>
<td>80</td>
<td>dB</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 $T_A = 25^\circ C$ and nominal power supply values.
2. D.U.T. decoder; impose 200 m$V_{P-P}$, 1.02 KHz on appropriate supply; measurement made at decoder output; decoder in idle channel conditions.
3. D.U.T. encoder; impose 200 m$V_{P-P}$, 1.02 KHz on appropriate supply; measurement made at encoder output; encoder in idle channel conditions.
4. $V_{FX}$ of D.U.T. encoder = 1.02 KHz, 0 dBm0. Decoder under quiet channel conditions; measurement made at decoder output.
5. $V_{FX}$ = 0 Vrms. Decoder = 1.02 KHz, 0 dBm0. Encoder under quiet channel conditions; measurement made at encoder output.
A.C. CHARACTERISTIC—TIMING SPECIFICATION\(^{(1)}\)

\[ T_A = 0^\circ C \text{ to } +70^\circ C, \ V_{DD} = +12V \pm 5\%, \ V_{CC} = +5V \pm 5\%, \ V_{BB} = -5V \pm 5\%, \ GRDA = 0V, \ GRDD = 0V, \]

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</td>
<td>30</td>
<td>ns</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_{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>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</td>
<td>180</td>
<td>ns</td>
</tr>
<tr>
<td>( t_{DHX} )</td>
<td>Data Hold Time</td>
<td>80</td>
<td>230</td>
<td>ns</td>
</tr>
<tr>
<td>( t_{HZX} )</td>
<td>Data Float on TS Exit</td>
<td>75</td>
<td>245</td>
<td>ns</td>
</tr>
<tr>
<td>( t_{SON} )</td>
<td>Timeslot X to Enable</td>
<td>30</td>
<td>220</td>
<td>ns</td>
</tr>
<tr>
<td>( t_{SOFF} )</td>
<td>Timeslot X to Disable</td>
<td>70</td>
<td>225</td>
<td>ns</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</td>
<td>150</td>
<td>ns</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 ( \frac{1}{16} )</td>
<td>9 ( \frac{1}{16} )</td>
<td>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</td>
<td>150</td>
<td>ns</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_{DHC} )</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 \( t_{HZX} \) 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.
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
- Telemetry
- Secure Communications Systems
- Signal Processing Systems

![Figure 1. Pin Configuration](image)

![Figure 2. Pin Names](image)

![Figure 3. Block Diagram](image)
<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 CLKC.</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 CLKC. TTL threestate 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 (FSx) into an 8-bit PCM word which is sent out on the Ox lead at the proper time. Similarly, on the receive link, 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 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 DC lead (data) and the CLKc lead (clock). The loading is synchronous 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.

**Figure 4. Typical Line Termination**

5-21
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_X and CLK_R). All Codecs tied to the same data bus receive identical framing pulses (FS_X and FS_R). 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_C and D_C for the desired transmit and receive timeslots according to the description in the Codec Control Section. All Codecs tied to the same D_R 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_X = CLK_R). 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_C–D_C interface in the microcomputer mode.

1. A complete timeslot assignment, consisting of eight negative transitions of CLK_C, 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_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 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 (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_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 (FS_X, FS_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 during 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**

In this example the Codec interface to the PCM highway then functions as shown below. (FS_X and FS_R may be asynchronous.)

**Figure 6. Microcomputer Mode PCM Highway Example**

**Direct Control Mode**

The direct mode of operation will be selected when the CLK_C pin is strapped to the +5V supply (VCC). In this mode, the D_C pin is an active low chip select. In other words, when D_C is low, the device transmits and receives in the timeslots which follow the appropriate framing pulses. With D_C high the device is in the power down state. Even though CLK_C 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 (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 D_C in order for the D_C 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 D_C 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 CLK_C 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 (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\textsubscript{X} lead. An internal switch samples the signal and the hold function is performed by the external capacitor connected to the CAP1\textsubscript{X} and CAP2\textsubscript{X} leads. The sampling and conversion is synchronized with the transmit timeslot. The PCM word is then output on the D\textsubscript{X} lead at the proper timeslot occurrence of the following frame. The A/D converter saturates at approximately ±2.2V RMS (±3.1V 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 VF\textsubscript{R} 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 D\textsubscript{C} and CLK\textsubscript{C} leads. These allow the Codec to be reactivated. In the microcomputer mode the Codec is placed into standby by loading a control word (D\textsubscript{C}) with a “1” in bits 1 and 2 locations. In the direct mode when D\textsubscript{C} 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 D\textsubscript{X} 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 V\textsubscript{CC} or V\textsubscript{DD} 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 V\textsubscript{CC} or V\textsubscript{DD} is the last contact made. It may be necessary to trim back the edge connector pins or fingers on V\textsubscript{CC} or V\textsubscript{DD} 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 TS\textsubscript{X} and D\textsubscript{X} during the application of power supplies.

The device is also tolerant of transients in the negative supply (V\textsubscript{BB}) so long as V\textsubscript{BB} remains more negative than −3.5V. V\textsubscript{BB} 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.

**CONVERSION LAW**

The conversion law is commonly referred to as the A Law.

\[
F(x) = \text{Sgn}(x) \left[ \frac{1 + \log_{10}(A|x|)}{1 + \log_{10}A} \right], \quad 1/A \leq |x| \leq 1
\]

\[
F(x) = \text{Sgn}(x) \left[ \frac{A|x|}{1 + \log_{10}A} \right], \quad 0 \leq |x| \leq 1/A
\]

where: 
- \( x \) = the input signal
- \( \text{Sgn}(x) \) = sign of the input signal
- \( A = 87.6 \) (defined by CCITT)

The Codec provides a piecewise linear approximation of the logarithmic law through 13 segments. Each segment is made of 16 steps with the exception of the first segment, which has 32 steps. In adjacent 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 corresponding decision levels. The output levels \( y_n \) are related to the input levels \( x_n \) by the expression:

\[
y_n = \frac{x_{n-1} + x_n}{2}, \quad 0 < n \leq 128
\]

**Figure 10. Codec Transfer Characteristic**
### 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>1</td>
<td>32 x 2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>16 x 4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>16 x 8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>16 x 16</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>16 x 32</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>16 x 64</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>16 x 128</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></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 $CAP_x$ should be 2000 pF ± 20%.

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.

$D_X$ 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 or 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.

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 ($VF_X$) 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 $VF_X$ and AUTO leads with the recommended values of $C_1 = 0.3 \ \mu F$, $R_1 = 150 \ \Omega$, $R_2 = 330 \Omega$, and $R_3 = 470 \ \Omega$. 

Figure 11. Circuit Interface—Without External Auto Zero

Figure 12. Circuit Interface—With External Auto Zero
**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:** This is a production data sheet. The specifications are subject to change without notice.

*WARNING: Stressing the device beyond the "Absolute Maximum Ratings" may cause permanent damage. These are stress ratings only. Operation beyond the "Operating Conditions" is not recommended and extended exposure beyond the "Operating Conditions" may affect device reliability.*

**D.C. CHARACTERISTICS**

\( T_A = 0°C \) to +70°C, \( V_{DD} = +12V \pm 5\% \), \( V_{CC} = +5V \pm 5\% \), \( V_{BB} = -5V \pm 5\% \), 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></td>
<td></td>
<td>Min</td>
<td>Typ(1)</td>
<td>Max</td>
</tr>
<tr>
<td>( I_{IL} )</td>
<td>Low Level Input Current</td>
<td>10</td>
<td>( \mu A )</td>
<td>( V_{IN} &lt; V_{IL} )</td>
</tr>
<tr>
<td>( I_{IH} )</td>
<td>High Level Input Current</td>
<td>10</td>
<td>( \mu A )</td>
<td>( V_{IN} &gt; V_{IH} )</td>
</tr>
<tr>
<td>( V_{IL} )</td>
<td>Input Low Voltage</td>
<td>0.6</td>
<td>V</td>
<td></td>
</tr>
<tr>
<td>( V_{IH} )</td>
<td>Input High Voltage</td>
<td>2.2</td>
<td>V</td>
<td></td>
</tr>
<tr>
<td>( V_{OL} )</td>
<td>Output Low Voltage</td>
<td>0.4</td>
<td>V</td>
<td>( D_X, I_{OL} = 4.0 , mA )</td>
</tr>
<tr>
<td>( V_{OH} )</td>
<td>Output High Voltage</td>
<td>2.4</td>
<td>V</td>
<td>( D_X, I_{OH} = 15 , mA )</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></td>
<td></td>
<td>Min</td>
<td>Typ(1)</td>
<td>Max</td>
</tr>
<tr>
<td>( Z_{AI} )</td>
<td>Input Impedance when Sampling, ( V_{FX} )</td>
<td>125</td>
<td>300</td>
<td>500 ( \Omega )</td>
</tr>
<tr>
<td>( Z_{AO} )</td>
<td>Small Signal Output Impedance, ( V_{FR} )</td>
<td>100</td>
<td>180</td>
<td>300 ( \Omega )</td>
</tr>
<tr>
<td>( V_{OR} )</td>
<td>Output Offset Voltage at ( V_{FR} )</td>
<td>(-50)</td>
<td>50</td>
<td>( \text{mV} )</td>
</tr>
<tr>
<td>( V_{IX} )</td>
<td>Input Offset Voltage at ( V_{FX} )</td>
<td>(-5)</td>
<td>5</td>
<td>( \text{mV} )</td>
</tr>
<tr>
<td>( V_{OL} )</td>
<td>Output Low Voltage at AUTO</td>
<td>( V_{BB} ) (( V_{BB} + 2 ))</td>
<td>V</td>
<td>400 K( \Omega ) to GRDA</td>
</tr>
<tr>
<td>( V_{OH} )</td>
<td>Output High Voltage at AUTO</td>
<td>( (V_{CC} - 2) )</td>
<td>V</td>
<td>400 K( \Omega ) to GRDA</td>
</tr>
</tbody>
</table>
D.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)

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_{DDO}</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_{BBO}</td>
<td>Standby Current</td>
<td>1.0</td>
<td>2.5</td>
<td>mA</td>
</tr>
<tr>
<td>I_{DDI}</td>
<td>Operating Current</td>
<td>11</td>
<td>16</td>
<td>mA</td>
</tr>
<tr>
<td>I_{CCI}</td>
<td>Operating Current</td>
<td>13</td>
<td>21</td>
<td>mA</td>
</tr>
<tr>
<td>I_{BBI}</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 $T_A = 25^\circ{}C$ and nominal power supply values.

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.

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. CCITT G.712 Method 2 (Half Channel)</td>
<td>37</td>
<td></td>
<td>dB</td>
</tr>
<tr>
<td></td>
<td></td>
<td>31</td>
<td></td>
<td>dB</td>
</tr>
<tr>
<td></td>
<td></td>
<td>26</td>
<td></td>
<td>dB</td>
</tr>
<tr>
<td>&amp;G</td>
<td>2911A Gain Tracking Deviation Half Channel</td>
<td>±0.25</td>
<td>±0.30</td>
<td>dB</td>
</tr>
<tr>
<td></td>
<td></td>
<td>±0.60</td>
<td>±0.70</td>
<td>dB</td>
</tr>
<tr>
<td></td>
<td></td>
<td>±1.5</td>
<td>±1.8</td>
<td>dB</td>
</tr>
<tr>
<td>&amp;GV</td>
<td>&amp;G Variation with Supplies Half Channel</td>
<td>±0.0002</td>
<td>±0.0004</td>
<td>dB/mV</td>
</tr>
<tr>
<td></td>
<td></td>
<td>±0.0004</td>
<td>±0.0008</td>
<td>dB/mV</td>
</tr>
<tr>
<td>&amp;GT</td>
<td>&amp;G Variation with Temperature Half Channel</td>
<td>±0.001</td>
<td>±0.002</td>
<td>dB/C</td>
</tr>
<tr>
<td></td>
<td></td>
<td>±0.002</td>
<td>±0.005</td>
<td>dB/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>HD</td>
<td>Harmonic Distortion (2nd or 3rd)</td>
<td>-48</td>
<td>-44</td>
<td>dB</td>
</tr>
<tr>
<td>IMD1</td>
<td>Intermodulation Distortion G.712(7.1)</td>
<td>-45</td>
<td>dB</td>
<td>CCITT G.712</td>
</tr>
<tr>
<td>IMD2</td>
<td>G.712(7.2)</td>
<td>-50</td>
<td>dBm0</td>
<td>Two Tone Method</td>
</tr>
</tbody>
</table>
A.C. CHARACTERISTICS

$T_A = 0°C \text{ to } +70°C$, $V_{DD} = +12V \pm 5\%$, $V_{CC} = +5V \pm 5\%$, $V_{BB} = -5V \pm 5\%$, $GRDA = 0V$, $GRDD = 0V$, 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>DmW</td>
<td>Digital Milliwatt Response</td>
<td>5.58</td>
<td>5.66</td>
<td>5.78</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></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></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DmWS</td>
<td>DmWO Variation with Supplies</td>
<td>±0.07</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>AIR</td>
<td>Input Dynamic Range</td>
<td>2.183</td>
<td>2.213</td>
<td>2.243</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>AIRT</td>
<td>Input Dynamic Range vs Temperature</td>
<td>-0.5</td>
<td>mVrms/°C</td>
<td></td>
</tr>
<tr>
<td>AIRS</td>
<td>Input Dynamic Range vs Supplies</td>
<td>±18</td>
<td>mVrms</td>
<td></td>
</tr>
<tr>
<td>AOR</td>
<td>Output Dynamic Range, $V_{FR}$</td>
<td>2.14</td>
<td>2.17</td>
<td>2.20</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>AORT</td>
<td>$A_{OR}$ Variation with Temperature</td>
<td>-0.5</td>
<td>mVrms/°C</td>
<td></td>
</tr>
<tr>
<td>AORS</td>
<td>$A_{OR}$ Variation with Supplies</td>
<td>±18</td>
<td>mVrms</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></td>
<td></td>
<td></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></td>
<td></td>
<td></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></td>
<td></td>
<td></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></td>
<td></td>
<td></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></td>
<td></td>
<td></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></td>
<td></td>
<td></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></td>
<td></td>
<td></td>
<td></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></td>
<td></td>
<td></td>
<td></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 $T_A = 25°C$ and nominal power supply values.
2. If the external auto zero is used $N_{0C}$ has a typical value of $-76$ dBm0.
3. Tested and guaranteed at 23°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 $V_{FX}$ 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. $V_{FX}$ of D.U.T encoder = 1.02 kHz, 0 dBm0. Decoder under quiet channel conditions; measurements made at decoder output.
9. $V_{FX} = 0$ Vrms. Decoder = 1.02 kHz, 0 dBm0. Encoder under quiet channel conditions; measurement made at encoder output.

5-30
Figure 13. Tracking Deviation ($\Delta G$) (Half Channel)

Figure 14. Signal to Total Distortion Ratio (Half Channel)
A.C. CHARACTERISTICS—TIMING SPECIFICATION

$T_A = 0^\circ C \text{ to } +70^\circ C$, $V_{DD} = +12V \pm 5\%$, $V_{CC} = +5V \pm 5\%$, $V_{BB} = -5V \pm 5\%$, $GRDA = 0V$, $GRDD = 0V$, unless otherwise specified.

### CLOCK SECTION

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Parameter</th>
<th>Limits</th>
<th>Unit</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>$t_{CY}$</td>
<td>Clock Period</td>
<td>485</td>
<td>ns</td>
<td>CLKX, CLKR (2.048 MHz systems), CLKC</td>
</tr>
<tr>
<td>$t_r, t_f$</td>
<td>Clock Rise and Fall Time</td>
<td>0</td>
<td>30</td>
<td>ns</td>
</tr>
<tr>
<td>$t_{CLK}$</td>
<td>Clock Pulse Width</td>
<td>215</td>
<td>ns</td>
<td>CLKX, CLKR, CLKC</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>$t_{VFX}$</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</td>
<td>180</td>
<td>ns</td>
</tr>
<tr>
<td>$t_{DHX}$</td>
<td>Data Hold Time</td>
<td>80</td>
<td>230</td>
<td>ns</td>
</tr>
<tr>
<td>$t_{HZX}$</td>
<td>Data Float on TS Exit</td>
<td>75</td>
<td>245</td>
<td>ns</td>
</tr>
<tr>
<td>$t_{SON}$</td>
<td>Timeslot X to Enable</td>
<td>30</td>
<td>185</td>
<td>ns</td>
</tr>
<tr>
<td>$t_{SOFF}$</td>
<td>Timeslot X to Disable</td>
<td>70</td>
<td>225</td>
<td>ns</td>
</tr>
<tr>
<td>$t_{FSO}$</td>
<td>Frame Sync Delay</td>
<td>15</td>
<td>150</td>
<td>ns</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>$t_{VFR}$</td>
<td>Analog Output Update</td>
<td>$9\frac{1}{16}$</td>
<td>$9\frac{1}{16}$</td>
<td>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>$t_{FSD}$</td>
<td>Frame Sync Delay</td>
<td>15</td>
<td>150</td>
<td>ns</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>$t_{DHC}$</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 $t_{HZX}$ and $t_{SOFF}$, 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(1)

TRANSMIT TIMING

RECEIVE TIMING

CONTROL TIMING

NOTE:
1. All timing parameters referenced to 1.5V, except \( t_{HZX} \) and \( t_{SOFF} \) which reference a high impedance state.
2912A
PCM TRANSMIT/RECEIVE FILTER

- Low Power Consumption:
  - 60 mW Typical without Power Amplifiers
  - 80 mW Typical with Power Amplifiers
  - 0.5 mW Typical Standby
- Low Idle Channel Noise:
  - 2 dB BrnC0 Typical, Receive
  - 6 dB BrnC0 Typical, Transmit
- Excellent Power Supply Rejection:
  - 40 dB Typical on VCC @ 50 KHz
  - 30 dB Typical on VBB @ 50 KHz
- Transmit Filter Rejects Low Frequency Noise:
  - 23 dB @ 60 Hz
  - 25 dB @ 50 Hz
  - 50 dB @ 16-2/3 Hz
- Adjustable Gain in Both Directions
- Fully Compatible with the Industry Standard Intel 2912
- D3/D4 and CCITT G712 Compatible
- Common Mode Op Amp Input Rejection 75 dB Typical
- Direct Interface to the Intel 2910A/2911A PCM Codecs Including Stand-By Power Down Mode
- Direct Interface with Transformer or Electronic Hybrids
- Fabricated with Reliable N-Channel MOS Process

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* 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, 2911A, 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.

---

**Pin Names**

<table>
<thead>
<tr>
<th>Pin Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>VFxI+, VFxI-</td>
<td>Analog Inputs</td>
</tr>
<tr>
<td>GSx</td>
<td>Gain Control</td>
</tr>
<tr>
<td>VFxO</td>
<td>Analog Output</td>
</tr>
<tr>
<td>VFai</td>
<td>Analog Input</td>
</tr>
<tr>
<td>VFao</td>
<td>Analog Output</td>
</tr>
<tr>
<td>PWRI</td>
<td>Driver Input</td>
</tr>
<tr>
<td>PWRO+, PWRO-</td>
<td>Driver Output</td>
</tr>
<tr>
<td>CLK</td>
<td>Clock Input</td>
</tr>
<tr>
<td>CLK0</td>
<td>Clock Selection</td>
</tr>
<tr>
<td>PDN</td>
<td>Power Down</td>
</tr>
<tr>
<td>VCC</td>
<td>Power (+5V)</td>
</tr>
<tr>
<td>VBB</td>
<td>Power (-5V)</td>
</tr>
<tr>
<td>GRDD</td>
<td>Digital Ground</td>
</tr>
<tr>
<td>GRDA</td>
<td>Analog Ground</td>
</tr>
</tbody>
</table>

---

**Figure 1. Block Diagram**

**Figure 2. Pin Configuration**

*AT&T is a registered trademark of American Telephone and Telegraph Corporation.
## 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>VFxl+</td>
<td>1</td>
<td>Input</td>
<td>Analog input of the transmit filter. The VFxl+ 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>VFxl−</td>
<td>2</td>
<td>Input</td>
<td>Inverting input of the gain adjustment operational amplifier on the transmit filter.</td>
</tr>
<tr>
<td>GSx</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>VFRO</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, VFRO is tied to PRWI and a dual balanced output is provided on pins PWRO+ and PWRO−.</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 VBB, 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−</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>VBB</td>
<td>8</td>
<td>Power</td>
<td>−5V ± 5% referenced to GRDA</td>
</tr>
<tr>
<td>VCC</td>
<td>9</td>
<td>Power</td>
<td>+5V ± 5% referenced to GRDA</td>
</tr>
<tr>
<td>VFRI</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 Sinx 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>CLKO(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, CLKO, 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>CLKO(1)</td>
<td>14</td>
<td>Input</td>
<td>Clock (pin 12, CLK) frequency selection. If tied to VBB, CLK should be 1.536 MHz. If tied to Ground, CLK should be 1.544 MHz. If tied to VCC, 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>VFxO</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>CLKO, Pin 14</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.536 MHz</td>
<td>192</td>
<td>1.536 MHz</td>
<td>VBB (−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>VCC (+5V)</td>
</tr>
</tbody>
</table>

5-35
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 an-}

alog 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 VFy± 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.

Figure 3. Typical Line Termination

Figure 4. Transmit Filter Gain Adjustment
Receive Filter Output

The VFRO lead is capable of driving high impedance electronic hybrids. The gain of the receive section from VFRI to VFRO is:

$$\frac{\pi f}{8000} \times \sin \left(\frac{\pi f}{8000}\right)$$

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 RLR on VFRO should not be less than 10K Ω.

![Figure 5. Receive Filter Output Gain Adjustment](270159-5)

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 VFRO is connected through gain setting resistors R1 and R2 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+ and PWRO−, the maximum voltage swing across the load is ±5.0 volts. The series combination of RS 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 VBB 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](270159-6)
**ABSOLUTE MAXIMUM RATINGS**

Temperature Under Bias ............ -10°C to +80°C
Storage Temperature ............... -65°C to +150°C
Supply Voltage with Respect to VBB ............... -0.3V to +14.0V
All Input and Output Voltages with Respect to VBB ............... -0.3V to +14.0V
All Output Currents .................. ± 50 mA
Power Dissipation .................... 1 Watt

**NOTICE:** This is a production data sheet. The specifications are subject to change without notice.

*WARNING: Stressing the device beyond the "Absolute Maximum Ratings" may cause permanent damage. These are stress ratings only. Operation beyond the "Operating Conditions" is not recommended and extended exposure beyond the "Operating Conditions" may affect device reliability.*

**D.C. CHARACTERISTICS**

\[ T_A = 0°C \text{ to } +70°C; \ V_{CC} = 5V \pm 5%; \ V_{BB} = -5V \pm 5%; \ GRDA = 0V; \]
GRDD = 0V; unless otherwise specified

**DIGITAL INTERFACE (CLK, CLK0, and PDN Pins)**

<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_{ILC}</td>
<td>Input Load Current, CLK</td>
<td>10</td>
<td>µA</td>
<td></td>
<td></td>
<td>VIN = GRDD to VCC</td>
</tr>
<tr>
<td>I_{ILO}</td>
<td>Input Load Current, CLK0</td>
<td>10</td>
<td>µA</td>
<td></td>
<td></td>
<td>VIN = VBB to VCC</td>
</tr>
<tr>
<td>I_{ILP}</td>
<td>Input Load Current, PDN</td>
<td>100</td>
<td>µA</td>
<td></td>
<td></td>
<td>VIN = GRDD to VCC</td>
</tr>
<tr>
<td>V_{IL}</td>
<td>Input Low Voltage (except CLK0)</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 CLK0)</td>
<td>2.0</td>
<td>V</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>V_{IL0}</td>
<td>Input Low Voltage, CLK0</td>
<td>V_{BB}</td>
<td>V_{BB}+0.5</td>
<td>V</td>
<td></td>
<td></td>
</tr>
<tr>
<td>V_{II0}</td>
<td>Input Intermediate Voltage, CLK0</td>
<td>V_{GRDD}−0.5</td>
<td>0.8</td>
<td>V</td>
<td></td>
<td></td>
</tr>
<tr>
<td>V_{IH0}</td>
<td>Input High Voltage, CLK0</td>
<td>V_{CC}−0.5</td>
<td>V</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**POWER DISSIPATION**

<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_{CC0}</td>
<td>V_{CC} Standby Current</td>
<td>50</td>
<td>100</td>
<td>µA</td>
<td></td>
<td>PDN = V_{IH} Min</td>
</tr>
<tr>
<td>I_{BB0}</td>
<td>V_{BB} Standby Current</td>
<td>50</td>
<td>100</td>
<td>µA</td>
<td></td>
<td>PDN = V_{IH} Min</td>
</tr>
<tr>
<td>I_{CC1}</td>
<td>V_{CC} Operating Current, Power Amplifiers Inactive</td>
<td>6</td>
<td>10</td>
<td>mA</td>
<td></td>
<td>PWRI = V_{BB}</td>
</tr>
<tr>
<td>I_{BB1}</td>
<td>V_{BB} Operating Current, Power Amplifiers Inactive</td>
<td>6</td>
<td>10</td>
<td>mA</td>
<td></td>
<td>PWRI = V_{BB}</td>
</tr>
<tr>
<td>I_{CC2}</td>
<td>V_{CC} Operating Current</td>
<td>8</td>
<td>14</td>
<td>mA</td>
<td></td>
<td></td>
</tr>
<tr>
<td>I_{BB2}</td>
<td>V_{BB} Operating Current</td>
<td>8</td>
<td>14</td>
<td>mA</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**NOTES:**
1. Typical values are for \[ T_A = 25°C \] and nominal power supply values.
2. To place the power amplifiers in the inactive mode PWRI must be tied to V_{BB} prior to power-up.
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_{BXI} )</td>
<td>Input Leakage Current, ( VF_X^+ ), ( VF_X^- )</td>
<td>100</td>
<td>nA</td>
<td>25</td>
<td>mV</td>
<td>(-2.2V &lt; V_{IN} &lt; 2.2V)</td>
</tr>
<tr>
<td>( R_{IXI} )</td>
<td>Input Resistance, ( VF_X^+ ), ( VF_X^- )</td>
<td>10</td>
<td>M(\Omega)</td>
<td>60</td>
<td>dB</td>
<td>(-2.2V &lt; V_{IN} &lt; 2.2V), 0 dBm0 = 1.1 ( V_{RMS} ), Input at ( VF_X^- )</td>
</tr>
<tr>
<td>( V_{OSXI} )</td>
<td>Input Offset Voltage, ( VF_X^+ ), ( VF_X^- )</td>
<td>25</td>
<td>mV</td>
<td>75</td>
<td>V</td>
<td>( R_L \geq 10 \Omega )</td>
</tr>
<tr>
<td>CMRR</td>
<td>Common Mode Rejection, ( VF_X^+ ), ( VF_X^- )</td>
<td>60</td>
<td>75</td>
<td>dB</td>
<td>V</td>
<td>( R_L \geq 10 \Omega )</td>
</tr>
<tr>
<td>( A_{VOL} )</td>
<td>DC Open Loop Voltage Gain, ( GS_X )</td>
<td>3000</td>
<td></td>
<td></td>
<td>V</td>
<td>( R_L \geq 10 \Omega )</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_X^O )</td>
<td>20</td>
<td>35</td>
<td>( \Omega )</td>
<td>( VF_X^+ ) Connected to GRDA, Input Op Amp at Unity Gain</td>
<td></td>
</tr>
<tr>
<td>( V_{OSX} )</td>
<td>Output DC Offset, ( VF_X^O )</td>
<td>100</td>
<td>mV</td>
<td>100</td>
<td>V</td>
<td>( VF_X^+ ) Connected to GRDA, Input Op Amp at Unity Gain</td>
</tr>
<tr>
<td>PSRR(_1)</td>
<td>Power Supply Rejection of ( V_{CC} ) at 1 KHz, ( VF_X^O )</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_X^O )</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_X^O )</td>
<td>25</td>
<td>pF</td>
<td>25</td>
<td>pF</td>
<td>Minimum ( R_L )</td>
</tr>
<tr>
<td>( R_{LX} )</td>
<td>Minimum Load Resistance, ( VF_X^O )</td>
<td>2.7</td>
<td>( \Omega )</td>
<td>2.7</td>
<td>( \Omega )</td>
<td>Minimum ( R_L )</td>
</tr>
<tr>
<td>( V_{OX1} )</td>
<td>Output Voltage Swing, 1 KHz, ( VF_X^O )</td>
<td>( \pm 3.2 )</td>
<td>V</td>
<td>( R_L \geq 10 \Omega ) or with 2910A or 2911A</td>
<td></td>
<td></td>
</tr>
<tr>
<td>( V_{OX2} )</td>
<td>Output Voltage Swing, 1 KHz, ( VF_X^O )</td>
<td>( \pm 2.5 )</td>
<td>V</td>
<td>( R_L \geq 2.7 \Omega )</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**NOTES:**
1. Typical values for \( T_A = 25^\circ C \) and nominal power supply values.
2. PSRR\(_1,2\) include op amp in transmit section.
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 Description</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, ( \text{VF}_{RI} )</td>
<td>3</td>
<td>( \mu\text{A} )</td>
<td>-3.2V &lt; ( V_{IN} ) &lt; 3.2V</td>
<td></td>
<td></td>
</tr>
<tr>
<td>( R_{IR} )</td>
<td>Input Resistance, ( \text{VF}_{RI} )</td>
<td>1</td>
<td>M( \Omega )</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( R_{OR} )</td>
<td>Output Resistance, ( \text{VF}_{RO} )</td>
<td>100</td>
<td>( \Omega )</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( V_{OSR} )</td>
<td>Output DC Offset ( \text{VF}_{RO} )</td>
<td>100</td>
<td>mV</td>
<td>( \text{VF}_{RI} ) Connected to GRDA</td>
<td></td>
<td></td>
</tr>
<tr>
<td>( PSRR_{3} )</td>
<td>Power Supply Rejection of ( V_{CC} ) at 1 KHz, ( \text{VF}_{RO} )</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, ( \text{VF}_{RO} )</td>
<td>30</td>
<td>35</td>
<td>dB</td>
<td></td>
<td></td>
</tr>
<tr>
<td>( C_{LR} )</td>
<td>Load Capacitance, ( \text{VF}_{RO} )</td>
<td>25</td>
<td>pF</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( R_{LR} )</td>
<td>Minimum Load Resistance, ( \text{VF}_{RO} )</td>
<td>10</td>
<td>K( \Omega )</td>
<td>Minimum ( R_L )</td>
<td></td>
<td></td>
</tr>
<tr>
<td>( V_{OR} )</td>
<td>Output Voltage Swing, ( \text{VF}_{RO} )</td>
<td>±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 Description</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, ( \text{PWRI} )</td>
<td>3</td>
<td>( \mu\text{A} )</td>
<td>-3.2V &lt; ( V_{IN} ) &lt; 3.2V</td>
<td></td>
<td></td>
</tr>
<tr>
<td>( R_{IRA} )</td>
<td>Input Resistance, ( \text{PWRI} )</td>
<td>10</td>
<td>M( \Omega )</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( R_{ORA} )</td>
<td>Output Resistance, ( \text{PWRO}+ ), ( \text{PWRO}− )</td>
<td>1</td>
<td>( \Omega )</td>
<td></td>
<td>(</td>
<td>I_{OUT}</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>(-3.0V &lt; V_{OUT} &lt; 3.0V )</td>
<td></td>
<td></td>
</tr>
<tr>
<td>( V_{OSRA} )</td>
<td>Output DC Offset, ( \text{PWRO}+ ), ( \text{PWRO}− )</td>
<td>50</td>
<td>mV</td>
<td>( \text{PWRI} ) Connected to GRDA</td>
<td></td>
<td></td>
</tr>
<tr>
<td>( C_{LRA} )</td>
<td>Load Capacitance, ( \text{PWRO}+ ), ( \text{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 ), ( \text{PWRO}+ ), ( \text{PWRO}− ) Single Ended Connection</td>
<td>±3.2</td>
<td>V</td>
<td>( R_L = 10 \text{K}\Omega )</td>
<td>( R_L ) Connected to GRDA to ( f_0 \geq 200 \text{Hz} )</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>( R_L = 600\Omega )</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>( R_L = 300\Omega )</td>
<td></td>
<td></td>
</tr>
<tr>
<td>( V_{ORA2} )</td>
<td>Differential Output Voltage Swing, ( \text{PWRO}+ ), ( \text{PWRO}− ) Balanced Output Connection</td>
<td>±6.4</td>
<td>V</td>
<td>( R_L = 20 \text{K}\Omega )</td>
<td>( R_L ) Connected between ( \text{PWRO}+ ) and ( \text{PWRO}− ) to ( f_0 \geq 200 \text{Hz} )</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>( R_L = 1200\Omega )</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>( R_L = 600\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  
\( T_A = 0^\circ C \) to \(+ 70^\circ C\); \( V_{CC} = 5V \pm 5\%; V_{BB} = -5V \pm 5\%; GRDA = 0V; \)  
GRDD = 0V; unless otherwise specified

Clock Input Frequency:  
CLK = 1.536 MHz ± 0.1\%; CLK0 = \( V_{IL0} \) (Tied to \( V_{BB} \))  
CLK = 2.048 MHz ± 0.1\%; CLK0 = \( V_{IH0} \) (Tied to \( V_{CC} \))  
CLK = 1.544 MHz ± 0.1\%; CLK0 = \( V_{II0} \) (Tied to GRDD)

TRANSMIT FILTER TRANSFER CHARACTERISTICS  
(See Transmit Filter Transmit 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></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( 16.67 \text{ Hz} )</td>
<td></td>
<td>(-56)</td>
<td>(-50)</td>
<td>dB</td>
<td>0 dBm0 Input Signal</td>
<td></td>
</tr>
<tr>
<td>( 50 \text{ Hz} )</td>
<td></td>
<td>(-25)</td>
<td>dB</td>
<td>Unity Gain</td>
<td></td>
<td></td>
</tr>
<tr>
<td>( 60 \text{ Hz} )</td>
<td></td>
<td>(-23)</td>
<td>dB</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( 200 \text{ Hz} )</td>
<td></td>
<td>(-1.8)</td>
<td>(-0.125)</td>
<td>dB</td>
<td>0 dBm0 Signal ( \equiv 1.1 \text{ V}_{\text{RMS}} )</td>
<td></td>
</tr>
<tr>
<td>( 300 \text{ Hz to 3000 Hz} )</td>
<td></td>
<td>(-0.125)</td>
<td>(0.125)</td>
<td>dB</td>
<td>Input at ( V_{FXl}^+ )</td>
<td></td>
</tr>
<tr>
<td>( 3300 \text{ Hz} )</td>
<td></td>
<td>(-0.35)</td>
<td>(0.03)</td>
<td>dB</td>
<td></td>
<td></td>
</tr>
<tr>
<td>( 3400 \text{ Hz} )</td>
<td></td>
<td>(-0.7)</td>
<td>(-0.1)</td>
<td>dB</td>
<td>0 dBm0 Signal ( \equiv 1.6 \text{ V}_{\text{RMS}} )</td>
<td></td>
</tr>
<tr>
<td>( 4000 \text{ Hz} )</td>
<td></td>
<td>(-14)</td>
<td>dB</td>
<td>Output at ( V_{FXO} )</td>
<td></td>
<td></td>
</tr>
<tr>
<td>( 4600 \text{ Hz and Above} )</td>
<td></td>
<td>(-32)</td>
<td>dB</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( G_{AX} )</td>
<td>Absolute Passband Gain at 1 KHz, ( V_{FXO} )</td>
<td>2.9</td>
<td>3.0</td>
<td>3.1</td>
<td>dB</td>
<td>( R_L = \infty (3) )</td>
</tr>
<tr>
<td>( G_{AXT} )</td>
<td>Gain Variation with Temperature at 1 KHz</td>
<td>0.0002</td>
<td>0.002</td>
<td>dB/°C</td>
<td>0 dBm0 Signal Level</td>
<td></td>
</tr>
<tr>
<td>( G_{AXS} )</td>
<td>Gain Variation with Supplies at 1 KHz</td>
<td>0.01</td>
<td>0.07</td>
<td>dB/V</td>
<td>0 dBm0 Signal Level, Supplies ±5%</td>
<td></td>
</tr>
<tr>
<td>( C_{RT} )</td>
<td>Cross Talk, Receive to Transmit, Measured at ( V_{FXO} )</td>
<td>(-75)</td>
<td>(-65)</td>
<td>dB</td>
<td>( V_{FR1} = 1.6 \text{ V}_{\text{RMS}}, 1 \text{ KHz} )</td>
<td></td>
</tr>
<tr>
<td>Connected through ( G_{SX} ), ( G_{SX} ) Connected through ( 10 \text{ K} \Omega ) to ( GRDA )</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, ( V_{FXO} )</td>
<td>6</td>
<td>11</td>
<td>dBm0</td>
<td>Gain Setting Op Amp at Unity Gain (Note 2)</td>
<td></td>
</tr>
<tr>
<td>( N_{CX2} )</td>
<td>Total C Message Noise at Output, ( V_{FXO} )</td>
<td>9</td>
<td>13</td>
<td>dBm0</td>
<td>Gain Setting Op Amp at 20 dB Gain (Note 2)</td>
<td></td>
</tr>
<tr>
<td>( D_{DX} )</td>
<td>Differential Envelope Delay, ( V_{FXO} ) 1 KHz to 2.6 KHz</td>
<td>60</td>
<td>( \mu\text{s} )</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( D_{AX} )</td>
<td>Absolute Delay at 1 KHz, ( V_{FXO} )</td>
<td>110</td>
<td>( \mu\text{s} )</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( D_{PX1} )</td>
<td>Single Frequency Distortion Products</td>
<td>(-48)</td>
<td>dB</td>
<td>0 dBm0 Input Signal at 1 KHz</td>
<td></td>
<td></td>
</tr>
<tr>
<td>( D_{PX2} )</td>
<td>Single Frequency Distortion Products at Maximum Signal Level of +3 dBm0 at ( V_{FXO} )</td>
<td>(-45)</td>
<td>dB</td>
<td>0.16 \text{ V}<em>{\text{RMS}} 1 \text{ KHz} Input Signal at ( V</em>{FXl}^+ ); Gain Setting Op Amp at 20 dB Gain. The +3 dBm0 Signal at ( V_{FXO} ) is 2.26 \text{ V}_{\text{RMS}}</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

---

(1) Typ = Typical  
(2) Note 2  
(3) \( R_L = \infty \)
A.C. CHARACTERISTICS  \( T_A = 0^\circ C \text{ to } +70^\circ C; V_{CC} = 5V \pm 5\%; V_{BB} = -5V \pm 5\%; \text{GRDA} = 0V; \text{GRDD} = 0V; \) unless otherwise specified (Continued)

Clock Input Frequency: \( \text{CLK} = 1.536 \text{ MHz} \pm 0.1\%; \text{CLK0} = \text{V}_{\text{ILD}} \) (Tied to \( V_{\text{BB}} \))
\( \text{CLK} = 1.544 \text{ MHz} \pm 0.1\%; \text{CLK0} = \text{V}_{\text{IIQ}} \) (Tied to \( \text{GRDD} \))
\( \text{CLK} = 2.048 \text{ MHz} \pm 0.1\%; \text{CLK0} = \text{V}_{\text{IHQ}} \) (Tied to \( V_{\text{CC}} \))

RECEIVE FILTER TRANSFER CHARACTERISTICS (See Receive Filter Transfer Characteristics, Figure 8)

<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_{RR} )</td>
<td>Gain Relative to Gain at 1 KHz with Sinx/x Correction of 2910A or 2911A</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Below 200 Hz</td>
<td></td>
<td>0.125 dB</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 dB</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4600 Hz and Above</td>
<td>-30 dB</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( G_{AR} )</td>
<td>Absolute Passband Gain at 1 KHz, ( V_{\text{FRO}} )</td>
<td>-0.1 dB</td>
<td>0 dB</td>
<td></td>
<td>dB</td>
<td>( R_L = \infty ) (3, 4)</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></td>
<td></td>
<td>0 dBm0 Signal Level</td>
</tr>
<tr>
<td>( G_{ARS} )</td>
<td>Gain Variation with Supplies at 1 KHz</td>
<td>0.01 dB</td>
<td>0.07 dB</td>
<td></td>
<td></td>
<td>0 dBm0 Signal Level, Supplies ± 5%</td>
</tr>
<tr>
<td>( CT_{TR} )</td>
<td>Cross Talk, Transmit to Receive, Measured at ( V_{\text{FRO}} ); 20 log (( V_{\text{FRO}}/V_{\text{FRO}} ))</td>
<td>-70 dB</td>
<td>-60 dB</td>
<td></td>
<td></td>
<td>( V_{\text{FRI}} = 1.1 V_{\text{RMS}}, 1 ) KHz Output, ( V_{\text{FRI}} ) Connected to ( \text{GRDA} )</td>
</tr>
<tr>
<td>( N_{CR} )</td>
<td>Total C Message Noise at Output, ( V_{\text{FRO}} )</td>
<td>2 dBm0</td>
<td>2 dBm0 (Note 2)</td>
<td></td>
<td></td>
<td>( V_{\text{FRO}} ) Output or ( \text{PWRO} + ) and ( \text{PWRO} - ) Connected with Unity Gain</td>
</tr>
<tr>
<td>( D_{DR} )</td>
<td>Differential Envelope Delay, ( V_{\text{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_{\text{FRO}} )</td>
<td>110 μs</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( DPR_{R1} )</td>
<td>Single Frequency Distortion Products</td>
<td>-48 dB</td>
<td></td>
<td></td>
<td></td>
<td>0 dBm0 Input Signal at 1 KHz</td>
</tr>
<tr>
<td>( DPR_{R2} )</td>
<td>Single Frequency Distortion Products at Maximum Signal Level of +3 dBm0 at ( V_{\text{FRO}} )</td>
<td>-45 dB</td>
<td></td>
<td></td>
<td></td>
<td>+3 dBm0 Signal Level of 2.26 ( V_{\text{RMS}}, 1 ) KHz Input at ( V_{\text{FRI}} )</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.
TRANSFER CHARACTERISTICS

Figure 7. Transmit Filter
**NOTES:**
1. Typical Transfer Function of the Receive Filter as a Separate Component.
POWER SUPPLY REJECTION TYPICAL VALUES OVER 3 RANGES

Figure 9. Transmit Filter

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
- Digital Echo Cancellers
- Secure Communications Systems
- Satellite Earth Stations

![Pin Configurations](image)

Figure 1. Pin Configurations
TRANSMIT SECTION

FILTER

SAMPLE AND HOLD AND DAC

COMPARATOR

SUCCESSIVE APPROXIMATION REGISTER

OUTPUT REGISTER

AUTO ZERO

ANALOG TO DIGITAL CONTROL LOGIC

REFERENCE

VFX1+

VFX1-

GSX

RECEIVE SECTION

FILTER

BUFFER

SAMPLE AND HOLD AND DAC

DIGITAL TO ANALOG CONTROL LOGIC

INPUT REGISTER

CONTROL LOGIC

CONTROL SECTIOn

GAIN SET

REFERENCE

GSr

GAIN SET

REFERENCE

PWRO+

PWRO-

VCC

VSS

GRDD

GRDA

FSr

CLKr

SIGr

DCLKr

SIGNAL/ASEL

CLKX

SIGX

DXX/DCLKX

210629-4

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+, PWRO−</td>
<td>Power Amplifier Outputs</td>
<td>VFx−, VFx+</td>
<td>Analog Inputs</td>
</tr>
<tr>
<td>GSr</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>SRG</td>
<td>Receive Signaling Output</td>
<td>TX</td>
<td>Timeslot Strobe/Buffer Enable</td>
</tr>
<tr>
<td>DCLKR</td>
<td>Receive Variable Data Clock</td>
<td>DCLKx</td>
<td>Transmit Variable Data Clock</td>
</tr>
<tr>
<td>DR</td>
<td>Receive PCM Input</td>
<td>DX</td>
<td>Transmit PCM Output</td>
</tr>
<tr>
<td>FSR</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>GSR</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 GSR.</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>
</tbody>
</table>
| CLKSEL | Input which must be pinline to reflect the master clock frequency at CLKx, CLKR. CLKSEL = VBB ....................... 2.048 MHz  
CLKSEL = GRDD .................... 1.544 MHz  
CLKSEL = VCC ....................... 1.536 MHz |
| LOOP   | Analog loopback. When this pin is TTL high, the analog output (PWRO+) is internally connected to the analog input (VFx+), GSR is internally connected to PWRO−, and VFx− is internally connected to GSx. A 0 dBm0 digital signal input at D= is returned as a + 3 dBm0 digital signal output at DX. |
| SIGR   | Signaling bit output, receive channel. In fixed data rate mode, SIGR outputs the logical state of the eighth bit of the PCM word in the most recent signaling frame. |
| DCLKR  | Selects the fixed or variable data rate mode. When DCLKR is connected to VBB, the fixed data rate mode is selected. When DCLKR is not connected to VBB, the device operates in the variable data rate mode. In this mode DCLKR becomes the receive data clock which operates at TTL levels from 64 Kb to 2.048 Mb data rates. |
Table 2. Pin Description (Continued)

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<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; CLKR in the fixed data rate mode and DCLKR 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>D_X</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 Kb to 2.048 Mb 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 D_X 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>VFX_1+</td>
<td>Non-inverting analog input to uncommitted transmit operational amplifier.</td>
</tr>
<tr>
<td>VFX_1-</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.

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

SWITCHING

CHANNEL BANKS

Figure 3. Typical Line Terminations
GENERAL OPERATION

System Reliability Features

The combochip can be powered up by pulsing \( FS_X \) and/or \( FS_R \) 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 \( VB_B \) or \( V_CC \) 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 \( D_X \) and \( TS_X \) are held in a high impedance state for approximately four frames (500 \( \mu \)s) after power up or application of \( VB_B \) or \( V_CC \). After this delay, \( D_X \), \( TS_X \), 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 \( SIG_R \) is also held low for a maximum of four frames after power up or application of \( VB_B \) or \( V_CC \). \( SIG_R \) will remain low thereafter until it is updated by a signaling frame.

To further enhance system reliability, \( TS_X \) and \( D_X \) will be placed in a high impedance state approximately 30 \( \mu \)s after an interruption of \( CLK_X \). Similarly, \( SIG_R \) will be held low approximately 30 \( \mu \)s after an interruption of \( CLK_R \). 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 \( FS_X \) and/or \( FS_R \). With both channels in the standby state, power consumption is reduced to the value shown in Table 3. If transmit only operation is desired, \( FS_X \) should be applied to the device while \( FS_R \) is held low. Similarly, if receive only operation is desired, \( FS_R \) should be applied while \( FS_X \) is held low.

Fixed Data Rate Mode

Fixed data rate timing, which is 2910A and 2911A compatible, is selected by connecting \( DCLK_R \) to \( VB_B \). It employs master clocks \( CLK_X \) and \( CLK_R \), frame synchronization clocks \( FS_X \) and \( FS_R \), and output \( TS_X \).

<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 )</td>
<td>5 mW</td>
<td>( TS_X ) and ( D_X ) are placed in a high impedance state and ( SIG_R ) is placed in a TTL low state within 10 ( \mu )s.</td>
</tr>
<tr>
<td>Standby Mode</td>
<td>( FS_X ) and ( FS_R ) are TTL Low</td>
<td>12 mW</td>
<td>( TS_X ) and ( D_X ) are placed in a high impedance state and ( SIG_R ) is placed in a TTL low state 300 milliseconds after ( FS_X ) and ( FS_R ) are removed.</td>
</tr>
<tr>
<td>Only Transmit Is on Standby</td>
<td>( FS_X ) is TTL Low</td>
<td>70 mW</td>
<td>( TS_X ) and ( D_X ) are placed in a high impedance state within 300 milliseconds.</td>
</tr>
<tr>
<td>Only Receive Is on Standby</td>
<td>( FS_R ) is TTL Low</td>
<td>110 mW</td>
<td>( SIG_R ) is placed in a TTL low state within 300 milliseconds.</td>
</tr>
</tbody>
</table>
CLK\textsubscript{X} and CLK\textsubscript{R} 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\textsubscript{X} and FS\textsubscript{R} 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\textsubscript{X} 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 D\textsubscript{X} on the first eight positive transitions of CLK\textsubscript{X} following the rising edge of FS\textsubscript{X}. Similarly, on the receive side, data is received on the first eight falling edges of CLK\textsubscript{R}. The frequency of CLK\textsubscript{X} and CLK\textsubscript{R} 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 DCLK\textsubscript{R} to the bit clock for the receive PCM highway rather than to V\textsubscript{BB}. It employs master clocks CLK\textsubscript{X} and CLK\textsubscript{R}, bit clocks DCLK\textsubscript{R} and DCLK\textsubscript{X}, and frame synchronization clocks FS\textsubscript{R} and FS\textsubscript{X}.

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\textsubscript{R} and DCLK\textsubscript{X} become the data clocks for the receive and transmit PCM highways. While FS\textsubscript{X} is high, PCM data from D\textsubscript{X} is transmitted onto the highway on the next eight consecutive positive transitions of DCLK\textsubscript{X}. Similarly, while FS\textsubscript{R} is high, each PCM bit from the highway is received by D\textsubscript{R} on the next eight consecutive negative transitions of DCLK\textsubscript{R}.

On the transmit side, the PCM word will be repeated in all remaining timeslots in the 125 \mu s frame as long as DCLK\textsubscript{X} is pulsed and FS\textsubscript{X} 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 (DCLK\textsubscript{R} = V\textsubscript{BB}). 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\textsubscript{X} 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\textsubscript{R} 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)](210629-7)
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+), GSR 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 combochips 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 companded 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.
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 VFx+ 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 D3/D4 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 D acl 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 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>OTLP1X</td>
<td>Zero Transmission Level Point</td>
<td>+ 276</td>
<td>dBm</td>
<td>Referenced to 600Ω</td>
</tr>
<tr>
<td></td>
<td>Transmit Channel (0 dBm0)</td>
<td>+ 1.00</td>
<td>dBm</td>
<td>Referenced to 900Ω</td>
</tr>
<tr>
<td>OTLP2X</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)</td>
<td>+ 1.03</td>
<td>dBm</td>
<td>Referenced to 900Ω</td>
</tr>
<tr>
<td>OTLP1R</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)</td>
<td>+ 4.00</td>
<td>dBm</td>
<td>Referenced to 900Ω</td>
</tr>
<tr>
<td>OTLP2R</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)</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 DR 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:

\[ V_{O+} \text{ at PWRO+} \]
\[ V_{O-} \text{ at PWRO-} \]
\[ V = (V_{O+}) - (V_{O-}) \text{(total differential response)} \]

R1 and R2 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 in-accuracies.

A is the gain of the power amplifiers, where

\[ A = \frac{1 + \left( \frac{R_1}{R_2} \right)}{4 + \left( \frac{R_1}{R_2} \right)} \]

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

\[ R_1/R_2 = \infty \text{ or } V(GSR) = V_{O-}; \]

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

\[ R_1/R_2 = 0 \text{ or } V(GSR) = V_{O+}; \]

i.e., GSR is tied to PWRO+.  

Figure 7. Gain Setting Configuration
**ABSOLUTE MAXIMUM RATINGS**

Temperature under Bias ............ -10°C to +80°C
Storage Temperature .............. -65°C to +150°C
V<sub>CC</sub> and GRDD with Respect to V<sub>BB</sub> ........... -0.3V to +15V
All Input and Output Voltages with Respect to V<sub>BB</sub> ........... -0.3V to +15V
Power Dissipation .................. 1.35W

**NOTICE:** This is a production data sheet. The specifications are subject to change without notice.

*WARNING:* Stressing the device beyond the "Absolute Maximum Ratings" may cause permanent damage. These are stress ratings only. Operation beyond the "Operating Conditions" is not recommended and extended exposure beyond the "Operating Conditions" may affect device reliability.

**D.C. CHARACTERISTICS**

T<sub>A</sub> = 0°C to 70°C, V<sub>CC</sub> = +5V ±5%, V<sub>BB</sub> = −5V ±5%, GRDA = 0V, GRDD = 0V, unless otherwise specified

Typical values are for T<sub>A</sub> = 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>I&lt;sub&gt;IL&lt;/sub&gt;</td>
<td>Low Level Input Current</td>
<td>10</td>
<td>μA</td>
<td>GRDD ≤ V&lt;sub&gt;IN&lt;/sub&gt; ≤ V&lt;sub&gt;IL&lt;/sub&gt;&lt;sup&gt;(1)&lt;/sup&gt;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>I&lt;sub&gt;IIH&lt;/sub&gt;</td>
<td>High Level Input Current</td>
<td>10</td>
<td>μA</td>
<td>V&lt;sub&gt;IH&lt;/sub&gt; ≤ V&lt;sub&gt;IN&lt;/sub&gt; ≤ V&lt;sub&gt;CC&lt;/sub&gt;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>V&lt;sub&gt;IL&lt;/sub&gt;</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&lt;sub&gt;IH&lt;/sub&gt;</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&lt;sub&gt;OL&lt;/sub&gt;</td>
<td>Output Low Voltage</td>
<td>0.4</td>
<td>V</td>
<td>I&lt;sub&gt;OL&lt;/sub&gt; = 3.2 mA at D&lt;sub&gt;X&lt;/sub&gt;, TS&lt;sub&gt;X&lt;/sub&gt; and SIG&lt;sub&gt;R&lt;/sub&gt;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>V&lt;sub&gt;OH&lt;/sub&gt;</td>
<td>Output High Voltage</td>
<td>2.4</td>
<td>V</td>
<td>I&lt;sub&gt;OH&lt;/sub&gt; = 9.6 mA at D&lt;sub&gt;X&lt;/sub&gt; I&lt;sub&gt;OH&lt;/sub&gt; = 1.2 mA at SIG&lt;sub&gt;R&lt;/sub&gt;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>V&lt;sub&gt;ILO&lt;/sub&gt;</td>
<td>Input Low Voltage, CLKSEL&lt;sup&gt;(2)&lt;/sup&gt;</td>
<td>V&lt;sub&gt;BB&lt;/sub&gt;</td>
<td>V&lt;sub&gt;BB&lt;/sub&gt; + 0.5</td>
<td>V</td>
<td></td>
<td></td>
</tr>
<tr>
<td>V&lt;sub&gt;IIO&lt;/sub&gt;</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&lt;sub&gt;IHO&lt;/sub&gt;</td>
<td>Input High Voltage, CLKSEL</td>
<td>V&lt;sub&gt;CC&lt;/sub&gt; − 0.5</td>
<td>V&lt;sub&gt;CC&lt;/sub&gt;</td>
<td>V</td>
<td></td>
<td></td>
</tr>
<tr>
<td>C&lt;sub&gt;OX&lt;/sub&gt;</td>
<td>Digital Output Capacitance&lt;sup&gt;(3)&lt;/sup&gt;</td>
<td>5</td>
<td>pF</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C&lt;sub&gt;IN&lt;/sub&gt;</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
TA = 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 (Continued)

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(5)</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>IBB0</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(4)</td>
<td>140</td>
<td>200</td>
<td>mW</td>
<td>PDN ≤ VIL; after 10 μs</td>
<td></td>
</tr>
<tr>
<td>PD0</td>
<td>Power Down Dissipation(4)</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(4)</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. SIGx and DCLKx are TTL level inputs between GRDD and VCC; they are also pin straps for mode selection when tied to VBB. Under these conditions VILO 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. 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>IBXI</td>
<td>Input Leakage Current, VFxl+, VFxl−</td>
<td>100</td>
<td>nA</td>
<td>−2.17V ≤ VIN ≤ 2.17V</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RIXI</td>
<td>Input Resistance, VFxl+, VFxl−</td>
<td>10</td>
<td>MΩ</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>VOSXI</td>
<td>Input Offset Voltage, VFxl+, VFxl−</td>
<td>25</td>
<td>mV</td>
<td>−2.17V ≤ VIN ≤ 2.17V</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CMRR</td>
<td>Common Mode Rejection, VFxl+, VFxl−</td>
<td>55</td>
<td>dB</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>AVOL</td>
<td>DC Open Loop Voltage Gain, GSX</td>
<td>5000</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>fC</td>
<td>Open Loop Unity Gain Bandwidth, GSX</td>
<td>1</td>
<td>MHz</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CLXI</td>
<td>Load Capacitance, GSX</td>
<td>50</td>
<td>pF</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RLXI</td>
<td>Minimum Load Resistance, GSX</td>
<td>10</td>
<td>KΩ</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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>RORA</td>
<td>Output Resistance, PWRO+, PWRO−</td>
<td>1</td>
<td>Ω</td>
<td></td>
<td>Relative to GRDA</td>
<td></td>
</tr>
<tr>
<td>VOSRA</td>
<td>Single-Ended Output DC Offset, PWRO+, PWRO−</td>
<td>75</td>
<td>±150</td>
<td>mV</td>
<td>Relative to GRDA</td>
<td></td>
</tr>
<tr>
<td>CLRA</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, 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.\(^2\) All output levels are (sin \(x\))/x corrected. Specifications are for synchronous operation. Typical values are for \(T_A = 25°C\) and nominal power supply values. \(T_A = 0°C\) to \(+70°C\); \(V_{CC} = +5V \pm 5\%\); \(V_{BB} = -5V \pm 5\%\); \(GRDA = 0V; GRDD = 0V\); 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 (\mu)-law; Signal input of 1.068 Vrms A-law; (T_A = 25°C); (V_{BB} = -5V); (V_{CC} = +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(<em>R). Signal input per CCITT Recommendation G.711. Output signal of 1000 Hz, (R_L = \infty); (T_A = 25°C); (V</em>{BB} = -5V); (V_{CC} = +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 (OTLP). This corresponds to an analog signal input of 1.064 Vrms or an output of 1.053 Vrms for \(\mu\)-law. See Table 4.
2. Unity gain input amplifier; \(GS_x\) is connected to \(VF_x-\), Signal input \(VF_x+\); Maximum gain output amplifier; \(GR_x\) 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>2913-1</td>
<td>2914-1</td>
<td>2913, 2914</td>
<td>Min</td>
</tr>
<tr>
<td>GT(_X)</td>
<td>Transmit Gain Tracking Error Sinusoidal Input; (\mu)-Law</td>
<td>±0.2</td>
<td>±0.3</td>
<td>±0.65</td>
<td>±0.25</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>dB</td>
</tr>
<tr>
<td></td>
<td></td>
<td>dB</td>
<td></td>
<td></td>
<td>-40 to -55 dBm0</td>
</tr>
<tr>
<td>GT(_X)</td>
<td>Transmit Gain Tracking Error Sinusoidal Input; A-Law</td>
<td>±0.2</td>
<td>±0.3</td>
<td>±0.65</td>
<td>±0.25</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>dB</td>
</tr>
<tr>
<td></td>
<td></td>
<td>dB</td>
<td></td>
<td></td>
<td>-50 to -55 dBm0</td>
</tr>
<tr>
<td>GT(_R)</td>
<td>Receive Gain Tracking Error Sinusoidal Input; (\mu)-Law</td>
<td>±0.2</td>
<td>±0.3</td>
<td>±0.65</td>
<td>±0.25</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>dB</td>
</tr>
<tr>
<td></td>
<td></td>
<td>dB</td>
<td></td>
<td></td>
<td>-50 to -55 dBm0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Measured at (PWRO+), (R_L = 300\Omega)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GT(_R)</td>
<td>Receive Gain Tracking Error Sinusoidal Input; A-Law</td>
<td>±0.2</td>
<td>±0.3</td>
<td>±0.65</td>
<td>±0.25</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>dB</td>
</tr>
<tr>
<td></td>
<td></td>
<td>dB</td>
<td></td>
<td></td>
<td>-50 to -55 dBm0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Measured at (PWRO+), (R_L = 300\Omega)</td>
<td></td>
<td></td>
<td></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\textsubscript{XC1}</td>
<td>Transmit Noise, C-Message Weighted</td>
<td>13</td>
<td>15</td>
<td>dBm0</td>
<td>VF\textsubscript{XL}+ = GRDA, VF\textsubscript{XL}− = GS\textsubscript{X}</td>
</tr>
<tr>
<td>N\textsubscript{XC2}</td>
<td>Transmit Noise, C-Message Weighted with Eighth Bit Signaling</td>
<td>16</td>
<td>18</td>
<td>dBm0</td>
<td>VF\textsubscript{XL}+ = GRDA, VF\textsubscript{XL}− = GS\textsubscript{X}; 6th Frame Signaling</td>
</tr>
<tr>
<td>N\textsubscript{XP}</td>
<td>Transmit Noise, Psophometrically Weighted</td>
<td>-77</td>
<td>-75</td>
<td>dBm0</td>
<td>VF\textsubscript{XL}+ = GRDA, VF\textsubscript{XL}− = GS\textsubscript{X}</td>
</tr>
<tr>
<td>N\textsubscript{RC1}</td>
<td>Receive Noise, C-Message Weighted: Quiet Code</td>
<td>8</td>
<td>11</td>
<td>dBm0</td>
<td>DR = 11111111</td>
</tr>
<tr>
<td>N\textsubscript{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\textsubscript{R} is zero code with sign bit toggle at 1 KHz rate</td>
</tr>
<tr>
<td>N\textsubscript{RP}</td>
<td>Receive Noise, Psophometrically Weighted</td>
<td>-82</td>
<td>-79</td>
<td>dBm0</td>
<td>DR = lowest positive decode level</td>
</tr>
<tr>
<td>N\textsubscript{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\textsubscript{1}</td>
<td>\textsubscript{VCC} 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\textsubscript{X}</td>
</tr>
<tr>
<td>PSRR\textsubscript{2}</td>
<td>\textsubscript{VBB} 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\textsubscript{X}</td>
</tr>
<tr>
<td>PSRR\textsubscript{3}</td>
<td>\textsubscript{VCC} 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\textsubscript{4}</td>
<td>\textsubscript{VBB} 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\textsubscript{TR}</td>
<td>Crosstalk, Transmit to Receive</td>
<td>-80</td>
<td>-71</td>
<td>dB</td>
<td>VF\textsubscript{XL}+ = 0 dBm0, 1.02 KHz, D\textsubscript{R} = lowest positive decode level, measure at PWRO+</td>
</tr>
<tr>
<td>CT\textsubscript{RT}</td>
<td>Crosstalk, Receive to Transmit</td>
<td>-80</td>
<td>-71</td>
<td>dB</td>
<td>D\textsubscript{R} = 0 dBm0, 1.02 KHz VF\textsubscript{XL}+ = GRDA, measure at D\textsubscript{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>SD1X</td>
<td>Transmit Signal to Distortion, μ-Law Sinusoidal Input; CCITT G.714-Method 2</td>
<td>36</td>
<td>30</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>SD2X</td>
<td>Transmit Signal to Distortion, A-Law Sinusoidal Input; CCITT G.714-Method 2</td>
<td>36</td>
<td>30</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>SD1R</td>
<td>Receive Signal to Distortion, μ-Law Sinusoidal Input; CCITT G.714-Method 2</td>
<td>36</td>
<td>30</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>SD2R</td>
<td>Receive Signal to Distortion, A-Law Sinusoidal Input; CCITT G.714-Method 2</td>
<td>36</td>
<td>30</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</td>
<td>Transmit Single Frequency Distortion Products</td>
<td>−46 dBm0</td>
<td>AT&amp;T Advisory #64 (3.8)</td>
<td>0 dBm0 Input Signal</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DP R</td>
<td>Receive Single Frequency Distortion Products</td>
<td>−46 dBm0</td>
<td>AT&amp;T Advisory #64 (3.8)</td>
<td>0 dBm0 Input Signal</td>
<td></td>
<td></td>
</tr>
<tr>
<td>IMD1</td>
<td>Intermodulation Distortion, End to End Measurement</td>
<td>−35 dB</td>
<td>CCITT G.712 (7.1)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>IMD2</td>
<td>Intermodulation Distortion, End to End Measurement</td>
<td>−49 dBm0</td>
<td>CCITT G.712 (7.2)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SOS</td>
<td>Spurious Out of Band Signals, End to End Measurement</td>
<td>−25 dBm0</td>
<td>CCITT G.712 (6.1)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SIS</td>
<td>Spurious in Band Signals, End to End Measurement</td>
<td>−40 dBm0</td>
<td>CCITT G.712 (9)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DAX</td>
<td>Transmit Absolute Delay</td>
<td>245 μs</td>
<td>Fixed Data Rate. CLKX = 2.048 MHz 0 dBm0, 1.02 kHz signal at VFXY + Measure at DX.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DDX</td>
<td>Transmit Differential Envelope Delay Relative to DAX</td>
<td>170 μs f = 500 – 600 Hz 95 μs f = 600 – 1000 Hz 45 μs f = 1000 – 2600 Hz 105 μs f = 2600 – 2800 Hz</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DAR</td>
<td>Receive Absolute Delay</td>
<td>190 μs</td>
<td>Fixed Data Rate, CLKR = 2.048 MHz; Digital input is DMW codes. Measure at PWRO + .</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DDR</td>
<td>Receive Differential Envelope Delay Relative to DAR</td>
<td>45 μs f = 500 – 600 Hz 35 μs f = 600 – 1000 Hz 85 μs f = 1000 – 2600 Hz 110 μs f = 2600 – 2800 Hz</td>
<td></td>
<td></td>
<td></td>
<td></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>GRX</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 VFx</td>
</tr>
<tr>
<td>16.67 Hz</td>
<td>-30 dB</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>50 Hz</td>
<td>-25 dB</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>60 Hz</td>
<td>-23 dB</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>200 Hz</td>
<td>-1.8 dB</td>
<td></td>
<td></td>
<td>-0.125 dB</td>
<td></td>
<td></td>
</tr>
<tr>
<td>300 to 3000 Hz</td>
<td>-0.125 dB</td>
<td></td>
<td>+0.125 dB</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3300 Hz</td>
<td>-0.35 dB</td>
<td></td>
<td></td>
<td>+ 0.03 dB</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3400 Hz</td>
<td>-0.7 dB</td>
<td></td>
<td></td>
<td>-0.10 dB</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4000 Hz</td>
<td>-14 dB</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4600 Hz and Above</td>
<td>-32 dB</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure 8. Transmit Channel

5-61
A.C. CHARACTERISTICS—TRANSMISSION PARAMETERS (Continued)

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>-0.5</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>-0.7</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 and Above</td>
<td></td>
<td>-30    dB</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure 9. Receive Channel

5-62
### 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, CLKₙ, CLKᵣ</td>
<td>488</td>
<td>ns</td>
<td></td>
<td>ns</td>
<td>( f_{CLKₙ} = f_{CLKᵣ} = 2.048 \text{ MHz} )</td>
</tr>
<tr>
<td>tCLK</td>
<td>Clock Pulse Width, CLKₙ, CLKᵣ</td>
<td>220</td>
<td>ns</td>
<td></td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>tDCLK</td>
<td>Data Clock Pulse Width</td>
<td>220</td>
<td>ns</td>
<td></td>
<td>ns</td>
<td>( 64 \text{ KHz} \leq f_{DCLK} \leq 2.048 \text{ MHz} )</td>
</tr>
<tr>
<td>tDCC</td>
<td>Clock Duty Cycle, CLKₙ, CLKᵣ</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(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>145</td>
<td>ns</td>
<td></td>
<td>( 0 &lt; C_{LOAD} &lt; 100 \text{ pF} )</td>
</tr>
<tr>
<td>tDDX</td>
<td>Data Delay from CLKₙ</td>
<td>0</td>
<td>145</td>
<td>ns</td>
<td></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>215</td>
<td>ns</td>
<td></td>
<td>( C_{LOAD} = 0 )</td>
</tr>
<tr>
<td>tSON</td>
<td>Timeslot X to Enable</td>
<td>0</td>
<td>145</td>
<td>ns</td>
<td></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>215</td>
<td>ns</td>
<td></td>
<td>( C_{LOAD} = 0 )</td>
</tr>
<tr>
<td>tFSD</td>
<td>Frame Sync Delay</td>
<td>100</td>
<td></td>
<td></td>
<td></td>
<td>( t_{CLK} )</td>
</tr>
<tr>
<td>tSS</td>
<td>Signal Setup Time</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>tSH</td>
<td>Signal Hold Time</td>
<td>0</td>
<td></td>
<td></td>
<td></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></td>
<td></td>
<td>( t_{CLK} )</td>
</tr>
<tr>
<td>tSIGR</td>
<td>SIGᵣ Update</td>
<td>0</td>
<td>2</td>
<td></td>
<td>( \mu s )</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 \( V_{IL} \).
### 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>( t_{TSDX} )</td>
<td>Timeslot Delay from DCLK(_{X})(^{(2)} )</td>
<td>140</td>
<td>( t_{DX} )</td>
<td>140</td>
<td>ns</td>
<td>( t_{DOFF} ) ( \leq ) 100 ns ( C_{LOAD} \leq 100 ) pF</td>
</tr>
<tr>
<td>( t_{FS} )</td>
<td>Frame Sync Delay</td>
<td>100</td>
<td>( t_{CY} )</td>
<td>100</td>
<td>ns</td>
<td>( t_{DOFF} ) ( \leq ) 50 ns ( C_{LOAD} \leq 100 ) pF</td>
</tr>
<tr>
<td>( t_{DDX} )</td>
<td>Data Delay from DCLK(_{X})</td>
<td>0</td>
<td>100</td>
<td>ns</td>
<td>( 0 &lt; C_{LOAD} &lt; 100 ) pF</td>
<td></td>
</tr>
<tr>
<td>( t_{DON} )</td>
<td>Timeslot to ( D_X ) Active</td>
<td>0</td>
<td>50</td>
<td>ns</td>
<td>( 0 &lt; C_{LOAD} &lt; 100 ) pF</td>
<td></td>
</tr>
<tr>
<td>( t_{DOFF} )</td>
<td>Timeslot to ( D_X ) Inactive</td>
<td>0</td>
<td>80</td>
<td>ns</td>
<td>( 0 &lt; C_{LOAD} &lt; 100 ) pF</td>
<td></td>
</tr>
<tr>
<td>( t_{DX} )</td>
<td>Data Clock Period</td>
<td>( )</td>
<td>488</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>140</td>
<td>ns</td>
<td></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>( t_{DR} )</td>
<td>140</td>
<td>ns</td>
<td>( t_{DOFF} ) ( \leq ) 100 ns</td>
</tr>
<tr>
<td>( t_{FS} )</td>
<td>Frame Sync Delay</td>
<td>100</td>
<td>( t_{CY} )</td>
<td>100</td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>( t_{DSR} )</td>
<td>Data Setup Time</td>
<td>10</td>
<td></td>
<td>ns</td>
<td></td>
<td></td>
</tr>
<tr>
<td>( t_{DHR} )</td>
<td>Data Hold Time</td>
<td>60</td>
<td></td>
<td>ns</td>
<td></td>
<td></td>
</tr>
<tr>
<td>( t_{DR} )</td>
<td>Data Clock Period</td>
<td>( )</td>
<td>488</td>
<td>15620</td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>( t_{SER} )</td>
<td>Timeslot End Receive Time</td>
<td>60</td>
<td></td>
<td>ns</td>
<td></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>ns</td>
<td></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>ns</td>
<td></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></td>
<td>10</td>
<td>( \mu s )</td>
<td></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

NOTE:
All timing parameters referenced to $V_{IH}$ and $V_{IL}$ except $t_{DON}$ and $t_{DOFF}$ 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".
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
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>$V_{BB}$</td>
<td>Power (-5V)</td>
<td>$GS_X$</td>
<td>Transmit Gain Control</td>
</tr>
<tr>
<td>PWRO +, PWRO -</td>
<td>Power Amplifier Outputs</td>
<td>$V_{FI}^-$</td>
<td>Analog Input</td>
</tr>
<tr>
<td>PDN</td>
<td>Power Down Select</td>
<td>GRDA</td>
<td>Analog Ground</td>
</tr>
<tr>
<td>DCLKR</td>
<td>Receive Variable Data Clock</td>
<td>$TS_X$</td>
<td>Timeslot Strobe/Buffer Enable</td>
</tr>
<tr>
<td>DR</td>
<td>Receive PCM Input</td>
<td>DCLKX</td>
<td>Transmit Variable Data Clock</td>
</tr>
<tr>
<td>FSR</td>
<td>Receive Frame Synchronization Clock</td>
<td>$D_X$</td>
<td>Transmit PCM Output</td>
</tr>
<tr>
<td>GRDD</td>
<td>Digital Ground</td>
<td>$FS_X$</td>
<td>Transmit Frame Synchronization Clock</td>
</tr>
<tr>
<td>$V_{CC}$</td>
<td>Power (+5V)</td>
<td>CLK</td>
<td>Master Clock</td>
</tr>
</tbody>
</table>
Table 2. Pin Description

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>$V_{BB}$</td>
<td>Most negative supply, input voltage is $-5 \text{ volts} \pm 5%$.</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>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>DCLKR</td>
<td>Selects the fixed or variable data rate mode. When DCLKR is connected to $V_{BB}$, the fixed data rate mode is selected. When DCLKR is not connected to $V_{BB}$, the device operates in the variable data rate mode. In this mode DCLKR becomes the receive data clock which operates at TTL levels from 64 Kb to 2.048 Mb data rates.</td>
</tr>
<tr>
<td>$D_R$</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>
</tr>
<tr>
<td>$FS_R$</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>
</tr>
<tr>
<td>GRDD</td>
<td>Digital ground for all internal logic circuits. Not internally tied to GRDA.</td>
</tr>
<tr>
<td>CLK</td>
<td>Master and data clock for the fixed data rate mode; 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>$D_X$</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>
</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 Kb to 2.048 Mb data rates.</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>Inverting analog input to uncommitted transmit operational amplifier.</td>
</tr>
<tr>
<td>$GS_X$</td>
<td>Output terminal of on-chip 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 $+5 \text{ volts} \pm 5%$.</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.

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.

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.

<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 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 within 300 milliseconds.</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>TSX and DX are placed in a high impedance state within 300 milliseconds.</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 FSR 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 DCLKR to the bit clock for the receive PCM highway rather than to VBB. It employs master clock CLK, bit clocks DCLKR and DCLKX, and frame synchronization clocks FSR and FSX.

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, DCLKR and DCLKX 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 DCLKX. Similarly, while FSR is high, each PCM bit from the highway is received by Dr on the next eight consecutive negative transitions of DCLKR.

On the transmit side, the PCM word will be repeated in all remaining timeslots in the 125 µs frame as long as DCLKX 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 (GSX) must be greater than 10 kilohms in parallel with less than 50 pF. The input signal on lead VFx — 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-url)
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 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 (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 D3 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</th>
<th>Value</th>
<th>Units</th>
<th>Test Conditions</th>
</tr>
</thead>
<tbody>
<tr>
<td>0TLP1x</td>
<td>Zero Transmission Level Point Transmit Channel (0dBm0) μ-law</td>
<td>+2.76</td>
<td>dBm</td>
<td>Referenced to 600Ω</td>
</tr>
<tr>
<td></td>
<td></td>
<td>+1.00</td>
<td>dBm</td>
<td>Referenced to 900Ω</td>
</tr>
<tr>
<td>0TLP2x</td>
<td>Zero Transmission Level Point Transmit Channel (0dBm0) A-law</td>
<td>+2.79</td>
<td>dBm</td>
<td>Referenced to 600Ω</td>
</tr>
<tr>
<td></td>
<td></td>
<td>+1.03</td>
<td>dBm</td>
<td>Referenced to 900Ω</td>
</tr>
<tr>
<td>0TLP1r</td>
<td>Zero Transmission Level Point Receive Channel (0dBm0) μ-law</td>
<td>+5.76</td>
<td>dBm</td>
<td>Referenced to 600Ω</td>
</tr>
<tr>
<td></td>
<td></td>
<td>+4.00</td>
<td>dBm</td>
<td>Referenced to 900Ω</td>
</tr>
<tr>
<td>0TLP2r</td>
<td>Zero Transmission Level Point Receive Channel (0dBm0) A-law</td>
<td>+5.79</td>
<td>dBm</td>
<td>Referenced to 600Ω</td>
</tr>
<tr>
<td></td>
<td></td>
<td>+4.03</td>
<td>dBm</td>
<td>Referenced to 900Ω</td>
</tr>
</tbody>
</table>
ABSOLUTE MAXIMUM RATINGS
Temperature Under Bias ........... -10°C to +80°C
Storage Temperature ............ -65°C to +150°C
VCC and GRDD with Respect
to VBB ........................ -0.3V to +15V
All Input and Output Voltages
with Respect to VBB ............ -0.3V to +15V
Power Dissipation ................ 1.35W

D.C. CHARACTERISTICS
(TA = 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></td>
<td>GRDD ≤ VIN ≤ VIL (Note 1)</td>
</tr>
<tr>
<td>IIH</td>
<td>High Level Input Current</td>
<td>10</td>
<td>µA</td>
<td></td>
<td></td>
<td>VIH ≤ VIN ≤ VCC</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></td>
<td>IOL = 3.2 mA at Dx, T<em>S</em></td>
</tr>
<tr>
<td>VOH</td>
<td>Output High Voltage</td>
<td>2.4</td>
<td>V</td>
<td></td>
<td></td>
<td>IOH = 9.6 mA at DX</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></td>
<td>PDN ≤ VIL; after 10 µs</td>
</tr>
<tr>
<td>IBB0</td>
<td>VBB Power Down Current</td>
<td>-0.5</td>
<td>1.0</td>
<td>mA</td>
<td></td>
<td>PDN ≤ VIL; after 10 µs</td>
</tr>
<tr>
<td>ICCS</td>
<td>VCC Standby Current</td>
<td>1.2</td>
<td>2.4</td>
<td>mA</td>
<td></td>
<td>FSX, FS_R ≤ VIL; after 300 ms</td>
</tr>
<tr>
<td>IBBS</td>
<td>VBB Standby Current</td>
<td>-1.2</td>
<td>2.4</td>
<td>mA</td>
<td></td>
<td>FSX, FS_R ≤ VIL; after 300 ms</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>PDD0</td>
<td>Power Down Dissipation(3)</td>
<td>5</td>
<td>10</td>
<td>mW</td>
<td></td>
<td>PDN ≤ VIL; after 10 µs</td>
</tr>
<tr>
<td>PST</td>
<td>Standby Power Dissipation(3)</td>
<td>12</td>
<td>25</td>
<td>mW</td>
<td></td>
<td>FSX, FS_R ≤ VIL</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.

2916/2917
### 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></td>
<td></td>
<td>nA</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></td>
<td></td>
<td>MΩ</td>
<td></td>
</tr>
<tr>
<td>V_{OSX1}</td>
<td>Input Offset Voltage, V_{FX1}--</td>
<td>25</td>
<td></td>
<td></td>
<td>mV</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></td>
<td></td>
<td>MHz</td>
<td></td>
</tr>
<tr>
<td>C_{LX1}</td>
<td>Load Capacitance, G_{SX}</td>
<td>50</td>
<td></td>
<td></td>
<td>pF</td>
<td></td>
</tr>
<tr>
<td>R_{LX1}</td>
<td>Minimum Load Resistance, G_{SX}</td>
<td>10</td>
<td></td>
<td></td>
<td>KΩ</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></td>
<td></td>
<td>Ω</td>
<td></td>
</tr>
<tr>
<td>V_{OSRA}</td>
<td>Single-Ended Output DC Offset, PWRO+, PWRO-</td>
<td>75</td>
<td></td>
<td></td>
<td>mV</td>
<td>Relative to GRDA</td>
</tr>
<tr>
<td>C_{LRA}</td>
<td>Load Capacitance, PWRO+, PWRO-</td>
<td>100</td>
<td></td>
<td></td>
<td>pF</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. (\(T_A = 0°C\) to +70°C; \(V_{CC} = +5V \pm 5\%; V_{BB} = -5V \pm 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>Units</th>
<th>Test Conditions</th>
</tr>
</thead>
<tbody>
<tr>
<td>E_mW</td>
<td>Encoder Milliwatt Response (\text{Transmit Gain Tolerance})</td>
<td>-0.18</td>
<td>±0.04</td>
<td>+0.18</td>
<td>dBm0</td>
<td>Signal Input of 1.064 V_rms (\mu)-law Signal Input of 1.068 V_rms (A)-law (T_A = 25°C, V_{BB} = -5V, V_{CC} = +5V)</td>
</tr>
<tr>
<td>E_{mWTS}</td>
<td>E_mW 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>D_mW</td>
<td>Digital Milliwatt Response (\text{Receive Gain Tolerance})</td>
<td>-0.18</td>
<td>±0.04</td>
<td>+0.18</td>
<td>dBm0</td>
<td>Measure Relative to 0TLP_R. 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>D_mW 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>

**NOTE:**

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 volts rms or an output of 1.503 volts rms \((\mu\text{law})\).
### GAIN TRACKING
Reference Level = -10 dBm0

<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><strong>GT1x</strong></td>
<td>Transmit Gain Tracking Error</td>
<td>±0.25</td>
<td>±0.5</td>
<td>±1.2</td>
<td>dB</td>
</tr>
<tr>
<td></td>
<td>Sinusoidal Input; μ-law</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><strong>GT2x</strong></td>
<td>Transmit Gain Tracking Error</td>
<td></td>
<td>±0.25</td>
<td></td>
<td>dB</td>
</tr>
<tr>
<td></td>
<td>Sinusoidal Input; A-law</td>
<td>±0.5</td>
<td>±1.2</td>
<td>dB</td>
<td>-40 to -50 dBm0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>dB</td>
</tr>
<tr>
<td><strong>GT1R</strong></td>
<td>Receive Gain Tracking Error</td>
<td>±0.25</td>
<td>±0.5</td>
<td>±1.2</td>
<td>dB</td>
</tr>
<tr>
<td></td>
<td>Sinusoidal Input; μ-law</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><strong>GT2R</strong></td>
<td>Receive Gain Tracking Error</td>
<td></td>
<td>±0.25</td>
<td></td>
<td>dB</td>
</tr>
<tr>
<td></td>
<td>Sinusoidal Input; A-law</td>
<td>±0.5</td>
<td>±1.2</td>
<td>dB</td>
<td>-40 to -50 dBm0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>dB</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>Min</td>
</tr>
<tr>
<td><strong>NXC1</strong></td>
<td>Transmit Noise, C-Message 15</td>
<td>15</td>
<td></td>
<td>dBrncO</td>
<td>Unity Gain</td>
</tr>
<tr>
<td></td>
<td>Weighted</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>NXP</strong></td>
<td>Transmit Noise, Psophometrically -75</td>
<td></td>
<td></td>
<td>dBm0p</td>
<td>Unity Gain</td>
</tr>
<tr>
<td></td>
<td>Weighted</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>NRC1</strong></td>
<td>Receive Noise, C-Message 11</td>
<td>11</td>
<td></td>
<td>dBrncO</td>
<td>DR = 11111111</td>
</tr>
<tr>
<td></td>
<td>Weighted Quiet Code</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>NRC2</strong></td>
<td>Receive Noise, C-Message 12</td>
<td>12</td>
<td></td>
<td>dBrncO</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Weighted Sign Bit Toggle</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>NRP</strong></td>
<td>Receive Noise, Psophometrically</td>
<td>-79</td>
<td></td>
<td>dBm0p</td>
<td>DR = Lowest Positive Decode Level</td>
</tr>
<tr>
<td></td>
<td>Weighted</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>NSF</strong></td>
<td>Single Frequency Noise</td>
<td>-50</td>
<td></td>
<td>dBm0</td>
<td>CCITT G.712.4.2</td>
</tr>
<tr>
<td></td>
<td>End to End Measurement</td>
<td></td>
<td></td>
<td></td>
<td>Measure at PWRO+</td>
</tr>
<tr>
<td><strong>PSRR1</strong></td>
<td>Vcc Power Supply Rejection, -30</td>
<td>-30</td>
<td>dB</td>
<td>Idle Channel; 200 mV P-P Signal on Supply; 0 to 50 KHz, Measure at DX</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Transmit Channel</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>PSRR2</strong></td>
<td>Vbb Power Supply Rejection, -30</td>
<td>-30</td>
<td>dB</td>
<td>Idle Channel; 200 mV P-P Signal on Supply; 0 to 50 KHz, Measure at DX</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Transmit Channel</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>PSRR3</strong></td>
<td>Vcc Power Supply Rejection, -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>
<td></td>
</tr>
<tr>
<td></td>
<td>Receive Channel</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>PSRR4</strong></td>
<td>Vbb Power Supply Rejection, -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>
<td></td>
</tr>
<tr>
<td></td>
<td>Receive Channel</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>CTTR</strong></td>
<td>Crosstalk, Transmit to Receive</td>
<td>-71</td>
<td></td>
<td>dB</td>
<td>Input = 0 dBm0, Unity Gain, 1.02 KHz, DR = Lowest Positive Decode Level, Measure at PWRO+</td>
</tr>
<tr>
<td><strong>CTR</strong></td>
<td>Crosstalk, Receive to Transmit</td>
<td>-71</td>
<td></td>
<td>dB</td>
<td>DR = 0 dBm0, 1.02 KHz, Measure at DX</td>
</tr>
</tbody>
</table>
## 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(X)</td>
<td>Transmit Signal to Distortion, (\mu)-Law Sinusoidal Input; CCITT G.714-Method 2 (2916)</td>
<td>36</td>
<td>dB</td>
<td>0 dBm0 to -30 dBm0</td>
<td>30</td>
<td>dB</td>
</tr>
<tr>
<td>SD2(X)</td>
<td>Transmit Signal to Distortion, A-Law Sinusoidal Input; CCITT G.714-Method 2 (2917)</td>
<td>36</td>
<td>dB</td>
<td>0 dBm0 to -30 dBm0</td>
<td>30</td>
<td>dB</td>
</tr>
<tr>
<td>SD1(R)</td>
<td>Receive Signal to Distortion, (\mu)-Law Sinusoidal Input; CCITT G.714-Method 2 (2916)</td>
<td>36</td>
<td>dB</td>
<td>0 dBm0 to -30 dBm0</td>
<td>30</td>
<td>dB</td>
</tr>
<tr>
<td>SD2(R)</td>
<td>Receive Signal to Distortion, A-Law Sinusoidal Input; CCITT G.714-Method 2 (2917)</td>
<td>36</td>
<td>dB</td>
<td>0 dBm0 to -30 dBm0</td>
<td>30</td>
<td>dB</td>
</tr>
<tr>
<td>DP(X)</td>
<td>Transmit Single Frequency Distortion Products (2916)</td>
<td>-46 dBm0</td>
<td>AT&amp;T Advisory #64 (3.8)</td>
<td>0 dBm0 Input Signal</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DP(R)</td>
<td>Receive Single Frequency Distortion Products (2916)</td>
<td>-46 dBm0</td>
<td>AT&amp;T Advisory #64 (3.8)</td>
<td>0 dBm0 Input Signal</td>
<td></td>
<td></td>
</tr>
<tr>
<td>IMD(_1)</td>
<td>Intermodulation Distortion, End to End Measurement</td>
<td>-35 dB</td>
<td>CCITT G.712 (7.1)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>IMD(_2)</td>
<td>Intermodulation Distortion, End to End Measurement</td>
<td>-49 dBm0</td>
<td>CCITT G.712 (7.2)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SOS</td>
<td>Spurious Out of Band Signals, End to End Measurement</td>
<td>-25 dBm0</td>
<td>CCITT G.712 (6.1)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SIS</td>
<td>Spurious In Band Signals, End to End Measurement</td>
<td>-40 dBm0</td>
<td>CCITT G.712 (9)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D(AX)</td>
<td>Transmit Absolute Delay</td>
<td>245 (\mu)s</td>
<td>Fixed Data Rate. CLK(_X) = 2.048 MHz; 0 dBm0, 1.02 KHz Input Signal, Unity Gain. Measure at D(_X).</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D(DX)</td>
<td>Transmit Differential Envelope Delay Relative to D(AX)</td>
<td>170 (\mu)s</td>
<td>f = 500 Hz to 600 Hz</td>
<td>95 (\mu)s</td>
<td>f = 600 Hz to 1000 Hz</td>
<td>45 (\mu)s</td>
</tr>
<tr>
<td>D(AR)</td>
<td>Receive Absolute Delay</td>
<td>190 (\mu)s</td>
<td>Fixed Data Rate, CLK = 2.048 MHz; Digital Input is DMW Codes. Measure at PWRO+</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D(DR)</td>
<td>Receive Differential Envelope Delay Relative to D(AR)</td>
<td>45 (\mu)s</td>
<td>f = 500 Hz to 600 Hz</td>
<td>35 (\mu)s</td>
<td>f = 600 Hz to 1000 Hz</td>
<td>85 (\mu)s</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 $V_{FLL}$</td>
</tr>
<tr>
<td></td>
<td>16.67 Hz</td>
<td>−30</td>
<td>dB</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>50 Hz</td>
<td>−25</td>
<td>dB</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>60 Hz</td>
<td>−23</td>
<td>dB</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>200 Hz</td>
<td>−1.8</td>
<td>dB</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>dB</td>
<td>+0.125</td>
<td>dB</td>
<td></td>
</tr>
<tr>
<td></td>
<td>3300 Hz</td>
<td>−0.35</td>
<td>dB</td>
<td>+0.03</td>
<td>dB</td>
<td></td>
</tr>
<tr>
<td></td>
<td>3400 Hz</td>
<td>−0.7</td>
<td>dB</td>
<td>−0.10</td>
<td>dB</td>
<td></td>
</tr>
<tr>
<td></td>
<td>4000 Hz</td>
<td>−14</td>
<td>dB</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>4600 Hz and Above</td>
<td>−32</td>
<td>dB</td>
<td></td>
<td></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&lt;sub&gt;R&lt;/sub&gt;</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&lt;sub&gt;R&lt;/sub&gt;</td>
</tr>
<tr>
<td></td>
<td>Below 200 Hz</td>
<td></td>
<td>+0.125</td>
<td></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 Description</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(1)

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Parameter Description</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>tDZX</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>tSOFF</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>tCLK</td>
<td>ns</td>
<td></td>
</tr>
</tbody>
</table>

RECEIVE SECTION, FIXED DATA RATE MODE

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Parameter Description</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_{SOFF} 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 $t_{DZX}$, $t_{SOFF}$ and $t_{HZX}$ 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(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>tTSDX</td>
<td>Timeslot Delay from DCLK&lt;sub&gt;X&lt;/sub&gt;(2)</td>
<td>140</td>
<td>t&lt;sub&gt;DX&lt;/sub&gt; - 140</td>
<td>ns</td>
<td></td>
<td></td>
</tr>
<tr>
<td>tFSD</td>
<td>Frame Sync Delay</td>
<td>100</td>
<td>t&lt;sub&gt;CY&lt;/sub&gt; - 100</td>
<td>ns</td>
<td></td>
<td></td>
</tr>
<tr>
<td>tDDX</td>
<td>Data Delay from DCLK&lt;sub&gt;X&lt;/sub&gt;</td>
<td>0</td>
<td>100</td>
<td>ns</td>
<td>0 &lt; C&lt;sub&gt;LOAD&lt;/sub&gt; &lt; 100 pF</td>
<td></td>
</tr>
<tr>
<td>tDON</td>
<td>Timeslot to D&lt;sub&gt;X&lt;/sub&gt; Active</td>
<td>0</td>
<td>50</td>
<td>ns</td>
<td>0 &lt; C&lt;sub&gt;LOAD&lt;/sub&gt; &lt; 100 pF</td>
<td></td>
</tr>
<tr>
<td>tDOFF</td>
<td>Timeslot to D&lt;sub&gt;X&lt;/sub&gt; Inactive</td>
<td>0</td>
<td>80</td>
<td>ns</td>
<td>0 &lt; C&lt;sub&gt;LOAD&lt;/sub&gt; &lt; 100 pF</td>
<td></td>
</tr>
<tr>
<td>tDX</td>
<td>Data Clock Period</td>
<td>488</td>
<td>15620</td>
<td>ns</td>
<td></td>
<td></td>
</tr>
<tr>
<td>tDFSX</td>
<td>Data Delay from FS&lt;sub&gt;X&lt;/sub&gt;</td>
<td>0</td>
<td>140</td>
<td>ns</td>
<td></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>tTSDR</td>
<td>Timeslot Delay from DCLK&lt;sub&gt;R&lt;/sub&gt;(3)</td>
<td>140</td>
<td>t&lt;sub&gt;DR&lt;/sub&gt; - 140</td>
<td>ns</td>
<td></td>
<td></td>
</tr>
<tr>
<td>tFSD</td>
<td>Frame Sync Delay</td>
<td>100</td>
<td>t&lt;sub&gt;CY&lt;/sub&gt; - 100</td>
<td>ns</td>
<td></td>
<td></td>
</tr>
<tr>
<td>tDSR</td>
<td>Data Setup Time</td>
<td>10</td>
<td></td>
<td>ns</td>
<td></td>
<td></td>
</tr>
<tr>
<td>tDHR</td>
<td>Data Hold Time</td>
<td>60</td>
<td></td>
<td>ns</td>
<td></td>
<td></td>
</tr>
<tr>
<td>tDR</td>
<td>Data Clock Period</td>
<td>488</td>
<td>15620</td>
<td>ns</td>
<td></td>
<td></td>
</tr>
<tr>
<td>tSRR</td>
<td>Timeslot End Receive Time</td>
<td>60</td>
<td></td>
<td>ns</td>
<td></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 Downtime</td>
<td>488</td>
<td></td>
<td>ns</td>
<td></td>
<td>FS&lt;sub&gt;X&lt;/sub&gt; is TTL High for Remainder of Frame</td>
</tr>
<tr>
<td>tFSLR</td>
<td>Receive Frame Sync Minimum Downtime</td>
<td>1952</td>
<td></td>
<td>ns</td>
<td></td>
<td>FS&lt;sub&gt;R&lt;/sub&gt; 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 tTSDX maximum spec for 64 KHz operation.
3. tFSLR minimum requirements overrides tTSDR maximum spec for 64 KHz operation.
VARIABLE DATA RATE TIMING

TRANSMIT TIMING

RECEIVE TIMING

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

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".
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 VFRI to receive input of the 2912A VFRI. 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, CLKR 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 lay out 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, 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**

**2911A/2912A 0 dBm0**
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
# DESIGNING SECOND-GENERATION DIGITAL TELEPHONY SYSTEMS USING THE INTEL 2913/14 CODEC/FILTER COMBOCHIP

## CONTENTS

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.0 INTRODUCTION</td>
<td>5-90</td>
</tr>
<tr>
<td>1.1 Background</td>
<td>5-90</td>
</tr>
<tr>
<td>1.2 Comparison of First- and Second-Generation Component Capabilities</td>
<td>5-90</td>
</tr>
<tr>
<td>2.0 DESIGN CONSIDERATIONS</td>
<td>5-93</td>
</tr>
<tr>
<td>2.1 Operating and Test Mode Selection</td>
<td>5-94</td>
</tr>
<tr>
<td>2.2 Data Rate Modes</td>
<td>5-94</td>
</tr>
<tr>
<td>2.3 Margin to Performance Specifications</td>
<td>5-95</td>
</tr>
<tr>
<td>2.4 Power Conservation</td>
<td>5-97</td>
</tr>
<tr>
<td>2.5 Elimination of Gain Trim in the Line Circuit</td>
<td>5-99</td>
</tr>
<tr>
<td>2.6 Power Up/Down Considerations</td>
<td>5-100</td>
</tr>
<tr>
<td>3.0 OPERATING MODES</td>
<td>5-100</td>
</tr>
<tr>
<td>3.1 Fixed Data Rate Mode</td>
<td>5-100</td>
</tr>
<tr>
<td>3.2 Variable Data Rate Mode</td>
<td>5-100</td>
</tr>
<tr>
<td>3.3 On-Line Test Modes</td>
<td>5-102</td>
</tr>
<tr>
<td>4.0 MULTIMODE TEST CAPABILITIES</td>
<td>5-102</td>
</tr>
<tr>
<td>4.1 Design/Prototype Testing</td>
<td>5-102</td>
</tr>
<tr>
<td>4.2 Production Testing</td>
<td>5-104</td>
</tr>
<tr>
<td>4.3 Optional On-Line Testing</td>
<td>5-106</td>
</tr>
</tbody>
</table>
Note: See data sheet for latest specifications. Values given in this application note are for reference only, and were considered correct at the time of publication (Feb. 1982).

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.
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>
<thead>
<tr>
<th>Table 1. Comparison between 2913/14 Combochip and the 2910A/11A/12A Single-Chip Codecs and Filters</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Features</strong></td>
</tr>
<tr>
<td>Power</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>Pins</td>
</tr>
<tr>
<td>Board Area Including Interconnects</td>
</tr>
<tr>
<td>Data Rates</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>Companding Law</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>PSRR</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>Gain Setting</td>
</tr>
<tr>
<td>Operating Modes</td>
</tr>
<tr>
<td>Timeslot Assign</td>
</tr>
<tr>
<td>On-Chip V&lt;sub&gt;REF&lt;/sub&gt;</td>
</tr>
<tr>
<td>ICN — Half Channel Improvement</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>S/D — Half Channel Improvement</td>
</tr>
<tr>
<td>GT — Half Channel Improvement</td>
</tr>
<tr>
<td>Power Down (Standby)</td>
</tr>
<tr>
<td>Signalling</td>
</tr>
<tr>
<td>Auto Zero</td>
</tr>
<tr>
<td>S &amp; H Caps</td>
</tr>
<tr>
<td>Test Modes</td>
</tr>
<tr>
<td>Encoder Implementation</td>
</tr>
<tr>
<td>Filter/Gain Trim</td>
</tr>
</tbody>
</table>
(a) Combochip Block Diagram

- **V_{BB}**: Power (-5V)
- **PWRO+, PWRO-**: Power Amplifier Outputs
- **GS_{R}**: Receive Gain Control
- **PDN**: Power Down Select
- **CLKSEL**: Master Clock Frequency
- **LOOP**: Analog Loop Back
- **SIG_{R}**: Receive Signaling Bit Output
- **DCLK_{R}**: Receive Variable Data Clock
- **D_{R}**: Receive PCM Input
- **F_{S_{R}}**: Receive Frame
- **GRDD**: Digital Ground
- **V_{CC}**: Power (+5V)

- **V_{F_{X_{L}}, V_{F_{X_{I}}}}**: Analog Inputs
- **G_{S_{X}}**: Transmit Gain Control
- **GRDA**: Analog Ground
- **NC**: No Connect
- **ASEL**: \(\mu\)- or A-law Select
- **TS_{X}**: Timeslot Strobe/Buffer Enable
- **DCLK_{X}**: Transmit Variable Data Clock
- **D_{X}**: Transmit PCM Output
- **F_{S_{X}}**: Transmit Frame
- **CLK_{X}**: Synchronization Clock
- **CLK_{R}**: Transmit Master Clock
- **F_{S_{R}}**: Receive 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 VREF
  - 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 high-speed serial PCM data bus over which the combochip transmits and receives its digitized voice code words. 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 2.048 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-

![Figure 3. 2914 Half Channel Gain Tracking Performance Measurements for Both Sine and Noise Testing](image-url)
Figure 4. 2914 Half Channel Signal to Distortion Ratio (S/D) Performance Measurements for Both Sine and Noise Testing

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.

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.

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.
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-de 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 6. Typical Power Dissipation Per Line Using 2914 Combochip

<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>
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
Table 7. Gain Trim Budget for Codec/Filter Functions

<table>
<thead>
<tr>
<th>Device</th>
<th>Manufacturing Uncertainty (Initial)</th>
<th>$\Delta T$</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></td>
<td></td>
</tr>
<tr>
<td>2912</td>
<td>$\pm 0.05$</td>
<td>$\pm 0.05$</td>
<td>$\pm 0.3$ dB</td>
<td>0 dB</td>
</tr>
<tr>
<td>2914</td>
<td>$\pm 0.15$</td>
<td>$\pm 0.15$</td>
<td></td>
<td></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 combo-chip. 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 $SIF_r$, $D_x$, and $TS_x$ approximately two or three frames after power up. Until this time, $SIG_r$ is held low and the other two signals are in a tri-state mode. During this time, $SIG_x$ 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 $CLK_x$ is interrupted. If this should happen, both $D_x$ and $TS_x$ 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- version, Switching, and Transmission using the Intel 2910A/2911A codec and 2912 PCM filter) also describes the basics of using the fixed data rate mode for first-generation codecs and filters which is essentially the same as for the 2913/14 second-generation combo-chip.

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 $FS_X$ and receiving data on the falling edge of the clock which is in the middle of the $D_X$ 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 $FS_X$.

According to the 2913/14 data sheet, the frame sync/data enable $FS_X$ must precede the clock ($DCLK_X$) by at least $T_{SDA}$ 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 $FS_X$ to lag up to 80 ns the $DCLK_X$ rising edge and recognize it as the first clock pulse in a 2.048 MHz system.

Once $FS_X$ has reached $V_{IH}$ of about 2V, the $D_X$ output will remain in the tri-state high-impedance mode for
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_{dsr}, 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 $D_X$ 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 $VF_XI$ (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 $\sin(\omega T/2)/\omega T/2$ correction factor must be included.
Table 9. 2914 Test Functions and Control Inputs for the Design Test Modes

<table>
<thead>
<tr>
<th>Input</th>
<th>Pin Function (24-Pin)</th>
<th>Test Function</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Pin 9 DR</td>
<td>Pin 17 TSx/DCLKx</td>
</tr>
<tr>
<td>O–V&lt;sub&gt;CC&lt;/sub&gt;</td>
<td>O–V&lt;sub&gt;CC&lt;/sub&gt;</td>
<td>DCLKR</td>
</tr>
<tr>
<td>V&lt;sub&gt;BB&lt;/sub&gt;</td>
<td>O–V&lt;sub&gt;CC&lt;/sub&gt;</td>
<td>—</td>
</tr>
<tr>
<td>O–V&lt;sub&gt;CC&lt;/sub&gt;</td>
<td>V&lt;sub&gt;BB&lt;/sub&gt;</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

The input to the receive filter first passes through a sample and hold. This is necessary to simulate the sin (ωt/2)/ωt/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 V<sub>BB</sub> 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".

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. If a signal above 4 KHz is sampled, those samples appear exactly as if the signal was at a frequency mirror imaged about 4 KHz. Two examples include signals at 5 KHz and 7 KHz which will result in samples that look like signals of 5–8 KHz = 3 KHz and 7–8 KHz = 1 KHz, respectively.

Conversely, the sampling process produces replicas (aliasing) of the sampled signal around multiples of 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.

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

\[
\text{Sinc} \left( \frac{\omega T}{2} \right) = \frac{\sin \left( \frac{\omega T}{2} \right)}{\frac{\omega T}{2}}
\]

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 \( f_t \) will have a weighting of

\[
W = \left( \frac{8000}{\pi f_t} \right) \sin \left[ \frac{\pi f_t}{8000} \right]
\]

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...
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[π f₁/8000] 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>Test Freq.</th>
<th>Transmit</th>
<th>Receive</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Freq.</td>
<td>Input</td>
</tr>
<tr>
<td>1</td>
<td>200</td>
<td>200</td>
</tr>
<tr>
<td>2</td>
<td>300</td>
<td>300</td>
</tr>
<tr>
<td>3</td>
<td>3000</td>
<td>3000</td>
</tr>
<tr>
<td>4</td>
<td>3300</td>
<td>3300</td>
</tr>
<tr>
<td>5</td>
<td>3400</td>
<td>3400</td>
</tr>
<tr>
<td>6</td>
<td>4000</td>
<td>4000</td>
</tr>
<tr>
<td>7</td>
<td>4600</td>
<td>4600</td>
</tr>
<tr>
<td>8</td>
<td>7000</td>
<td>7000</td>
</tr>
<tr>
<td>9</td>
<td>10000</td>
<td>10000</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 V_CC (TTL high). The result is to disconnect VF_x1+ and VF_x1− from the external circuitry and to connect internally PWRO+ to VF_x1+, GS_r to PWRO−, and VF_x1− to GS_X (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**
## DOMESTIC SALES OFFICES

<table>
<thead>
<tr>
<th>STATE</th>
<th>CITY</th>
<th>ADDRESS</th>
<th>PHONE</th>
<th>FAX</th>
</tr>
</thead>
<tbody>
<tr>
<td>ALABAMA</td>
<td>Montgomery</td>
<td>telnet corp. 5015 bradford dr.</td>
<td>(205) 639-0350</td>
<td>(205) 637-2640</td>
</tr>
<tr>
<td>ARIZONA</td>
<td>Phoenix</td>
<td>telnet corp. 140 north 44th st</td>
<td>(602) 231-0838</td>
<td>(602) 344-3446</td>
</tr>
<tr>
<td>CALIFORNIA</td>
<td>San Francisco</td>
<td>telnet corp. 9565 chesapeake dr</td>
<td>(415) 290-1220</td>
<td>(415) 290-8097</td>
</tr>
<tr>
<td>GEORGIA</td>
<td>Atlanta</td>
<td>telnet corp. 20 technology park</td>
<td>(404) 449-0541</td>
<td>(404) 602-9762</td>
</tr>
<tr>
<td>ILLINOIS</td>
<td>Chicago</td>
<td>telnet corp. 9655 pursil road</td>
<td>(312) 875-8942</td>
<td>(312) 875-1114</td>
</tr>
<tr>
<td>INDIANA</td>
<td>Indianapolis</td>
<td>telnet corp. 1900 st andrews dr</td>
<td>(317) 875-0623</td>
<td>(317) 875-8938</td>
</tr>
<tr>
<td>IOWA</td>
<td>Des Moines</td>
<td>telnet corp. 1930 st andrews dr</td>
<td>(515) 393-5510</td>
<td>(515) 393-5510</td>
</tr>
<tr>
<td>KANSAS</td>
<td>Kansas City</td>
<td>telnet corp. 1000 nodak road</td>
<td>(913) 345-2727</td>
<td>(913) 345-2727</td>
</tr>
<tr>
<td>MARYLAND</td>
<td>Baltimore</td>
<td>telnet corp. 10010 junction dr</td>
<td>(704) 568-8966</td>
<td>(704) 535-2236</td>
</tr>
<tr>
<td>MASSACHUSETTS</td>
<td>Westford center</td>
<td>telnet corp. 3 carlsfield st 2nd floor</td>
<td>(508) 692-9960</td>
<td>(508) 692-9960</td>
</tr>
<tr>
<td>MICHIGAN</td>
<td>Lansing</td>
<td>telnet corp. 7071 orchard lake rd</td>
<td>(517) 481-8891</td>
<td>(517) 481-8891</td>
</tr>
<tr>
<td>MINNESOTA</td>
<td>Bloomington</td>
<td>telnet corp. 3000 w 50th st</td>
<td>(612) 425-6722</td>
<td>(612) 425-6722</td>
</tr>
<tr>
<td>MISSOURI</td>
<td>Kansas City</td>
<td>telnet corp. 4500 earth city expressway</td>
<td>(816) 740-0111</td>
<td>(816) 740-0266</td>
</tr>
<tr>
<td>NEW JERSEY</td>
<td>Westfield</td>
<td>telnet corp. 1010 junction dr</td>
<td>(973) 748-1444</td>
<td>(973) 748-1444</td>
</tr>
<tr>
<td>NEW YORK</td>
<td>New York</td>
<td>telnet corp. 2950 express dr</td>
<td>(914) 723-2561</td>
<td>(914) 723-2561</td>
</tr>
<tr>
<td>PENNSYLVANIA</td>
<td>Norcross</td>
<td>telnet corp. 12000 ford road</td>
<td>(412) 875-3860</td>
<td>(412) 875-3860</td>
</tr>
<tr>
<td>PUERTO RICO</td>
<td>San Juan</td>
<td>telnet corp. 7322 s.w. freeway</td>
<td>(713) 988-8886</td>
<td>(713) 988-8886</td>
</tr>
<tr>
<td>TEXAS</td>
<td>Dallas</td>
<td>telnet corp. 5801 executive center dr</td>
<td>(512) 875-3860</td>
<td>(512) 875-3860</td>
</tr>
<tr>
<td>VIRGINIA</td>
<td>Richmond</td>
<td>telnet corp. 1504 santa rosa rd</td>
<td>(804) 282-5668</td>
<td>(804) 282-5668</td>
</tr>
<tr>
<td>WASHINGTON</td>
<td>Seattle</td>
<td>telnet corp. 1515 5th ave</td>
<td>(206) 451-9555</td>
<td>(206) 451-9555</td>
</tr>
<tr>
<td>WISCONSIN</td>
<td>Brookfield</td>
<td>telnet corp. 350 executive dr</td>
<td>(414) 796-8097</td>
<td>(414) 796-8097</td>
</tr>
<tr>
<td>CANADA</td>
<td>British Columbia</td>
<td>Tel Intel Semiconductor of Canada, Ltd.</td>
<td>(604) 259-6324</td>
<td>(604) 259-6324</td>
</tr>
<tr>
<td>ONTARIO</td>
<td>Toronto</td>
<td>Tel Intel Semiconductor of Canada, Ltd.</td>
<td>(416) 795-7472</td>
<td>(416) 795-7472</td>
</tr>
<tr>
<td>QUEBEC</td>
<td>Montreal</td>
<td>Tel Intel Semiconductor of Canada, Ltd.</td>
<td>(418) 675-2105</td>
<td>(418) 675-2105</td>
</tr>
<tr>
<td>VIRGINIA</td>
<td>Richmond</td>
<td>Tel Intel Semiconductor of Canada, Ltd.</td>
<td>(804) 282-5668</td>
<td>(804) 282-5668</td>
</tr>
<tr>
<td>WASHINGTON</td>
<td>Seattle</td>
<td>Tel Intel Semiconductor of Canada, Ltd.</td>
<td>(206) 451-9555</td>
<td>(206) 451-9555</td>
</tr>
<tr>
<td>WISCONSIN</td>
<td>Brookfield</td>
<td>Tel Intel Semiconductor of Canada, Ltd.</td>
<td>(414) 796-8097</td>
<td>(414) 796-8097</td>
</tr>
<tr>
<td>CANADA</td>
<td>British Columbia</td>
<td>Tel Intel Semiconductor of Canada, Ltd.</td>
<td>(604) 259-6324</td>
<td>(604) 259-6324</td>
</tr>
<tr>
<td>ONTARIO</td>
<td>Toronto</td>
<td>Tel Intel Semiconductor of Canada, Ltd.</td>
<td>(416) 795-7472</td>
<td>(416) 795-7472</td>
</tr>
<tr>
<td>QUEBEC</td>
<td>Montreal</td>
<td>Tel Intel Semiconductor of Canada, Ltd.</td>
<td>(418) 675-2105</td>
<td>(418) 675-2105</td>
</tr>
</tbody>
</table>

*Sales and Service Office

* Field Application Location
DOMESTIC DISTRIBUTORS
ALABAMA

Arrow Electronics. Inc.
1015 Henderson Road
Huntsville 35805
Tel: (205) 637-6955
FAX: 205-751-1581

Hamilton/Avnet Computer
4930 I Corporate Drive
Huntsville 35805

tArrow Electronics, Inc.
2961 Dow Avenue
Tustin 92680

tWyle Distribution Group
17872 Cowan Avenue
Irvine 92714

tPioneerfTechnologies Group, Inc.
337 Northlake Blvd .. Suite 1000
Alta Monte Springs 32701

Tel: (714) 836-5422
TWX: 910-595-2660

Tel: (714) 663-9953
TWX: 910-371-7127

Tel: (407) 634-9090
FAX: 407-834-0865

tWyle Distribution Group
26677 W. Agoura Rd.

PioneerfTechnoloQies Group, Inc.
674 S. Military Trail
Deerfield Beach 33442

Hamilton/Avnet Computer
3170 Pullman Street
Costa Mesa 92626

Calabasas 91302
Tel: (616) 660-9000
TWX: 372-0232

Hamilton/Avnet Computer
13618 West 190th Street
Gardena 90248

COLORADO

Huntsville 35805
Tel: (205) 837-7210
FAX: 205-721-0356

Hamilton/Avnet Computer
4103 Northgate Blvd.
Sacramento 95834

Arrow Electronics, Inc.
7060 South Tucson Way
Englewood 801 t 2

~9~oSt~~~~area~~ve

Hamilton/Avnet Computer
4545 Viewridge Avenue
San Diego 92123

Tel: (205) 630-9526
FAX: (205) 830-9557

Hamilton/Avnet Computer
1175 Bordeaux Drive
Sunnyvale 94089

Pioneer/Technolagies Group, Inc.
4825 University Square
Huntsville 35805

Hamilton/Avnet Electronics
21150 Caljfa Street
Woodland Hills 91367

Hamiltan/Avnet Electronics
4940 Research Drive

Suite 120
HUntsville 35806

Tel: (205) 637-9300
FAX: 205-637-9356
ALASKA

HamiitonfAvnet Computer
1400 W. Benson Blvd., Suite 400
Anchorage 99503
ARIZONA

fArrow Electronics, Inc,
4134 E. Wood Street
Phoenix 85040
Tel: (602) 437-0750

1WX: 910-951-1550
Hamilton/Avnet Computer
30 South McKemy Avenue
Chandler 85226

tHamilton/Avnet Electronics
3170 Pullman Street
Costa Mesa 92626
Tel: (714) 641-4150

1WX: 910-595-2636
tHamiltonlAvnet Electronics
1175 Bordeaux Drive
Sunnyvale 94086
Tel: (406) ·743-3300
TWX: 910-339-9332

tHamilton/Avnet Electronics
4545 Ridgeview Avenue
San Diego 92123
Tel: (619) 571-7500
TWX: 910-595-2636

tHamiiton/Avnet Electronics
21150 Calila 51.

Hamilton/Avnet Computer

Woodland Hills 91376

90 South McKemy Road

Chandler 85226

Tel: (816) 594-0404
FAX: 816-594-6233

tHamilton/Avnet Electronics
505 S. Madison Drive
Tempe 85281

tHamilton/Avnet Electronics
10950 W. Washington Blvd.
Culver City 20230

Tel: (602) 231-5140

1WX: 910-950-0077

Tel: (213) 556-2456
TWX: 910-340-6364

Hamilton/Avnet Electronics
30 South McKemy
Chandler 85226

tHamiiton/Avnet Electronics
1361 B West 190th Street
Gardena 90248

Tel: (602) 961-6669
FAX: 602-961-4073

Tel: (213) 217-6700
TWX: 910-340-6364

Wyle Distribution Group
4141 E. Raymond
Phoenix 85040

tHamilton/Avnet Electronics
4103 Northgale Blvd.

Sacramento 95834

Tel: (602) 249-2232

Tel: (916) 920-3150

1WX: 910-371-2871
CALIFORNIA

Pioneer/Technologies Group, Inc.
134 Rio Robles
San Jose 95134

Arrow Commercial System Group
1502 Crocker Avenue

Tel: (406) 954-9100
FAX: 406-954-9113

Hayward 94544
Tel: (415) 489-5371
FAX: (415) 469-9393

Wyle Distribution Group

Arrow Commercial System Group
14242 Chambers Road
Tustin 92680
Tel: (714) 544-0200
FAX: (714) 731-6436

tArrow Electronics, Inc.
19748 Dearbom Street
Chatsworth 91311
Tel: (213) 701-7500

Tel: (303) 790-4444

Hamilton/Avnet Computer
9605 Maroon Circle, Ste. 200
Engelwood 80112
tHamHton/Avnet Electronics
9605 Maroon Circle
Suite 200
Englewood 80112
Tel: (303) 799-{1663
TWX: 910-935-0767

twyle Distribution Group
451 E. 124th Avenue
Thornton 80241

1WX: 910-493-2066

Tel: (916) 638-5282

tArrow Electronics, Inc.
9511 Ridgehaven Court
San Diego 92123

tWyle Distribution Group
9525 Chesapeake Drive

~:r Psi~o5~~~~;71

Tel: (404) 497-1300

1WX: 810-766-0439
Hamilton/Avnet Computer
5825 D. Peachtree Corners E.
Norcross 30092

Tel: (404) 447-7500
TWX: 610-766-0432

tArrow Electronics, Inc.
12 Beaumont Road
Wallingford 06492
Tel: (203) 265-7741
TWX: 710-476-0162

Hamitton/Avnet Computer
Commerce Industrial Park
Commerce Drive
Danbury 06810
tHamilton/Avnet Electronics
Commerce Industrial Park
Commerce Drive
Danbury 06610
Tel: (203) 797·2600
TWX: 710-456-9974

tPioneer/Slandard Electronics
t 12 Main Street
Norwalk 06851

Pioneer{Technologies Group. Inc.
3100 F Northwoods Place
Norcross 30071
Tel: (404) 448-1711
FAX: 404-446-6270
ILLINOIS

tArrow Electronics, Inc.
1140 W. Thorndale
Itasca 60143
Tel: (706) 250-0500
TWX: 706-250-0916

Hamilton/Avnet Computer
1130 Thorndale Avenue
Bensenville 60106

Tel: (203) 853-1515
FAX: 203-638-9901

tHamilton/Avnet Electronics
1130 Thorndale Avenue
Bensenville 60106

FLORIDA

Tel: (706) 860-7760
1WX: 708-860-6530

tArrow Electronics, Inc.
400 Fairway Drive
Suite 102
Deerfield Beach 33441

MTI Systems Sales
1100 W. Thorndale

Tel: (305) 429-8200
FAX: 305-426-3991

tArrow Electronics, Inc.
37 Skyline Drive
Suite 3101

6601 NoW. 151h Way

tWyle Distribution Group
2951 Sunrise Blvd., Suite 175
Rancho Cordova 95742

tArrow Electronics, Inc.
4250 E. Rivergreen Parkylay
Deluth 30136 .

CONNECTICUT

124 Maryland Street

Garden Grove 92641

Delulh 30139
Tel: (404) 623-6625
FAX: (404) 623-8602

tHamilton/Avnet Electronics
5825 0 Peachtree Corners
Norcross 30092

~~~(~f~)dg~~o
~~~ g:~~U::~l~;~UP

GEORGIA

Arrow Commercial System Group
3400 C. Corporate Way

Tel: (303) 457-9953
TWX: 910-936-0770

Lake Marv 32746
Tel: (407) 323-0252
FAX: 407-323-3169

Tel: (714) 891-1717
FAX: 714-691-1621

Tel: (305) 426-6677
FAX: 305-481-2950

Hamilton/Avnet Computer
Ft. Lauderdale 33309
Hamilton/Avnet Computer
3247 Spring Forest Road
St. Peters~urg 33702
tHamilton/Avnef. Electronics
6801 NW. 15th Way
Ft. Lauderdale 33309
Tel: (305) 971-2900
FAX: 305-971-5420

tHamilton/Avnet Electronics
3197 Tech Drive North
SI. Petersburg 33702

Itasca 60143
Tel: (708) 773-2300

tPioneer/Standard Electronics
2171 Executive Dr., Suite 200
Addison 60101
Tel: (708) 495-9660
FAX: 706-495-9631
INDIANA

tArrow Electronics, Inc.
7108 Lakeview Parkway West Drive
Indianapolis 46268
Tel: (317) 299-2071
FAX: 317-299-{1255

Hamilton/Avnet Computer
485 Gradle Drive
Carmel 46032
Hamilton/Avnet Electronics
485 Gradle Drive
Carmel 46032
Tel: (317) 644-9333
FAX: 317-844-5921

Tel: (619) 565-4600
FAX: 619-279-6062

1WX: 910-335-1590

Tel: (613) 573-3930
FAX: 613-572-4329

tArrow Electronics, Inc.
521 Weddell Drive
Sunnyvale 94086

tWyle Distribution Group
3000 Bowers Avenue
Santa Clara 95051

tHamiiton/Avnet Electronics
6947 University Boulevard
Winter Park 32792

tPioneer/Standard Electronics
9350 Priority Way
West Drive
Indianapolis 46250

Tel: (406) 745-6600
TWX: 910-339-9371

Tel: (406) 727-2500

1WX: 406-966-2747

Tel: (407) 626-3666
FAX: 407-676-1676

Tel: (317) 573-0680
FAX: 317-573-{1979

tCertified Technical Distributor


DOMESTIC DISTRIBUTORS (Contd.)

IOWA
Hamilton Avnet Computer 913 33rd Avenue S Cedar Rapids 52404
Hamilton Avnet Electronics 913 33rd Avenue S Cedar Rapids 52404 Tel: (515) 362-4757

KANSAS
Arrow Electronics, Inc. 8208 Melrose Dr., Suite 210 Lenexa 66214 Tel: (913) 541-0942 FAX: 913-541-0328
Hamilton/Avnet Computer 15313 W. 95th Street Lenexa 66219 Tel: (913) 899-8500 FAX: 913-541-7951

KENTUCKY
Hamilton/Avnet Electronics 803 A. Newtown Church Lexington 40501 Tel: (859) 269-1470

MARYLAND
Arrow Electronics, Inc. 8300 Guilford Drive Suite H, River Center Columbia 21046 Tel: (301) 955-6002 FAX: 301-381-3904
Hamilton/Avnet Computer 6822 Oak Hall Lane Columbia 21045
Hamilton/Avnet Electronics 6822 Oak Hall Lane Columbia 21045 Tel: (301) 955-3905 FAX: 301-959-2393
Mesa Technology Corp. 9720 Patuxent Woods Dr. Columbia 21046 Tel: (301) 290-8100 FAX: 301-290-6974
Pioneer Technologies Group, Inc. 9100 Gallher Road Gaithersburg 20877 Tel: (301) 951-0960 FAX: 301-921-4255

MASSACHUSETTS
Arrow Electronics, Inc. 25 Upton Dr. Wilmington 01887 Tel: (508) 658-0000 TWX: 710-838-6770
Hamilton/Avnet Computer 10 D Centennial Drive Peabody 01960
Hamilton/Avnet Electronics 100 Centennial Drive Peabody 01960 Tel: (508) 532-9838 FAX: 508-596-7802
Pioneer/Standard Electronics 44 Hanwell Avenue Lexington 02173 Tel: (617) 931-0020 FAX: 617-803-1547
Wyde Distribution Group 15 Third Avenue Burlington 01803 Tel: (617) 272-7300 FAX: 617-272-6899

MICHIGAN
Arrow Electronics, Inc. 19880 Hagarty Road Livonia 48152 Tel: (313) 665-4100 TWX: 810-223-8020
Hamilton Avnet Computer 2215 S.E. A-5 Grand Rapids 49508
Hamilton Avnet Computer 41650 Garden Rd., Ste. 100 Novi 48050
Hamilton Avnet Electronics 2159 25th Street S.E. Space A Grand Rapids 49508 Tel: (616) 243-8805 FAX: 616-696-1831
Hamilton/Avnet Electronics 41650 Garden Brook Novi 48050 Tel: (313) 347-4271 FAX: 313-347-4021
Pioneer/Standard Electronics 4505 Broadmoor S.E. Grand Rapids 49508 Tel: (616) 699-1890 FAX: 616-696-1831
Pioneer/Standard Electronics 13485 Stamford Livonia 48150 Tel: (313) 525-1800 FAX: 313-427-3729

MINNESOTA
Arrow Electronics, Inc. 5200 W. 73rd Street Edina 55435 Tel: (612) 829-1800 TWX: 910-576-3125
Hamilton/Avnet Computer 12400 Whitewater Drive Minnetonka 55343
Hamilton/Avnet Electronics 12400 Whitewater Drive Minnetonka 55343 Tel: (612) 323-0900 TWX: 910-576-2730
Pioneer/Standard Electronics 7625 Golden Triangle Dr. Suite G Eden Prairie 55343 Tel: (612) 344-3355 FAX: 612-344-3794

MISSOURI
Arrow Electronics, Inc. 2300 Schulte St. Louis 63114 Tel: (314) 567-1104 FAX: 314-567-1114
Hamilton/Avnet Computer 739 Goddard Avenue Chesterfield 63005
Hamilton/Avnet Electronics 741 Goddard Chesterfield 63005 Tel: (314) 537-1600 FAX: 314-537-4248

NEW HAMPSHIRE
Hamilton/Avnet Computer 2 Executive Park Drive Bedford 03102
Hamilton/Avnet Computer 444 East Industrial Park Dr. Manchester 03103

NEW JERSEY
Arrow Electronics, Inc. 14 East Stow Road Unit 11 Marlton 08053 Tel: (609) 589-8000 FAX: 609-589-8632
Arrow Electronics 6 Century Drive Parsippany 07054 Tel: (201) 538-0900 FAX: 201-538-0900
Hamilton/Avnet Computer 1 Keystone Ave., Bldg. 36 Cherry Hill 08003

NEW YORK
Arrow Electronics, Inc. 3375 Brighton Henrietta Townline Rd. Rochester 14623 Tel: (716) 427-0300 TWX: 510-253-4766
Arrow Electronics, Inc. 20 Oser Avenue Hauppauge 11789 Tel: (516) 261-1000 TWX: 510-227-6623
Hamilton/Avnet Computer 933 Motor Parkway Hauppauge 11786
Hamilton/Avnet Computer 2060 Townline Rochester 14623
Hamilton/Avnet Electronics 933 Motor Parkway Hauppauge 11786 Tel: (516) 226-1800 TWX: 510-226-6166
Hamilton/Avnet Electronics 2060 Townline Road Rochester 14623 Tel: (716) 226-2744 TWX: 510-253-5470
Hamilton/Avnet Electronics 103 Twin Oaks Drive Syracuse 13206 Tel: (315) 437-9288 TWX: 710-541-1560

NORTH CAROLINA
Arrow Electronics, Inc. 5240 Greensboro Road Raleigh 27604 Tel: (919) 876-3132 TWX: 510-529-1856
Hamilton/Avnet Computer 3510 Spring Forest Road Raleigh 27604
Hamilton/Avnet Electronics 3510 Spring Forest Drive Raleigh 27604 Tel: (919) 876-0819 TWX: 510-928-1826
Pioneer/Technologies Group, Inc. 9401 L-Southern Pine Blvd. Charlotte 28210 Tel: (919) 527-8188 FAX: 704-522-8564

OHIO
Arrow Commercial System Group 284 Cramer Creek Road Dublin 43017 Tel: (614) 889-9347 FAX: 614-889-9860
Arrow Electronics, Inc. 6238 Cochran Road Solon 44139 Tel: (216) 248-3990 TWX: 810-427-9490
Hamilton/Avnet Computer 7764 Washington Village Dr. Dayton 45459
Hamilton/Avnet Computer 30253 Bainbridge Rd., Bldg. A Sonora 44319
Hamilton/Avnet Electronics 7760 Washington Village Dr. Dayton 45459 Tel: (513) 439-6733 FAX: 513-439-6711
Hamilton/Avnet Electronics 30253 Bainbridge Solon 44139 Tel: (216) 349-5100 TWX: 810-427-9452
Hamilton/Avnet Computer 777 Brookside Blvd. Westerville 43081 Tel: (614) 882-7004 FAX: 614-882-8650
Hamilton/Avnet Electronics 777 Brookside Blvd. Westerville 43081 Tel: (614) 882-7004
Hamilton/Avnet Electronics 23400 Commerce Park Road Brecksville 44141 Tel: (216) 464-6688
Pioneer/Standard Electronics 4433 Interpoint Boulevard Dayton 45424 Tel: (513) 236-9960 FAX: 513-236-8133
Pioneer/Standard Electronics 4900 E. 131st Street Cleveland 44104 Tel: (216) 587-3600 FAX: 216-963-1004

1Certified Technical Distributor
DOMESTIC DISTRIBUTORS (Contd.)

OKLAHOMA
Arrow Electronics, Inc.
4719 South Memorial Dr.
Tulsa 74145
1Hamilton/Avnet Electronics
12151 E. 51st St., Suite 102A
Tulsa 74146
Tel: (918) 252-7297

OREGON
1Almac Electronics Corp.
1865 N.W. 168th Place
Beaverton 97005
Tel: (503) 631-6000
FAX: 503-645-0611
Hamilton/Avnet Computer
9409 Southwest Nimbus Ave.
Beaverton 97005
1Hamilton/Avnet Electronics
9409 S.W. Nimbus Ave.
Beaverton 97005
Tel: (503) 627-0201
FAX: 503-641-4012
Wyle
9640 Sunshine Court
Bldg. G, Suite 200
Beaverton 97005
Tel: (503) 643-7900
FAX: 503-646-5466

PENNSYLVANIA
Arrow Electronics, Inc.
650 Seco Road
Montville 07827
Tel: (201) 396-7000
Hamilton/Avnet Computer
2800 Liberty Ave., Bldg. E
Pittsburgh 15222
Hamilton/Avnet Electronics
2800 Liberty Ave., Pittsburgh 15228
Tel: (412) 281-4150
Pioneer/Standard Electronics
259 Kappa Drive
Pittsburgh 15238
Tel: (412) 782-2300
FAX: 412-963-8255
1Pioneer/Technologies Group, Inc.
261 Gibralter Road
Horsham 19044
Tel: (215) 674-4000
FAX: 215-674-3107

TENNESSEE
Arrow Commercial System Group
3630 Knight Road
Memphis 38118
Tel: (901) 397-0540
FAX: (901) 397-0981

TEXAS
Arrow Electronics, Inc.
3220 Commander Drive
Carrollton 75006
Tel: (214) 380-6484
FAX: (214) 248-7208

1Certified Technical Distributor

Hamilton/Avnet Computer
1807A West Braier Lane
Austin 78758
Hamilton/Avnet Computer
Forum 2
4004 Beltline, Suite 200
Dallas 75244
Hamilton/Avnet Computer
4850 Wright Rd., Suite 190
Stafford 77477
1Hamilton/Avnet Electronics
1807 W. Braker Lane
Austin 78758
Tel: (512) 837-8201
FAX: 916-874-1319
1Hamilton/Avnet Electronics
4004 Beltline, Suite 200
Dallas 75234
Tel: (214) 308-1111
FAX: 916-860-5852
1Hamilton/Avnet Electronics
4850 Wright Rd., Suite 190
Stafford 77477
Tel: (713) 240-7725
FAX: 916-881-5523
1Pioneer/Standard Electronics
1926-D Kamar
Austin 78758
Tel: (512) 885-4000
FAX: 512-833-9809
1Pioneer/Standard Electronics
13710 Omega Road
Dallas 75244
Tel: (214) 396-7200
FAX: 214-450-6419
1Pioneer/Standard Electronics
10530 Rockey Road
Houston 77099
Tel: (713) 495-4700
FAX: 713-495-6542
1Wyle Distribution Group
1810 Greenville Avenue
Richardson 75081
Tel: (214) 235-5953
FAX: 214-644-5064

UTAH
Hamilton/Avnet Computer
1585 West 2100 South
Salt Lake City 84119
1Hamilton/Avnet Electronics
1585 West 2100 South
Salt Lake City 84119
Tel: (801) 972-2800
FAX: 910-908-4018
1Wyle Distribution Group
1325 West 2200 South
Suite E
West Valley 84119
Tel: (801) 974-9933

WASHINGTON
1Almac Electronics Corp.
14380 S.E. Eastgate Way
Bellevue 98007
Tel: (206) 643-9992
FAX: 206-643-7909
Hamilton/Avnet Computer
17761 Northeast 78th Place
Redmond 98052
Hamilton/Avnet Electronics
17761 N.E. 78th Place
Redmond 98052
Tel: (206) 881-6837
FAX: 206-867-0159
Wyle Distribution Group
15285 N.E. 92nd Street
Redmond 98052
Tel: (206) 881-1150
FAX: 206-881-1567

WISCONSIN
Arrow Electronics, Inc.
200 N. Patrick Blvd., Ste. 100
Brookfield 53005
Tel: (414) 792-0150
FAX: 414-792-0158
Hamilton/Avnet Computer
20875 Crossroads Circle
Suite 400
Wakesha 53166
1Hamilton/Avnet Electronics
29875 Crossroads Circle
Suite 400
Wakesha 53166
Tel: (414) 794-4010
FAX: 414-794-9509

CANADA
ALBERTA
Hamilton/Avnet Computer
2816 21st Street Northeast
Calgary T2E 6Z2
Hamilton/Avnet Electronics
2816 21st Street N.E. #3
Calgary T2E 6Z3
Tel: (403) 220-3586
FAX: 403-250-1591
Zentronics
6815 #6 Street N.E.
Suite 100
Calgary T2E 7H4
Tel: (403) 258-8181
FAX: 403-295-8714

BRITISH COLUMBIA
1Hamilton/Avnet Electronics
8610 Commerce Ct.
Burnaby V5A 4N6
Tel: (604) 420-4101
FAX: 604-437-4712
Zentronics
108-11400 Bridgeport Road
Richmond V6X 1T2
Tel: (604) 272-5375
FAX: 604-272-2913

ONTARIO
Arrow Electronics, Inc.
36 Antares Dr., Unit 100
Nepean K2E 7W5
Tel: (613) 226-6903
FAX: 613-723-2016
1Arrow Electronics, Inc.
1000 Meyerside, Unit 2
Mississauga L5T 1M4
Tel: (416) 573-7769
FAX: 416-672-0249
Hamilton/Avnet Computer
Canada Systems Engineering Group
3688 Nathua Drive
Units 7 & 9
Mississauga L4V 1M5
Hamilton/Avnet Computer
3688 Nathua Drive
Units 9 & 10
Mississauga L4V 1M5
Hamilton/Avnet Computer
6845 Flexwood Road
Units 3 & 4-5
Mississauga L4T 1R2
1Hamilton/Avnet Electronics
5780 Ashburn Way
Nepean K2E 7J5
1Hamilton/Avnet Electronics
6945 Flexwood Road
Units 3 & 4-5
Mississauga L4T 1R2
Tel: (416) 577-7432
FAX: 416-677-0340
1Hamilton/Avnet Electronics
190 Colonnade Road
Nepean K2E 7K1
1Hamilton/Avnet Electronics
155 Colonnade Road
Unit 17
Nepean K2E 7K1
Tel: (613) 226-1170
FAX: 613-226-1164
1Zentronics
1555 Mayside Drive
Mississauga L4T 1G9
Tel: (416) 564-9630
FAX: 416-564-9320
1Zentronics
155 Colonnade Road
Unit 17
Nepean K2E 7K1
Tel: (613) 226-6780
FAX: 613-226-6352

QUEBEC
Arrow Electronics Inc.
1100 Rte. du Roy
Dorval H9P 2T5
Tel: (514) 421-7411
FAX: 514-421-7430
Arrow Electronics, Inc.
500 Boutique, St-Jean-Baptiste
Suite 280
Quebec G2E 5G9
Tel: (418) 871-7500
FAX: 418-671-6816
Hamilton/Avnet Computer
2795 Rue Halpern
St. Laurent H4S 1P8
1Hamilton/Avnet Electronics
2795 Halpern
St. Laurent H4S 7K1
Tel: (514) 335-1000
FAX: 514-335-2481
1Zentronics
520 McCauley
St. Laurant H4S 1N3
Tel: (514) 737-9700
FAX: 514-737-5212
EUROPEAN SALES OFFICES
FINLAND

ITALY

SWEDEN

UNITED KINGDOM

Intel Finland OY
Ruosilantie 2
00390 Helsinki
Tel: (358) 0 544 644
TLX: 123332

Intel Corporation Iialia S.p.A.
Milanofiori Palazzo E
20094 Assago
Milano
Tel: (39) (02) 89200950
TLX: 341286

Intel Sweden A.B.
Dalvagen 24
171 36 Solna
Tel: (46) 8 734 01 00
TLX: 12261

Intel Corporation (U.K.) Ltd.
Pipers Way
Swindon, Wiltshire SN3 lRJ
Tel: (44) (0793) 696000
TLX: 444447/8

NETHERLANDS

SWIT2ERLAND

WES, GERMANY

Intel Semiconductor B.V.
Postbus 84130
3099 CC Rotterdam
Tel: (31) 10.407.11.11
TLX: 22283

Intel Semiconductor A.G.
Zuerichstrasse
6165 Winkel-Rueti be; Zuerich
Tel: (41) 01/860 62 62
TLX: 825977

Intel GmbH
Dornacher Strasse 1
8016 Feldkirchen bei Muenchen
Tel: (49) 089/90992-0
FAX: (49) 089/904/3948

FRANCE
Intel Corporation SAR.L
1, Rue Edison-BP 303
78054 81. Quentin-en-Yvelines
Cedex
Tel: (33) (1) 30 57 7000
TLX: 699016
ISRAEL
Intel Semiconductor Ltd.
Atidim Industrial Park-Neve Share!

P.O. Box 43202
Tel-Aviv 61430
Tel: (972) 03-498080
TLX: 371215

Intel GmbH
Abraham Uncoln Strasse 16-18
6200 Wiesbaden
Tel: (49) 06121/7605-0
TLX: 4-186183
Intel GmbH
Zettachring lOA
7000 Stuttgart 80
Tel: (49) 0711/7287-280
TLX: 7-254826

SPAIN
Intel Iberia SA
Zurbaran, 28
28010 Madrid
Tel: (34) (1) 308.25.52
TLX: 46880

EUROPEAN DISTRIBUTORS/REPRESENTATIVES
AUSTRIA

IRELAND

NORWAY

TURKEY

Bacher Electronics G.m.b.H.
Rotenmuehlgasse 26
1120 Wien
Tel: (43) (0222) 83 56 46
TLX: 31532

Micro Marketing Ltd.
Glenageary Office Park
Glenageary
Co. Dublin
Tel: (21) (353) (01) 856288
FAX: (21) (353) (01) 857364
TLX: 31584

Nordisk Elektronikk (Norge) A/S
Postboks 123
Smedsvingen 4
1364 Hvalstad
Tel: (47) (02) 84 62 10
TLX: 77546

EMPA Ele'ctronic
Lindwurmstrasse 95A
8000 Muenchen 2
Tel: (49) 089/53 80 570
TLX: 528573

BELGIUM
Inefco Belgium SA
Av. des Croix de Guerre 94
1120 Bruxelfes

?10~60~~~~~!rnlaan, 94
Tel: (32) (02) 21601 60
TLX: 64475 or 22090
DENMARK
ITT-Multikomponent
Naverland 29
2600 Glostrup
Tel: (45) (0) 2 45 66 45
TLX: 33 355

FINLAND
OY Fintronic AB
Melkonkatu 24A
00210 Helsinki
Tel: (358) (0) 6926022
TLX: 124224

FRANCE
Almex
Zone industrielle d'Antony
48, rue de l'Aubepine
BP 102
92164 Antony cedex
Tel: (33) (1) 46 66 21 12
TUe 250067
Jermyn
60, rue des Gemeaux
Sllie 580
94653 Rungis Cedex
Tel: (33) (1) 49 78 49 78
TLX: 261585

~~~~o~9~~~ieres
4, avo Laurent-Cely
92606 Asnieres Cedex
Tel: (33) (1) 47 90 62 40
TLX: 611448
Tekelec-Airtronic
Cite des Bruyeres
Rue Carle Vernet - BP 2
92310Sevres
Tel: (33) (1) 45 34 75 35
TLX: 204552

ISRAEL
Eastronics Ltd.
11 Rozanis Street
P.O.B. 39300
Tel-Aviv 61392
Tel: (972) 03-475151
TLX: 33638

ITALY
Intesi
Divisione ITT Industries GmbH
Viale Milanofiori
Palazzo E/5

~~R9(g~g~~1~31

TLX: 311351

Lasi Elettronica S.pA
V. Ie Fulvio Testi, 126
20092 Cinisello Balsamo (MI)
Tel: (39) 0212440012
TLX: 352040
Telcom S.r.l.
Via M. Civilali 75
20148 Milano
Tel: (39) 0214049046
TLX: 335654
In Multicomponents
Viale Milanofiori E/5

~~R~g9')g~fs~1~31

TLX: 311351

Silverstar
Via Dei Gracchi 20
20146 Milano
Tel: (39) 02149961
TLX: 332189

NETHERLANDS
Koning en Hartman
Elektroteehniek B.V.
Energieweg 1
2627 AP Delft
Tel: (31) (1) 15/609906
TLX: 38250

fflX: 837931
UNITED KINGDOM

PORTUGAL
ATO Portugal LOA
Rua Dos Lusiados, 5 Sala B
1300 Lisboa
Tel: (35) (1) 64 80 91
TLX: 61562
Ditram
Avenida Miguel Bombarda, 133
1000 Lisboa
Tel: (35) (1) 54 53 13
TLX: 14182

SPAIN
ATD Electronica, SA
Plaza Ciudad de Viena, 6
28040 Madrid
Tel: (34) (1) 2344000
TLX: 42477

ITI-SESA
Calle Miguel Angel, 21-3
28010 Madrid
Tel: (34) (1) 419 09 57 '
TLX: 27461
Metrologia Iberica, SA
Clra. de Fuenearral, n.80
28100 Alcobendas (Madrid)
Tel: (34) (1) 653 86 11

SWEDEN
Nordisk Elektronik AB
Torshamnsgatan 39
Box 36
16493 Kista
Tel: (46) 08-03 46 30
TLX: 10547

SWIT2ERLAND
Industrade A.G.
Hertistrasse 31
8304 Wallisellen
Tel: (41) (01) 8328111
TLX: 56788

Rapid Recall, Ltd.
Rapid House
Oxford Road
High Wycombe
Buckinghamshire HP11 2EE
Tel: (44) (0494) 26271
FAX: (44) (0494) 21860

Accent Electronic Components Ltd.
Jubilee House-, Jubilee Road
Letchworth, Herts SG6 1QH
Tel: (44) (0462) 670011
FAX: (44) (0462) 682467

fflX: 826505
Bytech Components Ltd.
12A Cedarwood
Chineham Business fark
Crockford Lane
Basingstoke
Hanls RG24 OWD
Tel: (0256) 707107
FAX: 0256-707162
Conformix
Unit 5
AIM Business Centre
Dixons Hilt Road
Welham Green
South Hatfield
Herts AL9 7JE
Tel: (07072) 73282
FAX: (07072) 61678
Bytech Systems
3 The Western Centre
Western Road
Braeknell RGt2 tRW
Tel: (44) (0344) 55333
FAX: (44) (0344) 867270

fflX: 849624
Jermyn
Vestry Estate
Otford Road
Sevendaks
Kent TN 14 5EU
Tel: (44) (0732) 450144
FAX: (44) (0732) 451251

fflX: 95142
MMDUd.
3 Bennet Court
Bennet Road
Reading
Berkshire RG2 OQX
Tel: (44) (0734) 313232
FAX: (44) (0734) 313255

fflX: 846669

Rapid Recall, Ltd.
28 High Street
Nantwich
Cheshire CW5 5AS
Tel: (0270) 627505
FAX: (0270) 629883

fflX: 36329
WEST GERMANY
Electronic 2000 AG

~6~~%~~~~~~n

' :2
Tel: (49) 089/42001-0
TLX: 522561
ITT Multikomponent GmbH
Postfach 1265
Bahnhofstrasse 44
7141 Moeglingen
Tet: (49) 07141/4879
TLX: 7264472
Jermyn GmbH
1m Dachsstueck 9
6250 Limburg
Tel: (49) 06431/508-0
TLX: 415257-0
Metrologie GmbH
Meglingerstrasse 49
8000 Muenchen 71
Tel: (49) 089/78042-0
TLX: 5213189
Proelectron Vertriebs GmbH
Max Planck Strasse 1-3
6072 Dreieleh
Tel: (49) 06103/30434-3
TLX: 417903

YUGOSLAVIA
H.R. Microelectronics Corp.
2005 de la Cruz Blvd., Ste.223
Santa Clara, CA 95050
U_S.A.
Tel: (1) (408) 988-0286
TLX: 387452
Rapido Electronic Components
S.p.a.
Via C. Beccaria, 8
34133 Trieste
lIalia
Tel: (39) 040/360555
TLX: 460461


INTERNATIONAL SALES OFFICES

AUSTRALIA
Intel Australia Pty. Ltd.
Unit 13
Allambee Grove Business Park
25 Frenchs Forest Road East
Frenchs Forest, NSW, 2086
Tel: 61-9975-3200
FAX: 61-9975-3375

BRAZIL
Intel Semicondutores do Brasil LTDA
Av. Paulista, 1159-CJF 404/405
01311 - Sao Paulo - S.P.
Tel: 55-11-287-5899
TLX: 391153149 ISOB
FAX: 55-11-287-5119

CHINA/HONG KONG
Intel PRC Corporation
15/F, Office 1, Cabell Bldg.,
Jian Guo Men Wai Streetrength
Beijing, PRC
Tel: (1) 500-4550
TLX: 22847 INTEL CN
FAX: (1) 500-2500

Intel Semiconductor Ltd.*
10/F East Tower
Bond Center
Queen'sway, Central
Hong Kong
Tel: (852) 844-4555
FAX: (852) 866-1998

INTERNATIONAL DISTRIBUTORS/REPRESENTATIVES

ARGENTINA
Defays S.R.L.
Chacabuco, 90-6 Piso
1069-Buenos Aires
Tel: 54-1-334-7725
FAX: 54-1-334-1871

AUSTRALIA
Email Electronics
15-17 Hume Street
Huntingdale, 3166
Tel: 011-61-3-544-2984
TLX: AA 30835
FAX: 011-61-3-543-8179

NSD-Australia
205 Middlerborough Rd.
Box Hill, Victoria 3128
Tel: 03 8900970
FAX: 03 8900919

BRAZIL
Eletra Componentes
Rua Geraldo Faustino Gomes, 78
7 Andar
04975 - Sao Paulo - S.P.
Tel: 55-11-534-9641
TLX: 55-11-54593/54591
FAX: 55-11-534-9424

CHINA/HONG KONG
Novel Precision Machinery Co., Ltd.
Room 208 Trade Square
681 Cheung Sha Wan Road
Kowloon, Hong Kong
Tel: 852-360-5005
TWX: 32032 NVTTL HX
FAX: 852-725-3695

INDIA
Micronic Devices
Arun Complex
No. 65 D.V.G. Road
Basaavangudi
Bangalore 500 024
Tel: 080-312-500-03
011-81-812-611-365
TLX: 903945332 MDGS

Micronic Devices
No. 516 9th Floor
Swastik Chambers
Son, Trombay Road
Chembur
Bombay 400 071
TLX: 9531 171447 MDEV

Micronic Devices
25th, 1st Floor
Bada Bazaar Marg
Old Rajinder Nagar
New Delhi 110 060
Tel: 011-91-11-5723509
011-91-11-5899771

Micronic Devices
6-3-649/2/1A Dwarkapuri Colony
Hyderabad 500 032
Tel: 011-91-462-226748

S&S Corporation
1567 Kosco Road
San Jose, CA 95118
Tel: (408) 978-6216
FAX: 978-8025

JAPAN
Asahi Electronics Co. Ltd.
KMM Bldg., 2-14-1 Asano
Kusakari-ku
Kita-yushu-shi 802
Tel: 03-511-6471
FAX: 03-551-7861

CTC Components Systems Co., Ltd.
4-8-1 Dobashi, Miyaimae-ku
Kawasumi-ku, Kanazawa 213
Tel: 049-802-5121
FAX: 049-877-4266

Da Semicon Systems, Inc.
Flower Hill Shinnami Higashi-kan
1-23-9 Shinnami, Setagaya-ku
Tokyo 154
Tel: 03-439-1600
FAX: 03-439-1601

Okaya Kohi
2-4-18 Sakae
Naka-ku, Nagoya-shi
460
Tel: 052-204-2901
FAX: 052-204-2901

Okawa Semi-Conductor Corp.
158-4, Sannomiya-Dori, Kangnamku
Seoul 136-080
Tel: (82) 557-8039
FAX: (82) 557-8039

SAMSUNG ELECTRONICS
South Korea

KOREA
J-Tek Corporation
Dong Sung Bldg. 9/F
158-4, Sannomiya-Dori, Kangnamku
Seoul 136-080
Tel: (82) 557-8039
FAX: (82) 557-8039

MEXICO
SSB Electronics, Inc.
375 Palmerton Street, Bldg. 4, Suite A
Chula Vista, CA 92011
Tel: (619) 585-3253
FAX: 619-385-3253

Dicoep S.A.
Tooth St 366 Frac. Ind. San Antonio Azcapotzalco
C.P. 02700-Mexico, D.F.
Tel: 52-5-561-3211
FAX: 52-5-561-3211

Phil S.A. de C.V.
Foa. Villa esq. Acauco s/n
Cuenca—Moralo
Tel: 52-72-13-9412
FAX: 52-72-13-5333

NEW ZEALAND
Electronic Components Corporation
20 Olive Road
Penrose, Auckland
Tel: 09-64-591-155
FAX: 09-64-592-681

SINGAPORE
Electronic Resources Pte. Ltd.
17 Harvey Road
#03-01 Singapore 1336
Tel: (65) 253-8888
TWX: RS 65641 ERS
FAX: 659-3207

SOUTH AFRICA
Electronics Building Elements
178 Erasmus St. (off Watermeyle St.)
Meyerspark, Pretoria, 0184
Tel: 012-212-803-7692
FAX: 012-212-803-8294

TANZANIA
Micro Electronics Corporation
12th Floor, 335
255 Nanking East Road
Taipei, R.O.C.
Tel: (886) 2-7198419
FAX: 886-2-7107916

Acer Seritek Inc.
15th Floor, Section 2
Chien Kuo North Rd.
Taipei 10679 R.O.C.
Tel: 886-6-501-096
FAX: 886-2-501-0521

*Field Application Location
Microcommunications

Microcommunications is the application of microelectronic or silicon technology to the field of communications. Intel, long recognized as an established leader in the microcommunications market, has a broad range of components covering local area networks (LANs) and modems.

This handbook contains product data sheets, design and applications information for Intel's entire line of microcommunications products, including Intel's first 32-bit LAN coprocessor, the 82596; twisted-pair Ethernet* products; and modem chips for desktop and laptop PCs. Technical information on Intel's communications boards and supercomponents is also included. Application articles explore Intel's entire line of microcommunications products. Charts are included to show parameters and test conditions, layout principles and much more.

*Ethernet is a registered trademark of Xerox Corp.