GET–6174

GE-PAC® 3010/2
COMPUTER SYSTEM

CENTRAL PROCESSOR REFERENCE MANUAL

GENERAL ELECTRIC

REVISION RECORD

PUBLICATION: GET-6174, GE-PAC 3010/2 Central Processor Reference Manual

REVISION OR PAGE NO.	DATE	REVISION OR PAGE NO.	DATE
ii	4/72		
iv			
2-8			
2-12			
2-13/2-14			
4-22			
5-12			
5-13			
5-16			
5-17			
5-23			
5-49 Added			
5-50 Added			
5-51 Added			
5-52 Added			
5-53 Added			
5-54 Added			
5-55 Added			
A5-1			
A5-2			
2-2	10/72		
2-12			
4-24			
5-5			
5-34			

GET-6174
10/72
$14.50

GE-PAC® 3010/2
COMPUTER SYSTEM

CENTRAL PROCESSOR REFERENCE MANUAL

General Electric reserves the right to make changes in the equipment or software, and its characteristics or functions, at any time without notice.

TABLE OF CONTENTS

Chapter			Page
1	INTRODUCTION		1-1
	1.1	Major Features	1-1
	1.2	Core Memory Parity Option	1-2
	1.3	Memory Protect Option	1-2
	1.4	Input/Output Methods	1-2
		1.4.1 Multiplexor Channel	1-2
		1.4.2 Multiplexor Channel I/O Techniques	1-2
		1.4.3 Interleaved Data Channel (IDC)	1-2
		1.4.4 Direct Memory Access Port	1-2
	1.5	Software	1-3
	1.6	Process I/O	1-3
	1.7	Peripheral Devices	1-3
2	SYSTEM DESCRIPTION		2-1
	2.1	Introduction	2-1
	2.2	Elements of the System	2-1
		2.2.1 Processor	2-1
		2.2.2 Core Memory	2-3
		2.2.2.1 Memory Bus	2-4
		2.2.2.2 Selector Channel	2-4
		2.2.2.3 Customer Designed Direct Memory Access Channel	2-4
		2.2.3 Multiplexor Channel	2-4
		2.2.3.1 Multiplexor Channel I/O Techniques	2-5
		2.2.3.2 Interleaved Data Channel	2-5
	2.3	Processor Operation	2-5
		2.3.1 Program Status Words	2-5
		2.3.2 Instruction Execution	2-6
		2.3.3 Core Memory Allocation	2-9
	2.4	Interrupt System	2-9
		2.4.1 Interrupt Procedure	2-9
		2.4.2 Internal Interrupts	2-10
		2.4.2.1 Fixed-Point Divide Fault Interrupt	2-10
		2.4.2.2 Floating-Point Arithmetic Fault Interrupt	2-10
		2.4.2.3 Machine Malfunction Interrupt	2-10
		2.4.2.4 Illegal Instruction Interrupt	2-11
		2.4.2.5 Protect Mode Violation Interrupt	2-11
		2.4.2.6 Supervisor Call (SVC) Interrupt	2-11

TABLE OF CONTENTS

Chapter Page

2.4.3 Input/Output Control Interrupts 2-12
2.4.3.1 External Interrupt 2-12
2.4.3.2 Automatic I/O Service 2-12
2.4.3.3 Automatic I/O Termination Interrupt. 2-12
2.4.3.4 Automatic I/O Termination Queue Overflow Interrupt. . . 2-13
2.4.4 Special Interrupts 2-13
2.4.4.1 Console Interrupt 2-13
2.4.4.2 Memory Protect Interrupt 2-13
2.4.4.3 Eight Line Priority Interrupt Module Interrupts 2-13

3 DATA AND INSTRUCTION FORMATS AND STORAGE ADDRESSING 3-1

3.1 Introduction 3-1

3.2 Data Formats 3-1
3.2.1 Hexadecimal Notation 3-1
3.2.2 Fixed-Point Data 3-1
3.2.2.1 2's Complement Notation 3-1
3.2.3 Floating-Point Data 3-3
3.2.4 Logical Data 3-4

3.3 Instruction Formats 3-5

3.4 General Register Usage 3-6

3.5 Storage Addressing 3-7

4 INSTRUCTION REPERTOIRE 4-1

4.1 Introduction 4-1

4.2 Fixed-Point Load/Store Instructions 4-3
4.2.1 Load Halfword 4-4
4.2.2 Load Multiple 4-5
4.2.3 Store Halfword 4-5
4.2.4 Store Multiple 4-6

4.3 Fixed-Point Arithmetic Instructions 4-7
4.3.1 Add Halfword 4-8
4.3.2 Add with Carry Halfword 4-9
4.3.3 Subtract Halfword 4-10
4.3.4 Subtract with Carry Halfword 4-11
4.3.5 Compare Logical Halfword 4-12
4.3.6 Compare Halfword 4-13
4.3.7 Multiply Halfword 4-14
4.3.8 Multiply Halfword Unsigned 4-14
4.3.9 Divide Halfword 4-15

4.4 Logical Instructions 4-16
4.4.1 AND Halfword 4-17
4.4.2 OR Halfword 4-18
4.4.3 Exclusive OR Halfword 4-19
4.4.4 Test Halfword Immediate 4-20

TABLE OF CONTENTS

Chapter			Page
	4.5	Byte Handling Instructions	4-21
		4.5.1 Load Byte	4-22
		4.5.2 Store Byte	4-22
		4.5.3 Exchange Byte	4-23
		4.5.4 Compare Logical Byte	4-23
	4.6	Floating-Point Instructions	4-24
		4.6.1 Floating-Point Load	4-25
		4.6.2 Floating-Point Store	4-25
		4.6.3 Floating-Point Add	4-26
		4.6.4 Floating-Point Subtract	4-27
		4.6.5 Floating-Point Compare	4-28
		4.6.6 Floating-Point Multiply	4-29
		4.6.7 Floating-Point Divide	4-30
	4.7	Shift/Rotate Instructions	4-31
		4.7.1 Shift Left Logical	4-32
		4.7.2 Shift Right Logical	4-33
		4.7.3 Rotate Left Logical	4-34
		4.7.4 Rotate Right Logical	4-35
		4.7.5 Shift Left Arithmetic	4-36
		4.7.6 Shift Right Arithmetic	4-37
	4.8	Branch Instructions	4-38
		4.8.1 Branch on True Condition	4-39
		4.8.2 Branch on False Condition	4-40
		4.8.3 Branch on Index	4-41
		4.8.4 Branch and Link	4-42
	4.9	List Instructions	4-43
		4.9.1 Add to Top/Bottom of List	4-44
		4.9.2 Remove From Top/Bottom of List	4-45
	4.10	Input/Output Instructions	4-46
		4.10.1 Acknowledge Interrupt	4-47
		4.10.2 Sense Status	4-48
		4.10.3 Output Command	4-49
		4.10.4 Read Data	4-49
		4.10.5 Write Data	4-50
		4.10.6 Read Block	4-51
		4.10.7 Write Block	4-52
		4.10.8 Read Halfword	4-53
		4.10.9 Write Halfword	4-54
		4.10.10 Autoload	4-55
	4.11	System Control Instructions	4-56
		4.11.1 Load Program Status Word	4-56
		4.11.2 Exchange Program Status	4-57
		4.11.3 Simulate Interrupt	4-57
		4.11.4 Supervisor Call	4-58
5	INPUT/OUTPUT SYSTEM		5-1
	5.1	Introduction	5-1
	5.2	Systems Interface	5-1
		5.2.1 Multiplexor Channel	5-1

TABLE OF CONTENTS

Chapter			Page
	5.2.2	Interleaved Data Channel	5-4
	5.2.3	Selector Channel	5-5
5.3		Input/Output Instructions	5-9
	5.3.1	Read Data (RD) Instruction	5-9
	5.3.2	Read Halfword (RH) Instruction	5-11
	5.3.3	Write Data (WD) Instruction	5-14
	5.3.4	Write Halfword (WH) Instruction	5-16
	5.3.5	Sense Status (SS) Instruction	5-19
	5.3.6	Output Command (OC) Instruction	5-21
	5.3.7	Acknowledge Interrupt (AI) Instruction	5-22
	5.3.8	Read Block (RB) Instruction	5-24
	5.3.9	Write Block (WB) Instruction	5-26
	5.3.10	Autoload (AL) Instruction	5-28
5.4		Device Controller Logic Design	5-30
	5.4.1	Multiplexor Bus	5-30
	5.4.2	Device Controller Addressing	5-34
	5.4.3	Data and Status Input/Output	5-34
		5.4.3.1 Data	5-34
		5.4.3.2 Status	5-36
	5.4.4	Data and Command Output	5-38
		5.4.4.1 Data	5-38
		5.4.4.2 Command	5-38
	5.4.5	Interrupt Control	5-38
	5.4.6	Multiplexor Bus Wiring	5-41
	5.4.7	Multiplexor Channel Timing	5-41
	5.4.8	Typical Device Controller Interface	5-44
	5.4.9	Data Channel Interface Design	5-44
5.5		Automatic I/O	5-49
	5.5.1	Automatic I/O Service Pointer Table	5-49
	5.5.2	Interrupt Service Block	5-49
	5.5.3	Automatic I/O Termination Queue	5-49
	5.5.4	General Operation	5-49
	5.5.5	Function Words	5-51
	5.5.6	Initialization	5-51
	5.5.7	I/O Operation	5-52
	5.5.8	Termination	5-53
	5.5.9	Example of Channel I/O Programming	5-54
6		CORE MEMORY	6-1
6.1		Introduction	6-1
6.2		Core Memory Modules	6-2
6.3		Parity Option	6-2
6.4		Memory Bus	6-3
	6.4.1	Memory Bus Priority	6-4
	6.4.2	Memory Bus Interfacing	6-4
	6.4.3	Memory Bus Timing	6-7

TABLE OF CONTENTS

Chapter — Page

7 CONTROL CONSOLE 7-1

7.1 Introduction 7-1

7.2 Control Console Description 7-1

7.2.1 Key Operated Security Lock 7-1
7.2.2 Control Switches 7-1
7.2.3 Function Switches 7-2

7.3 Control Console Operating Procedures 7-3
7.3.1 Power Up 7-3
7.3.2 Power Down 7-5
7.3.3 Program Loading 7-5
7.3.4 Program Execution 7-6
7.3.5 Program Termination 7-7
7.3.6 Manually Initiated Memory Operations 7-7
7.3.6.1 Memory Read 7-7
7.3.6.2 Memory Write 7-7

7.4 Programming Considerations 7-8
7.4.1 Control Console I/O 7-8
7.4.2 Console Interrupt 7-9
7.4.3 Wait State 7-9
7.4.4 Power Fail 7-9

Appendix

1 INSTRUCTION SUMMARY - ALPHABETICAL A1-1

2 INSTRUCTION SUMMARY - NUMERICAL A2-1

3 EXTENDED BRANCH MNEMONICS A3-1

4 OP CODE MAP A4-1

5 INSTRUCTION EXECUTION TIMES A5-1

6 ARITHMETIC REFERENCES A6-1

7 I/O AUTOMATIC SERVICE OPERATION AND TIMING DATA A7-1

8 I/O REFERENCES A8-1

CHAPTER 1
INTRODUCTION

The GE-PAC* 3010/2 process computer system is a flexible, modular process control system based upon a powerful and reliable Central Processor. A GE-PAC 3010/2 system may implement a full line of process input and output interfaces and peripheral devices. The Central Processor includes a Processor, a Core Memory, and an input/output Multiplexor Channel and Bus. The Central Processor is normally installed in a Central Systems Cabinet (CSC).

The advanced circuitry and packaging utilized in the system and the Central Processor provide an optimized price/performance ratio. The GE-PAC 3010/2 system is upwardly compatible with other GE-PAC computer systems, including the GE-PAC 30-1, 30-2, 30-2E and 3010. The user instruction sets, interrupt handling scheme, input and output formats, and control sequences are similar in all of these machines. Programs which run on these earlier systems will run on 3010/2 systems with no changes or with only minor changes. Absolute upward program compatibility from the 3010 to the 3010/2 is maintained.

The basic Central Processor consists of only five printed wire boards which provide the control logic, a read-only memory (ROM) and 8,192 bytes of Core Memory. This simple Central Processor has a functional capability similar to much larger and more complex computers because the non-volatile ROM allows great simplification of the Processor logic, by executing micro-instruction subroutines, which in turn, emulate the user instructions, and execute other valuable functions simplifying I/O and housekeeping operations.

Core Memory is addressable at the 8-bit byte or 16-bit halfword level. This memory is expandable from the basic 8,192 bytes to 65,536 bytes. All of core is directly addressable with the user instructions, and no paging or indirect addressing is required.

The Processor hardware includes sixteen 16-bit General Registers which may be used as accumulators. Fifteen of the General Registers may also be used as index registers. Several register to register user instructions permit direct register-to-register operations, eliminating redundant loading and storing in Core Memory.

Up to four Direct Memory Access Channels may be implemented in the Central Processor. These channels operate over a common Core Memory bus on a cycle stealing basis, through a Direct Memory Access Port, which is built into the Processor. Two types of Direct Memory Access Channels may be implemented: The first is the Selector Channel, which permits direct data transfers between any standard device controller and core. The second direct channel may be custom designed for special applications.

1.1 MAJOR FEATURES

The user instruction set consists of 113 instructions including:

- Floating-Point Instructions
- Arithmetic and Logical Instructions
- Byte Processing Instructions
- Both Single-Word and Double-Word Shift Instructions
- List Processing Instructions
- Short Branch Instructions for local branches
- Single-Byte, Double-Byte, and Block I/O Instructions
- Simulate Interrupt Instruction, for sophisticated I/O control
- Supervisor Call Instruction, for communication with the operating system (RTMOS-30)

The instruction words are 16 bits and 32 bits in length. Data transfers are in the form of 8-bit bytes, 16-bit halfwords and 32-bit fullwords.

The Core Memory features a 1.0 microsecond cycle time and the access time is 300 nanoseconds. Because the Read-Only Memory is faster than in earlier GE-PAC mini-based computer systems, instruction execution times in this machine are considerably reduced.

*Registered Trademark of General Electric Company

The Processor may respond to up to 256 interrupt levels. The interrupt scheme and the micro-program are combined effectively to permit an Automatic I/O Service mode, wherein much of the overhead required for I/O service is handled by the Processor, simplifying programming and saving memory space.

The Central Processor includes a compact but complete Control Console which permits manual operation of the system by programming and maintenance personnel. The console permits reading from and writing in core, displays any of the General Purpose Registers, allows an operator to manually step through programs, and features a keyswitch to prohibit unauthorized tampering with the controls.

1.2 CORE MEMORY PARITY OPTION

Preservation of memory integrity is provided by a memory Parity option. During Write operations, an odd parity bit is generated, and is appended to each 16-bit halfword. When data is fetched from memory, parity is again calculated and compared with the parity bit originally stored during the Write operation. In the event the parity bit originally stored and the recalculated parity bit are different, the Processor is interrupted via the Machine Malfunction Interrupt.

1.3 MEMORY PROTECT OPTION

The preservation of specific programs and data stored in Core Memory is provided by the Memory Protect option. The Memory Protect option permits available memory to be divided into blocks of 512, 1, 024, or 2, 048 bytes. Any combination of from 1 up to 64 blocks may be selected for protection under program control. The Memory Protect option permits the Processor to read data from any core location, but to write data only into unprotected locations. Any attempt to write into a protected block of memory is aborted and the Memory Protect hardware generates an I/O interrupt as an indication to the Processor that an illegal write was attempted.

1.4 INPUT/OUTPUT METHODS

1.4.1 Multiplexor Channel

The Multiplexor Channel provides an extremely reliable Input/Output System capable of communicating directly with up to 255 peripheral devices. Data transfers over the Multiplexor Channel can be either 8-bit or 16-bit to accommodate both byte and halfword oriented devices.

1.4.2 Multiplexor Channel I/O Techniques

The flexible structure of the Multiplexor Channel provides a repertoire of built-in I/O techniques. These techniques provide an important benefit in the ease of use, since an appropriate I/O technique can be chosen and matched to the speed and transfer requirements for a variety of peripheral devices. The various Multiplexor Channel I/O techniques include: Program-Controlled I/O, Interrupt Driven I/O, Automatic I/O Service, and Burst Mode I/O.

1.4.3 Interleaved Data Channel (IDC)

The Interleaved Data Channel is a high speed autonomous memory port. It provides a convenient method for customer-designed device controllers, requiring direct access to memory, to transfer data at 440K bytes in the Burst Mode or 280K bytes in the Single Cycle Mode.

1.4.4 Direct Memory Access Port

The built-in Direct Memory Access Port accepts up to four DMA Channels. This port operates on a cycle-steal basis with core memory, allowing simultaneous processing and I/O data transfers. The maximum transfer rate over a DMA Channel is 2, 000, 000 bytes per second. The Direct Memory Access Port may be used through the Selector Channel, which permits block transfers between memory and up to 16 devices, or by interfacing special devices directly to the port.

1.5 SOFTWARE

The GE-PAC 3010/2 system features a full line of standard software packages to support the available functional subsystems. The Real-Time Operating System, RTMOS-30, supports all process I/O and peripheral device operations on a real-time multiprogrammed basis. RTMOS-30 operates in a protected mode, wherein the operating system is protected from user level programs by the Processor hardware, which implements a "privileged instruction" concept. The privileged instructions are the system control and I/O instruction subsets, which are reserved for use by the operating system. User level programs interface with the operating system via a subset of Supervisor Call instructions (SVC).

In addition, core areas occupied by RTMOS-30 and other critical programs and data may be protected if the Memory Protect Option is implemented (1.3).

The standard software includes both on-line and off-line loaders, assemblers, editors, and debugging programs. The standard software packages are described in the GE-PAC 30/3010 Programming Reference Manual, GET-6171.

A complete line of hardware test programs is also provided with 3010/2 systems. These programs are part of the maintenance documentation provided with each system.

1.6 PROCESS I/O

An Analog Input/Output Subsystem, which includes an Analog Input Scanner, Absolute Analog Output Voltages, and Variable Analog Output Current Pulses, is available. The Digital I/O Subsystem provides digital inputs, digital outputs, and pulse train outputs. All process I/O subsystems are described in detail in the 3010/2 General Description.

1.7 PERIPHERAL DEVICES

The peripheral device complement may include TermiNet* 300 Printers, teletypewriters, a paper tape reader, a card reader, a paper tape punch, video display terminals, and bulk memory units. Serial-bit communications interfaces, suitable for remote communications via digital data sets are available as is a High Speed Data Link, which provides a parallel 8-bit byte interface with another GE-PAC 3010 or 4000 series computer. Refer to the GE-PAC 3010/2 General Description for detailed descriptions of these devices.

* Trademark of General Electric Company

CHAPTER 2
SYSTEM DESCRIPTION

2.1 INTRODUCTION

The GE-PAC 3010/2 is a 16-bit halfword-oriented digital computer useful in a wide variety of applications such as scientific, industrial process control, business, data collection, communications, and general purpose processing. The Processor is modularly constructed using the latest Large Scale Integrated (LSI) Circuits and state of the art computer technology.

The GE-PAC 3010/2 provides the user with more computing power per dollar than any machine available today. The Processor uses 16- and 32-bit instructions for efficient coding and optimum core utilization.

The Processor is designed around highly reliable 4,096 by 16-bit core memory modules. An optional 4,096 by 17-bit parity core memory is available. Both types of memory modules may be intermixed for a total of 65,536 bytes of storage (eight modules).

A Direct Memory Access Port is built into the Memory System. Up to four Direct Memory Access (DMA) Channels may be plugged into the port. The DMA Channels cycle-steal memory from the Processor at a maximum rate of 2,000,000 bytes per second. Two kinds of DMA Channels can be used with the Memory System; a Selector Channel available from General Electric for use with standard peripheral device controllers, and a direct Memory Connection custom designed by the user for special applications.

Input/Output operations between the GE-PAC 3010/2 and the peripheral device controllers ordinarily occur over the Multiplexor Bus. Up to 255 device controllers may be connected to this bus. The Processor implements eight Input/Output instructions plus Read Block and Write Block Instructions for high speed data transfer over the Multiplexor Bus. The Multiplexor Bus is fully hardware plug compatible with all peripheral device controllers developed and field proven on the existing line of GE-PAC 3010 Processors.

A parallel, character buffered, Teletypewriter adapter is standard on the GE-PAC 3010/2 Processor. This provides a very economical method for interfacing to a Teletypewriter.

The Processor is presented in this chapter in block diagram form. A System block diagram is shown in Figure 2-1 and a general block diagram of the Processor is shown in Figure 2-2. The major elements of the system are described in the following paragraphs.

2.2 ELEMENTS OF THE SYSTEM

2.2.1 Processor

The various elements of the system are organized around the Processor, as shown in Figure 2.1. The Processor contains facilities for:

1. Sequencing of instructions in the required order.
2. Arithmetic and logical data processing.
3. Initiating or controlling communications with external devices.
4. Servicing interrupts.

The basic elements of the Processor are: a set of 16 General Registers, an Arithmetic and Logic unit, a Control unit, and connections to the Memory and I/O Buses. See Figure 2-2.

The Processor operates under the direction of the Control unit which has a micro-program contained in a Read-Only Memory (ROM). The micro-program is a sequence of micro-operations which fetches the Processor instructions, decodes them, and processes the operands located in the General Registers and core memory locations. Such micro-programs are often referred to as "firmware" because they operate in a plane between "hardware" and "software".

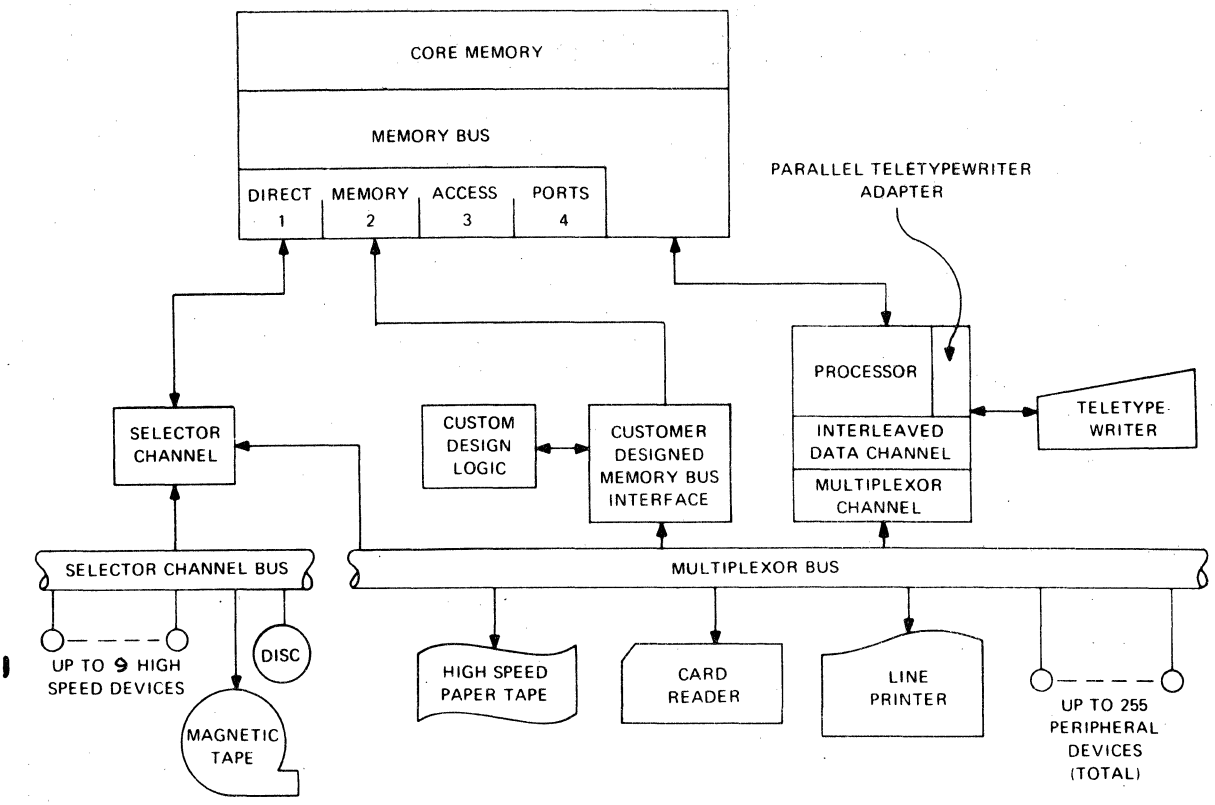


Figure 2-1. System Block Diagram

The instruction set of the GE-PAC 3010/2 is described in Chapter 4. All memory is directly addressable with the primary instruction word; no paging or indirect addressing is required. The 16-bit General Registers can be used as fixed-point accumulators, link registers for subroutine returns, or pointers for program branching. Of the 16 General Registers, 15 can be used as index registers for address modification.

The Protect Mode is enabled in the Processor under program control. In this mode, the Memory Protect is activated, and the Privileged instructions are detected and their execution is prevented. Privileged instructions are I/O instructions and any Control instruction whose execution could change the status of the system. In the Protect Mode, the execution of any Privileged instruction causes an Illegal Instruction Interrupt. Privileged instructions are discussed in detail in Chapter 4.

In general, fixed-point operations are performed upon one operand in a General Register, with the other operand in either a General Register or a core memory location.

Multiple-precision arithmetic operations are possible using two's complement representation, and by recognition of the carry/borrow from one operation to another.

The Standard Floating-Point instructions manipulate floating-point data using eight unique Floating-Point Registers which are resident in core memory.

The standard format for 32-bit single-precision floating-point data is identical to that used in the IBM System/360. This format represents numbers in the range of 5.4×10^{-79} to 7.2×10^{75}, with six digits of precision.

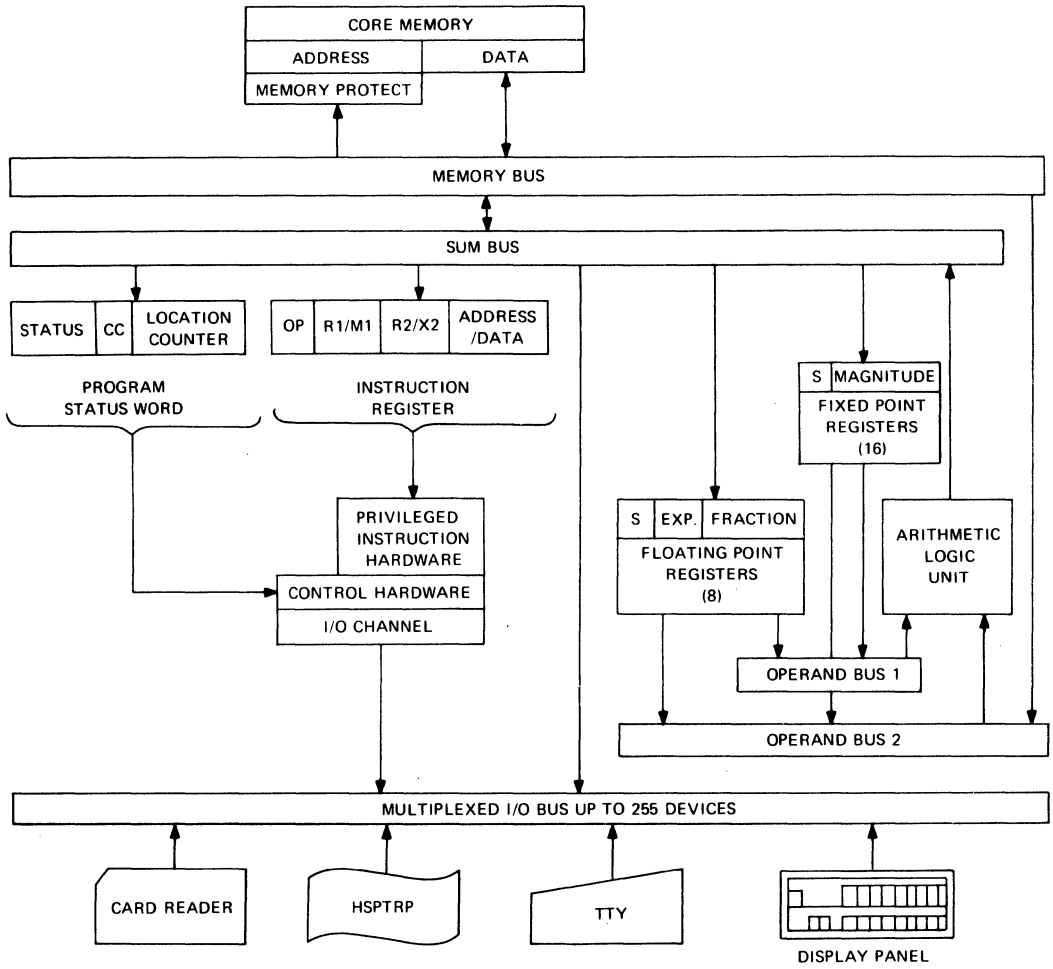


Figure 2-2. Processor Block Diagram

2.2.2 Core Memory

The GE-PAC 3010/2 can have from 8K to 65K bytes of Core Memory, where a byte refers to eight binary bits. The byte designation is used to describe memory size, since the memory reference instructions address memory at the byte level. The hardware memory modules are actually 16-bit oriented, and each Read or Write operation on the memory transfers 16-bits in one memory cycle. The maximum memory size, therefore, is 65,536/8-bit bytes or 32,768/16-bit halfwords.

When executing instructions, all 16-bit instructions and 16-bit data are handled in a single memory cycle. Multiple halfword data requires an additional memory cycle for each 16-bit halfword. Byte operations are performed by selectively manipulating the right or left 8-bits of the 16-bit halfword.

The Memory Write Control can be divided into two types. These two types have meaningful differences when used in conjunction with the Memory Protect. The first type is Standard Memory Write, the performance of which is subject to Memory Protect. That is, if the Processor is in the Protect Mode, any memory write into a protected area is inhibited, and a Memory Protect Controller Interrupt is generated. Memory, in this case is not altered. This type of memory write is used by all instructions which store data of any kind into memory. In Chapter 4, all instructions of this type are noted as being "subject to Memory Protect". Note that the Memory Protect does not affect reading from memory.

The other type of Memory Write Control, referred to as Privileged Write, overrides the Memory Protect circuit. This type of control is used by various internal Processor functions which must be allowed to write

into core memory, even in the Protect Mode. Examples of this operation are Processor access of dedicated registers in low core, and the register save sequence which is used on power failure. Other Privileged Write operations are noted where applicable in this manual.

In systems equipped with less than 65K bytes of memory, it is possible to address memory at an address greater than that of the last actual memory location. In this case, memory Read operations cause all zero data to be read, and Write operations proceed normally, but the data is lost.

2.2.2.1 Memory Bus

The Memory Bus provides the path for communication between the memory modules, the Processor, and up to four parallel Direct Memory Access Ports. The ports operate on a cycle stealing basis where the maximum latency time is one memory cycle or 1.0 microsecond. Memory service is granted on a priority basis where the Processor is always lowest in priority. The assigning of priority for a port is on a serial daisy-chain basis with the port closest to the Processor having the highest priority. The Selector Channel is used with standard device controllers for high speed devices.

2.2.2.2 Selector Channel

The Selector Channel is a Direct Memory Access device which provides high speed block-oriented data transfer at a rate of up to 2000K bytes. Up to 16 devices can be accommodated by the Selector Channel. Once initiated, the block transfer proceeds on a cycle stealing basis independent of the Processor. To initiate a data transfer, the Processor specifies the device address, the starting address in memory, the type of operation (Read or Write), and the ending address in memory. The data transfer is then completed without further direction or intervention by the Processor. Upon completion of the data transfer or termination due to a fault, the Processor is notified via a Selector Channel generated external interrupt.

2.2.2.3 Customer Designed Direct Memory Access Channel

Customers wishing to design their own Direct Memory Access Channel (DMAC) interfaces directly to the Memory Bus can do so with data transfer rates of up to 2,000,000 bytes per second in the Burst Mode through a custom-built DMAC. This is accomplished by using a general purpose wire wrapped circuit board available from General Electric. In addition to the circuits required for the Memory Bus and the user's device, his DMAC must include a 16-bit Memory Address Register and a 16-bit Memory Data Register, which are switched onto the Memory Bus when the DMAC captures the daisy-chain response for a request for service.

2.2.3 Multiplexor Channel

The Multiplexor Channel provides an Input/Output system for communicating directly with up to 255 peripheral device controllers. Operation over the Multiplexor Channel can be to either 8-bit byte devices or 16-bit halfword devices. The Multiplexor Channel consists of 30 lines:

- Sixteen bi-directional data lines for transferring either bytes or halfwords of data between the device and Processor.
- Eight control lines are used to identify the type of data transferred over the bi-directional data lines, such as status, command, address, or data.
- Five test lines are used to interrupt the Processor, and to synchronize the Input/Output system with the Processor.
- One initialize line is used to initalize the Input/Output system when power is turned On to the system.

The Multiplexor Channel operates on a request/response basis to allow simple reliable device controller design. Priority on the Multiplexor Channel is assigned on a party line basis where the device controller with the closest electrical proximity to the Processor has the highest priority. The Processor responds to an interrupt by swapping Program Status Words. The interrupt response time is 8.0 microseconds. The latency time for most instructions except Store Multiple and Load Multiple (which have a latency time of 20 microseconds) is 4 microseconds to 8 microseconds.

2.2.3.1 Multiplexor Channel I/O Techniques

The task-oriented structure of the Multiplexor Channel provides a repertoire of built-in I/O techniques. These techniques provide an important benefit in the ease of use of the GE-PAC 3010/2 since the appropriate I/O technique can be chosen and matched to the speed and transfer requirements of various devices. The Multiplexor Channel I/O techniques include:

Program-Controlled I/O Using normal programmed I/O, a program can interrogate the status of any device, and control the transfer of data (either 8-bit or 16-bit) to or from a device when the device indicates it is ready.

Interrupt Driven I/O Provides an interrupt facility with which any device can indicate a device "ready" condition to the Processor by interrupting the running program. Interrupt acknowledgements are achieved using "daisy chain" hardware logic, which avoids any programmed device polling to determine which device interrupted. Interrupt programming, therefore, can efficiently transfer data to or from multiple devices simultaneously.

Automatic I/O Interrupt programming is simplified by an Automatic I/O Service Mode, in which the Processor performs much of the overhead associated with each interrupt. This Automatic I/O Service Mode can be disabled under program control, which makes the I/O compatible with the other GE-PAC 30's and 3010's. With the Automatic I/O Service in use, the I/O Channel capability can also be used. The I/O Channels, which are commanded via specified Channel Control Blocks (CCB) in memory, perform signal counting or data transfers without interrupting the running program. When the I/O channel completes a specified sequence, a variety of automatic command chaining and interrupt queuing operations can take place.

Burst Mode I/O This incorporates Read Block/Write Block instructions. These instructions permit the Processor to achieve transfer rates in excess of 330K bytes by momentary dedication to a particular device.

2.2.3.2 Interleaved Data Channel

The Interleaved Data Channel (IDC) is an inexpensive direct to memory I/O system, which operates on an instruction steal basis, as compared with a memory cycle steal technique for the Direct Memory Access Channels. Data is transferred over the Multiplexor Channel autonomously with respect to the current program. Internal registers in the Processor are used as buffer registers between memory and the data channel device controller.

The Interleaved Data Channel controller is set up by the program over the Multiplexor Channel. Once initiated, the data transfer proceeds without further direction from the Program. For each Interleaved Data Channel cycle, the Processor inputs a 16-bit memory address from the Interleaved Data Channel controller along with a command to Read or Write. If the command is Write, the Processor reads the the memory location and outputs the 16-bit halfword of data to the Interleaved Data Channel controller. If the command is Read, the Processor inputs the 16-bit halfword of data and writes that data to memory.

The Interleaved Data Channel operates in the Burst Mode at 440K bytes, or at speeds up to 280K bytes in the Single Cycle Mode.

2.3 PROCESSOR OPERATION

2.3.1 Program Status Words

The focal point of control for the Processor is the Current Program Status Word (PSW). This 32-bit register contains the information required to direct program execution; a 12-bit Status field, a 4-bit Condition Code field, and a 16-bit Location Counter. See Figure 2-3.

The left half of the PSW defines Program Status, the right half is the Location Counter. The Current PSW controls instruction sequencing, and maintains the status of the system in relation to the program currently being executed. A program can change the Processor status by loading a New PSW. This is accomplished

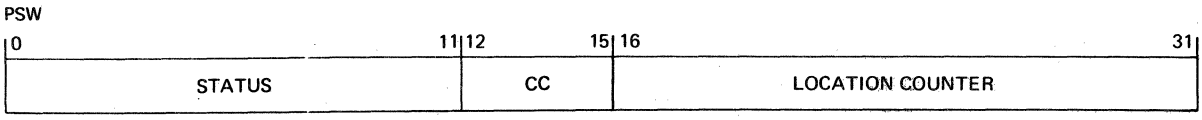


Figure 2-3. Program Status Word Format

by executing a Load Program Status Word (LPSW) or Exchange Program Status (EPSR) instruction. These instructions are described in Chapter 4.

The interrupt mechanism also involves the PSW. When an interrupt takes place, the Current PSW is stored at a unique four-byte location called the Old PSW. After the Current PSW is stored, the Current PSW register is loaded from another four-byte location called the New PSW. Each and every interrupt class has a unique set of Old and New PSWs. The PSW swap takes place automatically, and after the PSW swap, program execution will begin at the location specified by the Location Counter of the New PSW. The reserved core locations for Old and New PSWs for all interrupts are defined in Section 2.3.3.

The meaning of each bit in the left half of the PSW, Status and Condition Code, is explained in Table 2-1, and shown in Figure 2-4. The particular meaning or function of each bit applies when the bit is 1.

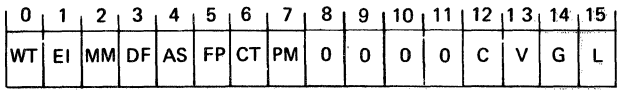


Figure 2-4. Program Status Bits

2.3.2 Instruction Execution

The 16-bit Location Counter field of the Program Status Word specifies the location of the next instruction to be fetched and processed. The 16-bit address field has the capability of directly addressing the maximum core memory of 65K bytes, or 32K halfwords.

Note that since instructions are aligned on halfword boundaries the value of the Location Counter must be even. That is, Bit 15 of the Location Counter must be zero.

During the normal processing of a program, an instruction is fetched from the location specified by the Location Counter, the instruction is executed, the Location Counter is incremented, and another fetch and execute cycle begins. After instruction execution, (except for Branch or Control instructions) the Location Counter is incremented by two if the executed instruction is of the halfword (RR or SF) 16-bit format, or by four if the executed instruction is of the fullword (RX or RS) 32-bit format.

Following Branch instructions or System Control instructions, the Location Counter is adjusted as a function of the particular instruction. See Section 4.8 for a summary of Branch instructions, and Section 4.11 for a summary of System Control instructions.

The sequencing of instructions during program execution is also changed if an interrupt occurs. In this case, the PSW swap procedure saves the Current PSW in core memory so that, after an interrupt is processed, execution can resume at the correct location.

TABLE 2-1. PROGRAM STATUS BIT DEFINITIONS

Bit	Name		Comments
0	WT	Wait State	The Wait bit is set to halt program execution. When this bit is set in the Current PSW, no program execution takes place, but the Processor will respond to all I/O and Machine Malfunction Interrupts, if they are enabled.
1	EI	External Interrupt Enable	The External Interrupt Enable bit is set to make the Processor responsive to interrupt signals from the Multiplexor Bus. External interrupts are discussed in detail in Section 2.4.3.1.
2	MM	Machine Malfunction Interrupt Enable	The Machine Malfunction Enable bit allows an interrupt to occur if a power fail is detected, if the machine is equippped with the Memory Parity Option and a memory parity error occurs, or during the restart process following a power down. See Section 2.4.2.3.
3	DF	Fixed Point Divide Fault Interrupt Enable	The Divide Fault Interrupt Enable bit allows the Processor to interrupt when a Fixed-Point Divide instruction is attempted and the result cannot be expressed in 16-bits. See Section 2.4.2.1.
4	AS	Automatic Input/ Output Service Enable	The Automatic I/O Service Enable bit allows the Processor to acknowledge I/O Interrupts and service them automatically as described in Section 2.4.3.2.
5	FP	Floating-Point Arithmetic Fault Interrupt Enable	The Floating-Point Arithmetic Fault Interrupt Enable bit allows the Processor to interrupt if exponent overflow or underflow occurs during any floating-point operation. See Section 2.4.2.2.
6	CT	Channel Termination Interrupt Enable	Channel Termination Interrupt Enable bit pertains to the Automatic I/O Channel, which can be used in conjunction with the Automatic I/O Service as described in Section 2.4.3.3.
7	PM	Protect Mode	The Protect Mode bit enables Memory Protect and detection of Privileged instructions. When the Protect Mode is not enabled the Processor is said to be in the Supervisor Mode. See Section 2.4.4.2.
8-11		Unused	Must be zero.
12	C	Carry/Borrow	The Condition Code bits are set or adjusted after the execution of instructions by the Processor. See Chapter 4 for details.
13	V	Overflow	
14	G	Greater than Zero	
15	L	Less than Zero	

TABLE 2-2
CORE MEMORY ALLOCATION

Function	Hexadecimal Memory Address	Assignment
Floating-Point Registers	00-03	Floating-Point Register, R0
	04-07	Floating-Point Register, R2
	08-0B	Floating-Point Register, R4
	0C-0F	Floating-Point Register, R6
	10-13	Floating-Point Register, R8
	14-17	Floating-Point Register, R10
	18-1B	Floating-Point Register, R12
	1C-1F	Floating-Point Register, R14
Power-Fail Locations	20-21	Unassigned
	22-23	Register Save Pointer
	24-27	Current PSW Save Area
Interrupt PSWs	28-2B	Old PSW FLPT Arithmetic Fault Interrupt
	2C-2F	New PSW FLPT Arithmetic Fault Interrupt
	30-33	Old PSW Illegal Instruction Interrupt
	34-37	New PSW Illegal Instruction Interrupt
	38-3B	Old PSW Machine Malfunction Interrupt
	3C-3F	New PSW Machine Malfunction Interrupt
	40-43	Old PSW External Interrupt
	44-47	New PSW External Interrupt
	48-4B	Old PSW Fixed-Point Divide Fault Interrupt
	4C-4F	New PSW Fixed-Point Divide Fault Interrupt
Reserved	50-7F	Bootstrap Loader and Device Definition Table
Automatic I/O Termination Parameters	80-81	Termination Queue Pointer
	82-85	**Old PSW Automatic I/O Termination Interrupt**
	86-89	**New PSW Automatic I/O Termination Interrupt**
	8A-8B	Overflow Termination Pointer
	8C-8F	Old PSW Termination Queue Overflow Interrupt
	90-93	New PSW Termination Queue Overflow Interrupt
Supervisor Call Parameters	94-95	Supervisor Call Argument Pointer
	96-99	Old PSW Supervisor Call
	9A-9B	New PSW (Status and Condition Code) Supervisor Call
	9C-9D	New PSW (Location Counter) Supervisor Call 0
	9E-9F	New PSW (Location Counter) Supervisor Call 1
	A0-A1	New PSW (Location Counter) Supervisor Call 2
	A2-A3	New PSW (Location Counter) Supervisor Call 3
	A4-A5	New PSW (Location Counter)Supervisor Call 4
	A6-A7	New PSW (Location Counter) Supervisor Call 5
	A8-A9	New PSW (Location Counter) Supervisor Call 6
	AA-AB	New PSW (Location Counter) Supervisor Call 7
	AC-AD	New PSW (Location Counter) Supervisor Call 8
	AE-AF	New PSW (Location Counter) Supervisor Call 9
	B0-B1	New PSW (Location Counter) Supervisor Call 10
	B2-B3	New PSW (Location Counter) Supervisor Call 11
	B4-B5	New PSW (Location Counter) Supervisor Call 12
	B6-B7	New PSW (Location Counter) Supervisor Call 13
	B8-B9	New PSW (Location Counter) Supervisor Call 14
	BA-BB	New PSW (Location Counter) Supervisor Call 15
	BC-CF	Reserved
Automatic I/O Service Table	D0-D1	Service Pointer, Device 0
	D2-D3	Service Pointer, Device 1
	D4-D5	Service Pointer, Device 2
	•	
	•	
	•	
	•	
	2CC-2CD	Service Pointer, Device 254
	2CE-2CF	Service Pointer, Device 255

2.3.3 Core Memory Allocation

The GE-PAC 3010/2 Processor requires certain locations in core memory for Floating-Point Registers, register save areas, and interrupt processing. These locations are defined in Table 2-2 and described in the following paragraphs.

Floating-Point Registers - These eight 32-bit registers are used by the Floating-Point instructions. See Section 4.6 for details of the Floating-Point instructions.

Power Fail Locations - The Register Save Pointer at location X'22', points to the first of 16 consecutive halfword locations in memory where the General Registers are saved in the event of power failure. When power is restored, the General Registers are restored automatically from these locations. The Current PSW is saved and restored in similar fashion from location X'24'-X'27'.

Interrupt PSWs - These locations are reserved for the Old and New PSWs for the various internal and external interrupts. These are discussed further in Sections 2.4.2 and 2.4.3.1 respectively.

Bootstrap Loader - The Bootstrap Loader is called the 50 sequence and is used to load the more sophisticated loaders.

Channel I/O Termination Parameters - These locations are used in conjunction with Termination interrupts from channel I/O operation. Refer to Sections 2.4.3.3 and 2.4.3.4 for details.

Supervisor Call Parameters - These locations are used for the PSW exchange associated with the Supervisor Call (SVC) instruction. This instruction is described in Section 4.11.4.

Automatic I/O Service Table - This table of 256 halfwords is used in the Automatic I/O Service Mode of operation. The Processor uses this table to uniquely service each interrupting device. See Section 2.4.3.2.

2.4 INTERRUPT SYSTEM

2.4.1 Interrupt Procedure

The Interrupt structure of the GE-PAC 3010/2 provides rapid response to internal and external events that require service by special software routines. In the interrupt response procedure, the Processor preserves the current state of the machine, and branches to the required service routine. The service routine may optionally restore the previous machine state upon completion of its service. The types of interrupts with their associated enable/disable PSW bits are listed in Table 2-3. Interrupts without a controlling PSW bit are always enabled.

Interrupts can occur at various times during processing. The Arithmetic Fault Interrupts occur during execution of user instructions. The Illegal instruction and Protect Mode Interrupt occur as soon as the offending instruction is recognized. The Supervisor Call Interrupt occurs as part of the execution of the Supervisor Call instruction. The Machine Malfunction and I/O Service Interrupts occur following instruction execution. The Channel Termination Interrupt can also occur during a Load Program Status Word or Exchange Program Status instruction.

The Interrupt Procedure is based on the concepts of Old, Current, and New Program Status Words. The Current PSW, contained in a hardware register, defines the operating status of the machine. When this status must be interrupted, the Current PSW becomes an Old PSW and is stored in a core location dedicated

TABLE 2-3. INTERRUPTS

Interrupt	PSW Control Bit
External	1
Machine Malfunction	2
Fixed Point Divide Fault	3
Automatic I/O Service	4
Floating-Point Arithmetic Fault	5
Channel Termination	6
Protect Mode	7
Illegal Instruction	Cannot be disabled
Channel Termination Queue Overflow	Cannot be disabled
Supervisor Call	Cannot be disabled

to the type of interrupt that has occurred. The New PSW becomes the Current PSW by being loaded from a dedicated core location into the hardware PSW Register. The status portion of the Current PSW now contains the operating status for the interrupt service routine, and the Location Counter points to the first instruction in the service routine. New Program Status Words for interrupts controlled by PSW bits should disable interrupts of their own class. Interrupts controlled by Bits 1 and 6 must disable interrupts of their own class to prevent the Processor from going into an endless loop. The dedicated core locations for Old and New Program Status Words are shown in Table 2-2 of Section 2.3.2. The Program Status Word exchange procedure does not change the contents of the New PSW location, and subsequent interrupts of the same type are treated in the same way.

2.4.2 Internal Interrupts

The GE-PAC 3010/2 can generate six Internal Interrupts. Of these, the Illegal instruction and the Supervisor Call cannot be inhibited. Inhibited Internal Interrupts are not queued.

2.4.2.1 Fixed-Point Divide Fault Interrupt

The Fixed-Point Divide Fault Interrupt, enabled by Bit 3 of the Program Status Word, is indicative of division by zero or quotient overflow. Quotient overflow is defined as quotient magnitude greater than $2^{15}-1$. The interrupt takes place before modification of the operand registers. After a Fixed-Point Divide Fault Interrupt, the Old PSW Location Counter points to the next instruction following the Divide instruction.

2.4.2.2 Floating-Point Arithmetic Fault Interrupt

The Floating-Point Arithmetic Fault Interrupt enabled by Bit 5 of the Current PSW, occurs on exponent overflow or underflow as well as on division by zero. In the case of division by zero, the interrupt takes place prior to alteration of the operand register. An exponent overflow sets the result to ±X'7FFF FFFF'. An exponent underflow sets the result to X'0000 0000'. The Location Counter of the Old PSW points to the next instruction. Refer to Section 4.6 for an explanation of Floating-Point instructions.

2.4.2.3 Machine Malfunction Interrupt

Bit 2 of the Current Program Status Word controls the Machine Malfunction Interrupt. This error can occur on either a primary power failure, a memory parity error, or during the restart process following a power shutdown. If the memory is equipped with the Parity Option, the parity bit of each memory word is set to maintain odd parity. This bit is recomputed during each memory read; if the computed bit is not equal to the bit read out of memory and if Bit 2 of the current PSW is set, the Current Program Status Word is stored at the Machine Malfunction Old PSW location, and the Current PSW is loaded from the Machine Malfunction New PSW

location. The Condition Code field of the Current PSW is then adjusted by setting the G flag (PSW 14) if the parity error occurred on instruction read, or setting the V flag (PSW 13) if the error occurred on an operand read. It is not possible to guarantee programmed recovery from a parity error.

> NOTE
>
> The Condition Code field of the Machine Malfunction New PSW location in memory must be zero.

If enabled by Bit 2 of the PSW, a Machine Malfunction Interrupt will occur on power fail. Power fail occurs when the optional Primary Power fail detector senses a low voltage, when the Initialize switch is depressed, or when the key-operated power switch is turned off. After the PSW swap, the L flag of the Current PSW is set. The software service subroutine for Machine Malfunction can distinguish power fail from parity errors by conditional branch instructions. The user is allowed approximately 1 millisecond before the system is shut down. On shutdown, the Processor stores the Current Program Status Word in locations X'0024' through X'0027', and the General Registers in the consecutive locations starting at the address contained in location X'0022'. When power is restored, the registers are reloaded and the Current PSW is restored from locations X'0024' through X'0027'. If Bit 2 (Machine Malfunction) of this PSW is set, the Processor makes a PSW exchange from the Machine Malfunction location. The software service routine for the Machine Malfunction Interrupt can differentiate between the memory parity error, power failure, and power up conditions by testing the condition code. Note that the V flag is set for parity error on operand fetch, the G flag is set for parity error on instruction fetch, the L flag is set on power fail, and no flags are set on power restore. Additionally, the Location Counter of the Machine Malfunction Old PSW, may be compared with the contents of locations X'0026' and X'0027' (Power Fail PSW save area). If they are equal, a power failure and restore sequence has occurred.

2.4.2.4 Illegal Instruction Interrupt

The Illegal Instruction Interrupt is not represented by an enabling bit in the PSW, and is therefore always operative. An illegal instruction is defined as an operation code that is not in the GE-PAC 3010/2 repertoire. No attempt is made to execute the illegal instruction, nor is the Location Counter of the Current PSW incremented. The Old PSW, stored as a result of an Illegal Instruction Interrupt, points to the address of the illegal instruction.

2.4.2.5 Protect Mode Violation Interrupt

The Protect Mode Violation Interrupt is enabled when Bit 7 of the Current PSW is set, which puts the Processor in the Protect Mode. The interrupt occurs, in this mode, when an attempt is made to execute a Privileged instruction. Privileged instructions are all I/O instructions, and System Control instructions: Load Program Status Word, Exchange Program Status, and Simulate Interrupt, which are described in Chapter 4. When such an instruction is attempted in this mode, the instruction is not executed, and the Illegal instruction Interrupt procedure takes place, as described above. The Location Counter is not incremented, so that the Old PSW points to the Privileged instruction that caused the interrupt. PSW Bit 7, when set, also enables Memory Protect.

2.4.2.6 Supervisor Call (SVC) Interrupt

This interrupt occurs as the result of an SVC instruction, which is used to communicate between running programs and operating systems. The Supervisor Call Interrupt is not inhibitable. When an SVC instruction is executed, the following action takes place:

1. Current PSW is stored at the Supervisor Call Old PSW location, location X'0096'.
2. The effective address from the SVC instruction is stored at the Supervisor Call argument pointer, location X'0094'.
3. The status portion of the Current PSW is loaded from the Supervisor Call New PSW Status location, location X'009A'.
4. The Current Location Counter is loaded from one of the Supervisor Call New PSW Location Counter locations. Refer to Section 4.11.4 for details on the Supervisor Call instruction.

2.4.3 Input/Output Control Interrupts

If individually enabled by the program, a peripheral device is allowed to request Processor service when the device itself is ready to transfer data via an interrupt. The Processor may respond to this signal in several ways depending on the setting of certain bits in the Program Status Word. The GE-PAC 3010/2 has two classes of interrupts directly related to peripheral device handling. These are External Interrupt and Immediate Interrupt. Two other classes, the Channel Termination Interrupt and the Channel Queue Overflow Interrupt can occur upon termination of a channel I/O sequence. PSW Bits 1 and 4, in combination, control the External and Immediate Interrupts.

If Bit 1 is reset, I/O Device Interrupt signals are ignored. The signal remains pending, however, until PSW Bit 1 is set and the signal is acknowledged. Bit 6 of PSW controls the Channel Termination Interrupt. The Channel Termination Queue Overflow Interrupt is always enabled.

2.4.3.1 External Interrupt

If Bit 1 of the Current PSW is set, and Bit 4 is reset, an I/O Device Interrupt signal results in the following action: The Current PSW is stored at the Input/Output Interrupt Old PSW location. The Current PSW is loaded from the Input/Output New PSW location. From this point, software must acknowledge the interrupt, identify the device, and take appropriate action. Note that the New PSW for External Interrupts must have Bit 1 reset.

2.4.3.2 Automatic I/O Service

If both Bit 1 and Bit 4 of the Current PSW are set, an interrupt signal from a peripheral device results in the following Automatic I/O Service. The signal is automatically acknowledged and the device number returned is used to index into the Automatic Service Pointer Table in locations X'00D0' to X'02CF'. See Table 2-2. The Service Pointer obtained is the address of either an Old PSW save area or a Channel Command Word for an Automatic I/O operation. If Bit 15 of the Service Pointer is reset, the Current PSW is stored in the fullword location whose address is contained in the Service Pointer Table. The halfword whose address is the contents of the Service Pointer Table plus four contains the New PSW Status and Condition Code fields. The Location Counter is set to a value equal to the contents of the Service Pointer Table plus six and instruction execution resumes.

Current PSW (0:31) ⟶ [(SERVICE POINTER)]
Current PSW (0:15) ⟵ [(SERVICE POINTER) + 4]
Current PSW (16:31) ⟵ (SERVICE POINTER) + 6

Through this Immediate Interrupt mechanism, a unique service routine for any device number can be automatically entered. Exit from the routine is made by executing a Load Program Status Word instruction specifying the Old PSW location (Service Pointer) at the origin of the subroutine.

If Bit 15 of the Service Pointer is set, the address contained is that of an Interrupt Service Block and indicates that Automatic I/O service is required. This Processor activity is described in Chapter 5.

2.4.3.3 Automatic I/O Termination Interrupt

The termination of an Automatic I/O operation may result in the storing of a termination pointer in the circular list located at the address specified by the Queue Pointer location. If at this time, Bit 6 of the Current PSW is set, the Current PSW is stored at the Automatic I/O Termination Old PSW location, and the Current PSW is loaded from the Automatic I/O Termination New PSW location. In this way, the control software is notified of the completion of a channel I/O operation. Whenever the Processor executes a Load Program Status Word instruction or an Exchange Program Status instruction, it checks Bit 6 of the newly loaded PSW. If Bit 6 of the loaded PSW is set, and there is an entry in the queue, this interrupt is taken. This is described in detail in Chapter 5.

2.4.3.4 Automatic I/O Termination Queue Overflow Interrupt

If the Processor attempts to enter an Automatic I/O Termination Pointer in the Termination Queue and the queue is already full, it stores the termination pointer at location X'008A', the Overflow Termination Pointer location; stores the Current PSW in location X'008C', the Queue Overflow Old PSW location; and loads the Current PSW from location X'90', the Queue Overflow New PSW location. This action allows the software to clear out the queue before any automatic I/O terminations are lost. This interrupt cannot be disabled.

2.4.4 Special Interrupts

The GE-PAC 3010/2 Processor provides two classes of special interrupts. These are the Console Interrupt and the Memory Protect Interrupt.

2.4.4.1 Console Interrupt

The GE-PAC 3010/2 provides for operator intervention in the following manner. If Bit 4 of the Current PSW is set, the rotary Function switch is in the OFF position, and the RUN Function switch is depressed, depressing the EXECUTE switch causes an interrupt signal from Device 01. Servicing this signal can be accomplished through the Immediate Interrupt or the Channel I/O service.

2.4.4.2 Memory Protect Interrupt

If Bit 7 (Protect Mode) of the Current PSW is set, the Processor is said to be in the Protect Mode. Should the program attempt to store into a protected core area (as defined by the mask in the Memory Protect Controller) while the Processor is in the Protect Mode, an Interrupt signal is generated by the Memory Protect Controller. Furthermore, if Bit 1 (External Interrupt Enable) of the Current PSW is set, this interrupt is recognized and appropriate action can be taken. Refer to Appendix 8 for details on the Memory Protect Controller. Note that if the External Interrupt is disabled, the Memory Protect Interrupt cannot occur, however, the store operation is ignored and execution resumes at the next instruction.

2.4.4.3 Eight Line Priority Interrupt Module Interrupts

An 8-Line Interrupt Module is available as a hardware option with the GE-PAC 3010/2. There are 8 levels of priority provided on each module with enable/disable control over each level or the entire module. Each of the 8 levels respond to the Multiplexor Channel with a unique device address for that level. The interrupt module hardware assures that higher priority levels respond before lower priority levels. The lowest priority bit in the enable/disable register can be optionally used to mask off all other lower-priority interrupt signals from other devices on the Multiplexor Channel. Refer to Appendix 8 for details on the 8-Line Interrupt Module.

CHAPTER 3
DATA AND INSTRUCTION FORMATS AND STORAGE ADDRESSING

3.1 INTRODUCTION

A program is a set of instructions which directs the Processor to perform a specific task. Ordinarily, program instructions are stored in sequential memory locations. During the normal processing of a Program, an instruction is fetched from the location specified by the Location Counter, the instruction is executed, the Location Counter is incremented, and another fetch and execute cycle begins.

3.2 DATA FORMATS

The Instruction Set manipulates data of three different word lengths: 8-bit bytes, 16-bit halfwords, and 32-bit fullwords. This data may represent a fixed-point number, a floating-point number, or logical data. The data is used as operands for the instructions, and is manipulated as indicated by the particular instruction being executed.

3.2.1 Hexadecimal Notation

Binary information is expressed in hexadecimal notation (base 16) for purposes of simplicity. All references to binary instructions, data, or addresses in 3 0 1 0 / 2 software are made in hexadecimal notation. Four binary bits of information are conveniently expressed by a single hexadecimal digit. Thus, byte information is expressed by two hexadecimal digits, halfword information by four hexadecimal digits, and fullword information by eight hexadecimal digits. Table 3-1 lists hexadecimal, binary and decimal equivalents.

3.2.2 Fixed-Point Data

The basic Fixed-Point Arithmetic operand is the 16-bit halfword. In multiply and divide operations, 32-bit fullwords are manipulated. See Figure 3-1.

Fixed-point data is treated as signed 15-bit integers in the halfword format, and as signed 31-bit integers in the fullword format. Positive numbers are expressed in true binary form with a sign bit of zero. Negative numbers are represented in two's complement form with a sign bit of one. Refer to Section 3.2.2.1 for further details.

The numerical value of zero is always represented with all bits zero. Table 3-2 shows several examples of the fixed-point number representation used in 3 0 1 0 / 2 Systems.

Since halfword arithmetic operands are 16-bit values, Fixed-Point Arithmetic instructions can be used for address arithmetic. Logical, and Shift instructions can also be used for address manipulation or computation.

For details on manipulating fixed-point quantities, refer to Section 4.3.

3.2.2.1 2's Complement Notation

Negative numbers are represented in 2's complement notation. A fixed-point number is negative only if Bit 0 is set. To change the sign of a number, the 2's complement of the number is produced in a two-step procedure:

1. Change all zeros to ones, and change all ones to zeros (complement every bit).
2. Add 1 to the number.

Example: The number five is represented in binary form as
0000 0000 0000 0101

Step 1. 1111 1111 1111 1010 (complement) = 1's complement of 5

Step 2. 1111 1111 1111 1011 (add one) = 2's complement of 5

The result is the 2's complement of 5, representing -5.

TABLE 3-1
HEXADECIMAL, BINARY, AND DECIMAL CROSS-REFERENCE

Hexadecimal	Binary	Decimal
0	0000	0
1	0001	1
2	0010	2
3	0011	3
4	0100	4
5	0101	5
6	0110	6
7	0111	7
8	1000	8
9	1001	9
A	1010	10
B	1011	11
C	1100	12
D	1101	13
E	1110	14
F	1111	15
Example: The binary 2-byte number (0001111110100101) can be expressed in hex notation as X'1FA5'		

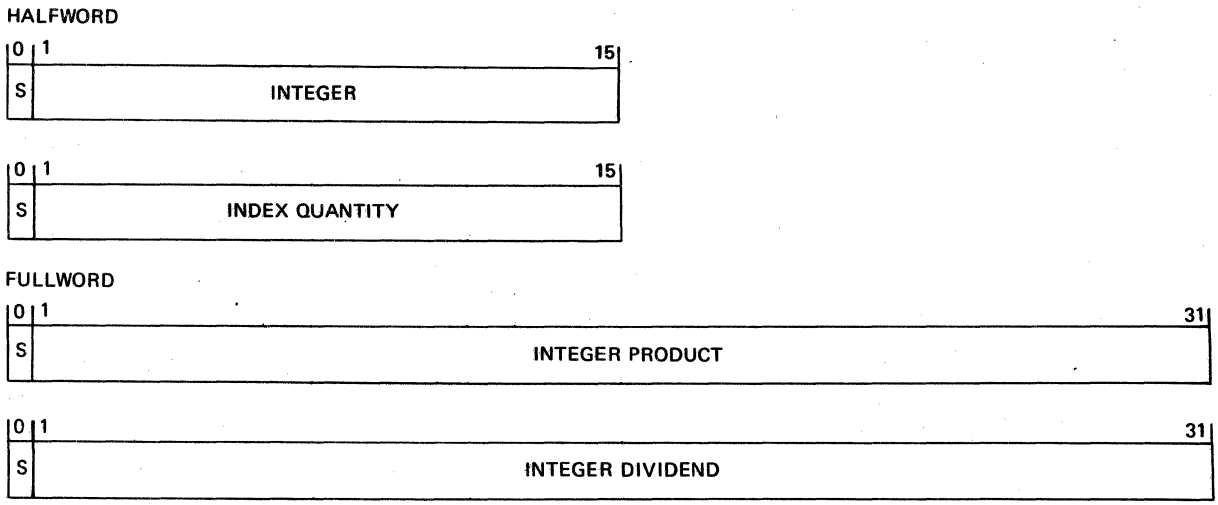


Figure 3-1. Fixed-Point Word Formats

TABLE 3-2
EXAMPLES OF FIXED-POINT REPRESENTATION

Number	Decimal	Binary				Hexadecimal
$2^{15}-1$	32767	0111	1111	1111	1111	7FFF
2^0	1	0000	0000	0000	0001	0001
0	0	0000	0000	0000	0000	0000
-2^0	-1	1111	1111	1111	1111	FFFF
-2^{15}	-32768	1000	0000	0000	0000	8000

3.2.3 Floating-Point Data

A floating-point number consists of a signed exponent and a signed fraction. The quantity expressed by this number is the product of the fraction and the number 16 raised to the power of the exponent. Each floating-point value requires two halfwords. The floating-point format is shown in Figure 3-2.

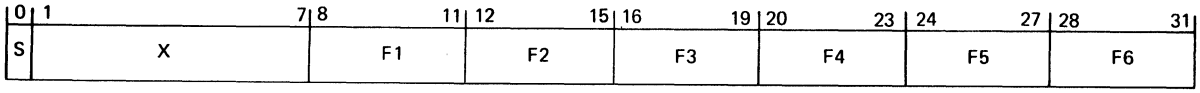


Figure 3-2. Floating-Point Word Format

Sign and magnitude representation is used, in which the sign bit S is zero for positive values, and one for negative values. The exponent X is expressed in excess 64 binary notation; that is, field X contains the true value of the exponent +64.

The fraction contains six hexadecimal digits F1-F6. The value of a floating-point fraction can be expressed as follows: $F_1 . 16^{-1}+F_2 . 16^{-2}+F_3 . 16^{-3}+\ldots . +F_6 . 16^{-6}$.

A normalized floating-point number has a non-zero high-order hexadecimal fraction digit (F_1). If the high-order hexadecimal fraction digit (F_1) is zero, the number is said to be unnormalized. The range of the magnitude (M) of a normalized floating-point number is:

$$16^{-65} \leq M \leq (1 - 16^{-6}) \cdot 16^{63}$$

or approximately

$$5.4 \cdot 10^{-79} \leq M \leq 7.2 \cdot 10^{75}$$

Table 3-3 shows several examples of the floating-point number representation used in GE-PAC 3010/2 Systems.

All floating-point numbers are assumed to be normalized prior to their use as operands. No pre-normalization is performed, all results are post-normalized. The Floating-Point Load instruction will normalize unnormalized floating-point numbers.

Exponent overflow is defined as a resultant exponent greater than 63. Exponent underflow is defined as a resultant exponent less than -64. The Overflow flag is set whenever exponent overflow or underflow is detected. The Greater Than flag is set on positive overflow, the Less Than flag is set on negative overflow, and both flags are reset on underflow. On overflow, the exponent and fraction of the result are set to all ones. The sign of the result is not affected by the overflow. On underflow, the sign, exponent and fraction of the sum are set to zero.

TABLE 3-3
EXAMPLES OF FLOATING-POINT REPRESENTATION

Decimal Value	Binary				Hexadecimal Value
1.0	0100	0001	0001	0000	4110
	0000	0000	0000	0000	0000
-1.0	1100	0001	0001	0000	C110
	0000	0000	0000	0000	0000
9.5	0100	0001	1001	1000	4198
	0000	0000	0000	0000	0000
-0.5	1100	0000	1000	0000	C080
	0000	0000	0000	0000	0000
$-(1-16^{-6}).16^{63}$	1111	1111	1111	1111	FFFF
	1111	1111	1111	1111	FFFF
-16^{-65}	1000	0000	0001	0000	8010
	0000	0000	0000	0000	0000
$0.1 + 16^{-6}$	0100	0000	0001	1001	4019
	1001	1001	1001	1010	999A

The floating-point value in which all data bits are zero is called true zero. A true zero may arise as the result of an arithmetic operation because of exponent underflow, or when a resultant fraction is zero because of loss of significance. In general, zero values participate as normal numbers in all arithmetic operations.

There are eight 32-bit Floating-Point Registers, which are addressed with the even numbers 0, 2, 4, ..., 14. The Floating-Point Registers are reserved core memory locations and are addressable only by the Floating-Point instructions, which are described in Section 4.6

3.2.4 Logical Data

Logical operations manipulate 8-bit bytes, 16-bit halfwords, and 32-bit fullwords. All bits participate in logical operations. The data words have the format shown in Figure 3-3.

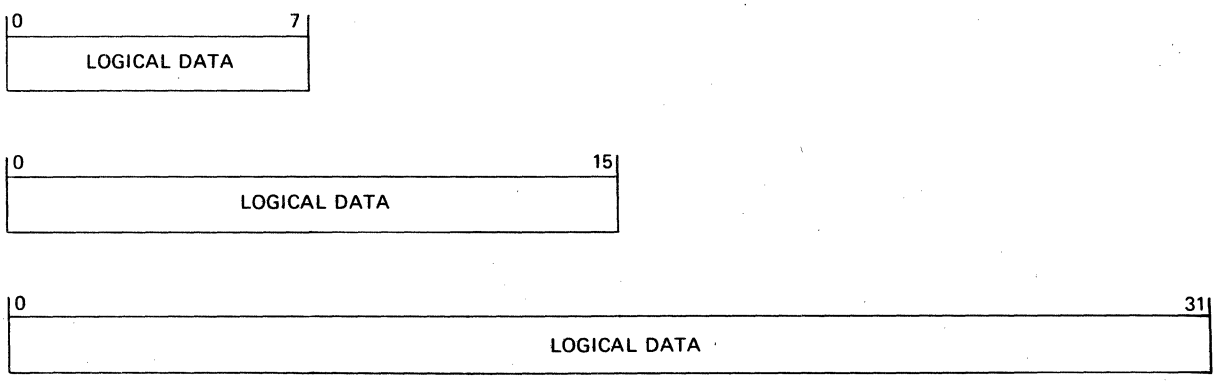


Figure 3-3. Logical Data Word Formats

For upward compatibility with future machines, boundary conventions for halfwords and fullwords should be observed.

3.3 INSTRUCTION FORMATS

GE-PAC 3010/2 Instructions represent one of four formats designated Register to Register (RR), Short Format (SF), Register to Indexed Memory (RX) and Register to Storage (RS) instructions.

In general, each format specifies three things: The operation to be performed, the address of the first operand, and the address of the second operand. The first operand is normally the contents of a General Register. The second operand is normally the contents of another General Register, the contents of a core memory location, or a data constant from the instruction word itself.

A 16-bit halfword format is used for Register-to-Register and Short Format instructions. The Short Format instructions may be used to manipulate small quantities or execute short branches relative to the present Location Counter. A 32-bit fullword format is used for the Register to Indexed Memory, and the Register to Storage formats. The specific formats are shown in Figure 3-4.

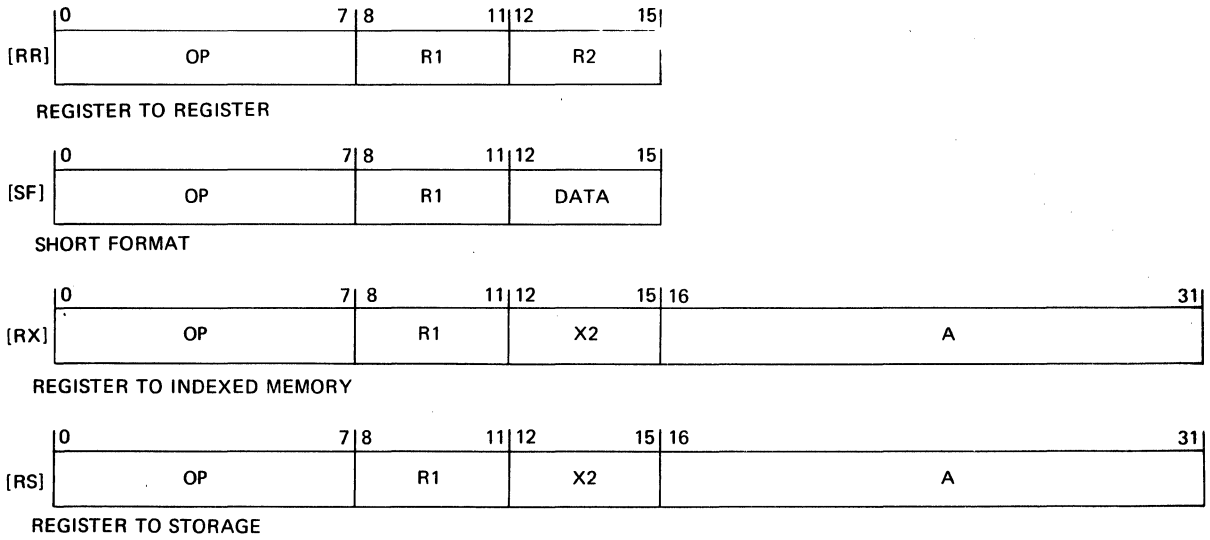


Figure 3-4. Instruction Word Formats

The eight-bit OP field in all formats specifies the machine operation to be performed. Operation codes are represented as two hexadecimal characters.

The four-bit R1 field in the instruction formats specifies the address of the first operand. The R1 field is normally the address of a General Register.

The four-bit R2 field in the RR instruction format specifies the address of the second operand, which is normally a register address.

The four-bit data field of the SF instructions supplies data in the case of Fixed-Point Arithmetic instructions, or a displacement from the current Location Counter in the case of Branch instructions.

A non-zero X2 field in the RX and RS formats specifies a General Register whose contents is used as an index value. The index value (X2) may be positive or negative. If X2 is zero, no address modification takes place. General Registers 1 through 15 can optionally be used for indexing, but General Register 0 can never be used for indexing.

The 16-bit Address field specifies a memory address in the RX format, or contains a value (Data) to be used as an immediate operand in the RS format.

The RR instructions are used for operations between registers. The first operand is the contents of the register specified by the R1 field of the instruction word. The second operand is the contents of the register specified by the R2 field.

The SF instructions are used for; short immediates, in which the data field specifies a four-bit data value; short shifts, in which the data field specifies the shift count; and short branches, in which the data field specifies a displacement (in halfwords) from the current instruction address.

The RX instructions are used for operations between register and memory with the option of indexing. The first operand is the register specified by the R1 field of the instruction word. The second operand is the contents of the memory location specified by the A field of the instruction word or the sum of the A field and the contents of the General Register specified by the X2 field if indexing is specified.

In the RS instructions, the first operand is the contents of the General Register specified by the R1 field of the instruction word. The second operand is the number contained in the A field of the instruction, or the sum of the A field and the contents of the General Register specified by the X2 field if indexing is specified. The second operand of an RS instruction specifies the number of bit positions in Shift instructions, or forms the second operand in Immediate instructions.

There are some exceptions to the first operand-second operand nomenclature used above. For example, with Branch on Condition instructions, the R1 field of the instruction is a four-bit mask (M1) which is ANDed with the Condition Code in the Current PSW. These instructions are discussed in Section 4.8. For all Input/Output instructions, the contents of the register specified by R1 specifies the device number for the I/O operation. For the Supervisor Call instruction, the R1 field specifies 1 out of 16 possible types of supervisor call. With the Load Program Status Word (LPSW), Simulate Interrupt (SINT) and Auto Load (AL) instructions, the R1 field must be zero. These instructions are described in Chapter 4.

Table 3-4 summarizes the first and second operand designations for each instruction format.

TABLE 3-4
DESIGNATIONS FOR FIRST AND SECOND OPERANDS

First Operand:	The contents of the register specified by the R1 Field (R1).	RR, RX, RS and SF
	The M1 Field	RR, RX, and SF Branch on Condition
	The actual value of the R1 Field.	SVC
Second Operand:	The contents of the register specified by the R2 Field (R2).	RR
	The contents of the address derived by adding the A field and the contents of the General Register specified by the X2 Field. [A + (X2)]	RX
	The A field plus the contents of the General Register specified by the X2 Field A + (X2)	RS
	The actual value of the R2 Field	SF

3.4 GENERAL REGISTER USAGE

The 16 General Registers function as accumulators or Index Registers in all arithmetic and logical operations. Each General Register is a 16-bit halfword consisting of two 8-bit bytes. For arithmetic operations, bit zero (leftmost position) is considered the sign bit using two's complement representation.

The General Registers are numbered from zero to fifteen (decimal), and written in hexadecimal notation as; 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, and F. The General Registers have not been given specific functional assignments. However, the following operational restrictions should be noted:

1. It is not possible to use General Register 0 as an Index Register. In the RX and RS instruction formats, a zero entry in the X2 field indicates that no indexing is to take place.
2. For Fixed-Point Multiply, Divide, and fullword Shift and Rotate instructions, the R1 field must specify an even numbered General Register. See Sections 4.3 and 4.7.
3. For Branch on Index instructions, the R1 field specifies the first of 3 consecutive General Registers, and the value of the R1 therefore, should be equal or less than 13. See Section 4.8.3.
4. For Floating-Point instructions the R1 field must be an even value, and specify one of the Floating-Point Registers rather than one of the General Registers.
5. With any RR type instruction, the R1 field and the R2 field can specify the same register, but special attention should be given to note what the instruction will do. For example, with the EPSR instruction, if the R1 field equals the R2 field, the program status is stored in a General Register, but the program status is unchanged.

3.5 STORAGE ADDRESSING

The GE-PAC 3010/2 Instruction Set manipulates data of three different word lengths: 8-bit bytes, 16-bit halfwords, or 32-bit fullwords. In each case, the bits are numbered from left to right, starting with the number zero. The format for each word length is shown in Figure 3-5.

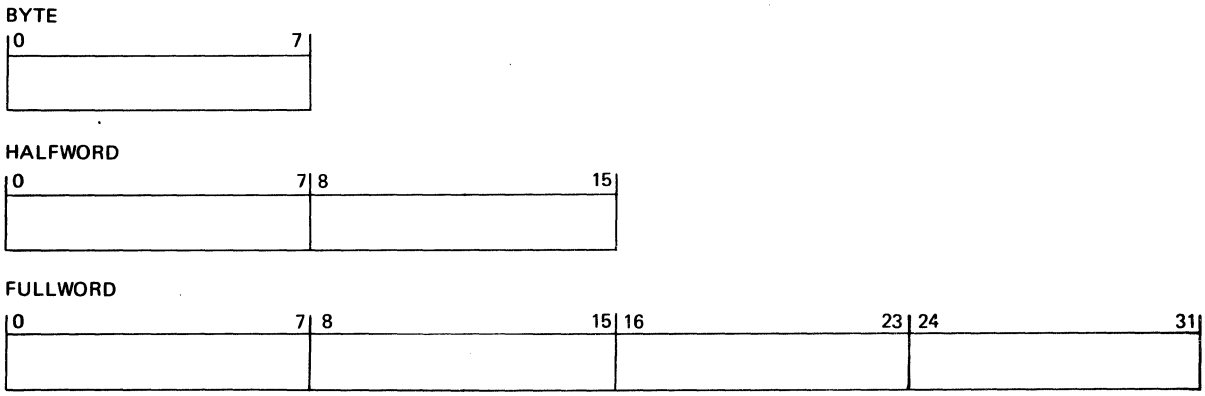


Figure 3-5. Data Word Formats

Core memory locations are numbered consecutively, beginning at 0000, for each 8-bit byte. Operands in memory are addressed by the RX type instructions. Since the address portion (A) of an RX instruction is 16-bits wide, it is possible to directly address 65,536 bytes.

The GE-PAC 3010/2 transfers binary information between memory and the Processor as 16-bit halfwords. The instruction being performed determines if the address specified is that of a byte, a halfword, or a fullword. If a byte of information is desired, either the left or right byte of the halfword read from memory is manipulated as determined by the specific address. If a halfword of information is desired, the entire 16-bits read from memory are used. If a fullword is desired, a second 16-bits are read from memory and combined with the original halfword.

NOTE

Bytes of information are addressed by their specific hexadecimal address. A group of bytes combined to form a halfword or a fullword are addressed by the leftmost byte in the group. Halfword or fullword operands must be positioned at an address which is a multiple of 2. Table 3-5 illustrates the addressing scheme used to maintain compatibility with other GE-PAC 30 and 3010 Systems. Fullword operands must be positioned at an address which is a multiple of 4.

TABLE 3-5
MEMORY ADDRESSING EXAMPLE

Address	0050	0051	0052	0053	0054	0055	0056	0057
Contents	01	23	45	67	89	AB	CD	EF
Operand Length and Position	Byte	Byte	Byte	Byte	Byte	Byte	Byte	Byte
	←Halfword→		←Halfword→		←Halfword→		←Halfword→	
	←Fullword→				←Fullword→			

For example, if the address referenced in Table 3-5 is 0050_{16}, then:

A Byte-Oriented instruction would extract the value 01_{16}, as an operand.

A Halfword-Oriented instruction would extract the value 0123_{16} as an operand.

A Fullword instruction would extract the value 01234567_{16} as an operand.

CHAPTER 4
INSTRUCTION REPERTOIRE

4.1 INTRODUCTION

The instruction repertoire has been grouped by function in this chapter. The use and operation of each instruction is presented in the following format:

1. An instruction word chart for each instruction including: Mnemonic operation code, and first and second operand designations in the correct assembler format. The format type designated by [SF], [RR], [RS], and [RX]. An instruction diagram with hexadecimal operation code and the locations of all fields is also provided, for example:

SIS R1,N [SF]

0 – 7	8 – 11	12 – 15
27	R1	N

SHR R1,R2 [RR]

0 – 7	8 – 11	12 – 15
0B	R1	R2

SH R1,A(X2) [RX]

0 – 7	8 – 11	12 – 15	16 – 31
4B	R1	X2	A

SHI R1,A(X2) [RS]

0 – 7	8 – 11	12 – 15	16 – 31
CB	R1	X2	A

2. A description of instruction operation.

3. An example of a diagrammatic representation of instruction operation is shown below.

SIS: $(R1) \leftarrow (R1) - N$
SHR: $(R1) \leftarrow (R1) - (R2)$
SH: $(R1) \leftarrow (R1) - [A + (X2)]$
SHI: $(R1) \leftarrow (R1) - A - (X2)$

4. A chart illustrating the possible variations of the Condition Code in the Current Program Status Word as a result of performing the instruction: a one indicates set, a zero indicates reset. It is important to note that any instruction which changes the Condition Code can change all four bits. The conditions listed on the chart are only those conditions which are meaningful after a particular instruction. Other bits may be changed, but their condition is not meaningful, for example:

Resulting Condition Code:

12 C	13 V	14 G	15 L	
		0	0	DIFFERENCE IS ZERO.
		0	1	DIFFERENCE IS LESS THAN ZERO.
		1	0	DIFFERENCE IS GREATER THAN ZERO.
	1			ARITHMETIC OVERFLOW.
1				BORROW.

5. A programming note to provide additional pertinent or clarifying information. All privileged instructions and those instructions which may cause a memory protect violation are so noted, for example:

Programming Note:

The Subtract Immediate Short (SIS) instruction, causes the four-bit second operand N to be subtracted from the contents of the General Register specified by R1. This instruction is useful for decrementing a register by a small value (e.g. X'2').

The Subtract Halfword Immediate (SHI) instruction, produces a value which is the difference between the first operand General Register (R1), less the sum of the address field itself, and the content of a General Register index (X2).

The symbols and abbreviations used in the instruction diagrams are defined as follows:

( ) [ ]	Parentheses or Brackets. Read as "the content of ...".
←	Arrow. Read as "is replaced by ..." or "replaces ...".
A	The 16-bit halfword address which is a part of the RX and RS instructions.
R1	The address of a General Register the content of which is the first operand.
M1	Mask of four-bits specifying Branch on Condition testing.
R2	The address of a General Register the content of which is the second operand of an RR instruction.
X2	The address of a General Register the content of which is used as an index value.
N	The four-bit second operand used with Short Format Immediate instructions.
D	The four-bit displacement value used with Short Format Branch instructions.
(0:7) (8:15) (16:31)	A bit grouping within a byte, a halfword, or a fullword. Read as "0 thru 7 inclusive" "Bits 8 through 15 inclusive", etc.
PSW	Program Status Word of 32 bits containing the Status, Condition Code, and current instruction address.
CC	Condition Code of four-bits contained in the PSW.
C	Carry Bit contained in the Condition Code (Bit 12 of PSW).
V	Overflow Bit contained in the Condition Code (Bit 13 of PSW).
G	Greater Than Bit contained in the Condition Code (Bit 14 of PSW).
L	Less Than Bit contained in the Condition Code (Bit 15 of PSW).
+ - * / :	Arithmetic operations - Add, Subtract, Multiply, and Divide respectively. Logical comparison, when used, e.g., R1:R2.

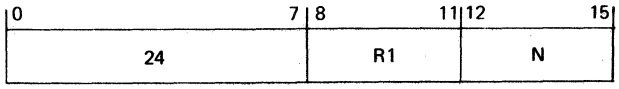
4.2 FIXED-POINT LOAD/STORE INSTRUCTIONS

The Fixed-Point Load/Store instructions are used to transfer fixed-point (see 3.2.2) data between the General Registers and core memory. The instructions described in this section are:

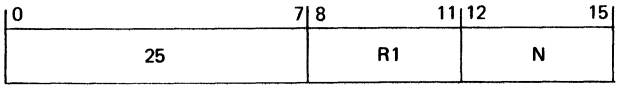
4.2.1	LIS	Load Immediate Short
	LCS	Load Complement Short
	LHR	Load Halfword RR
	LH	Load Halfword
	LHI	Load Halfword Immediate
4.2.2	LM	Load Multiple
4.2.3	STH	Store Halfword
4.2.4	STM	Store Multiple

4.2.1 Load Halfword

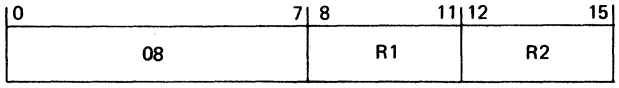
LIS R1,N [SF]



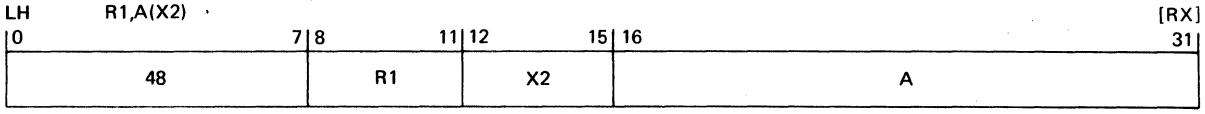
LCS R1,N [SF]



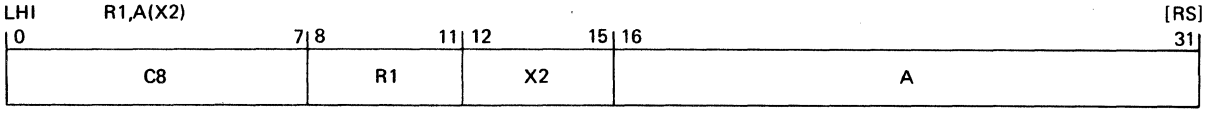
LHR R1,R2 [RR]



LH R1,A(X2) [RX]



LHI R1,A(X2) [RS]



The second operand is loaded into the General Register specified by R1.

LIS: (R1) ← N
LCS: (R1) ← -N
LHR: (R1) ← (R2)
LH: (R1) ← [A + (X2)]
LHI: (R1) ← A + (X2)

Resulting Condition Code:

12	13	14	15	
C	V	G	L	
		0	0	OPERAND IS ZERO.
		0	1	OPERAND IS LESS THAN ZERO.
		1	0	OPERAND IS GREATER THAN ZERO.

Programming Note:

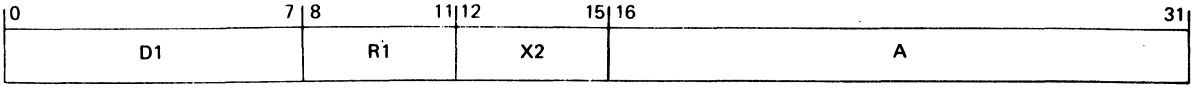
The Load Immediate Short (LIS) instruction, causes the four-bit second operand to be expanded to a 16-bit halfword with high order bits set to zero. This halfword is loaded into the General Register specified by R1.

The Load Complement Short (LCS) instruction, causes the four-bit second operand to be expanded to a 16-bit halfword with high order bits set to zero. The two's complement of this halfword is loaded into the General Register specified by R1.

These instructions may be used to preset a register with an index value, load a register with the first operand for a subsequent arithmetic operation (e. g. add, multiply), or set the Condition Code for supplemental testing by a Branch on Condition instruction.

4.2.2 Load Multiple

LM R1,A(X2) [RX]



Sequential halfwords from memory are loaded into successive General Registers, beginning with the General Register specified by the R1 field. The first halfword is defined by A + (X2). The operation is terminated when R15 is loaded from memory. Note that any number of sequential General Registers can be loaded in this manner.

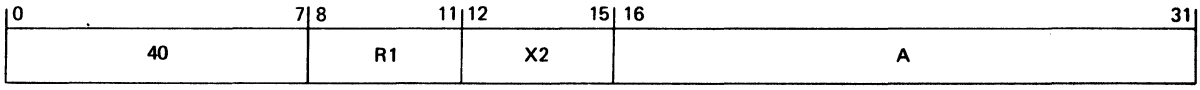
1. (R1) ← [A + (X2)]
2. R1: X'F'
   if R1 = X'F', the instruction is finished
   if R1 ≠ X'F', then:
3. R1 ← R1 + 1
4. A ← A + 2, return to Step 1

Resulting Condition Code:

Unchanged.

4.2.3 Store Halfword

STH R1,A(X2) [RX]



The 16-bit first operand is stored in the core memory location specified by the second operand. The first operand is unchanged.

STH: (R1) → [A + (X2)]

Resulting Condition Code:

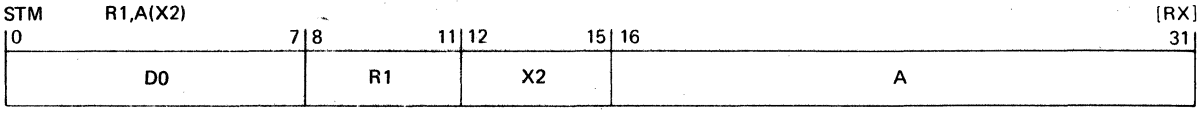
Unchanged.

Programming Note:

This instruction is subject to Memory Protect.

4.2.4 Store Multiple

STM R1,A(X2) [RX]



Successive General Registers are stored sequentially into memory, beginning with the General Register specified by the R1 field. The first storage address is determined by A + (X2). The operation is terminated when R15 is stored in memory. Note that any number of sequential General Registers can be transferred in this manner.

1. (R1) $\longrightarrow$ [A + (X2)]
2. R1: X'F'
   if R1 = X'F', then instruction is finished
   if R1 $\neq$ X'F', then:
3. R1 $\longleftarrow$ R1 + 1
4. A $\longleftarrow$ A + 2, return to step 1

Resulting Condition Code:

Unchanged.

Programming Note:

This instruction is subject to Memory Protect.

The Store Multiple (STM) instruction in conjunction with the Load Multiple (LM) instruction is an aid to subroutine execution. They permit the easy saving and restoring of the registers required by the subroutine. The Store Multiple instruction can be used upon entering the subroutine, and the Load Multiple would be the last instruction executed before returning from the subroutine.

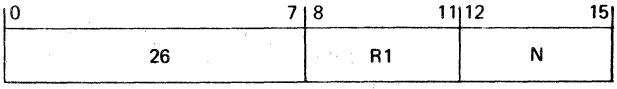
4.3 FIXED-POINT ARITHMETIC INSTRUCTIONS

The Fixed-Point Arithmetic instructions provide for addition, subtraction, multiplication and division of fixed-point data (see 3.2.2) contained in the General Registers and/or core memory. Also included are logical and arithmetic compare operations. The instructions described in this section are:

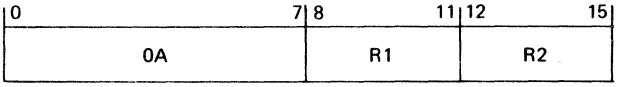
4.3.1	AIS	Add Immediate Short
	AHR	Add Halfword RR
	AH	Add Halfword
	AHI	Add Halfword Immediate
	AHM	Add Halfword to Memory
4.3.2	ACHR	Add with Carry Halfword RR
	ACH	Add with Carry Halfword
4.3.3	SIS	Subtract Immediate Short
	SHR	Subtract Halfword RR
	SH	Subtract Halfword
	SHI	Subtract Halfword Immediate
4.3.4	SCHR	Subtract with Carry Halfword RR
	SCH	Subtract with Carry Halfword
4.3.5	CLHR	Compare Logical Halfword RR
	CLH	Compare Logical Halfword
	CLHI	Compare Logical Halfword Immediate
4.3.6	CHR	Compare Halfword RR
	CH	Compare Halfword
	CHI	Compare Halfword Immediate
4.3.7	MHR	Multiply Halfword RR
	MH	Multiply Halfword
4.3.8	MHUR	Multiply Halfword Unsigned RR
	MHU	Multiply Halfword Unsigned
4.3.9	DHR	Divide Halfword RR
	DH	Divide Halfword

4.3.1 Add Halfword

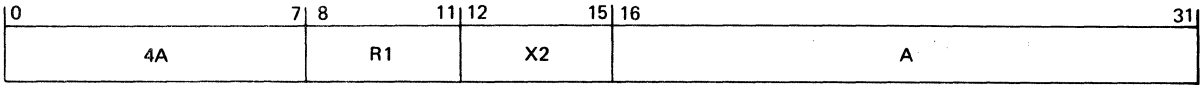
AIS R1,N [SF]



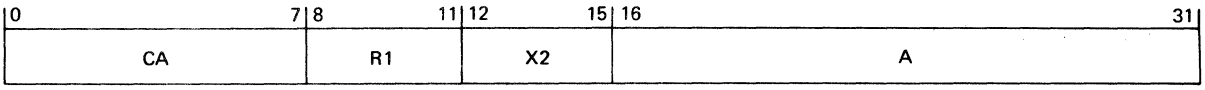
AHR R1,R2 [RR]



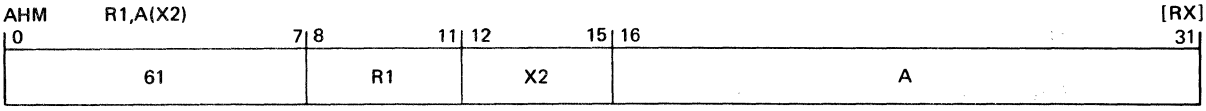
AH R1,A(X2) [RX]



AHI R1,A(X2) [RS]



AHM R1,A(X2) [RX]



The second operand is added algebraically to the contents of the General Register specified by R1.

AIS: $(R1) \leftarrow (R1) + N$
AHR: $(R1) \leftarrow (R1) + (R2)$
AH: $(R1) \leftarrow (R1) + [A + (X2)]$
AHI: $(R1) \leftarrow (R1) + A + (X2)$
AHM: $[A + (X2)] \leftarrow (R1) + [A + (X2)]$

Resulting Condition Code:

12 C	13 V	14 G	15 L	
		0	0	SUM IS ZERO.
		0	1	SUM IS LESS THAN ZERO.
		1	0	SUM IS GREATER THAN ZERO.
	1			ARITHMETIC OVERFLOW.
1				CARRY.

Programming Note:

The Add Immediate Short (AIS) instruction, causes the four-bit second operand N to be added to the contents of the General Register specified by R1. The result replaces the contents of R1.

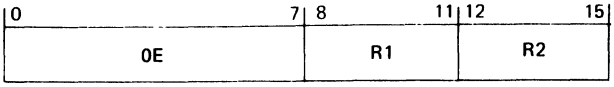
The Add Halfword Immediate (AHI) instruction, produces a value which is the algebraic sum of the address field itself, the content of a General Register index (X2), and the first operand General Register (R1).

The Add Halfword to Memory (AHM) instruction, causes the second operand $[A + (X2)]$ to be added to the contents of the General Register specified by R1. The result of the addition does not replace the contents of R1, but instead is stored in core memory at the address specified by A + (X2). The first operand (R1) remains unchanged. This instruction effectively permits every location in core memory to be used as a counter.

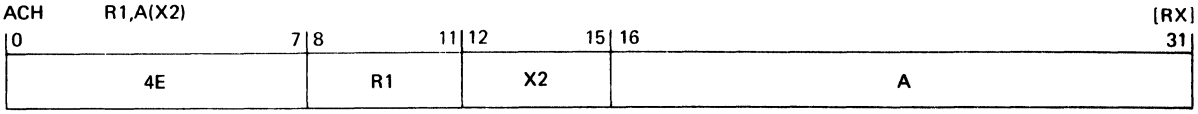
This instruction is subject to Memory Protect.

4.3.2 Add with Carry Halfword

ACHR R1,R2 [RR]



ACH R1,A(X2) [RX]



The 16-bit second operand and the Carry Bit of the Condition Code (PSW 12) are added algebraically to the General Register specified by R1. The resulting sum is contained in R1. The second operand is unchanged.

ACHR: (R1) ← (R1) + (R2) + C
ACH: (R1) ← (R1) + [A + (X2)] + C

Resulting Condition Code:

12 C	13 V	14 G	15 L	
		0	0	SUM IS ZERO.
		0	1	SUM IS LESS THAN ZERO.
		1	0	SUM IS GREATER THAN ZERO.
	1			ARITHMETIC OVERFLOW.
1				CARRY.

Programming Note:

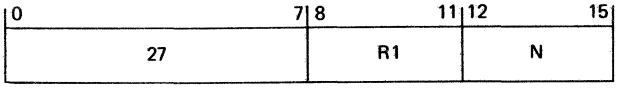
Multiple precision addition operations require a Carry forward from the least significant operands to the most significant. To accomplish this, the locations containing the least significant portions of the two operands are summed, using the Add Halfword (AH) instruction. A Carry forward, if it occurs, is retained in the Carry Bit position of the Condition Code (PSW 12).

The locations containing the next least significant portions of the two operands are then summed, using the Add with Carry Halfword (ACH) instruction. The Carry Bit contained in the Condition Code (set from the previous addition) participates in this sum; the Carry Bit position is then set to reflect the new result.

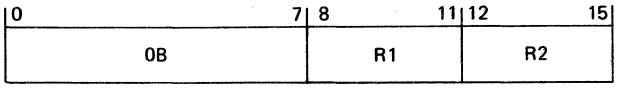
The Add with Carry Halfword (ACH) instruction, is used on succeeding pairs of operands until the most significant operands of the multiple precision words have been summed. The resulting Condition Code is valid for testing the multiple precision word.

4.3.3 Subtract Halfword

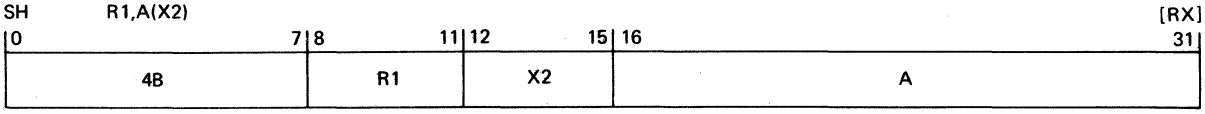
SIS R1,N [SF]



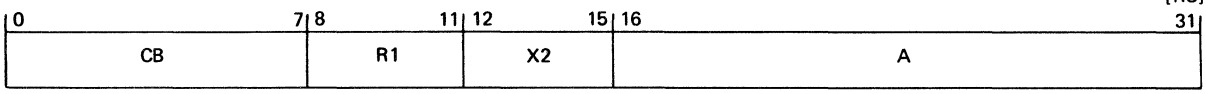
SHR R1,R2 [RR]



SH R1,A(X2) [RX]



SHI R1,A(X2) [RS]



The second operand is subtracted from the General Register specified by R1. The difference is contained in R1. The second operand is unchanged.

SIS: (R1) ← (R1) - N
SHR: (R1) ← (R1) - (R2)
SH: (R1) ← (R1) - [A + (X2)]
SHI: (R1) ← (R1) - A - (X2)

Resulting Condition Code:

12	13	14	15	
C	V	G	L	
		0	0	DIFFERENCE IS ZERO.
		0	1	DIFFERENCE IS LESS THAN ZERO.
		1	0	DIFFERENCE IS GREATER THAN ZERO.
	1			ARITHMETIC OVERFLOW.
1				BORROW.

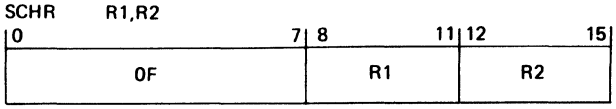
Programming Note:

The Subtract Immediate Short (SIS) instruction, causes the four-bit second operand N to be subtracted from the contents of the General Register specified by R1. This instruction is useful for decrementing a register by a small value (e.g. X'2').

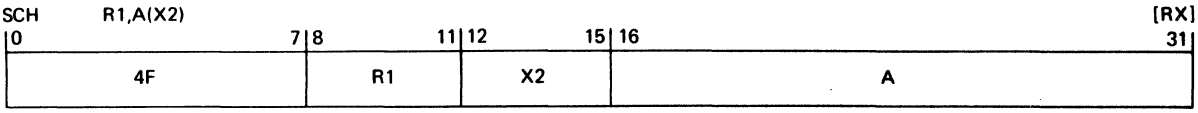
The Subtract Halfword Immediate (SHI) instruction, produces a value which is the difference between the first operand General Register (R1), less the sum of the address field itself, and the content of a General Register index (X2).

4.3.4 Subtract with Carry Halfword

SCHR R1,R2 [RR]



SCH R1,A(X2) [RX]



The 16-bit second operand with the Carry (borrow) Bit is subtracted from the General Register specified by R1. The difference is contained in R1. The second operand is unchanged.

SCHR: (R1) ← (R1) - (R2) - C
SCH: (R1) ← (R1) - [A + (X2)] - C

Resulting Condition Code:

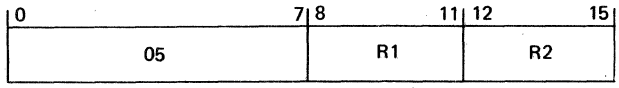
12 C	13 V	14 G	15 L	
		0	0	DIFFERENCE IS ZERO.
		0	1	DIFFERENCE IS LESS THAN ZERO.
		1	0	DIFFERENCE IS GREATER THAN ZERO.
	1			ARITHMETIC OVERFLOW.
1				BORROW.

Programming Note:

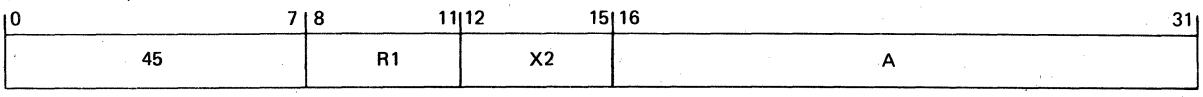
See Add with Carry Halfword 4.3.2.

4.3.5 Compare Logical Halfword

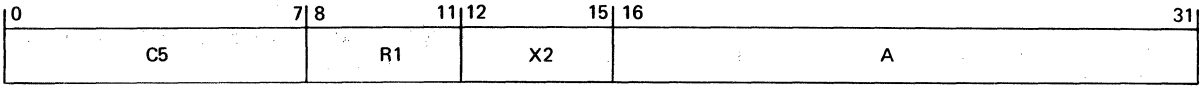
CLHR R1,R2 [RR]



CLH R1,A(X2) [RX]



CLHI R1,A(X2) [RS]



The first operand specified by R1 is compared logically to the 16-bit second operand. The result is indicated by the setting of the Condition Code [PSW (12:15)] . Both operands remain unchanged.

CLHR: (R1) : (R2)
CLH: (R1) : [A + (X2)]
CLHI: (R1) : A + (X2)

Resulting Condition Code:

12	13	14	15	
C	V	G	L	
		0	0	FIRST OPERAND EQUAL TO SECOND OPERAND.
		0	1	FIRST OPERAND NOT EQUAL TO SECOND OPERAND.
		1	0	
1				FIRST OPERAND LESS THAN SECOND OPERAND.
0				FIRST OPERAND EQUAL TO OR GREATER THAN SECOND OPERAND.

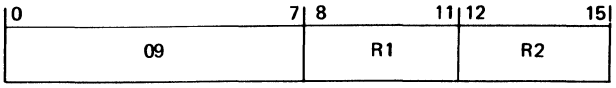
Programming Note:

The logical comparison is performed by subtracting the second operand from the first operand. The result is in the Condition Code setting, the operands are not modified.

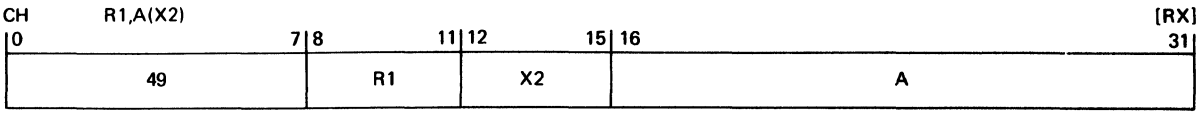
The Compare Logical Halfword Immediate (CLHI) instruction, produces a value which is the logical comparison of the address field itself plus the content of a General Register index (X2) with the first operand General Register (R1).

4.3.6 Compare Halfword

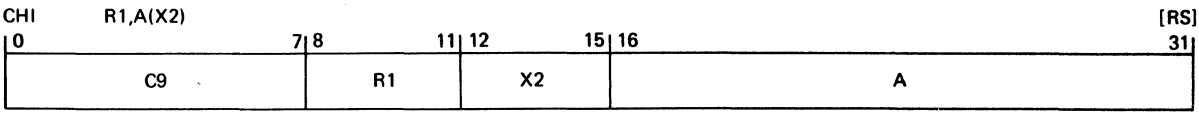
CHR R1,R2 [RR]



CH R1,A(X2) [RX]



CHI R1,A(X2) [RS]



The first operand specified by R1 is compared to the 16-bit second operand. The comparison is algebraic, taking into account the sign and magnitude of each number. The result is indicated by the setting of the Condition Code [PSW (12:15)]. Both operands remain unchanged.

CHR: (R1) : (R2)
CH: (R1) : [A + (X2)]
CHI: (R1) : A + (X2)

Resulting Condition Code:

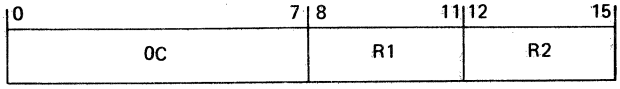
12 C	13 V	14 G	15 L	
		0	0	FIRST OPERAND EQUAL TO SECOND OPERAND.
		0	1	FIRST OPERAND LESS THAN SECOND OPERAND.
		1	0	FIRST OPERAND GREATER THAN SECOND OPERAND.
1				FIRST OPERAND LESS THAN SECOND OPERAND.
0				FIRST OPERAND EQUAL TO OR GREATER THAN SECOND OPERAND.

Programming Note:

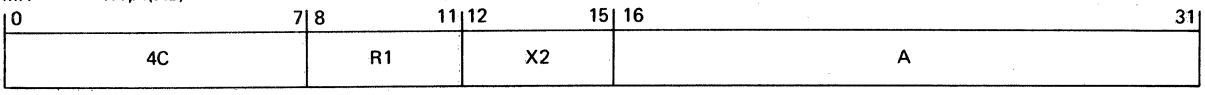
The Compare Halfword (CH) instructions, permit arithmetic comparison of signed two's complement 16-bit integers. They facilitate fast comparisons for DO loop, and IF statement processing in FORTRAN.

4.3.7 Multiply Halfword

MHR R1,R2 [RR]



MH R1,A(X2) [RX]



The 16-bit second operand is multiplied by the contents of the General Register specified by R1 + 1. The R1 field of the instruction must specify an even numbered register. The resulting 32-bit product is contained in R1 and R1 + 1, an even-odd pair; the second operand is unchanged. The sign of the product is determined by the rules of algebra.

MHR: (R1, R1 + 1) ⟵ (R1 + 1)*(R2)
MH: (R1, R1 + 1) ⟵ (R1 + 1)*[A + (X2)]

Resulting Condition Code:

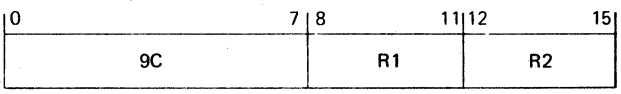
Unchanged.

Programming Note:

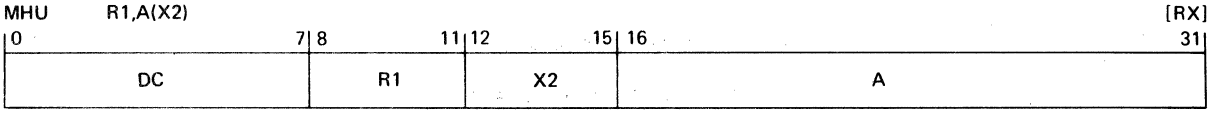
After multiplication, the most significant 15 bits with sign are contained in R1. The least significant 16 bits are contained in R1 + 1.

4.3.8 Multiply Halfword Unsigned

MHUR R1,R2 [RR]



MHU R1,A(X2) [RX]



The 16-bit second operand is multiplied by the contents of the General Register specified by R1 + 1. All 16-bits of both operands are considered to be magnitude. The resulting 32-bit product is contained in R1 and R1 + 1, the second operand is unchanged. The R1 field of the instruction must specify an even numbered register.

MHUR: (R1, R1 + 1) ⟵ (R1 + 1)*(R2)
MHU: (R1, R1 + 1) ⟵ (R1 + 1)*[A + (X2)]

Resulting Condition Code:

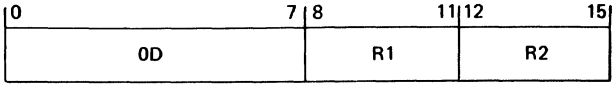
Unchanged.

Programming Note:

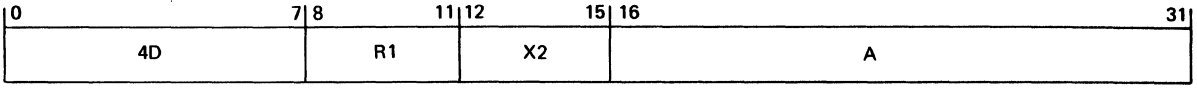
This instruction is most useful in applications requiring multiple precision multiply capability. Typically, a Multiply Halfword (MH) instruction would be used with the most significant halfwords of the two operands, after the least significant parts of the two operands were multiplied using the Multiply Halfword Unsigned (MHU) instruction. The partial products could then be summed.

4.3.9 Divide Halfword

DHR R1,R2 [RR]



DH R1,A(X2) [RX]



The 16-bit second operand is divided into the 32-bit dividend contained in the General Register specified by R1 and R1 + 1. The first operand, R1, must specify an even numbered register. The resulting 15-bit quotient with sign is contained in R1 + 1; a 15-bit remainder with sign is contained in R1, the second operand is unchanged. The sign of the result is determined by the rules of algebra; the sign of the remainder is the same as the sign of the dividend.

DHR: (R1 + 1)←(R1, R1 + 1)/(R2)
(R1)←Remainder

DH: (R1 + 1)←(R1, R1 + 1)/ [A + (X2)]
(R1)←Remainder

Resulting Condition Code:

Unchanged.

Programming Note:

Attempted division by zero or a quotient which would be greater than X'8000' causes a Fixed-Point Divide Fault Interrupt, if enabled by Bit 3 of the Program Status Word. The operands remain unchanged.

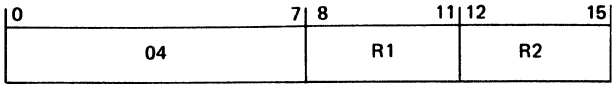
4.4 LOGICAL INSTRUCTIONS

The Logical instructions manipulate logical data (see 3.2.4) such that each bit of the first operand is logically combined with the corresponding bit in the second operand. The instructions described in this section are:

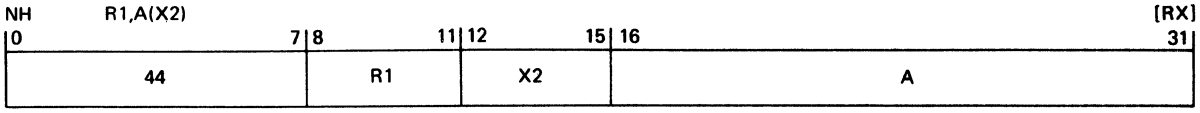
4.4.1	NHR	AND Halfword RR
	NH	AND Halfword
	NHI	AND Halfword Immediate
4.4.2	OHR	OR Halfword RR
	OH	OR Halfword
	OHI	OR Halfword Immediate
4.4.3	XHR	Exclusive OR Halfword RR
	XH	Exclusive OR Halfword
	XHI	Exclusive OR Halfword Immediate
4.4.4	THI	Test Halfword Immediate

4.4.1 AND Halfword

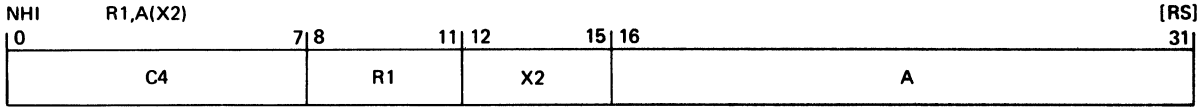
NHR R1,R2 [RR]



NH R1,A(X2) [RX]



NHI R1,A(X2) [RS]



The logical product of the 16-bit second operand and the content of the General Register specified by R1, replaces the content of R1. The 16-bit product is formed on a bit-by-bit basis.

NHR: (R1) ← (R1) AND (R2)
NH: (R1) ← (R1) AND [A + (X2)]
NHI: (R1) ← (R1) AND A + (X2)

Resulting Condition Code:

12 C	13 V	14 G	15 L	
		0	0	LOGICAL PRODUCT IS ZERO.
		0	1	LOGICAL PRODUCT IS NOT ZERO.
		1	0	

Programming Note:

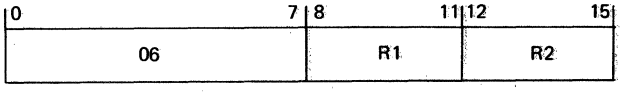
The AND Halfword Immediate (NHI) instruction, produces a value which is the logical product of the address field itself plus the content of a General Register index (X2) with the first operand General Register (R1).

The truth table for the AND function is:

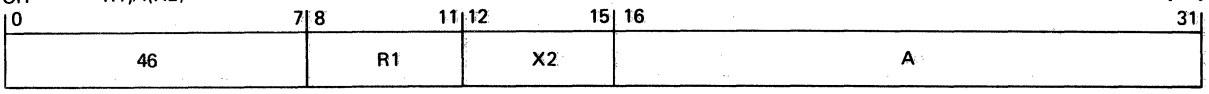
0 AND 0 = 0
0 AND 1 = 0
1 AND 0 = 0
1 AND 1 = 1

4.4.2 OR Halfword

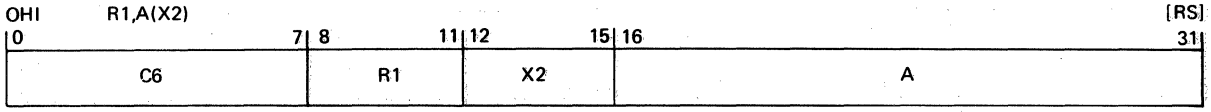
OHR R1,R2 [RR]



OH R1,A(X2) [RX]



OHI R1,A(X2) [RS]



The logical sum of the 16-bit second operand and the content of the General Register specified by R1, replaces the content of R1. The 16-bit sum is formed on a bit-by-bit basis.

OHR: (R1) ← (R1) OR (R2)
OH: (R1) ← (R1) OR [A + (X2)]
OHI: (R1) ← (R1) OR A + (X2)

Resulting Condition Code:

12 C	13 V	14 G	15 L	
		0	0	LOGICAL SUM IS ZERO.
		0	1	LOGICAL SUM IS NOT ZERO.
		1	0	

Programming Note:

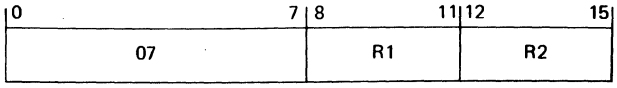
The OR Halfword Immediate (OHI) instruction, produces a value which is the logical sum of the address field itself plus the content of the General Register index (X2) with the first operand General Register (R1).

The truth table for the OR function is:

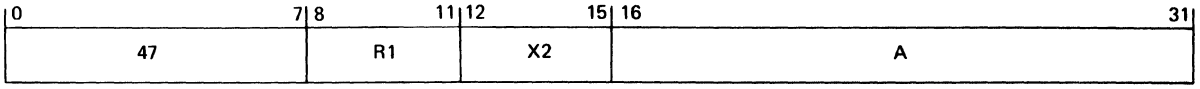
0 OR 0 = 0
0 OR 1 = 1
1 OR 0 = 1
1 OR 1 = 1

4.4.3 Exclusive OR Halfword

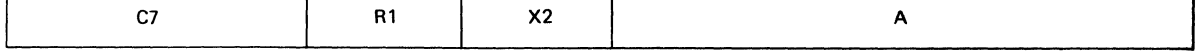
XHR R1,R2 [RR]



XH R1,A(X2) [RX]



XHI R1,A(X2) [RS]



The logical difference of the 16-bit second operand and the General Register specified by R1, replaces the content of R1. The 16-bit difference is formed on a bit-by-bit basis.

XHR: (R1) ← (R1) XOR (R2)
XH: (R1) ← (R1) XOR [A + (X2)]
XHI: (R1) ← (R1) XOR A + (X2)

Resulting Condition Code:

12	13	14	15	
C	V	G	L	
		0	0	LOGICAL DIFFERENCE IS ZERO.
		0	1	LOGICAL DIFFERENCE IS NOT ZERO.
		1	0	LOGICAL DIFFERENCE IS NOT ZERO.

Programming Note:

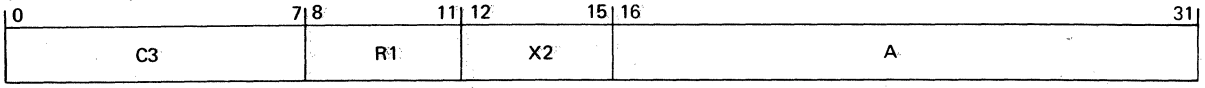
The Exclusive OR Halfword Immediate (XHI) instruction, produces a value which is the logical difference of the address field itself plus the content of the General Register index (X2) with the first operand General Register (R1).

The truth table for the Exclusive OR function is:

0 XOR 0 = 0
0 XOR 1 = 1
1 XOR 0 = 1
1 XOR 1 = 0

4.4.4 Test Halfword Immediate

THI R1,A(X2) [RS]



Each bit in the 16-bit second operand is logically ANDed with the corresponding bit in the General Register specified by R1. The contents of R1 and the second operand remain unchanged.

THI: (R1) AND A + (X2)

Resulting Condition Code:

12	13	14	15	
C	V	G	L	
		0	0	NONE OF THE BITS OF THE RESULT SET.
		0	1	BIT 0 OF THE RESULT SET.
		1	0	ONE OR MORE OF BITS 1-15 OF THE RESULT SET.

Programming Note:

The Test Halfword Immediate (THI) instruction can be used to test the state of individual bits or combinations of bits in a General Register. For example, to test the state of Bit 6 in Register 3, use THI 3,X'0200'.

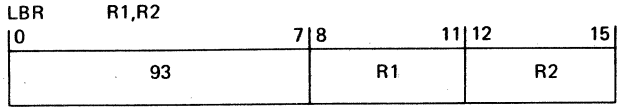
4.5 BYTE HANDLING INSTRUCTIONS

The Byte Handling instructions provide for transferring bytes between core memory and the General Registers. Compare Logical Byte is useful for testing a particular byte within memory. The instructions described in this section are:

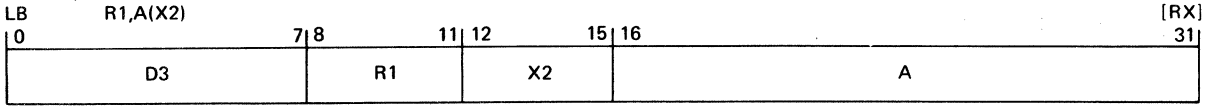
4.5.1	LBR	Load Byte RR
	LB	Load Byte
4.5.2	STBR	Store Byte RR
	STB	Store Byte
4.5.3	EXBR	Exchange Byte RR
4.5.4	CLB	Compare Logical Byte

4.5.1 Load Byte

LBR R1,R2 [RR]



LB R1,A(X2) [RX]



The eight-bit second operand is loaded into the right-most (least significant) eight-bits of the General Register specified by R1. The left-most (most significant) eight-bits of R1 are set to zero. The second operand is unchanged.

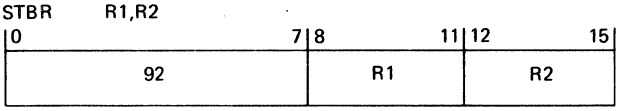
LBR: R1 (8:15) ← [R2 (8:15)]
R1 (0:7) ← Zero
LB: R1 (8:15) ← [A + (X2)]
R1 (0:7) ← Zero

Resulting Condition Code:

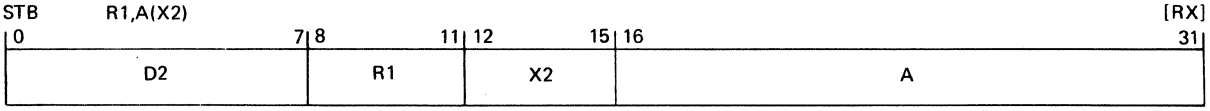
Unchanged.

4.5.2 Store Byte

STBR R1,R2 [RR]



STB R1,A(X2) [RX]



The right-most (least significant) eight-bit byte of the first operand is stored in the General Register or core memory location specified by the second operand. The first operand is unchanged.

STBR: [R1 (8:15)] → R2 (8:15)
STB: [R1 (8:15)] → [A + (X2)]

Resulting Condition Code:

Unchanged.

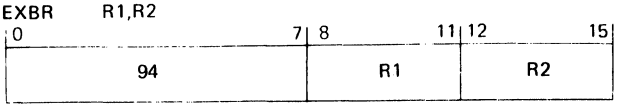
Programming Note:

In the Register-to-Register (RR) form of this instruction, the left-most byte of R2, (0:7), is unchanged.

The RX Store Byte (STB) instruction is subject to Memory Protect.

4.5.3 Exchange Byte

EXBR R1,R2 [RR]



The two eight-bit bytes of the second operand are exchanged and loaded into the General Register specified by R1.

EXBR: R1 (0:7) ← R2 (8:15)
R1 (8:15) ← R2 (0:7)

Resulting Condition Code:

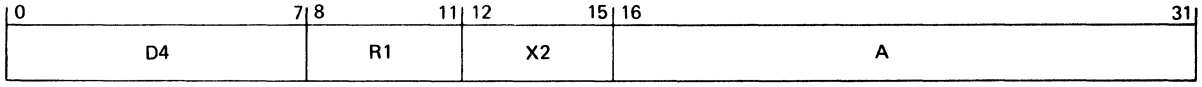
Unchanged.

Programming Note:

R1 and R2 may specify the same General Register.

4.5.4 Compare Logical Byte

CLB R1,A(X2) [RX]



The least significant eight-bit byte of the first operand is logically compared to the eight-bit second operand. The result is indicated by the setting of the Condition Code [PSW (12:15)]. Neither operand is changed.

CLB: R1(8:15) : [A + (X2)]

Resulting Condition Code:

12 C	13 V	14 G	15 L	
		0	0	FIRST OPERAND EQUALS SECOND OPERAND.
		0	1	FIRST OPERAND DOES NOT EQUAL SECOND OPERAND.
		1	0	
1				FIRST OPERAND IS LESS THAN SECOND OPERAND.
0				FIRST OPERAND IS EQUAL TO OR GREATER THAN SECOND OPERAND.

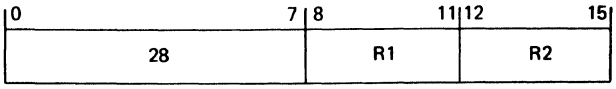
4.6 FLOATING-POINT INSTRUCTIONS

The Floating-Point instructions provide for loading, storing, adding, subtracting, multiplying, dividing, and comparing of floating-point operands. The Arithmetic instructions assume normalized floating-point operands and produce a normalized result. The Floating-Point Load instruction normalizes an un-normalized floating-point number. The data format for the Floating-Point instructions is identical to that of the IBM 360 single-precision floating-point number (see 3.2.3). The R1 and R2 fields of the Floating-Point instructions must specify even Floating-Point Registers (0, 2, 4, 6, etc.). Note that the Floating-Point Registers are reserved core memory locations. Quantities in Floating-Point Registers should be manipulated only with Floating-Point instructions. The instructions described in this section are:

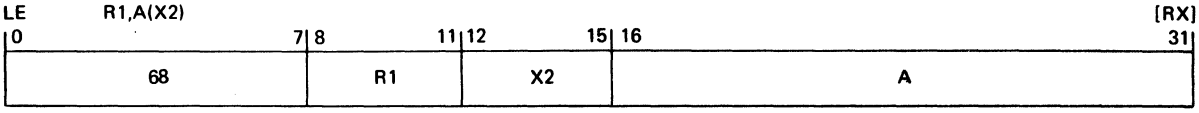
4.6.1	LER	Floating-Point Load RR
	LE	Floating-Point Load
4.6.2	STE	Floating-Point Store
4.6.3	AER	Floating-Point Add RR
	AE	Floating-Point Add
4.6.4	SER	Floating-Point Subtract RR
	SE	Floating-Point Subtract
4.6.5	CER	Floating-Point Compare RR
	CE	Floating-Point Compare
4.6.6	MER	Floating-Point Multiply RR
	ME	Floating-Point Multiply
4.6.7	DER	Floating-Point Divide RR
	DE	Floating-Point Divide

4.6.1 Floating-Point Load

LER R1,R2 [RR]



LE R1,A(X2) [RX]



The floating-point second operand is normalized and placed in the Floating-Point Register specified as the first operand. During normalization, the fraction is shifted left hexadecimally (four-bits at a time) until the most significant hexadecimal digit is not zero. The exponent is decremented by one for each hexadecimal shift required. Zeros are shifted into the least significant hexadecimal digit of the fraction. The second operand is unchanged.

If the normalization causes exponent underflow, the entire floating-point result is set to zero, and the Overflow (V) flag is set.

LER: (R1) ← (R2)
LE: (R1) ← [A + (X2)]

Resulting Condition Code:

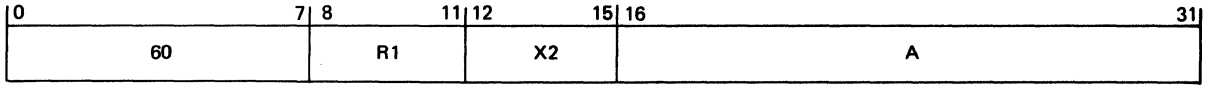
12 C	13 V	14 G	15 L	
		0	0	ZERO.
		0	1	LESS THAN ZERO.
		1	0	GREATER THAN ZERO.
	1	0	0	EXPONENT UNDERFLOW.

Programming Note:

In the event of underflow, the Floating-Point Arithmetic Fault Interrupt is caused, if enabled by Bit 5 of the PSW.

4.6.2 Floating-Point Store

STE R1,A(X2) [RX]



The floating-point first operand is placed in the core memory location specified by A + (X2). The first operand is unchanged.

STE: (R1) → [A + (X2)] .

Resulting Condition Code:

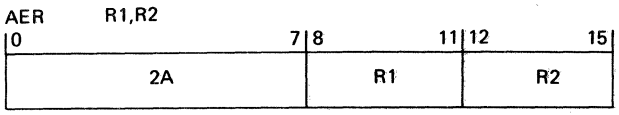
Unchanged.

Programming Note:

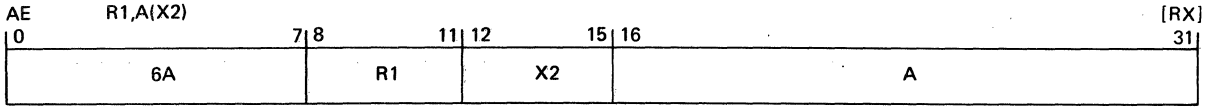
This instruction is subject to Memory Protect.

4.6.3 Floating-Point Add

AER R1,R2 [RR]



AE R1,A(X2) [RX]



The exponents of the two operands are compared. If the exponents differ, the fraction with the smaller exponent is shifted right hexadecimally (four-bits at a time), and its exponent is incremented by one for each hexadecimal shift until the two exponents agree. The fractions are then added algebraically. If a Carry results, the exponent of the sum in incremented by one and the fraction (result) is shifted right one hexadecimal position (four-bits). The Carry is shifted back into the most significant hexadecimal digit of the fraction. If an exponent overflow results, the exponent and fraction of the result are set to all ones, and the Overflow flag is set. The sign of the result is not affected by the overflow.

If no Carry results from the addition of fractions, the sum is normalized. During normalization, the fraction is shifted left hexadecimally (four-bits at a time) until the most significant hexadecimal digit is not zero. The exponent is decremented by one for each hexadecimal shift required. Zeros are shifted into the least significant hexadecimal digit of the fraction.

If the normalization causes exponent underflow; the sign, exponent, and fraction of the sum are set to zero, and the Overflow flag is set. If a zero sum is generated from adding two equal magnitudes with unlike signs, the entire floating-point result is zeroed.

AER: (R1)←(R1) + (R2)
AE: (R1)←(R1) + [A + (X2)]

Resulting Condition Code:

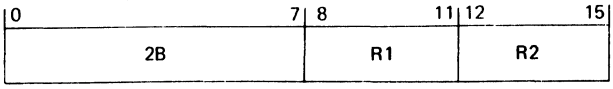
12	13	14	15	
C	V	G	L	
		0	0	SUM IS ZERO.
		0	1	SUM IS LESS THAN ZERO.
		1	0	SUM IS GREATER THAN ZERO.
	1	X	X	EXPONENT OVERFLOW.
	1	0	0	EXPONENT UNDERFLOW.

Programming Note:

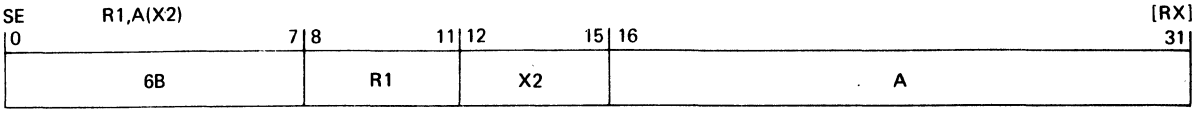
In the event of overflow or underflow, the Floating-Point Arithmetic Fault Interrupt is caused, if enabled by Bit 5 of the PSW.

4.6.4 Floating-Point Subtract

SER R1,R2 [RR]



SE R1,A(X2) [RX]



The exponents of the two operands are compared. If the exponents differ, the fraction with the smaller exponent is shifted right hexadecimally (four-bits at a time), and its exponent is incremented by one for each hexadecimal shift until the two exponents agree. The fractions are then subtracted algebraically. If a Carry results, the exponent of the difference is incremented by one and the fraction (result) is shifted right one hexadecimal position (four-bits). The Carry is shifted into the most significant hexadecimal digit of the fraction. If an exponent overflow occurs, the exponent and fraction of the result are set to all ones, and the Overflow flag is set. The sign of the result is not affected by the overflow.

If no Carry results from the subtraction of fractions, the difference is normalized by shifting the fraction left hexadecimally (four-bits at a time) until the most significant hexadecimal digit is not zero. The exponent is decremented by one for each hexadecimal shift required. Zeros are shifted into the least significant hexadecimal digit of the fraction.

If the normalization causes exponent underflow, the entire floating-point result is set to zero, and the Overflow flag is set.

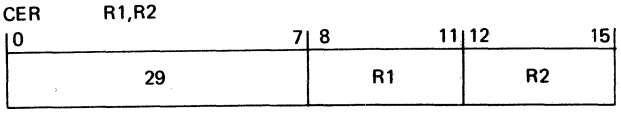
SER: (R1) ◄——— (R1) - (R2)
SE: (R1) ◄——— (R1) - [A + (X2)]

Resulting Condition Code:

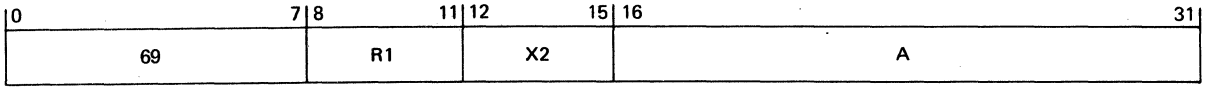
12	13	14	15	
C	V	G	L	
		0	0	DIFFERENCE IS ZERO.
		0	1	DIFFERENCE IS LESS THAN ZERO.
		1	0	DIFFERENCE IS GREATER THAN ZERO.
	1	X	X	EXPONENT OVERFLOW.
	1	0	0	EXPONENT UNDERFLOW.

4.6.5 Floating-Point Compare

CER R1,R2 [RR]



CE R1,A(X2) [RX]



The first operand is compared to the second operand. Comparison is algebraic, taking into account the sign, fraction, and exponent of each number. The result is indicated by the setting of the Condition Code [PSW (12:15)]. Both operands remain unchanged.

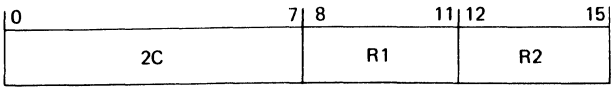
CER: (R1) : (R2)
CE: (R1) : [A + (X2)]

Resulting Condition Code:

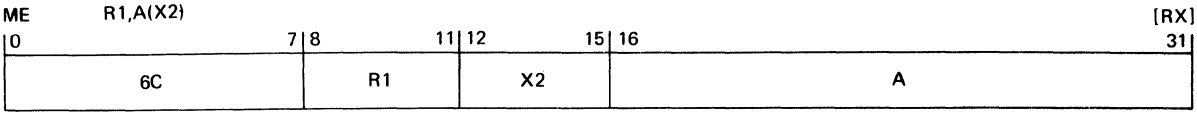
12 C	13 V	14 G	15 L	
		0	0	FIRST OPERAND EQUALS SECOND OPERAND.
		0	1	FIRST OPERAND IS LESS THAN THE SECOND OPERAND.
		1	0	FIRST OPERAND IS GREATER THAN THE SECOND OPERAND.
		0		FIRST OPERAND IS LESS THAN OR EQUAL TO THE SECOND OPERAND.
0				FIRST OPERAND IS GREATER THAN OR EQUAL TO THE SECOND OPERAND.
1				FIRST OPERAND IS LESS THAN THE SECOND OPERAND.

4.6.6 Floating-Point Multiply

MER R1,R2 [RR]



ME R1,A(X2) [RX]



The exponents of the two operands are added to produce the exponent of the result. The resultant exponent is readjusted to excess 64 notation. If an exponent overflow occurs, the exponent and fraction of the product are set to ones, and the Overflow flag is set. The sign of the product is determined by the rules of algebra. If an exponent underflow occurs, the entire floating-point result is set to zero, and the Overflow flag is set. In either event, the Floating-Point Arithmetic Fault Interrupt is caused, if enabled by Bit 5 in the PSW.

If an exponent overflow or underflow does not occur, the multiplication takes place. If the product is zero, the entire floating-point result is zero. If the result is not zero, normalization may occur. During normalization, the fraction is shifted left hexadecimally (four-bits at a time) until the most significant hexadecimal digit is not zero. The exponent of the result is decremented by one for each hexadecimal shift required. After normalization, the product is rounded to 24 bits.

If normalization causes the exponent to underflow, the entire floating-point result is set to zero, and the Overflow flag is set.

MER: $(R1) \leftarrow (R1)*(R2)$
ME: $(R1) \leftarrow (R1)*[A + (X2)]$

Resulting Condition Code:

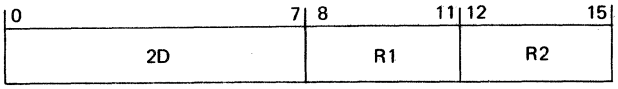
12	13	14	15	
C	V	G	L	
		0	0	PRODUCT IS ZERO.
		0	1	PRODUCT IS LESS THAN ZERO.
		1	0	PRODUCT IS GREATER THAN ZERO.
	1	X	X	EXPONENT OVERFLOW.
	1	0	0	EXPONENT UNDERFLOW.

Programming Note:

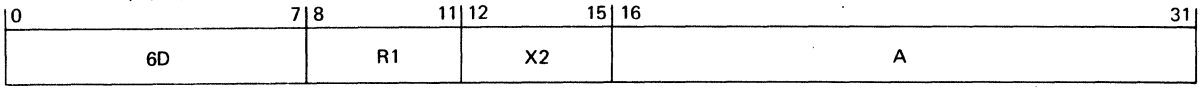
The sum of the exponents of the two operands must be less than 64, or overflow occurs, producing the maximum possible value as a product. For example, the multiplication $1/2 \times 16^{63} * 1 = 1/2 \times 16^{63} * 1/16 \times 16' = 1/32 \times 16^{64}$ causes an overflow, rather than the result $1/2 \times 16^{63}$.

4.6.7 Floating-Point Divide

DER R1,R2 [RR]



DE R1,A(X2) [RX]



The exponents of the two operands are subtracted to produce the exponent of the result. The resultant exponent is readjusted to excess 64 notation. If an exponent overflow occurs, the exponent and fraction of the quotient are set to all ones, and the Overflow flag is set. The sign of the quotient is determined by the rules of algebra. If an exponent underflow occurs, the entire floating-point result is set to zero, and the Overflow flag is set. If the divisor (the second operand) is zero, the operands are unchanged. In the event of exponent overflow, underflow, or division by zero; the Floating-Point Arithmetic Fault Interrupt is caused, if enabled by Bit 5 of the PSW.

If the exponent overflow or underflow does not occur, and if the divisor is not zero, the second operand is divided into the first operand. Division continues until the quotient is normalized, adjusting the exponent for each additional division required. If an exponent underflow occurs, the entire floating-point result is set to zero, and the Overflow flag is set.

No remainder is returned to the user. The quotient is rounded to compensate for the loss of the remainder.

DER: $(R1) \leftarrow (R1)/(R2)$
DE: $(R1) \leftarrow (R1)/[A + (X2)]$

Resulting Condition Code:

12	13	14	15	
C	V	G	L	
		0	0	QUOTIENT IS ZERO.
		0	1	QUOTIENT IS LESS THAN ZERO.
		1	0	QUOTIENT IS GREATER THAN ZERO.
	1	X	X	EXPONENT OVERFLOW.
	1	0	0	EXPONENT UNDERFLOW.
1	1	0	0	DIVISOR EQUAL TO ZERO.

Programming Note:

Division by zero, overflow, or underflow cause a Floating-Point Arithmetic Fault Interrupt, if enabled by Bit 5 of the PSW. Inspection of the Condition Code of the Old PSW indicates the actual cause of the interrupt. If the Carry flag is set, then the divisor was zero. If the Carry flag is not set, then either overflow or underflow caused the interrupt. In this case, if the Greater than (G) or Less than (L) flag is set, the interrupt was caused by an overflow. If the G or L flag is reset, the interrupt was caused by an underflow.

The difference of the exponents of the two operands must be less than 64, or overflow occurs, producing the maximum possible value as a quotient, even when normalization of the computed mantissa would bring the resultant exponent within range.

4.7 SHIFT/ROTATE INSTRUCTIONS

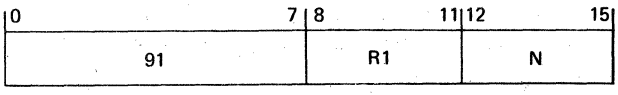
The Shift/Rotate instructions provide for arithmetic and logical manipulation of information contained in the General Registers. Bits shifted out of the high or low order and of a General Register are passed through the Carry Bit position of the Condition Code (PSW 12). After execution of a Shift instruction, the last bit which was shifted out is contained in the Carry position.

A shift of zero positions causes the Condition Code to be set properly with no alteration to the information contained in the General Register. The instructions described in this section are:

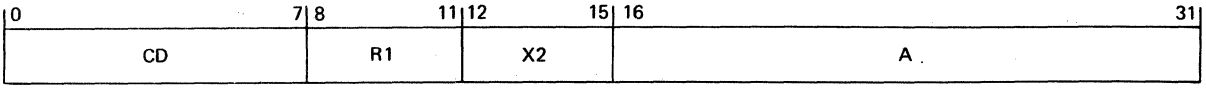
4.7.1	SLLS	Shift Left Logical Short
	SLHL	Shift Left Halfword Logical
	SLL	Shift Left Logical
4.7.2	SRLS	Shift Right Logical Short
	SRHL	Shift Right Halfword Logical
	SRL	Shift Right Logical
4.7.3	RLL	Rotate Left Logical
4.7.4	RRL	Rotate Right Logical
4.7.5	SLHA	Shift Left Halfword Arithmetic
	SLA	Shift Left Arithmetic
4.7.6	SRHA	Shift Right Halfword Arithmetic
	SRA	Shift Right Arithmetic

4.7.1 Shift Left Logical

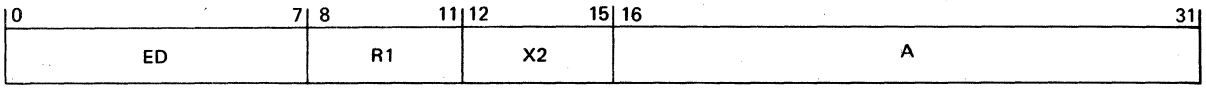
SLLS R1,N [SF]



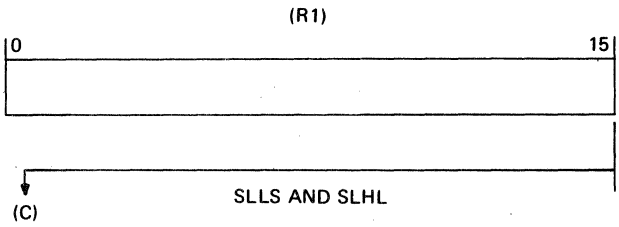
SLHL R1,A(X2) [RS]

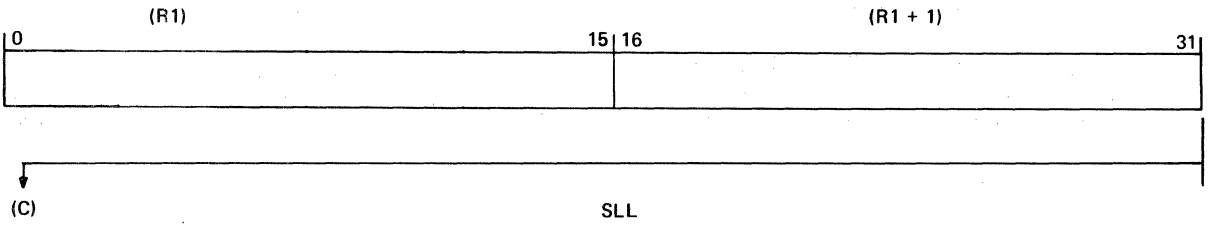


SLL R1,A(X2) [RS]



The content of the first operand is shifted left the number of positions specified by the second operand. High order bits shifted out of Position 0 are shifted through the Carry Bit of the PSW and then lost. Zeros are shifted into the low order bit position.





Resulting Condition Code:

12	13	14	15	
C	V	G	L	
		0	0	RESULT IS ZERO.
		0	1	RESULT IS LESS THAN ZERO.
		1	0	RESULT IS GREATER THAN ZERO.
0				LAST BIT THAT WAS SHIFTED OUT WAS A ZERO.
1				LAST BIT THAT WAS SHIFTED OUT WAS A ONE.

Programming Note:

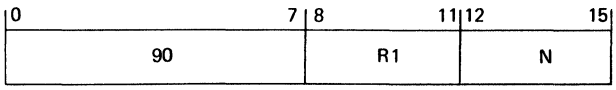
For the Shift Left Logical Short (SLLS) instruction, the N field (Bits 12 through 15) of the instruction specify the number of positions the content of R1 is to be shifted.

For the Shift Left Halfword Logical (SLHL) instruction, only the low order four-bits (12 through 15) of A + (X2) are used for the shift count.

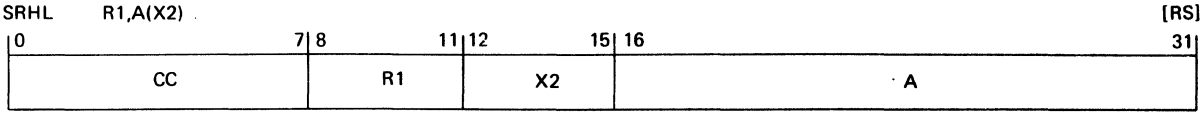
The Shift Left Logical (SLL) instruction, shifts Registers R1 and R1 + 1, an even-odd pair. The R1 field of the instruction must specify an even register. The shift count is specified by the low order five-bits (11 through 15) of the value A + (X2). The Carry is formed by the output of R1.

4.7.2 Shift Right Logical

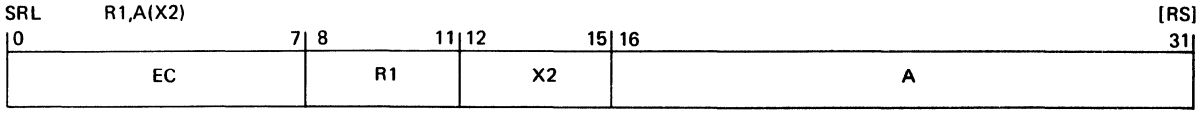
SRLS R1,N [SF]



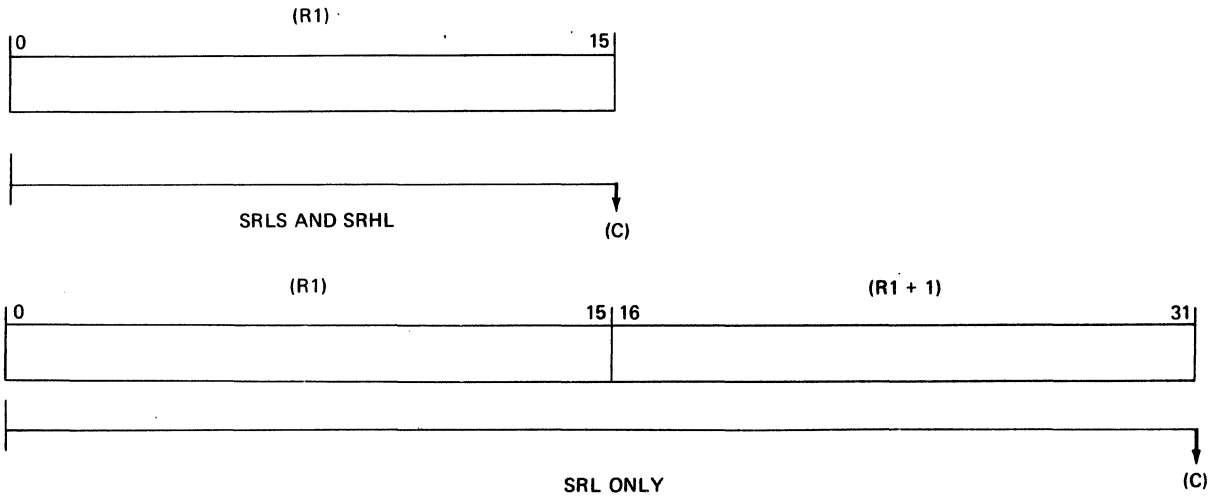
SRHL R1,A(X2) [RS]



SRL R1,A(X2) [RS]



The content of the first operand is shifted right the number of bit positions specified by the second operand. Low order bits shifted out of Position 15 are shifted thru the Carry Bit of the PSW and then lost. Zeros are shifted into Position 0.



Resulting Condition Code:

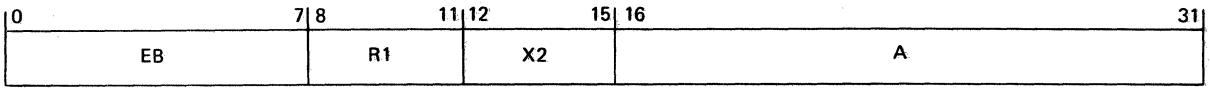
12	13	14	15	
C	V	G	L	
		0	0	RESULT IS ZERO.
		0	1	RESULT IS LESS THAN ZERO.
		1	0	RESULT IS GREATER THAN ZERO.
0				LAST BIT THAT WAS SHIFTED OUT WAS A ZERO.
1				LAST BIT THAT WAS SHIFTED OUT WAS A ONE.

Programming Note:

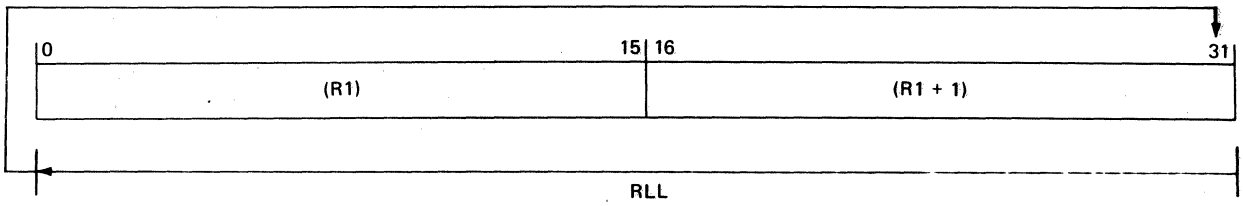
See Shift Left Logical 4.7.1.

4.7.3 Rotate Left Logical

RLL R1,A(X2) [RS]



The 32-bit first operand specified by R1 is shifted left, end around, the number of positions specified by the low order five bits of the value A + (X2). All 32 bits of the fullword are shifted. Bits shifted out of Position 0 are shifted into Position 31. A shift specification of 16-bits interchanges the two halves (R1, R1 + 1) of the first operand.



Resulting Condition Code:

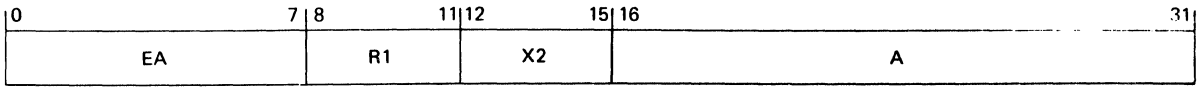
12	13	14	15	
C	V	G	L	
		0	0	RESULT IS ZERO.
		1	0	RESULT IS GREATER THAN ZERO.
		0	1	RESULT IS LESS THAN ZERO.

Programming Note:

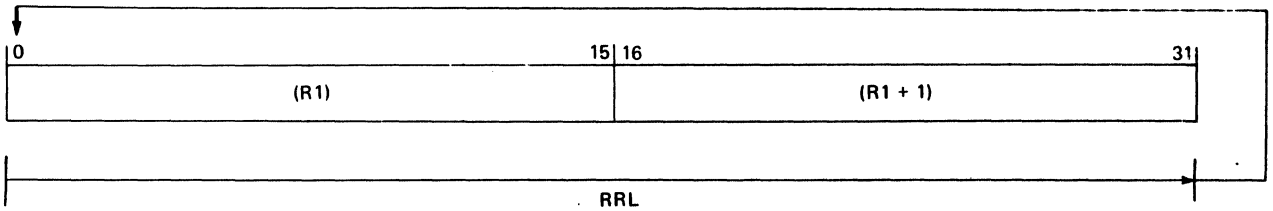
The Rotate Left Logical (RLL) instruction, rotates Registers R1 and R1 + 1, an even-odd pair. The R1 field of the instruction must specify an even register.

4.7.4 Rotate Right Logical

RRL R1,A(X2) [RS]



The 32-bit first operand specified by R1 is shifted right, end around, the number of positions specified by the low order five bits of the value A + (X2). All 32 bits of the fullword are shifted. Bits shifted out of position 31 are shifted into Position 0. A shift specification of 16-bits interchanges the two halves (R1, R1 + 1) of the first operand.



Resulting Condition Code:

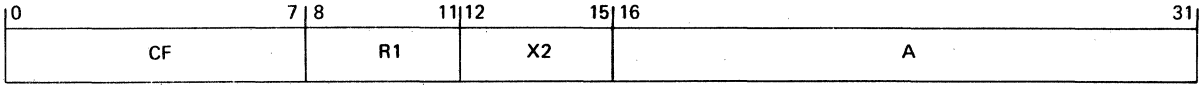
12	13	14	15	
C	V	G	L	
		0	0	RESULT IS ZERO.
		1	0	RESULT IS GREATER THAN ZERO.
		0	1	RESULT IS LESS THAN ZERO.

Programming Note:

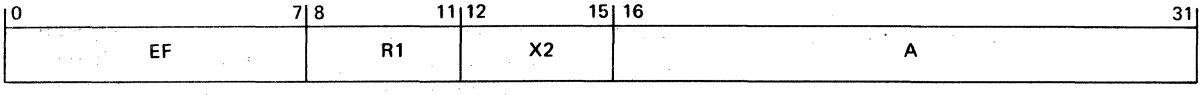
The Rotate Right Logical (RRL) instruction, rotates Registers R1 and R1 + 1, an even-odd pair. The R1 field of the instruction must specify an even register.

4.7.5 Shift Left Arithmetic

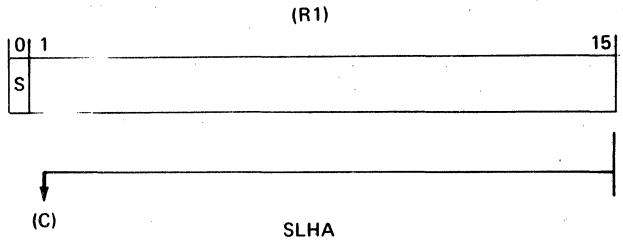
SLHA R1,A(X2) [RS]

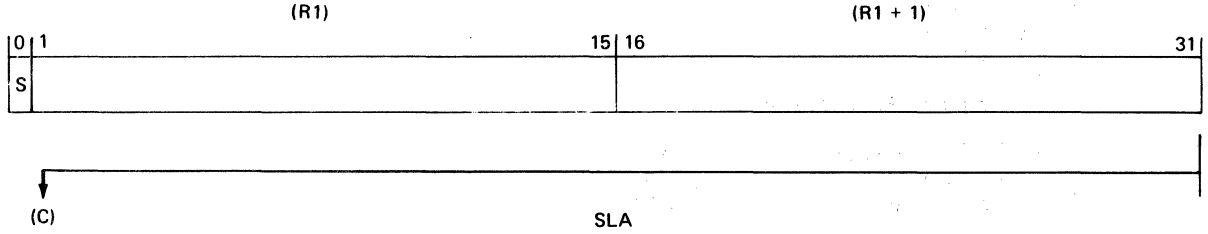


SLA R1,A(X2) [RS]



The content of the first operand is shifted left the number of bit positions specified by the second operand. The Sign Bit is unchanged. High order bits shifted out of Position 1 are shifted through the Carry Bit of the PSW and then lost. Zeros are shifted into the low order bit position.





Resulting Condition Code:

12	13	14	15	
C	V	G	L	
		0	0	RESULT IS ZERO.
		0	1	RESULT IS LESS THAN ZERO.
		1	0	RESULT IS GREATER THAN ZERO.
0				LAST BIT THAT WAS SHIFTED OUT WAS A ZERO.
1				LAST BIT THAT WAS SHIFTED OUT WAS A ONE.

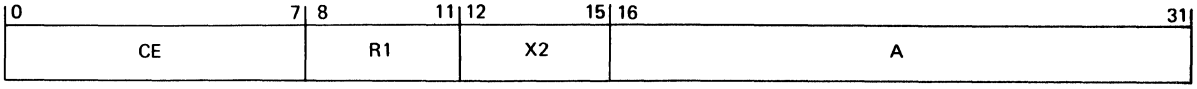
Programming Note:

For the Shift Left Halfword Arithmetic (SLHA) instruction, the shift count is specified by the low order four-bits (12 through 15) of the value of A + (X2).

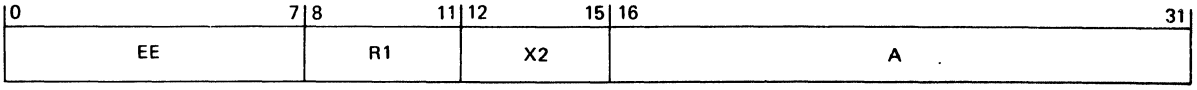
The Shift Left Arithmetic (SLA) instruction, shifts Registers R1 and R1 + 1, an even-odd pair. R1 must specify an even register. The shift count is specified by the low order five-bits (11 through 15) of the value of A + (X2).

4.7.6 Shift Right Arithmetic

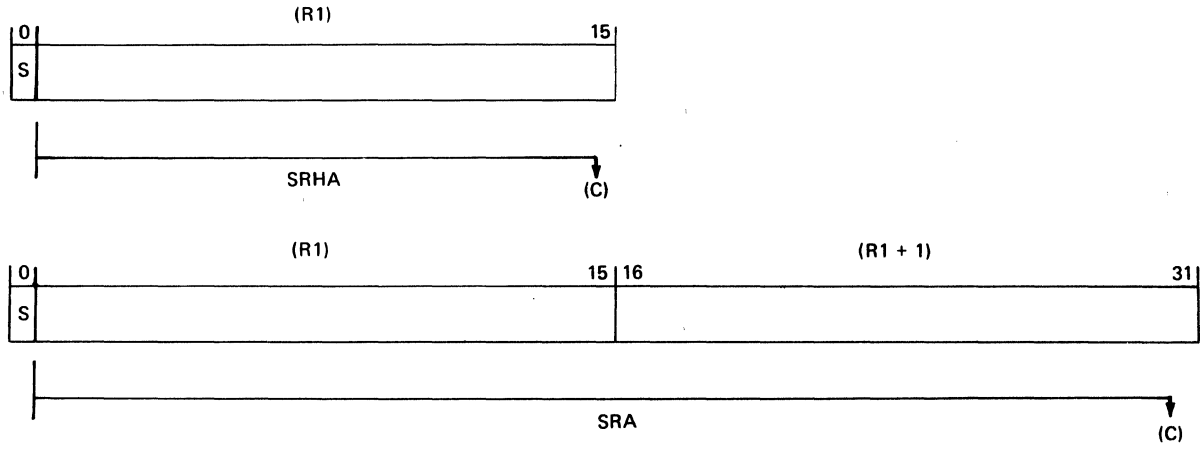
SRHA R1,A(X2) [RS]



SRA R1,A(X2) [RS]



The content of the first operand is shifted right the number of bit positions specified by the second operand. The Sign Bit, Bit 0, of R1 is unchanged and is shifted right into Bit 1; therefore, Bit 0, is propagated right as many positions as specified by the second operand. Low order bits of the first operand are shifted through the Carry Bit of the PSW and then lost.



Resulting Condition Code:

12	13	14	15	
C	V	G	L	
		0	0	RESULT IS ZERO.
		0	1	RESULT IS LESS THAN ZERO.
		1	0	RESULT IS GREATER THAN ZERO.
0				LAST BIT THAT WAS SHIFTED OUT WAS A ZERO.
1				LAST BIT THAT WAS SHIFTED OUT WAS A ONE.

Programming Note:

For the Shift Right Halfword Arithmetic (SRHA) instruction, the shift count is specified by the low order four-bits (12 through 15) of the value of A + (X2).

The Shift Right Arithmetic (SRA) instruction, shifts Registers R1 and R1 + 1, an even odd pair. R1 must specify an even register. The shift count is specified by the low order five-bits (11 through 15) of the value of A + (X2). The Carry is formed by the output of R1 + 1 instead of R1.

4.8 BRANCH INSTRUCTIONS

Branch instructions are programmed decisions providing entry to subprograms, as well as testing the result of arithmetic, logical, or indexing operations.

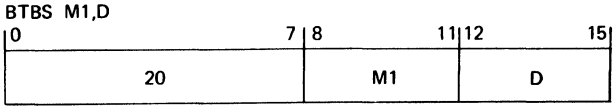
Many Processor operations result in setting of the Condition Code in the Program Status Word [PSW (12:15)]. The Branch on Condition instructions implement the testing of the Condition Code through use of a mask field contained in the instruction itself (M1 field).

The four-bit M1 field is not a register address, but rather an image of the Condition Code to be tested. The instructions described in this section are:

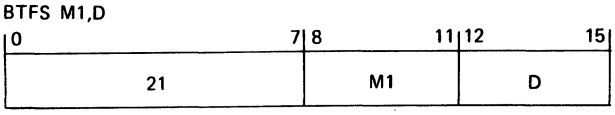
4.8.1	BTBS	Branch on True Backward Short
	BTFS	Branch on True Forward Short
	BTCR	Branch on True Condition RR
	BTC	Branch on True Condition
4.8.2	BFBS	Branch on False Backward Short
	BFFS	Branch on False Forward Short
	BFCR	Branch on False Condition RR
	BFC	Branch on False Condition
4.8.3	BXH	Branch on Index High
	BXLE	Branch on Index Low or Equal
4.8.4	BALR	Branch and Link RR
	BAL	Branch and Link

4.8.1 Branch on True Condition

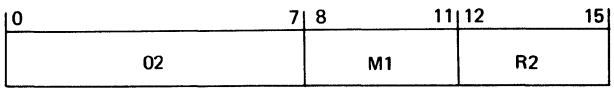
BTBS M1,D [SF]



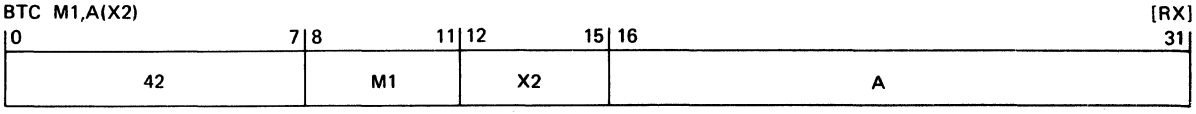
BTFS M1,D [SF]



BTCR M1,R2 [RR]



BTC M1,A(X2) [RX]



The Condition Code field of the Program Status Word PSW (12:15) is tested for the condition specified by the Mask Field (M1). If any of the conditions tested are found to be true, a Branch is executed to the 16-bit address specified by the second operand. If none of the conditions tested are found to be true the next sequential instruction is executed.

Tested Condition True:

BTBS: [PSW (16:31)] ← [PSW (16:31)]-2D
BTFS: [PSW (16:31)] ← [PSW (16:31)]+2D
BTCR: [PSW (16:31)] ← (R2)
BTC: [PSW (16:31)] ← A + (X2)

Tested Condition False:

BTBS:
BTFS: } [PSW (16:31)] ← [PSW (16:31)]+2
BTCR:
BTC: [PSW (16:31)] ← PSW (16:31)+4

Programming Note:

A logical AND is performed between each bit in the Condition Code and its corresponding bit in the M1 field. If any resultant bit is a one, the Branch will occur. The Condition Code [PSW (12:15)] is not changed. For example, if the Condition Code is 1010 and the M1 field is 1000, the Branch occurs with Branch on True instructions.

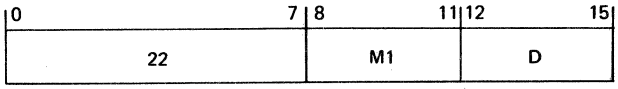
The Branch on True Backward Short (BTBS) instruction, causes a Branch to an address relative to the present Location Counter when the tested condition is true. The displacement is specified by the D field (Bits 12 through 15) of the instruction. The D field (times two) is subtracted from the present Location Counter to generate the address of the next instruction.

The Branch on True Forward Short (BTFS) instruction, causes a Branch to an address relative to the present Location Counter when the tested condition is true. The displacement is specified by the D field (Bits 12 through 15) of the instruction. The D field (times two) is added to the present Location Counter to generate the address of the next instruction.

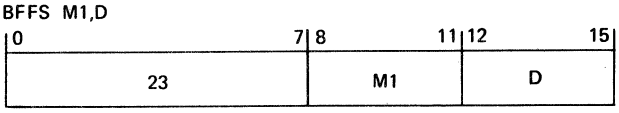
The Short Branch instructions (e.g. BTBS), are appropriate for Branches which specify small displacements from the present Location Counter, for example, in sense status loops used for program controlled I/O.

4.8.2 Branch on False Condition

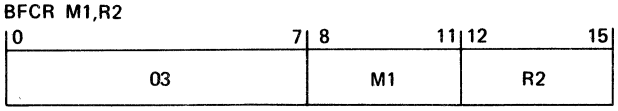
BFBS M1,D [SF]



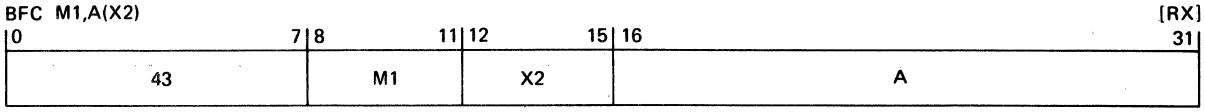
BFFS M1,D [SF]



BFCR M1,R2 [RR]



BFC M1,A(X2) [RX]



The Condition Code field of the Program Status Word [PSW (12:15)] is tested for the condition specified by the mask field (M1). If all conditions tested are found to be false, a Branch is executed to the 16-bit address specified by the second operand. If any of the conditions tested are found to be true the next sequential instruction is executed.

Tested Condition False

BFBS: PSW (16:31) ← [PSW (16:31)] -2D
BFFS: PSW (16:31) ← [PSW (16:31)] +2D
BFCR: PSW (16:31) ← (R2)
BFC: PSW (16:31) ← A + (X2)

Tested Condition True

BFBS:
BFFS: } PSW (16:31) ← [PSW (16:31)] +2
BFCR:
BFC: PSW (16:31) ← [PSW (16:31)] +4

Programming Note:

A logical AND is performed between each bit in the Condition Code and its corresponding bit in the M1 field. If any resultant bit is a one, the Branch will not occur. The Condition Code [PSW (12:15)] is not changed. For example, if the Condition Code is 1010 and the M1 field is 1100, the Branch does not occur with the Branch on False instruction.

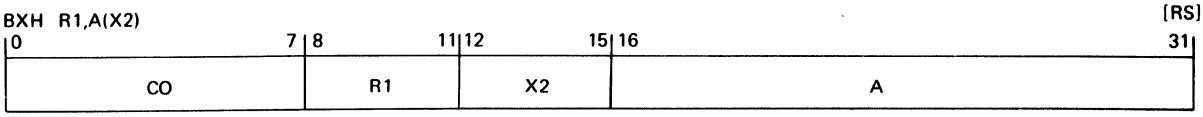
The Branch on False Backward Short (BFBS) instruction, causes a Branch to an address relative to the present Location Counter when the tested condition is false. The displacement is specified by the D field (Bits 12 through 15) of the instruction. The D field (times two) is subtracted from the present Location Counter to generate the address of the next instruction.

The Branch on False Forward Short (BFFS) instruction, causes a Branch to an address relative to the present Location Counter when the tested condition is false. The displacement is specified by the D field (Bits 12 through 15) of the instruction. The D field (times two) is added to the present Location Counter to generate the address of the next instruction.

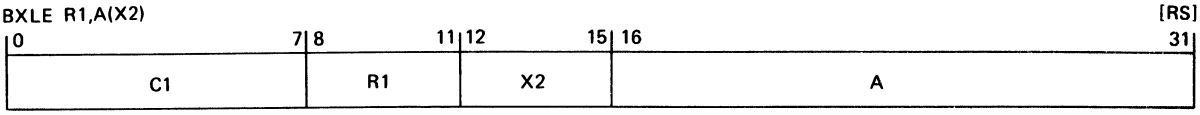
Branch on False Condition with a mask of 0 is an Unconditional Branch.

4.8.3 Branch on Index

BXH R1,A(X2) [RS]



BXLE R1,A(X2) [RS]



Prior to execution of this instruction, the General Register specified by the first operand (R1) must contain a 16-bit starting index value, R1 + 1 must contain a 16-bit increment value, and R1 + 2 must contain a 16-bit comparand (limit or final value). All values may be signed.

Execution of this instruction causes the index (R1) to be incremented by (R1 + 1) and logically compared to the index limit, (R1 + 2).

BXH: (R1) ← (R1) + (R1 + 1)
(R1) : (R1 + 2)
if (R1) > (R1 + 2), then [PSW (16:31)] ← A + (X2)
if (R1) ≤ (R1 + 2), then [PSW (16:31)] ← [PSW (16:31)]+4

BXLE: (R1) ← (R1) + (R1 + 1)
(R1) : (R1 + 2)
if (R1) ≤ (R1 + 2), then [PSW (16:31)] ← A + (X2)
if (R1) > (R1 + 2), then [PSW (16:31)] ← [PSW (16:31)]+4

Resulting Condition Code:

Unchanged.

Programming Note:

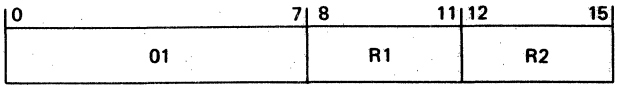
For the Branch on Index High (BXH) instruction, the contents of R1 + 1 should be negative. As long as the index (R1) is greater than the limit (R1 + 2), the 16-bit address specified by the second operand is transferred to the instruction address field of the Program Status Word [PSW (16:31)] The next instruction executed will be accessed from the location specified by the new instruction address. When the count is not greater than the index limit, the instruction following Branch on Index High will be executed.

For the Branch on Index Low or Equal (BXLE) instruction, the contents of R1 + 1 should be positive. As long as the index (R1) is equal to or less than the limit (R1 + 2), the 16-bit address specified by the second operand is transferred to the instruction address field of the Program Status Word [PSW (16:31)] . The next instruction executed will be accessed from the location specified by the new instruction address. When the count is greater than the limit, the instruction following Branch on Index Low will be executed.

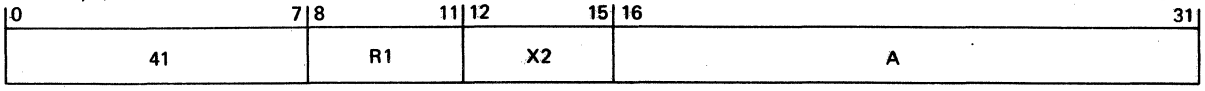
The Branch on Index High and Branch on Index Low instructions are appropriate for rapid loop control, particularly when one or more of the instructions in the loop is indexed.

4.8.4 Branch and Link

BALR R1,R2 [RR]



BAL R1,A(X2) [RX]



The address of the next sequential instruction is saved in the General Register specified by the first operand (R1), and an unconditional branch is executed to the 16-bit address specified by the second operand. In all 3000 series Processors except the GE-PAC 30-1, the effective second operand is derived before the contents of register R1 are changed. See note below.

BALR: (R1) ← [PSW (16:31)] +2
PSW (16:31) ← (R2)

BAL: (R1) ← [PSW (16:31)] +4
PSW (16:31) ← A + (X2)

Condition Code:

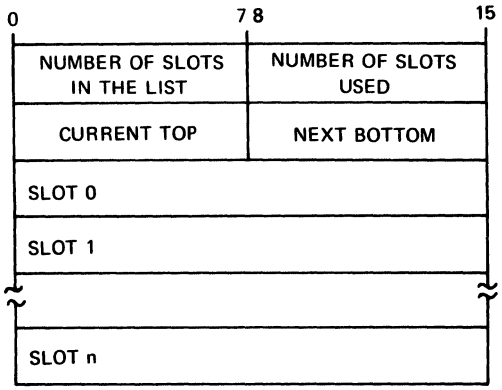
Unchanged.

Programming Note:

The Branch and Link instruction may be used for entry to sub-programs. It differs from the Branch Unconditional instruction in that the incremented Location Counter value is preserved in a specified General Register to be used as the sub-program exit address. Exit from the sub-program is effected by a Branch Unconditional instruction through the General Register in which the exit address has been maintained. Note that in the 30-1, if the same register is specified in both the first and second operands of the BALR instruction (R1 = R2), R1 will be loaded with the saved address of the next instruction before it is used to derive the second operand. In all other GE-PAC 30 and 3010 Processors, including the 3010/2, the effective second operand is derived before the contents of R1 are changed.

4.9 LIST INSTRUCTIONS

The List instructions manipulate a circular list defined as follows:



The first two halfwords contain the list parameters. Immediately following the parameter block is the list itself. The first halfword in the list is designated Slot 0. The remaining slots are designated 1, 2, 3, etc. up to a maximum slot number which is equal to the number in the list minus one. An absolute maximum of 255 halfword slots is specifiable. (Maximum slot designation is equal to X'FE'.)

The first parameter byte indicates the number of slots (halfwords) in the entire list. The second parameter byte indicates the current number of slots being used. When this byte equals zero, the list is empty; when this byte equals the number of slots in the list, the list is full. Once initialized, this byte is maintained automatically. It is incremented when elements are added to the list and decremented when elements are removed.

The third and fourth bytes of the list parameters specify the current top of the list and the next bottom of the list respectively. These pointers are also updated automatically. See Figure 4-1.

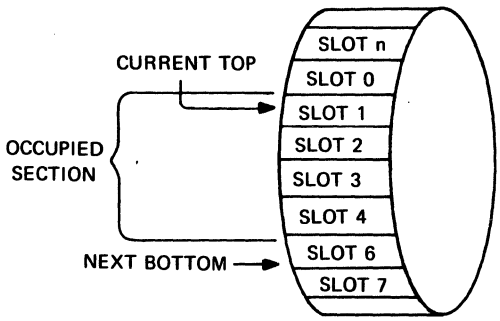


Figure 4-1. Circular List

The instructions described in this section are:

4.9.1 ATL Add to Top of List

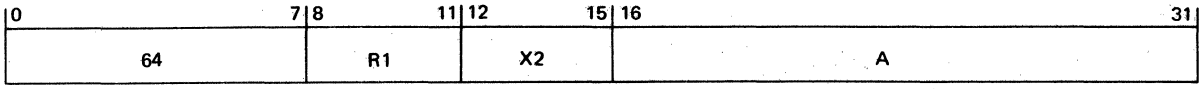
ABL Add to Bottom of List

4.9.2 RTL Remove from Top of List

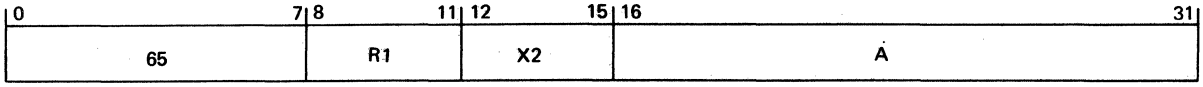
RBL Remove from Bottom of List

4.9.1 Add to Top/Bottom of List

ATL R1,A(X2) [RX]



ABL R1,A(X2) [RX]



The General Register specified by R1 contains the element to be added to the list. The second operand, A + (X2), specifies the address of the list. The number of slots used tally is compared to the number of slots in the list as specified by the first byte of the list. If the number of slots used tally is equal to the number of slots in the list an overflow condition exists. The element is not added to the list and the instruction terminates with the V flag set in the PSW. If the number of slots used tally is less than the number of slots in the list; it is incremented by one, the appropriate pointer is changed, the element is added to the list, and the instruction terminates with a Condition Code of zero.

Resulting Condition Code:

12	13	14	15	
C	V	G	L	
0	1	0	0	LIST OVERFLOW.
0	0	0	0	ELEMENT ADDED SUCCESSFULLY.

Programming Note:

The Add to Top of List (ATL) instruction, manipulates the Current Top Pointer in the list. If no overflow occurred, the Current Top Pointer, which points to the last element added to the top of the list, is decremented by one and the element is inserted in the slot pointed to by the new Current Top Pointer. If the Current Top Pointer was zero on entering this instruction the Current Top Pointer is set to the maximum slot number in the list. This condition is referred to as list wrap.

The Add to Bottom of List (ABL) instruction, manipulates the Next Bottom Pointer. If no overflow occurred, the element is inserted in the slot pointed to by the Next Bottom Pointer, and the Next Bottom Pointer is incremented by one. If the incremented Next Bottom Pointer is greater than the maximum slot number in the list, the Next Bottom Pointer is set to zero. This condition is referred to as list wrap.

This instruction is subject to Memory Protect.

4.9.2 Remove From Top/Bottom of List

RTL R1,A(X2) [RX]

0 – 7	8 – 11	12 – 15	16 – 31
66	R1	X2	A

RBL R1,A(X2) [RX]

0 – 7	8 – 11	12 – 15	16 – 31
67	R1	X2	A

The element removed from the list is placed in the General Register specified by R1. The second operand, A + (X2), specifies the address of the list. If, on entering the instruction the number of slots used tally is zero, the list is already empty and the instruction terminates with V flag set in the PSW. This condition is referred to as list underflow. If underflow does not occur the number of slots used tally is decremented by one, the appropriate pointer is changed, and the element is extracted and placed in R1. The instruction terminates with the Condition Code equal to zero if the list is now empty, or with the G flag set if the list is not yet empty.

Resulting Condition Code:

12	13	14	15	
C	V	G	L	
0	1	0	0	LIST WAS ALREADY EMPTY.
0	0	0	0	LIST IS NOW EMPTY.
0	0	1	0	LIST IS NOT YET EMPTY.

Programming Note:

The Remove from Top of List (RTL) instruction, manipulates the Current Top Pointer. If no underflow occurred, the Current Top Pointer points to the element to be extracted. The element is extracted and placed in R1. The Current Top Pointer is incremented and compared to the maximum slot number. If the Current Top Pointer is greater than the maximum slot number, the Current Top Pointer is set to zero. This condition is referred to as list wrap.

The Remove from Bottom of List (RBL) instruction, manipulates the Next Bottom Pointer. If no underflow occurred, and the Next Bottom Pointer is zero it is set to the maximum slot number (list wrap); otherwise it is decremented by one and the element now pointed to is extracted and placed in R1.

This instruction is subject to Memory Protect.

4.10 INPUT/OUTPUT INSTRUCTIONS

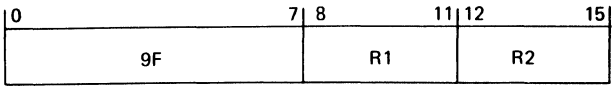
The I/O instructions provide for the transfer of data between the Processor and the peripheral devices on the Multiplexor Bus. All of the instructions described in this section are privileged and, if executed with the Processor in Protect Mode (PSW Bit 7 set), result in an Illegal Instruction Interrupt.

Following most I/O instructions, the V flag in the Condition Code indicates an instruction time-out. That is, due to an improper device response - either the addressed device does not exist, or it did not respond correctly - the specified I/O operation was not performed. Following Sense Status or Acknowledge Interrupt instructions, the Condition Code (CVGL) also reflects Bits 4 through 7 of the device status. With standard 3 0 1 0 / 2 device controllers, Bit 5 of the status byte, which is reflected in the V flag in the Condition Code, is defined as Examine Status. This means that status byte should be examined. Following Sense Status and Acknowledge Interrupt instructions, therefore, the occurrence of the V flag with status Bits 0 through 3 equal zero indicates instruction time-out. For a complete definition of the bits in either command bytes, or status bytes, refer to documentation on the device in question. The instructions described in this section are:

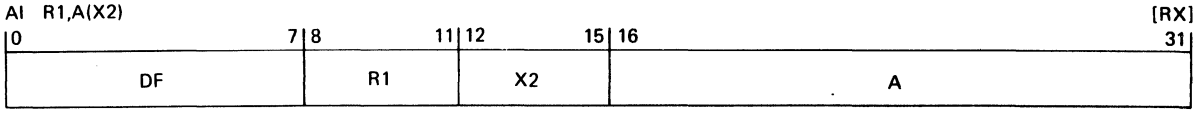
4.10.1	AIR	Acknowledge Interrupt RR	4.10.6	RBR	Read Block RR
	AI	Acknowledge Interrupt		RB	Read Block
4.10.2	SSR	Sense Status RR	4.10.7	WBR	Write Block RR
	SS	Sense Status		WB	Write Block
4.10.3	OCR	Output Command RR	4.10.8	RHR	Read Halfword RR
	OC	Output Command		RH	Read Halfword
4.10.4	RDR	Read Data RR	4.10.9	WHR	Write Halfword RR
	RD	Read Data		WH	Write Halfword
4.10.5	WDR	Write Data RR	4.10.10	AL	Autoload
	WD	Write Data			

4.10.1 Acknowledge Interrupt

AIR R1,R2 [RR]



AI R1,A(X2) [RX]



The address of the interrupting device replaces the content of the 16-bit General Register specified by the first operand (R1). The eight-bit device status byte replaces the content of the location specified by the second operand. The Condition Code is set equal to the right-most four bits in the device status byte. The device interrupt condition is then cleared.

AIR: [R1 (8:15)] ← Device address
[R1 (0:7)] ← Zero
[R2 (8:15)] ← Status byte
[R2 (0:7)] ← Zero
[PSW (12:15)] ← Status byte (4:7)

AI: [R1 (8:15)] ← Device number
[R1 (0:7)] ← Zero
[A + (X2)] ← Status byte
[PSW (12:15)] ← Status byte (4:7)

Resulting Condition Code:

12	13	14	15	
C	V	G	L	
1				DEVICE BUSY (BSY)
	1			EXAMINE STATUS (EX) OR TIME OUT
		1		END OF MEDIUM (EOM)
			1	DEVICE UNAVAILABLE (DU)

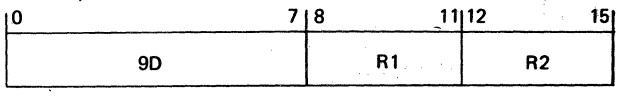
Programming Note:

These instructions are privileged.

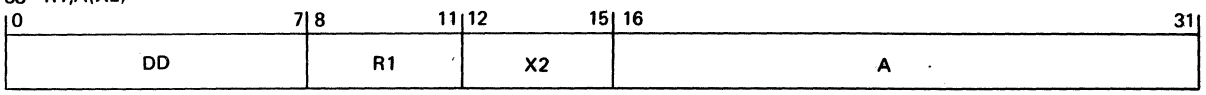
The RX form (AI) is subject to Memory Protect.

4.10.2 Sense Status

SSR R1,R2 [RR]



SS R1,A(X2) [RX]



The 16-bit General Register specified by the first operand (R1) contains the device address. The device is addressed and the eight-bit device status byte replaces the content of the location specified by the second operand. The Condition Code is set equal to the right-most four bits of the device status byte. The first operand is unchanged.

SSR: [R2 (8:15)] ← Status byte
[R2 (0:7)] ← Zero
[PSW (12:15)] ← Status byte (4:7)

SS: [A + (X2)] ← Status byte
[PSW (12:15)] ← Status byte (4:7)

Resulting Condition Code:

12 C	13 V	14 G	15 L	
1				DEVICE BUSY (BSY)
	1			EXAMINE STATUS (EX) OR TIME OUT
		1		END OF MEDIUM (EOM)
			1	DEVICE UNAVAILABLE (DU)

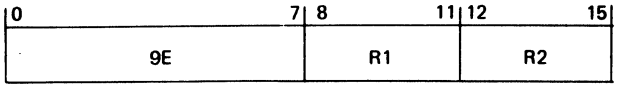
Programming Note:

These instructions are privileged.

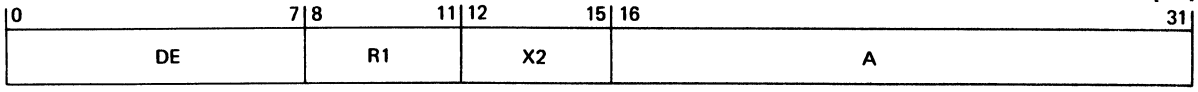
The RX form (SS) is subject to Memory Protect.

4.10.3 Output Command

OCR R1,R2 [RR]



OC R1,A(X2) [RX]



The 16-bit General Register specified by the first operand (R1) contains the device address. The device is addressed and the eight-bit device command byte specified by the second operand is transmitted to the addressed device. Both operands remain unchanged.

OCR: Device ← [R2 (8:15)]
OC: Device ← [A + (X2)]

Resulting Condition Code:

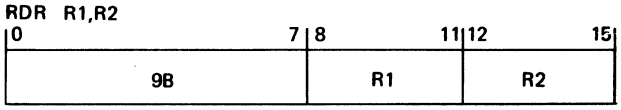
12	13	14	15	
C	V	G	L	
0	1	0	0	INSTRUCTION TIME OUT

Programming Note:

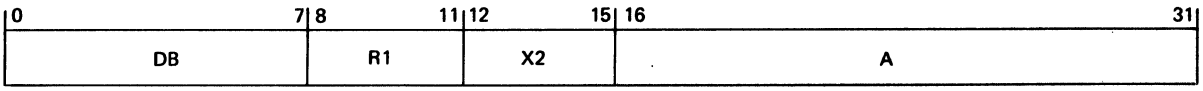
The Examine Status bit is set if the device cannot complete the command action. These instructions are privileged.

4.10.4 Read Data

RDR R1,R2 [RR]



RD R1,A(X2) [RX]



The 16-bit General Register specified by the first operand (R1) contains the device address. The device is addressed and a single eight-bit data byte is transmitted from the device replacing the content of the location specified by the second operand.

RDR: [R2 (8:15)] ← Data byte
[R2 (0:7)] ← Zero
RD: [A + (X2)] ← Data byte

(Continued on next page)

Resulting Condition Code:

12	13	14	15	
C	V	G	L	
	1			INSTRUCTION TIME OUT

Programming Note:

These instructions are privileged.
The RX form (RD) is subject to Memory Protect.

These instructions should not be used with 16-bit oriented device controllers. Note that standard 3010/2 peripheral devices use 8-bit oriented device controllers. For 16-bit oriented devices, use Read Halfword/Write Halfword instructions.

4.10.5 Write Data

WDR R1,R2 [RR]

0	7	8	11	12	15
9A		R1		R2	

WD R1,A(X2) [RX]

0	7	8	11	12	15	16	31
DA		R1		X2		A	

The 16-bit General Register specified by the first operand (R1) contains the device address. The device is addressed and a single eight-bit data byte is transmitted to the device. Both operands remain unchanged.

WDR: [R2 (8:15)] ⟶ (Device)
WD: [A + (X2)] ⟶ (Device)

Resulting Condition Code:

12	13	14	15	
C	V	G	L	
	1			INSTRUCTION TIME OUT

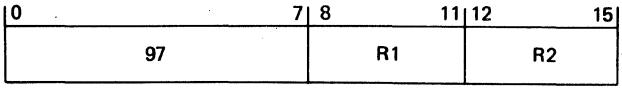
Programming Note:

These instructions are privileged.

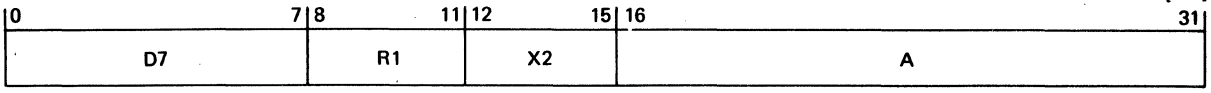
These instructions should not be used with 16-bit oriented device controllers. Note that standard 3010/2 peripheral devices use 8-bit oriented device controllers. For 16-bit oriented devices, use Read Halfword/Write Halfword instructions.

4.10.6 Read Block

RBR R1,R2 [RR]



RB R1,A+(X2) [RX]



The 16-bit General Register specified by the first operand (R1) contains the device address. The 16-bit second operand location, (R2) or [A + (X2)] contains the starting address of the data buffer to be transferred. The next sequential halfword, (R2 + 1) or [A + (X2) + 2] contains the ending address of the data buffer. The starting address must be equal to, or less than, the ending address. Data transfer is inclusive of the buffer limits.

The Read Block instruction causes transfer of eight-bit data bytes from a device to consecutive memory locations. No other instructions are executed during transfer of the data block.

The Condition Code portion of the Program Status Word [PSW (12:15)] will be set to zero after a normal transfer. In the event of an abnormal block data transfer, the Condition Code will not be zero.

Resulting Condition Code:

12 C	13 V	14 G	15 L	
0	0	0	0	BLOCK DATA TRANSFER COMPLETE CORRECTLY.
1				DEVICE BUSY (BSY)
	1			EXAMINE STATUS (EX) OR TIME OUT
		1		END OF MEDIUM (EOM)
			1	DEVICE UNAVAILABLE (DU)

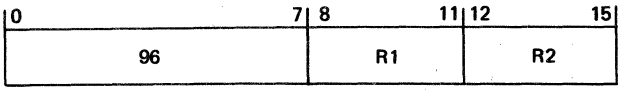
Programming Note:

These instructions are privileged.
These instructions are subject to Memory Protect.

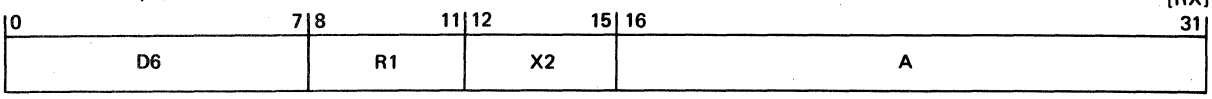
These instructions should not be used with 16-bit oriented device controllers. Note that standard 3010/2 peripheral devices use 8-bit oriented device controllers. For 16-bit oriented devices, use Read Halfword/Write Halfword instructions.

4.10.7 Write Block

WBR R1,R2 [RR]



WB R1,A(X2) [RX]



The 16-bit General Register specified by the first operand (R1) contains the device address. The 16-bit second operand location, (R2) or [A + (X2)] contains the starting address of the data buffer to be transferred. The next sequential halfword, (R2 + 1) or [A + (X2) + 2] contains the ending address of the data buffer. The starting address must be equal to, or less than, the ending address. Data transfer is inclusive of the buffer limits.

The Write Block instruction causes transfer of eight-bit data bytes from consecutive memory locations to a device. No other instructions are executed during transfer of the data block. The Condition Code portion of the Program Status Word [PSW (12:15)] will be set to zero after a normal transfer. In the event of an abnormal block data transfer, the Condition Code will not be zero.

Resulting Condition Code:

12	13	14	15	
C	V	G	L	
0	0	0	0	BLOCK DATA TRANSFER COMPLETED CORRECTLY.
1				DEVICE BUSY (BSY)
	1			EXAMINE STATUS (EX) OR TIME OUT.
		1		END OF MEDIUM (EOM).
			1	DEVICE UNAVAILABLE (DU).

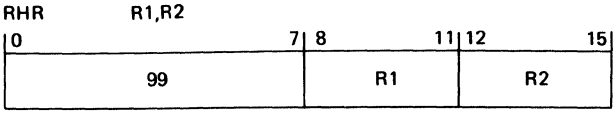
Programming Note:

These instructions are privileged.

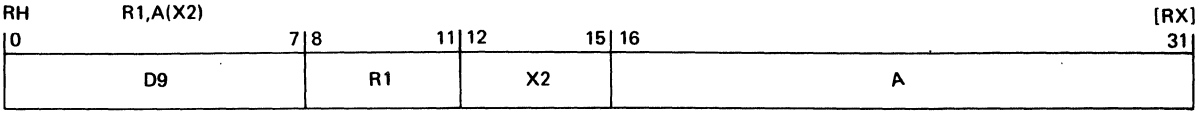
This instruction should not be used with 16-bit oriented device controllers. Note that standard 3010/2 peripheral devices use 8-bit oriented device controllers. For 16-bit oriented devices, use Read Halfword/Write Halfword instructions.

4.10.8 Read Halfword

RHR R1,R2 [RR]



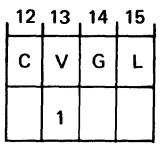
RH R1,A(X2) [RX]



The 16-bit General Register specified by R1 contains the device address. The device is addressed and a 16-bit halfword is received from the device replacing the contents of the second operand. The Read Halfword instruction is implemented such that it can work with both 8-bit byte oriented device controllers and with 16-bit halfword oriented device controllers. If the controller is byte oriented the Processor inputs two 8-bit bytes, if the controller is halfword oriented the Processor inputs one 16-bit halfword.

RHR: [R2 (0:7)] ← First Data Byte } 8-bit oriented device controller
[R2 (8:15)] ← Second Data Byte }
[R2 (0:15)] ← Halfword of Data 16-bit oriented device controller

RH: [A + (X2)] ← First Data Byte } 8-bit oriented device controller
[A + (X2) + 1] ← Second Data Byte }
[A + (X2)] ← Halfword of Data 16-bit oriented device controller



INSTRUCTION TIME OUT

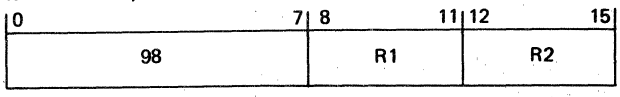
Programming Note:

These instructions are privileged.
The RX form (RH) is subject to Memory Protect.

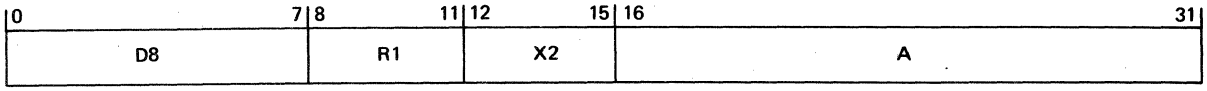
With the RX form (RH), the effective address A+(X2) should be an even value.

4.10.9 Write Halfword

WHR R1,R2 [RR]



WH R1,A(X2) [RX]



The 16-bit General Register specified by R1 contains the device address. The device is addressed and a 16-bit halfword is transmitted to the device from the location specified by the second operand. The Write Halfword instruction is implemented such that it can work with both 8-bit byte oriented device controllers and with 16-bit halfword oriented device controllers. If the controller is byte oriented the Processor outputs two 8-bit bytes, if the controller is halfword oriented the Processor outputs one 16-bit halfword.

WHR:	[R2 (0:7)] → Device	8-bit oriented device controller
	[R2 (8:15)] → Device	
	[R2 (0:15)] → Device	16-bit oriented device controller
WH:	[A + (X2)] → Device	8-bit oriented device controller
	[A + (X2) + 1] → Device	
	[A + (X2)] → Device	16-bit oriented device controller

Resulting Condition Code:

12	13	14	15	
C	V	G	L	
	1			INSTRUCTION TIME OUT

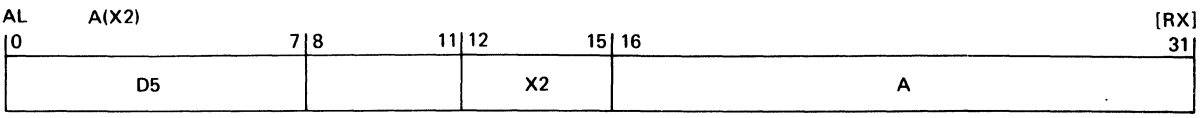
Programming Note:

The Read Halfword and Write Halfword instructions are useful with devices requiring two bytes per transfer. Since the transfer is accomplished with one instruction instead of two, both time and core are saved. Some examples of devices with which these instructions can be used are Halfword I/O Module, 16-line Interrupt Module, conversion equipment (i.e. D/A and A/D Converters), card reader, and Control Panel.

With the RX form (WH), the effective address A+(X2) should be an even value.

These instructions are privileged.

4.10.10 Autoload



The Autoload instruction loads memory with a block of data from a byte oriented input device (e.g., teletypewriter, photo-electric Paper Tape Reader, Magnetic Tape, etc.). The data is read a byte at a time and stored in successive memory locations starting with location X'80'. The last byte is loaded into the memory location specified by the address of the second operand, A + (X2). Any blank or zero bytes that are input prior to the first non zero byte are considered to be leader and are therefore ignored; all other zero bytes are stored as data. The input device is specified by memory location X'78'. The device command code is specified by memory location X'79'.

1. n ← 0
2. (X'80' + n) ← byte
3. n ← n + 1
4. If A + (X2) < X'80' + n, instruction is finished, otherwise return to equation 2.

Resulting Condition Code:

12	13	14	15	
C	V	G	L	
0	0	0	0	DATA TRANSFER COMPLETED CORRECTLY.
1				DEVICE BUSY (BSY)
	1			EXAMINE STATUS (EX) OR TIME OUT.
		1		END OF MEDIUM (EOM)
			1	DEVICE UNAVAILABLE (DU).

Programming Note:

This instruction is privileged.

This instruction is subject to Memory Protect.

The R1 field of an Autoload machine instruction must contain 0.

This instruction should not be used with 16-bit oriented device controllers. Note that standard 3010/2 peripheral devices use 8-bit oriented device controllers. For 16-bit oriented devices, use Read Halfword/Write Halfword instructions.

4.11 SYSTEM CONTROL INSTRUCTIONS

The System Control instructions provide a means for the program to set the Program Status Word, swap PSW's, trigger special interrupt handling, and communicate with a supervisor program. Some of these instructions are privileged and may be executed only with the Processor in the Supervisor Mode (i.e., Bit 7 of the PSW reset). Any attempt to execute these instructions in the Protect Mode results in an Illegal Instruction Interrupt. The instructions described in this section are:

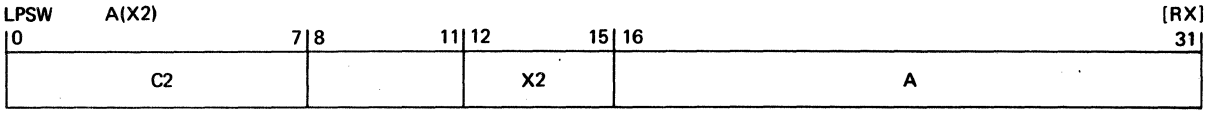
4.11.1 LPSW Load Program Status Word

4.11.2 EPSR Exchange Program Status

4.11.3 SINT Simulate Interrupt

4.11.4 SVC Supervisor Call

4.11.1 Load Program Status Word



A 32-bit operand is loaded into the Current Program Status Word. The second operand is unchanged.

[PSW (0:31)] ←——— [A + (X2)]

Resulting Condition Code:

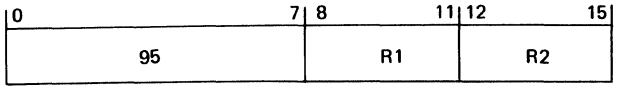
Determined by PSW loaded by the instruction.

Programming Note:

This instruction is privileged.
The R1 field of a Load PSW instruction must contain 0.

4.11.2 Exchange Program Status

EPSR R1,R2 [RR]



The Current Program Status, PSW (0:15), is stored into the register specified by R1. The content of R2 then becomes the Current Program Status, [PSW (0:15)] . Note that if R1 = R2, this results in the Program Status being copied into R1, but otherwise remaining unchanged. This instruction is useful for capturing the running Program Status, enabling or disabling interrupts, or loading the Condition Code with a specified value.

EPSR: [PSW (0:15)] → R1
[PSW (0:15)] ← R2

Resulting Condition Code:

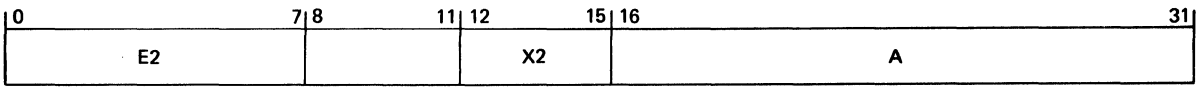
Determined by New PSW.

Programming Note:

This instruction is privileged.

4.11.3 Simulate Interrupt

SINT A(X2) [RS]



The least significant eight-bits of the second operand, A + (X2), is presented to the Interrupt Handler as a device number. The device number indexes into the Service Pointer Table at X'00D0' and results in either an Immediate Interrupt or an I/O Channel operation.

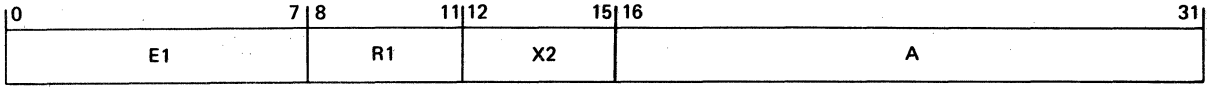
Programming Note:

This instruction is privileged.

The R1 field of a Simulate Interrupt instruction must contain 0.

4.11.4 Supervisor Call

SVC R1,A(X2) [RX]



The Supervisor Call instruction is used to initiate certain functions in the Supervisor program. The second operand address, A + (X2), may be a pointer to the core location of the parameters the Supervisor program will need to complete the function specified.

The value, A + (X2), is stored in core location X'0094'. The Current Program Status Word is stored in the fullword core location at X'0096'. Core location X'009A' contains the New Program Status value. Core locations X'009C' through X'00BB' contain sixteen new Location Counter values, one for each type of Supervisor call.

The type of Supervisor call is specified in the R1 field of the instruction. Sixteen different calls are provided for. Return from the Supervisor is made by executing a Load Program Status Word instruction specifying the stored "Old" PSW in location X'0096'.

(X'0094') ← A + (X2)
(X'0096') ← [PSW (0:31)]
(X'009A') → [PSW (0:15)]
(X'009C' + 2 *R1) → [PSW (16:31)]

Resulting Condition Code:

Defined by New PSW.

Programming Note:

This instruction provides a convenient means of switching from the Protect Mode to the Supervisor Mode. Return to the Protect Mode is accomplished by a Load Program Status Word or Exchange Program Status instruction.

CHAPTER 5
INPUT/OUTPUT SYSTEM

5.1 INTRODUCTION

This chapter discusses the GE-PAC 3010/2 Input/Output (I/O) System. There are several methods of communication between the Processor and peripheral devices and/or other system elements. The methods vary in speed, sophistication, and the amount of attention required by the Processor. Thus, the systems interface may be tailored to the individual user's needs and it may be gracefully expanded as the user's requirements grow.

There are two primary purposes for this chapter; (1) to familiarize the user with the GE-PAC 3010/2 systems interface, and (2) to provide the data required to permit the user to effectively interface peripheral equipment to the GE-PAC 3010/2 System. A functional description of each I/O subsystem is given later in this chapter, followed by a detailed description of each I/O instruction, and rules and specifications for designing interfaces to the System.

5.2 SYSTEMS INTERFACE

Figure 5-1 is a block diagram of a 3010/2 System emphasizing the systems interface capability. Note that there are four separate methods of interfacing to peripheral devices or system elements:

1. Multiplexor Channel
2. Interleaved Data Channel
3. Selector Channel
4. Direct Memory Access Channel (See Chapter 6)

The following paragraphs describe each of the interface methods.

5.2.1 Multiplexor Channel

The Multiplexor Channel is a byte or halfword oriented I/O system which communicates with up to 255 peripheral devices. The Multiplexor Bus consists of 30 lines; 16 bi-directional Data Lines, 8 Control Lines, 5 Test Lines, and an Initialize Line.

The lines in the Multiplexor Bus are:

Data Lines	D00:15	(Processor ⟷ Device)	16 Lines
Control Lines	SR	(⟶)	1 Line
	DR	(⟶)	1 Line
	CMD	(⟶)	1 Line
	DA	(⟶)	1 Line
	ADRS	(⟶)	1 Line
	ACK	(⟶)	1 Line
	DACK	(⟶)	1 Line
	CL07	(⟶)	1 Line
Test Lines	ATN	(⟵)	1 Line
	SYN	(⟵)	1 Line
	HW	(⟵)	1 Line
	DC	(⟵)	1 Line
	DCR	(⟵)	1 Line
Initialize Line	SCLRO	(⟶)	1 Line

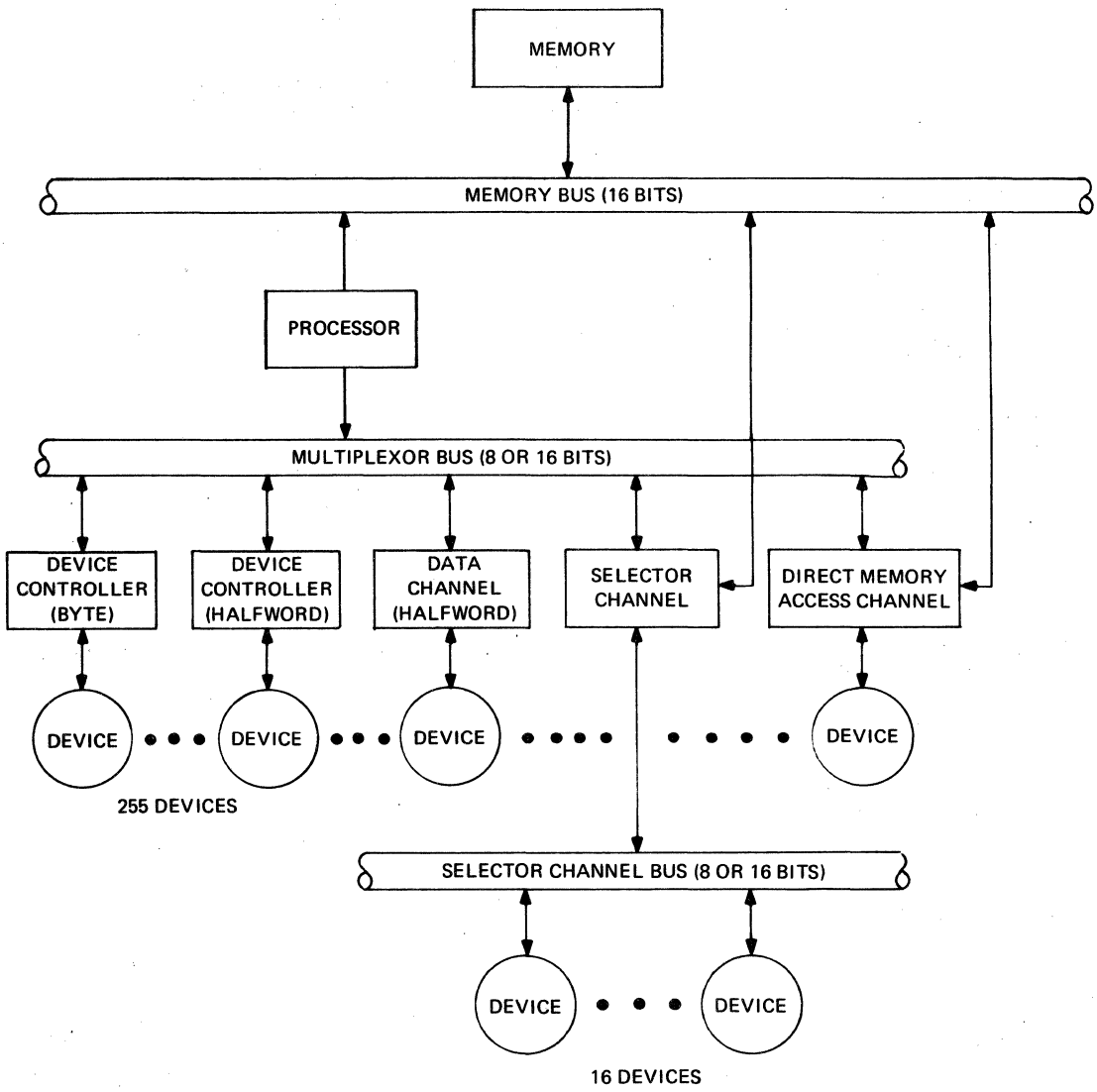


Figure 5-1. System Interface, Block Diagram

Figure 5-2 is a block diagram of the Multiplexor Channel. The following general definitions apply to the lines in the Multiplexor Bus.

Data Lines D00:15

The data lines are used to transfer one 8-bit byte or one 16-bit halfword of data between the Processor and the device. One byte of address or command is transferred from the Processor to the device over Data Lines 8:15 (D08:15) when accompanied by either an Address (ADRS) or a Command (CMD) control line. One byte of data or one halfword of data is transferred from the Processor to the device when accompanied by the Data Available (DA) control line. The device, in response to an Acknowledge (ACK) control line or a Sense Status (SR) control line, sends one byte of address or status information to the Processor over D08:15. In response to a Data Request (DR) control line, the device sends either an 8-bit byte or a 16-bit halfword of data to the Processor. The device always sends a Synchronize (SYN) signal to the Processor to indicate that it has either received the data from the Processor or that it has sent the data to the Processor. The SYN signal is removed immediately after the Processor removes the control line.

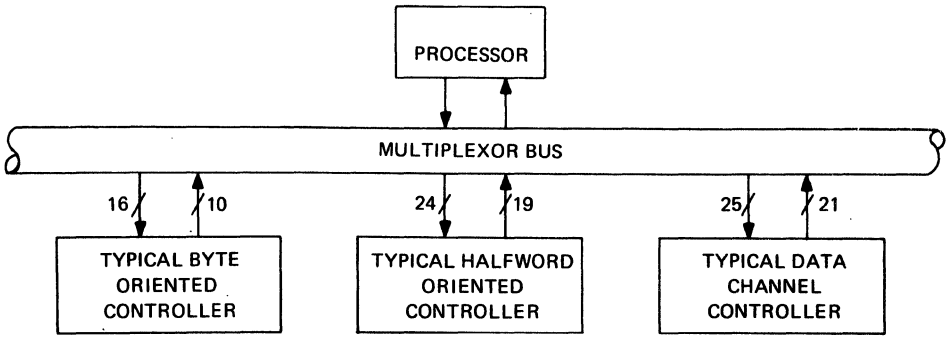


Figure 5-2. Multiplexor Channel, Block Diagram

Control Lines

SR — Status Request. The device controller must present device status to Data Lines D08:15, followed by a SYN.

DR — Data Request. The device controller presents data to Data Lines 8:15 or 0:15 (D08:15 or D00:15), followed by a SYN. If a Halfword (HW) of data is presented, the HW test line is also active.

ACK — Acknowledge. The interrupting device controller presents its address on D08:15, followed by a SYN.

DA — Data Available. The Processor presents data on D00:15 for transfer to the device. The device controller accepts the low byte or the entire halfword and responds with a SYN.

CMD — Command. The Processor presents a Command Byte on D08:15. The device controller accepts the Command Byte and responds with a SYN.

ADRS — Address. The Processor presents an Address Byte on D08:15. ' The device controller accepts the Address Byte and responds with a SYN.

DACK — Data Channel Acknowledge. The Processor presents an address of zero on D08:15. The ADRS control line and the DACK control line are simultaneously active. The interrupting Data Channel device controller becomes selected and responds with a SYN. As a result of addressing device zero (a null address), only the selected data channel device controller remains addressed.

CL070 — This control line is activated by the Processor when a Power Fail condition is detected by the Processor, if the Power Fail option is equipped. This line is held active until the SCLR0 signal occurs.

Test Lines

ATN — Attention. Any device desiring to interrupt the Processor will activate the ATN line and hold this line until an ACK is received from the Processor.

HW — Halfword. The HW line is activated by a halfword oriented device controller whenever it is communicating normally with the Processor. The HW line is not activated when a device controller is operating in the Interleaved Data Channel mode.

DC — Data Channel Request. Any Data Channel device desiring to interrupt the Processor will activate the DC line and hold this line until a DACK is received from the Processor.

DCR — Data Channel Read. The selected Data Channel device controls the state of the DCR line, high for read, low for write, from the device.

SYN — Synchronize. This signal is generated by the device to inform the Processor that it has properly responded to a control line.

Initialize Line

SCLR — System Clear. This is a metallic contact to ground which closes on a power-down, power-up, or hardware initialize sequence.

> NOTE
>
> All control lines, except ACK and DACK, are connected in parallel to all devices. These lines are activated by the Processor in response to an external interrupt. The ACK line is connected in series with all devices. If no interrupt is pending in the first controller when the ACK or DACK signal arrives, the signal is passed on daisy chain fashion to the next controller, and so on until it is captured by the interrupting controller. See definition of ACK and DACK.

Communication over the Multiplexor Bus is performed on a request/response basis where each sequence of events is controlled by the micro-program contained in the Processor's Read-Only-Memory. A typical sequence to perform an I/O instruction with a device controller is:

1. The Processor addresses the device controller by placing an eight-bit address on the data lines and activating the ADRS control line. The device controller whose address corresponds to the Address Byte on the data lines responds by setting its Address flip-flop and returning SYN to the Processor. (All other device controllers reset their Address flip-flops.) Once a device controller is addressed it remains so until another device is addressed or until the system is initialized. The addressed device controller responds to all subsequent activity on the Multiplexor Bus.
2. If the I/O instruction involves transferring data from the Processor to the device controller, the Processor places the data on the data lines and activates the appropriate control line. The addressed device controller responds with a SYN after it has received the data, the Processor then removes the control line.
3. If the I/O instruction involves transferring data to the Processor from the device controller, the Processor activates the appropriate control line, and waits for the device controller to respond by placing the data on the data lines and activating SYN. When the Processor receives SYN, it accepts the data and removes the control line.
4. In all cases the device controller removes the SYN whenever the Processor removes the control line.

The sequence described here is somewhat simplified for the sake of clarity. The exact sequence for each I/O instruction is listed later in this chapter.

Whenever a device controller desires, it may interrupt the Processor by activating the ATN test line. This may be done by any device controller at any time, regardless of whether it is addressed or not. If interrupts are enabled by the Current Program Status Word, the Processor responds to ATN by interrupting the currently running program and directing the Processor to a new program (or a new micro-program) which identifies and services the interrupt as required.

5.2.2 Interleaved Data Channel

The Interleaved Data Channel provides high speed low cost autonomous memory access on an instruction steal basis. Data transfer between the device controller and memory is accomplished over the Multiplexor

Bus at data rates of 440,000 bytes per second in the Burst Mode. Internal Processor registers provide the necessary buffering between the memory and the device controller. A typical sequence to perform an Interleaved Data Channel cycle is:

1. When the device controller is ready it requests a Data Channel cycle by activating the DC test line. This line is separate from, and of higher priority than, the ATN line. The Processor responds to this line by addressing device zero, and at the same time activating the DACK control line. The DACK line is "daisy chained" through all Data Channel devices until it is captured by the highest priority controller requiring Data Channel service. That device controller becomes the "on line" device.
2. The on line device controller controls the DCR test line which specifies either a Read or a Write operation to memory. If the Data Channel operation is Read from memory, the Processor inputs a 16-bit memory address from the device controller, reads the contents of that address from memory, and outputs the 16-bits read out to the device controller.
3. If the Data Channel operation is Write to memory, the Processor inputs a 16-bit memory address and a 16-bit data halfword, and then it writes that data into the memory at the specified address.

The sequence described here is somewhat simplified for the sake of clarity. The exact sequence for each kind of Data Channel operation is discussed later in this chapter.

5.2.3 Selector Channel

The optional Selector Channel provides block data transfer between one of up to 9 I/O devices, and memory. Once initiated, the transfer is independent of the Processor. The program specifies the device address, the type of operation (Read or Write), the starting address in memory, and the final memory address of the transfer. The Selector Channel then completes the transfer, cycle stealing from the Processor, without further direction by the Processor. Upon completion of the transfer, or termination of the transfer due to a fault, the Selector Channel Busy condition is dropped and the Processor is notified via an interrupt.

Figure 5-3 is a block diagram of the Selector Channel. Address lines to, and data lines to and from the Memory Bus are shown on the right side. The Memory Bus Control Logic (one of several arbitrary functional groupings used only for purposes of this block diagram) gates an address to the Memory Bus and data to or from the bus depending upon the direction of transfer. The Selector Channel Data Register (DR) stores the 16-bit data halfword to/from memory. The Transfer Control Logic gates the data between the Selector Bus (shown on the bottom) and the Data Register in either 8-bit bytes or 16-bit halfwords, depending on the device. The address circuits are shown in the upper right area. The Final Address Register (FR) is loaded in two bytes from the Multiplexor Bus. The Address Register (AR) is loaded with the starting address in two bytes. After each byte of data transfers to/from the device, the Address Register is incremented and its contents is compared to the contents of the Final Address Register. When AR=FR, a terminate signal is sent to the Transfer Control Logic. If AR≠FR, the next data transfer is initiated to/from the device.

The Multiplexor Bus is shown on the left side of Figure 5-3. Note that the Multiplexor Bus may be gated to any one of six places. The gates are functionally represented by a six position rotary switch. With the gating as shown, and assuming the Transfer Control Logic is also as shown, the Multiplexor Bus including all 16 Data Lines is gated directly to the Selector Bus. This is the condition which exists when the Selector Channel has not been addressed. Thus, all devices on the Selector Channel may be used via the Multiplexor Channel if the Selector Channel is not in use. Four of the remaining five points onto which the Multiplexor Bus may be gated, are the Upper and Lower halfs of FR and AR. The sixth point is designated Command and Sense Logic. Commands from the Processor are decoded in this block to produce control signals for the Transfer Control Logic and the Multiplexor Input Control Logic.

The following is a typical sequence of operation for a Selector Channel I/O operation. Figure 5-4 is a flow chart of Selector Channel operation. Circled numbers on Figure 5-4 refer to steps in the following sequence:

1. The device controller is addressed and the appropriate command sent to it (for example, Read Tape Forward).

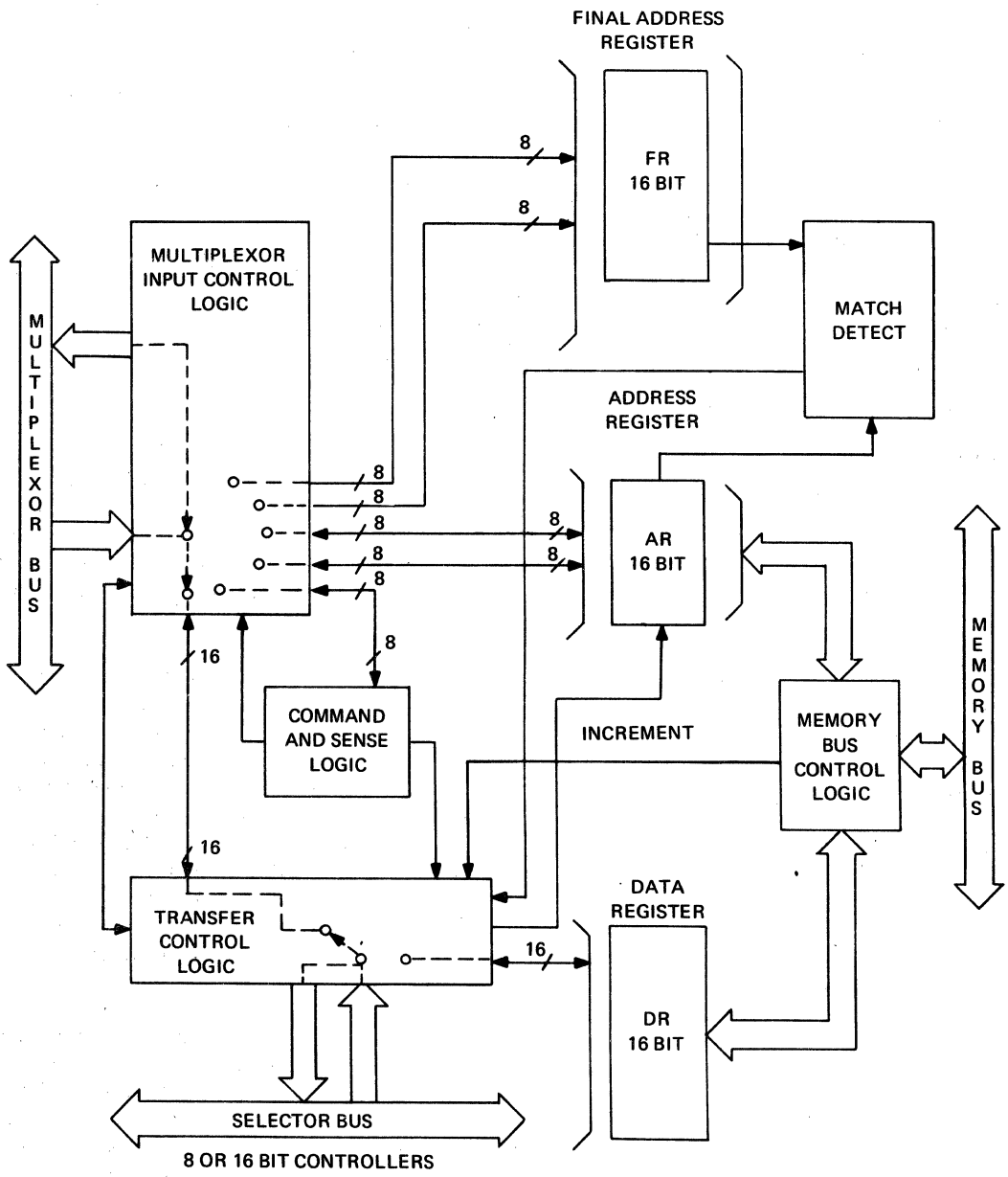


Figure 5-3. Selector Channel, Block Diagram

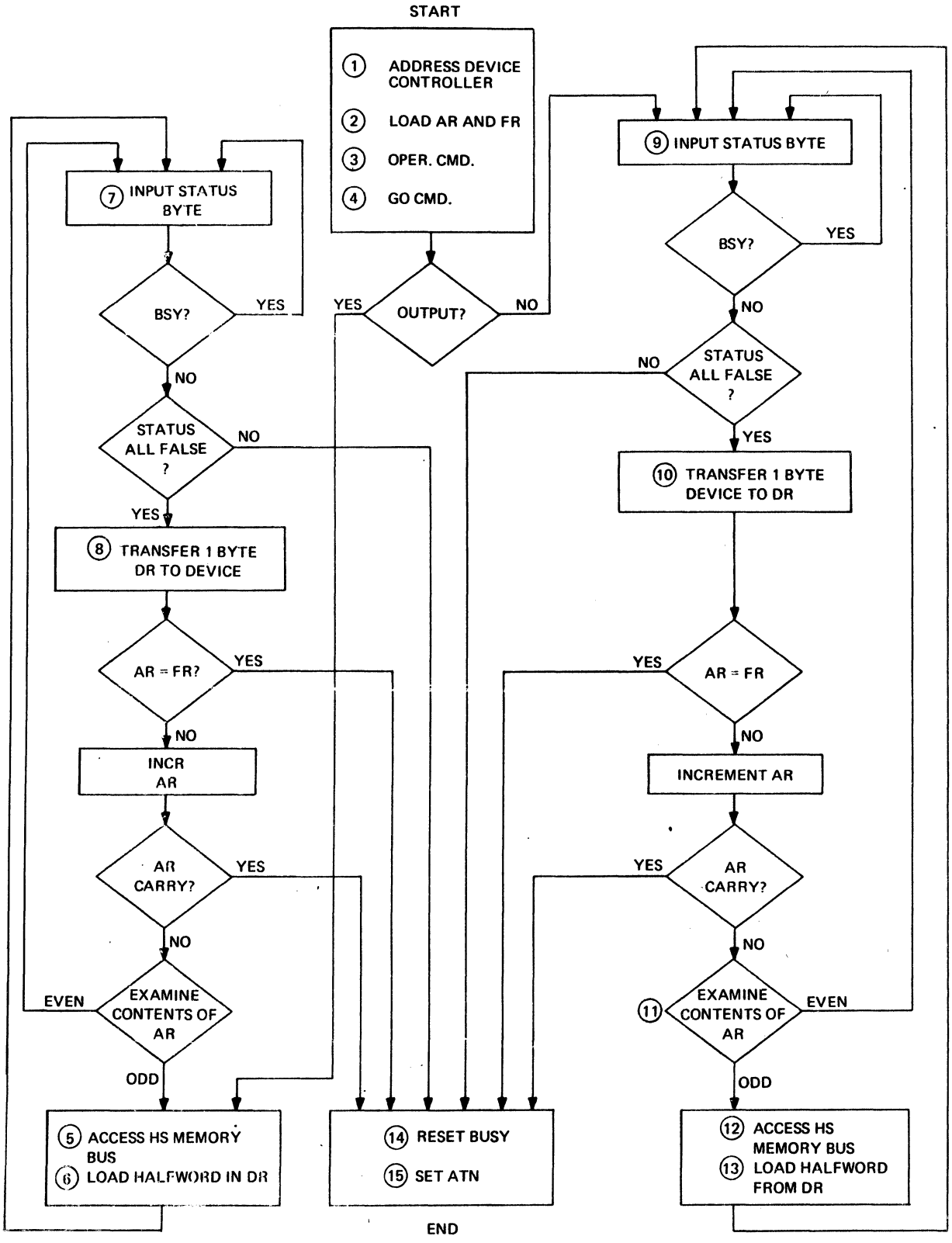


Figure 5-4. Selector Channel, Flow Chart

2. The Selector Channel AR and FR are loaded via four byte transfers from the Multiplexor Channel. The FR may be odd or even.

> NOTE
>
> Steps 1 and 2 may be reversed.

3. A command which specifies if this is an input operation (information received from the device) is sent to the Selector Channel from the Multiplexer Channel. (The Selector Channel is initialized and returns to the output state - information to the device - whenever a Read Data or Write Data instruction is issued to the Selector Channel.)

4. A GO Command which starts the transfer operation is sent from the Multiplexor Channel to the Selector Channel.

> NOTE
>
> The Processor is now free to continue its program while the block I/O transfer is performed by the Selector Channel on a cycle-stealing basis. Steps 5 through 8 apply solely to Read operations (memory to device). Steps 9 through 12 apply to Write operations (device to memory).

5. If this is a Read operation, the Selector Channel requests memory service via the built-in memory port in the Processor. When service is granted, the Selector Channel fetches a halfword from memory.

6. When memory data becomes available, it is gated to the DR.

7. The Status Byte from the device is examined. If the device Busy Bit (Bit 4) is true, the Selector Channel waits for it to become false. If any of the Status Bits 5, 6, or 7 are true, the transfer is terminated.

8. Data (either an 8-bit byte or a 16-bit halfword depending on the controller) is transferred from the DR to the device when the device is not busy. If AR=FR, the transfer is terminated. If not, the AR is incremented, and if there is a carry (AR=0) the transfer is terminated. If the contents of the AR is even another byte is transferred to the device.

9. If this is a Write operation, the Status Byte is input from the device. If the Busy is true, the Selector Channel waits for it to become false. If any of the other three bits in the status code are true, the transfer sequence is terminated. If all bits are false, the sequence continues.

10. Data is transferred from the device to the DR when the device is not busy. If AR=FR, the transfer is terminated. If not, the AR is incremented, and if there is a carry (AR=0) the transfer is terminated.

11. If the contents of the AR is even, another byte is transferred to the device. (The sequence returns to Step 9.)

12. The Selector Channel requests memory service.

13. The halfword in DR is written into the addressed memory location when granted memory.

Steps 14 through 16 describe the termination sequence. Any of the following conditions will cause termination:

a. AR=FR

b. AR increments to Zero (carry out of AR).

c. A status failure from the device (EX, EOM, or DU).

d. A Stop Command from the Processor.

NOTE

If the Selector Channel is in a memory cycle when the Stop Command is received from the Processor, execution of the Stop Command will be delayed until the completion of the memory cycle.

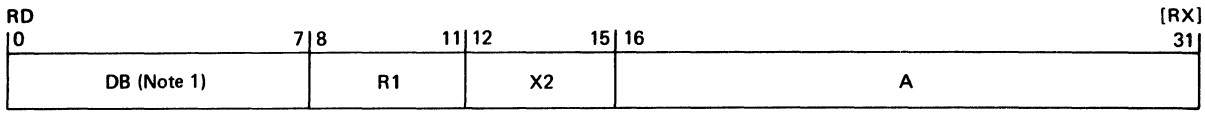
14. Reset the Selector Channel Busy indication.
15. Set the Selector Channel Attention flip-flop to generate an interrupt to the Processor.
16. After the Processor acknowledges the interrupt and addresses the Selector Channel, it will send a Status Request to the Selector Channel. The status received will be the status of the active device on the SELCH Bus with the Busy Bit forced to zero. To determine whether the transfer terminated normally, the contents of AR may be Read by issuing two Data Requests to compare its contents with the final memory address.

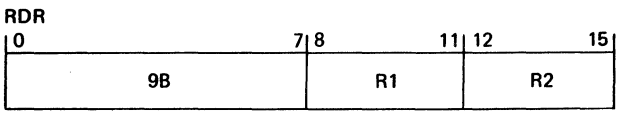
The Selector Channel is complete on one mother-board which is mounted in any even numbered Universal Expansion Slot.

5.3 INPUT/OUTPUT INSTRUCTIONS

The following paragraphs describe each of the Input/Output (I/O) instructions. Each instruction is implemented by a sequence of operations controlled by the Processor's micro-program. A discussion of the micro-programmed sequence and the related device controller logic for each I/O instruction is provided.

5.3.1 Read Data (RD) Instruction





Note 1: The operation codes in this figure and all similar figures which follow in this chapter are given in hexadecimal notation.

Execution of the RD instruction transfers an eight-bit byte of data from the device specified by the contents of General Register R1 to the memory byte address specified by A, indexed by the contents of General Register X2. If X2 equals zero, the byte is transferred directly to the memory address specified by A. The transfer takes place via the Multiplexor Channel. Refer to Figures 5-5 and 5-6 during the following description of the RD instruction sequence of operation. The gate and flip-flop designations listed in this chapter reference the figures only. They have no hardware significance.

1. The first thing the micro-program does after decoding an RD instruction, is to place the contents of R1 (the device number) on the Data Lines (D080 through D150).

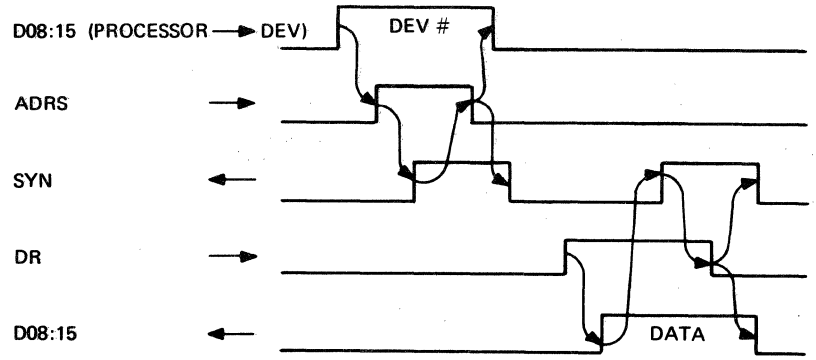


Figure 5-5. Read Data Instruction Timing

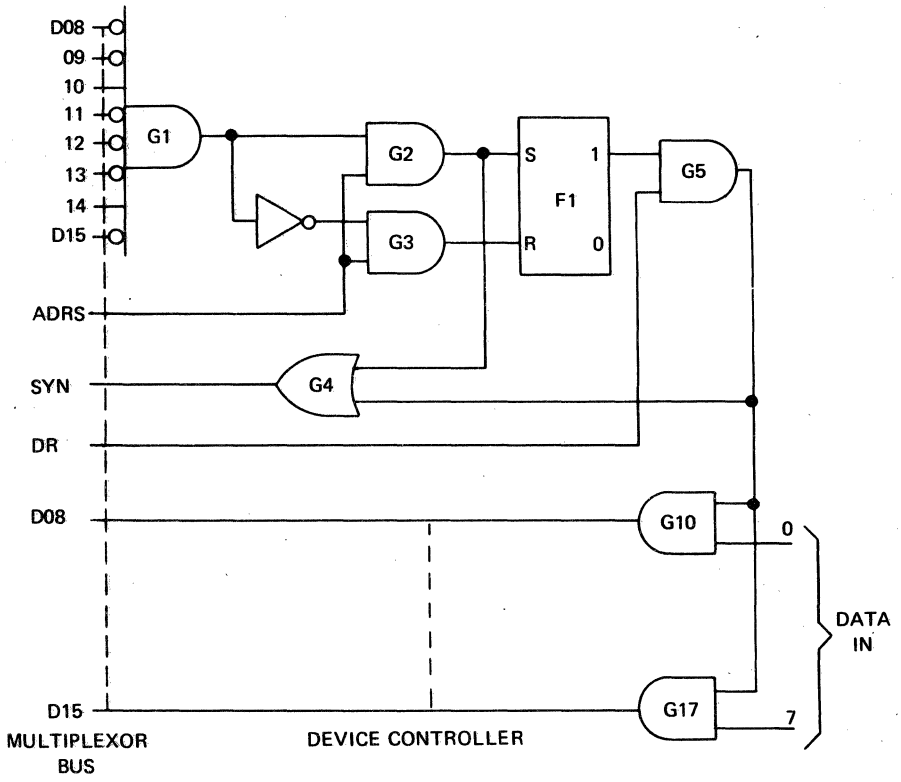


Figure 5-6. Device Controller Logic for Read Data Instruction

NOTE

A "0" or "1" is appended to GE-PAC 3010/2 signal designations to indicate the state of the signal. Thus, the designation D080 through D150 indicates that the eight Data Lines (D08 through D15) are low when active; in other words, this is a false bus.

2. The Address (ADRS) control line is raised.
3. The device controller which decodes its address via Gate G1 (arbitrary hexadecimal address 22 on Figure 5-6) sets the Address flip-flop (F1) through Gate G2.

4. The output from Gate G2, via OR Gate G4, also raises the Synchronization (SYN) response from the device controller to the Processor. The SYN signal at this time indicates that the device controller has decoded its address and has received an ADRS control line.

5. When the SYN signal is received, the Processor removes the ADRS control line and the device number. The device controller, in turn, lowers the SYN signal.

6. The Processor next raises the Data Request (DR) control line. This line is ANDed with the Address flip-flop (F1) in Gate G5. The output from Gate G5 enables the byte of data from the device to the Processor (Gate G10 through G17). Data to the Processor is sent on the Data Lines (D080 through D150).

7. The output from Gate G5 also raises the SYN line to the Processor to indicate that the data is ready.

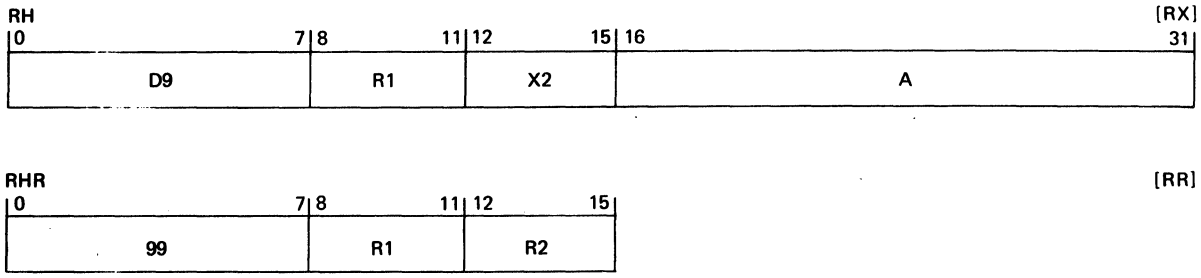
8. The Processor gates the data to the designated byte address (location A indexed by the contents of X2, if specified).

9. When the byte has been received, the Processor lowers the DR control line. The device controller then lowers SYN and removes the data from the Data Lines (D080 through D150). The sequence for one byte is now completed. The Address flip-flop (F1) remains set until another device is addressed, or until a System Clear (SCLRO) signal is generated. When another device controller is addressed, F1 is reset through Gate G3. Resetting F1 effectively disconnects the device controller from the Multiplexor Bus.

A Time Out feature is provided in the Processor to prevent locking up the computer on a malfunctioning device or a non-existent device. The Time Out signal is generated if the device controller fails to return the SYN response within 25 to 50 microseconds of a request. The Time Out feature also sets the V flag in the Program Status Word (PSW) Condition Code. The programmer may therefore branch on the Time Out condition, typically to an error message print-out routine.

The Read Data to Register (RDR) instruction is executed in the same manner as the RD instruction. The only difference is that the data is stored in General Register R2 instead of A.

5.3.2 Read Halfword (RH) Instruction



Execution of the RH instruction transfers a 16-bit halfword of data from the device specified by the contents of General Register R1 to the memory halfword specified by A, indexed by the contents of General Register X2. If X2 is zero, the halfword is transferred directly to the memory address specified by A. The RH instruction is implemented such that it can work with both 8-bit byte oriented device controllers and with 16-bit halfword oriented device controllers. If the controller is byte oriented the Processor inputs two eight-bit bytes, if the controller is halfword oriented the Processor inputs one 16-bit halfword. Refer to Figures 5-7 and 5-8 during the description of the sequence of operations for byte oriented controllers.

1. The device controller is addressed and connected exactly as described in Steps 1 through 5 of the RD sequence, Section 5.3.1.

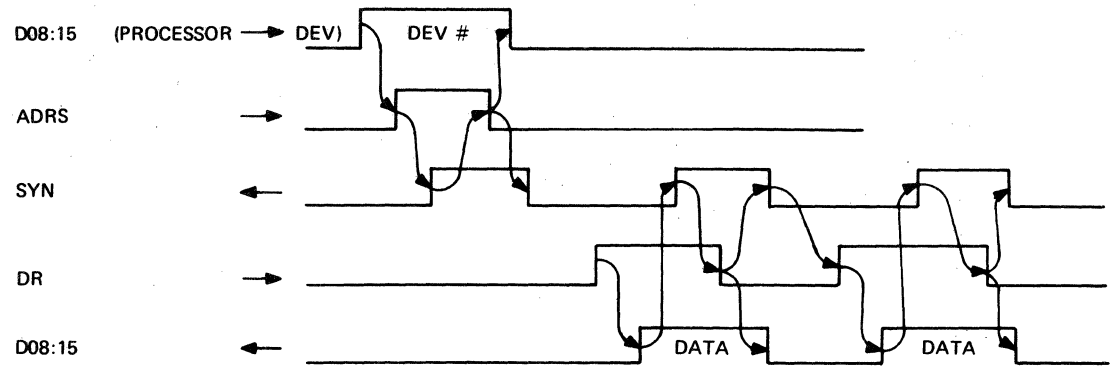


Figure 5-7. Read Halfword Instruction Timing for Byte Oriented Controllers

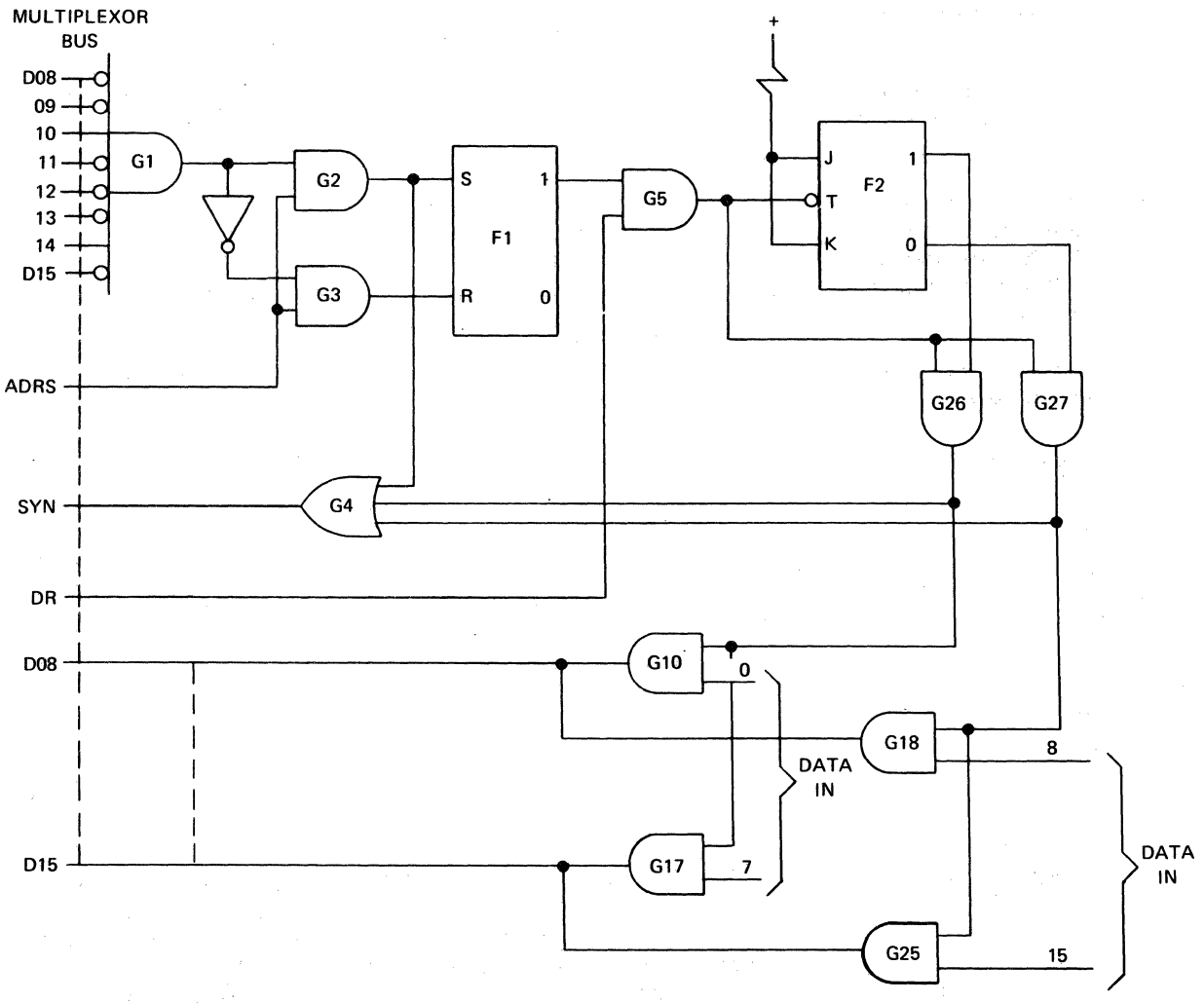


Figure 5-8. Byte Oriented Device Controller Logic for Read Halfword Instruction

2. **The Processor next raises the Data Request (DR) control line. This line is ANDed with the Address flip-flop (F1) in Gate G5. The output of Gate G5 is input to Gates G26 and G27 and to the Byte Counter flip-flop (F2). When flip-flop F2 is set, Gate G26 enables the data bits 0 through 7 onto the Data Lines (D080 through D150) through Gates G10 to G17.**

3. The output of Gate 26 also raises the SYN line to the Processor to indicate that the first byte of data is ready.

4. The Processor accepts the first byte of data when it receives SYN.

5. When the byte has been received, the Processor lowers the DR control line which causes the device controller to remove the data from the Data Lines (D080 through D150), to remove SYN, and to toggle flip-flop F2 reset.

6. When the device controller removes SYN, the Processor raises the DR control line a second time. With flip-flop F2 reset, Gate 27 enables the data Bits 8 through 15 onto the Data Lines (D080 through D150).

7. The output from Gate 27 also raises the SYN line to the Processor to indicate that the second byte of data is ready.

8. The Processor accepts the second byte of data when it receives SYN. The Processor now gates the full 16-bit halfword (the first and second bytes) to the designated memory address (location A indexed by the contents of X2, if specified).

Refer to Figures 5-9 and 5-10 during the following description of the sequence of operations for halfword oriented device controllers.

1. The device controller is addressed and connected exactly as described in Steps 1 through 5 of the RD sequence, Section 5.3.1.

2. If the device is a halfword oriented device it raises the Halfword (HW) test line upon being addressed through Gate G6. This stays active until some other device becomes addressed.

3. The Processor next raises the Data Request (DR) control line. This line is ANDed with the Address flip-flop (F1) in Gate G5. The output from Gate G5 enables the halfword of data from the device to the Processor (Gates G10 through G25). Data to the Processor is sent on the Data Lines (D000 through D150).

4. The output from Gate G5 also raises the SYN line to the Processor to indicate that the data is ready.

5. The Processor gates the data to the designated address (location A indexed by the contents of X2, if specified).

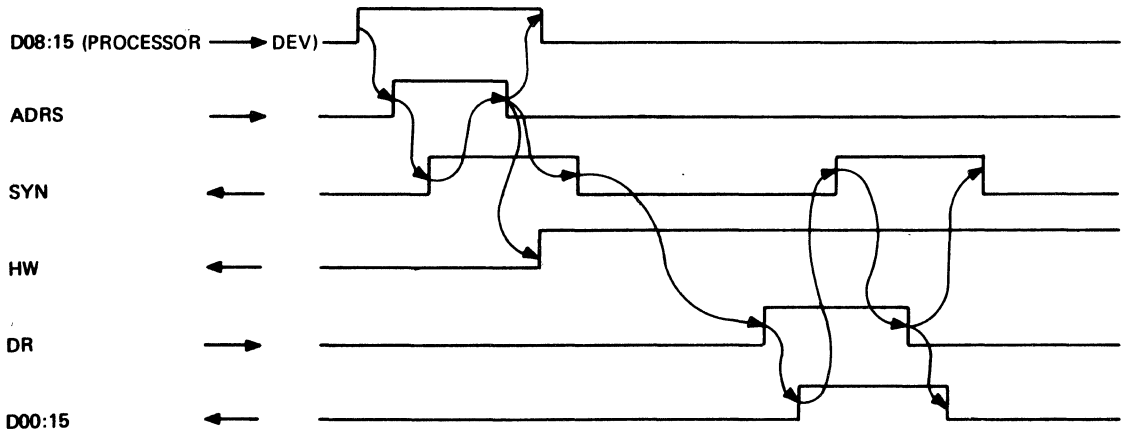


Figure 5-9. Read Halfword Instruction Timing for Halfword Oriented Controller

The device controller Address flip-flop and the Time Out feature are as described in Section 5.3.1 for the RD instruction. The Read Halfword to Register (RHR) instruction is executed in the same manner as the RH instruction. The only difference is that the halfword of data is stored in General Register R2 instead of A.

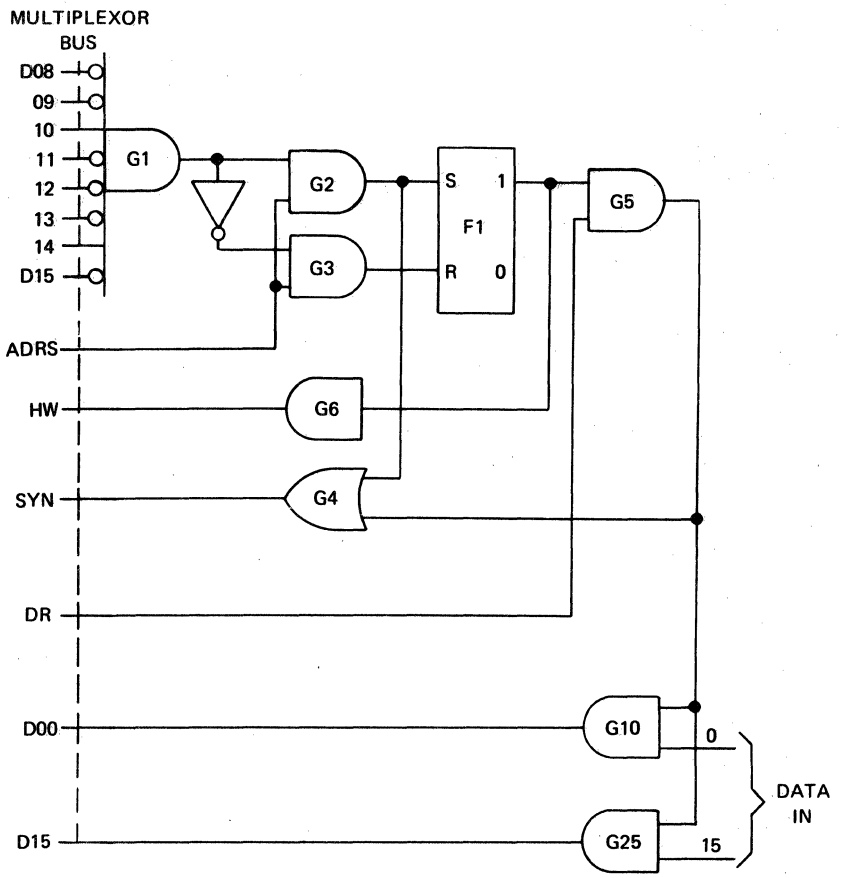
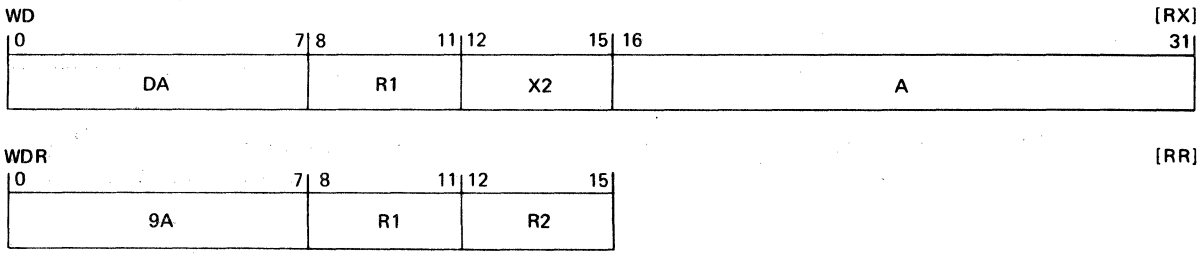


Figure 5-10. Halfword Oriented Controller Logic for Read Halfword Instruction

5.3.3 Write Data (WD) Instruction



Execution of the WD instruction transfers a byte of data from memory address A, indexed by the contents of General Register X2, to the device number specified by the contents of General Register R1. Refer to Figures 5-11 and 5-12 during the following sequence of operations.

1. The device controller is addressed and connected exactly as described in Steps 1 through 5 of the RD sequence, Section 5.3.1.
2. The contents of the byte address (A indexed by X2) is placed on the Data Lines (D080 through D150).
3. The Data Available (DA) control line is then raised.
4. The DA signal is ANDed with the output from Address flip-flop (F1) by Gate G6. The output from Gate G6 strobes the data into the device controller register (F2 through F9).
5. The output from Gate G6 also generates a SYN response the Processor to indicate that the data has been accepted.

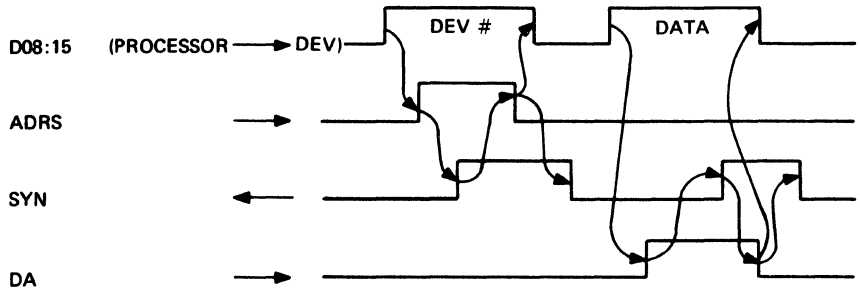


Figure 5-11. Write Data Instruction Timing

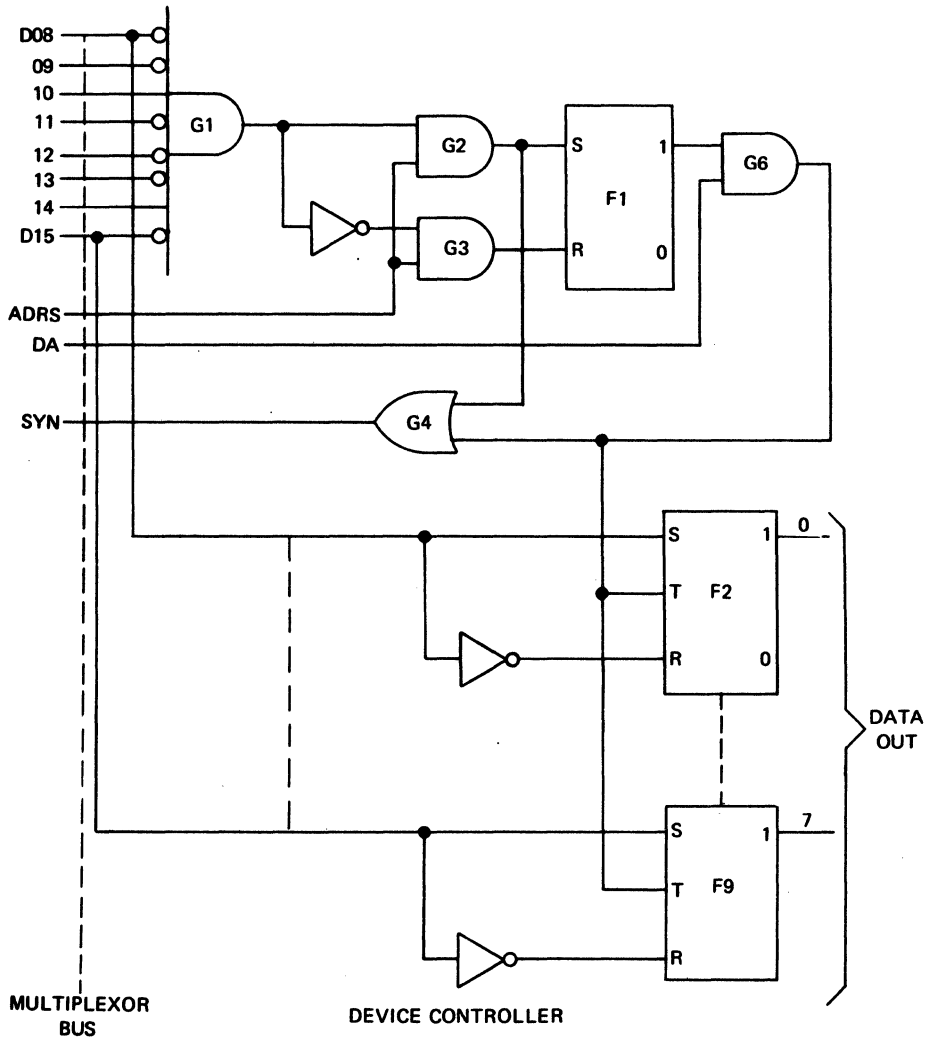


Figure 5-12. Device Controller Logic for Write Data Instruction

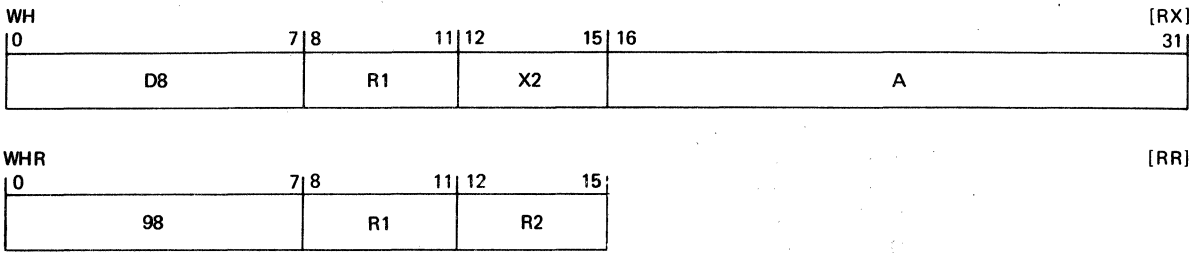
6. When it receives the SYN signal, the Processor lowers the DA line and removes the data from the Data Lines (D080 through D150).

7. The device controller then lowers the SYN line.

As described in Section 5.3.1, the device controller remains selected until another device controller is selected, or a System Clear is generated. The Time Out feature is also exactly as described in Section 5.3.1 for the RD instruction.

The WDR instruction is similar to the WD instruction, except that the byte which is transferred originates in General Register R2 rather than a memory address.

5.3.4 Write Halfword (WH) Instruction



Execution of the WH instruction transfers a 16-bit halfword of data to the device specified by the contents of General Register R1 from the memory halfword specified by A, indexed by the contents of General Register X2. If X2 is zero, the halfword is transferred directly from the address specified by A. The WH instruction is implemented such that it can work with both 8-bit byte oriented device controllers and with 16-bit halfword oriented device controllers. If the controller is byte oriented the Processor outputs two eight-bit bytes, if the controller is halfword oriented the Processor outputs a 16-bit halfword in parallel. Refer to Figures 5-13 and 5-14 during the description of the sequence for byte oriented controllers.

1. The device controller is addressed and connected exactly as described in Steps 1 through 5 of the RD sequence, Section 5.3.1.
2. The Processor places the first byte of the memory address specified (A indexed by X2) on the Data Lines (D080 through D150).
3. The Data Available (DA) control line is then raised.
4. The output from Gate G6 is ANDed with the output from flip-flop F18 by Gates G8 and G7. If flip-flop F18 is set, Gate G8 strobes the first byte of the halfword of data into the device controller register (F2 through F9).
5. The output from Gate G8 also generates SYN to the Processor to indicate that the first byte has been received.
6. When the Processor receives SYN, it lowers the DA control line which causes flip-flop F18 to toggle set, and removes the first byte of data from the Data Lines (D080 through D150).

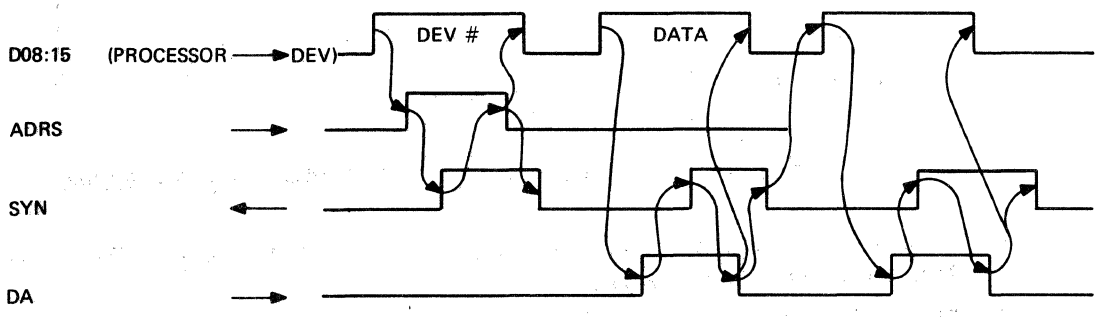


Figure 5-13. Write Halfword Instruction Timing for Byte Oriented Controllers

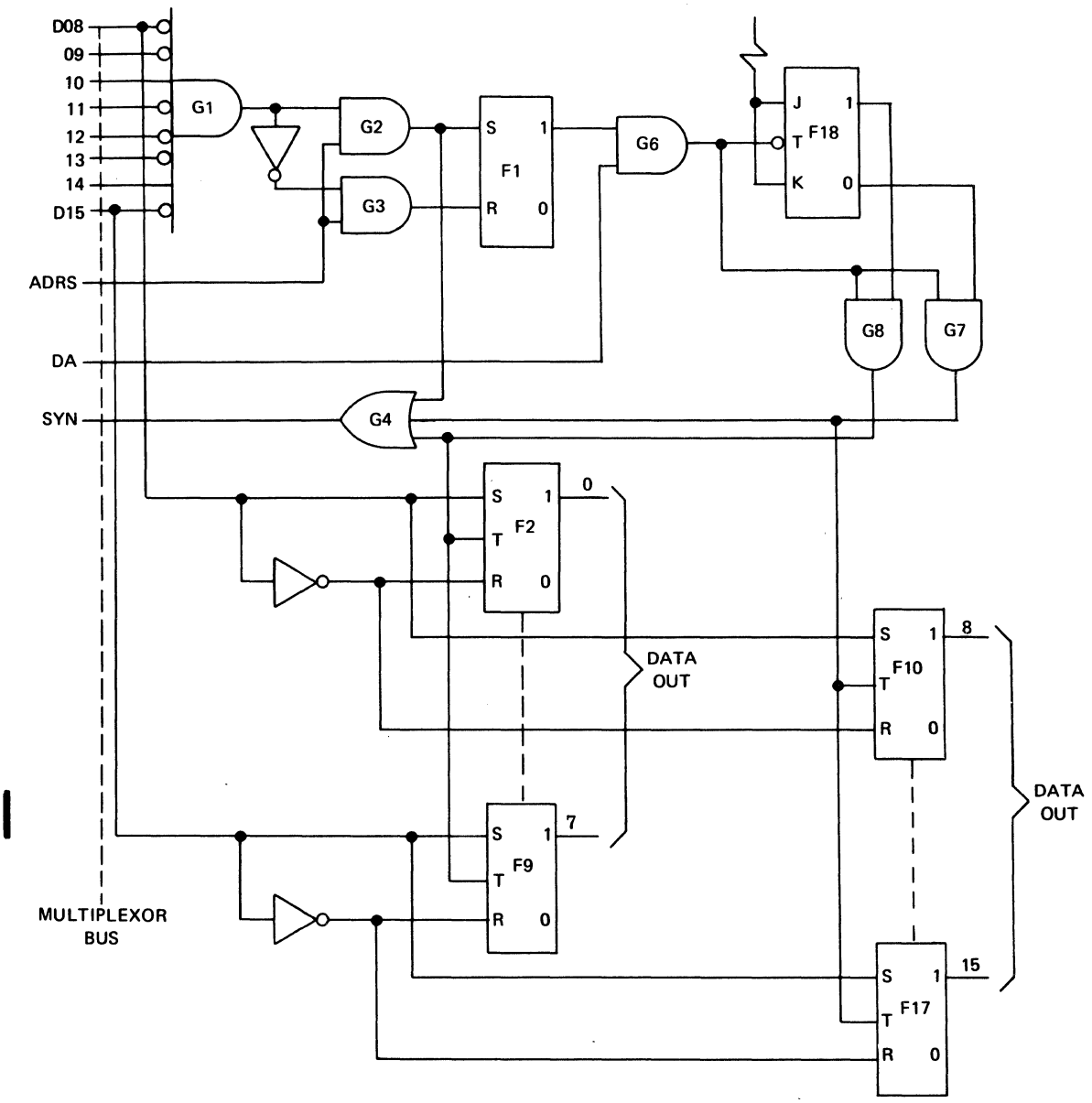


Figure 5-14. Byte Oriented Device Controller Logic for Write Halfword Instruction

7. The device controller than removes SYN.

8. Steps 2 through 7 are then repeated, except that the Processor now sends the second byte of data and Gate G7 strobes the second byte of data into the device controller register (F10 through F17).

Refer to Figures 5-15 and 5-16 during the following description of the sequence of operation for halfword oriented device controllers.

1. The device controller is addressed and connected as described in Steps 1 through 5 of the RD sequence, Section 5.3.1.

2. If the device is a halfword oriented device, it raises the Halfword (HW) test line upon being addressed through Gate G6. HW stays active until some other device becomes addressed.

3. The Processor places the halfword of data on the Data Lines (D000 through D150.)

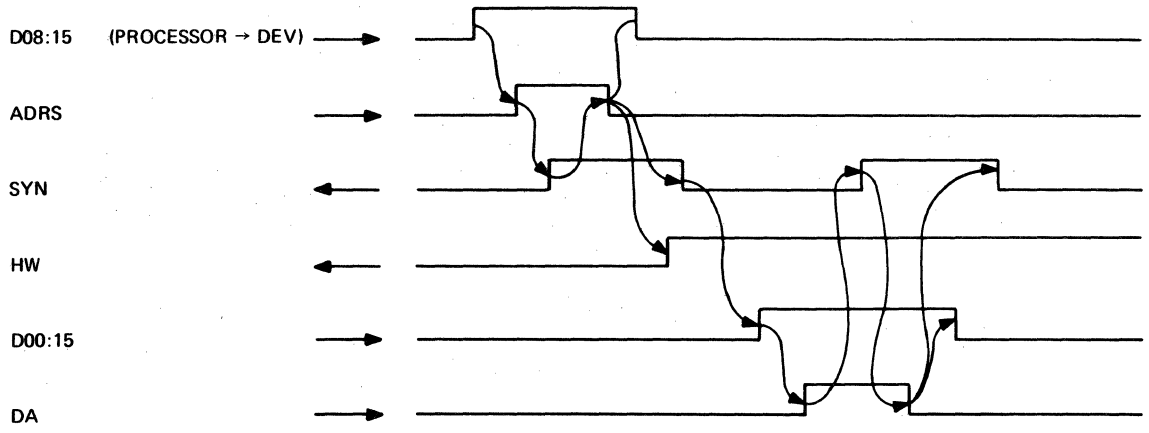


Figure 5-15. Write Halfword Instruction Timing for Halfword Oriented Devices

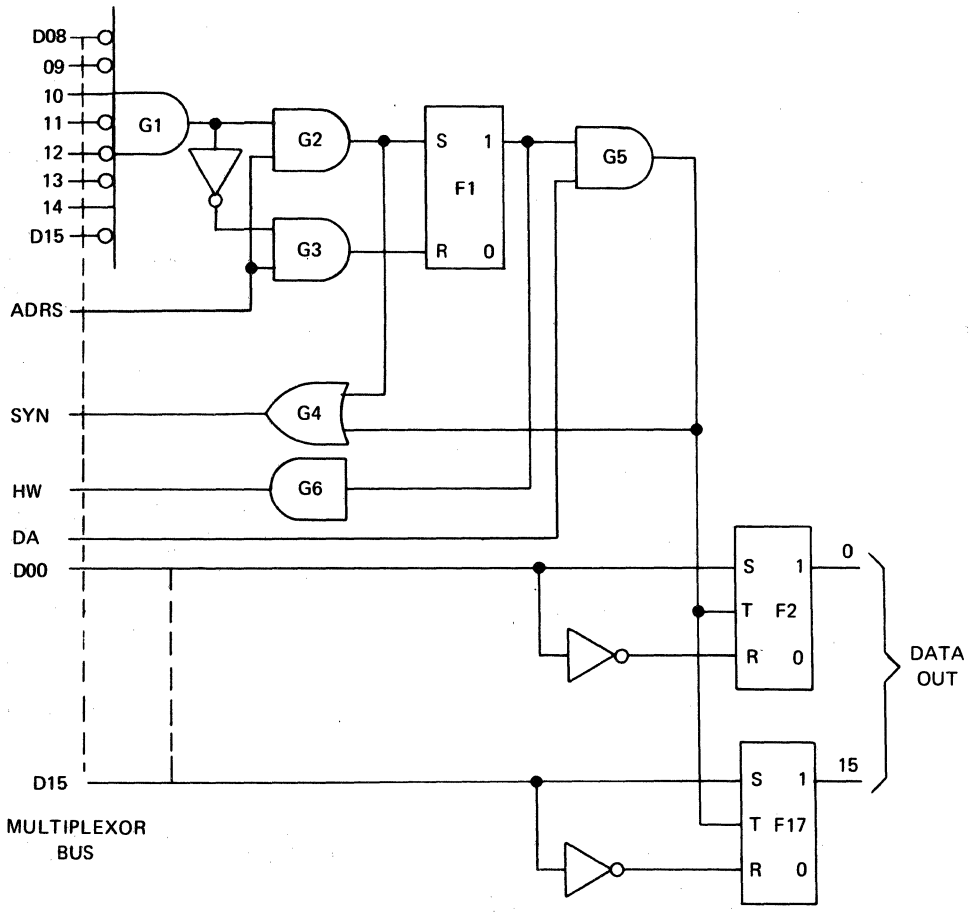


Figure 5-16. Halfword Oriented Device Controller Logic for Write Halfword

4. The Data Available (DA) control line is then raised.
5. The DA control line is ANDed with flip-flop F1 by Gate G5. Gate G5 strobes the halfword of data into the device controller register (F2 through F17).

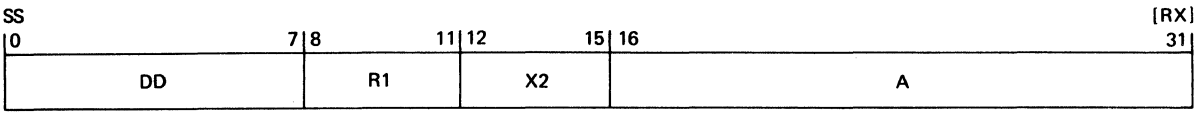
6. The output from Gate G5 also generates SYN to the Processor indicating that the halfword of data has been received.

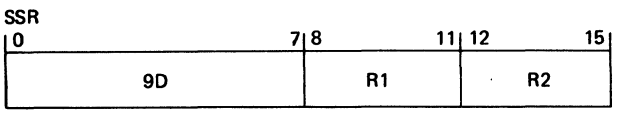
7. When the Processor receives SYN it lowers the DA control line; which removes the halfword of data from the Data Lines (D000 through D150).

8. The device controller removes SYN.

The device controller Address flip-flop and the Time Out feature are as described in Section 5.3.1 for the RD instruction. The Write Halfword from Register (WHR) instruction is executed in the same manner as the WH instruction, except that the byte which is transferred originates in General Register R2 rather than the memory halfword specified by A.

5.3.5 Sense Status (SS) Instruction





Execution of the SS instruction transfers an 8-bit Status Code from the device specified by General Register R1 to memory location A, indexed by the contents of General Register X2. In addition, the four least significant bits are placed in the four bit Condition Code of the Program Status Word (PSW).

C = Device Busy - Indicates that the device is not ready to transfer data.

V = Examine Status (EX) - Indicates that the device has detected a condition which is indicated by the most significant four bits of the Status Condition Code.

G = End of Medium (EOM) - Indicates that the device has reached the end of its data. For example, the Card Reader has reached the end of a card.

L = Device Unavailable (DU) - Indicates that the device is either not connected or not ready.

Thus, the program, after executing an SS instruction, may branch directly on any of the above conditions. Normally if the V flag is set, the program examines the other four bits of the Status Condition Code. The Status Byte is stored at memory location A, and the four bits may be assigned any significance which is appropriate for the particular device.

Refer to Figures 5-17 and 5-18 during the following sequence of operations for the SS instruction.

1. The device controller is addressed and connected as described in Steps 1 through 5 of the RD sequence, Section 5.3.1.

2. The Processor next raises the Status Request (SR) control line.

3. The output from Gate G7 enables the Status Byte from the device to the Processor via the Data Lines (D080 through D150), and sends a SYN response to the Processor to indicate that the data is on the bus.

4. When it receives the SYN signal, the Processor transfers the Status Byte Bits 0 through 7 to Address A, and Bits 4 through 7 only to the Condition Code of the PSW.

5. The Processor then lowers SR, which causes the device controller to lower SYN and remove the Status Byte from the bus.

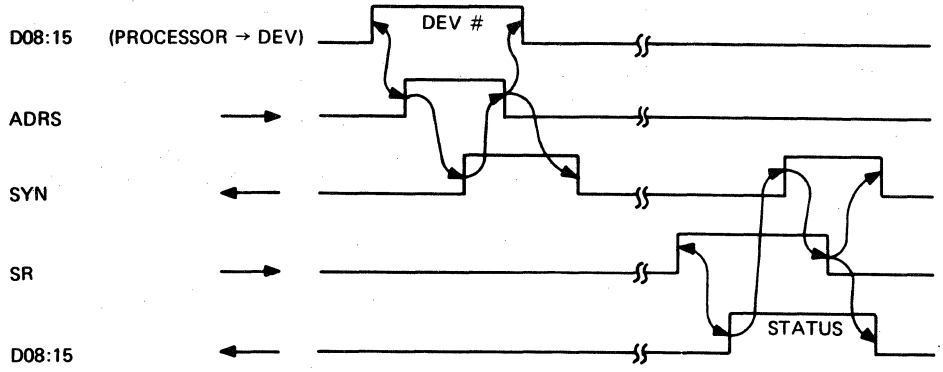


Figure 5-17. Sense Status Instruction Timing

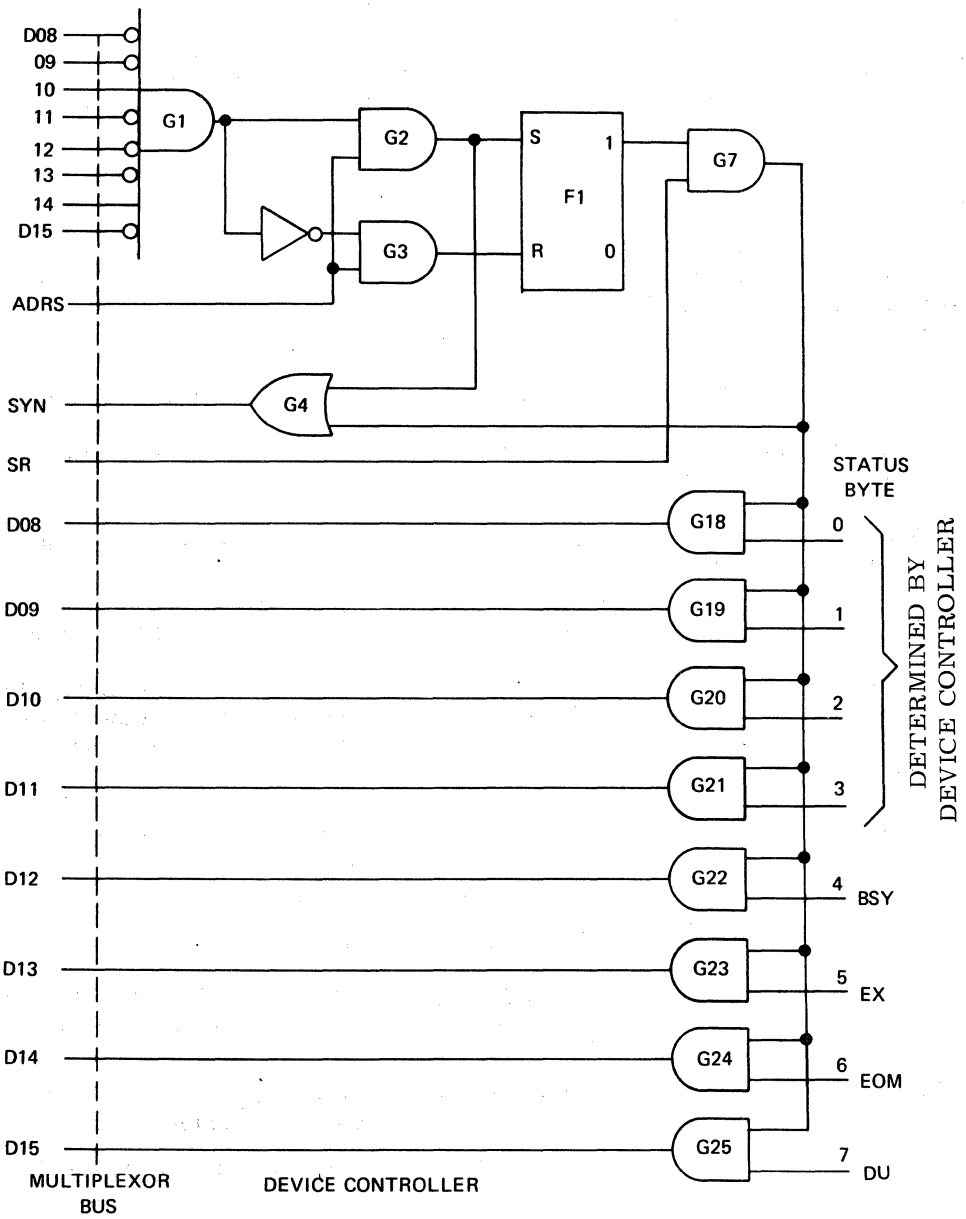
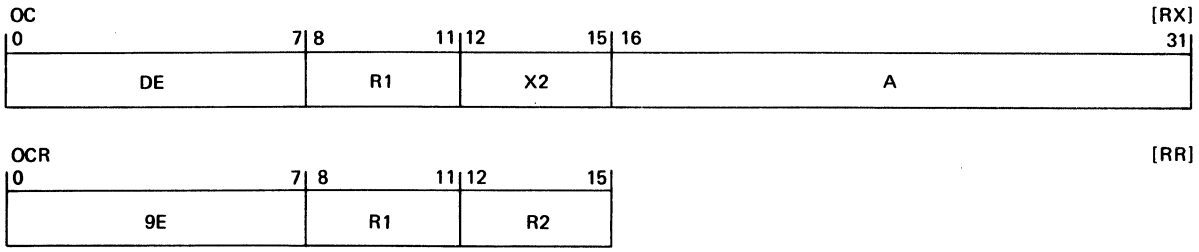


Figure 5-18. Device Controller Logic for Sense Status Instruction

The device controller Address flip-flop and the Time Out feature are as described in Section 5.3.1 for the RD instruction. The SSR instruction is similar to the SS instruction except that the Status Byte is stored in General Register R2 instead of memory address A.

5.3.6 Output Command (OC) Instruction



Execution of the OC instruction transfers an 8-bit Output Command from address A, indexed by the contents of General Register X2, to the device specified by the contents of General Register R1. Command line coding is normally assigned to either device control function or device controller modes of operation. None of the command bits are preassigned a specific function. The OC instruction is therefore a powerful instruction which may be tailored to the specific requirements of a particular system.

Refer to Figures 5-19 and 5-20 during the following sequence of operation description.

1. The device controller is addressed and connected as described in Steps 1 through 5 of the RD sequence, Section 5.3.1.
2. The Processor next places the contents of the byte address A (the command word) on the Data Lines (D080 through D150).
3. The Processor than raises the Command (CMD) control line. The CMD signal is ANDed with the Address flip-flop (F1) by Gate G8.
4. The output from Gate G8 strobes the Command word on the Data Lines (D080 through D150) into the Command Control Register in the device controller (F10 through F17). The output from Gate G8 also sends a SYN response to the Processor to indicate that the device has stored the Command word.
5. The Processor then lowers CMD and removes the Command word from the Data Lines (D080 through D150).
6. Finally, the device controller lowers its SYN line.

The device controller Address flip-flop and the Time Out feature are as described for the RD instruction in Section 5.3.1. The OCR instruction is similar to the OC instruction except that the Command Byte is from General Register R2 instead of Address A.

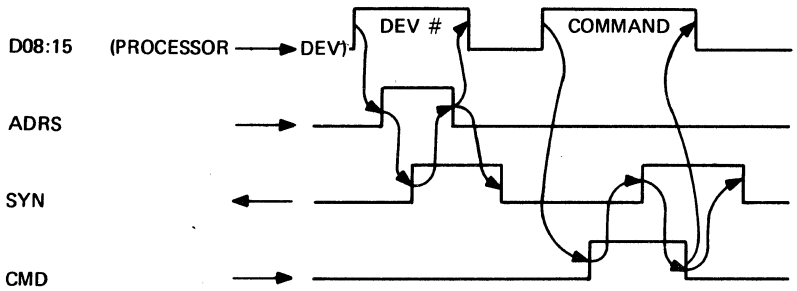


Figure 5-19. Output Command Instruction Timing

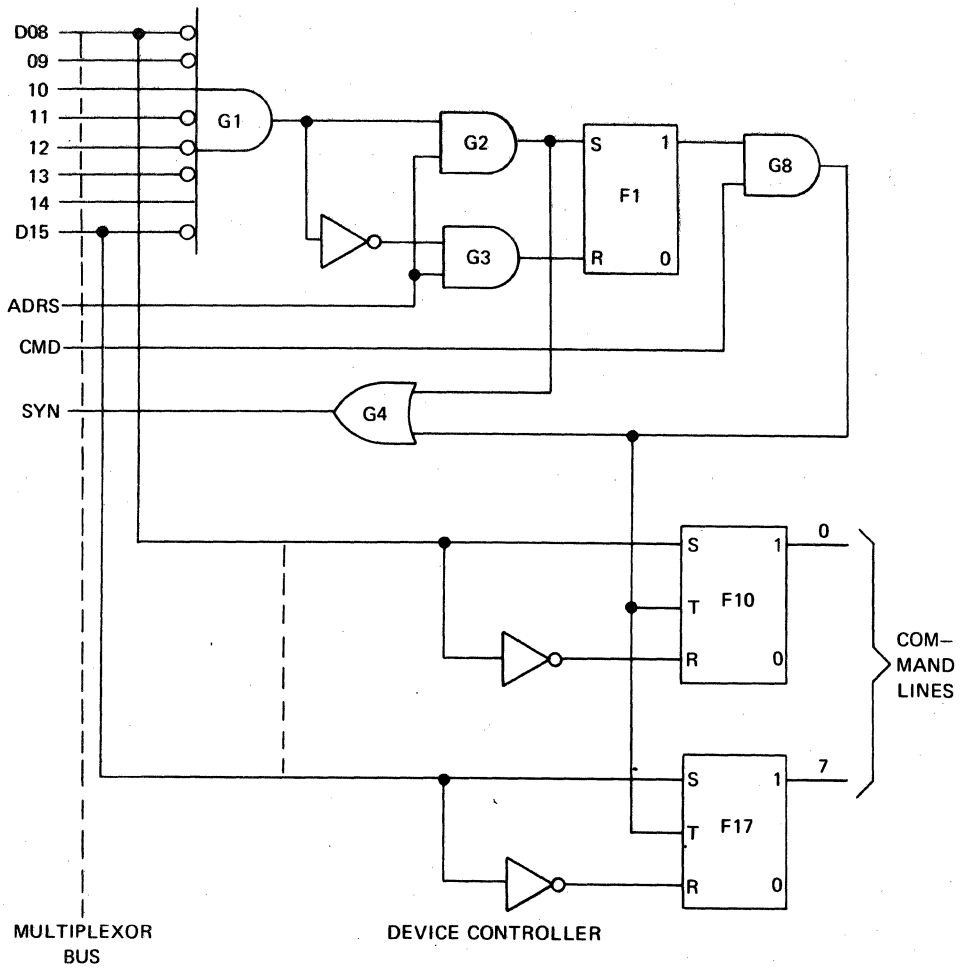
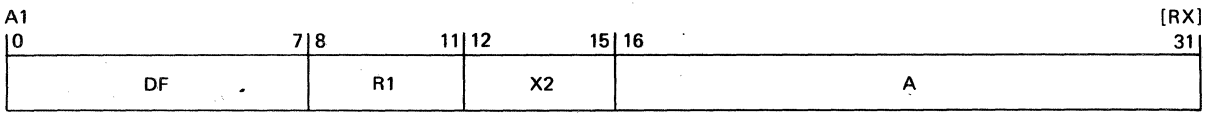
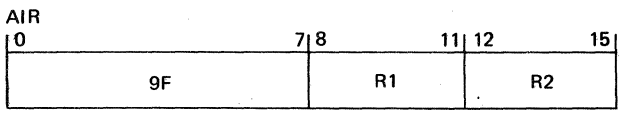


Figure 5-20. Device Controller Logic for Output Command Instruction

5.3.7 Acknowledge Interrupt (AI) Instruction





To understand the AI instruction, the reader should have a working knowledge of the 3010/2 Interrupt System. A brief description of the Interrupt System is provided here; for more detail refer to Chapter 2. Interrupts are provided to detect events both within, and external to, the Processor. The occurrence of the interrupt may cause the program to Branch from its normal sequence to an interrupt subroutine which performs some operation appropriate to the interrupt. For example, an external device may interrupt to indicate that it is ready to transfer another byte of data. The interrupt subroutine would then initiate another transfer. After the interrupt is serviced (the interrupt subroutine is completed), the main program is resumed at the point at which it was interrupted.

The AI instruction is normally the first instruction in the interrupt subroutine which services interrupts from external devices. When executed, the device number of the interrupting device is placed in General Register R1, and the Status Byte of the interrupting device is placed in Address A, indexed by the contents of General Register X2. The least significant four bits of the Status Byte are placed in the Condition Code of the Program Status Word (PSW) exactly as described in Section 5.3.5 for the SS instruction. Thus, for example, with a single AI instruction a magnetic tape reader may be identified, and the fact that it has reached an EOR gap determined (via the EOM Status flag).

Refer to Figures 5-21 and 5-22 during the following sequence of operation:

1. The device interrupts and sets flip-flop F18 in the device controller. The output from F18 generates an Attention (ATN) signal to the Processor, if interrupts have been enabled by a prior Output Command causing ENABLE to be active.
2. The Processor responds by raising the Acknowledge (ACK) control line.

> NOTE
>
> The ACK line is received by the first device controller in the line as Receive Acknowledge (RACK). See Figure 5-22. If the Interrupt flip-flop (F18) in the first controller is not set, the RACK signal is gated out of this controller as Transmit Acknowledge (TACK). The next controller receives it as RACK. Thus the ACK signal "daisy chains" through the device controllers until it finds one with its Interrupt flip-flop (F18) set. The '0' output from F18 inhibits the propagation of TACK to the next device controller.

3. The F18 high output and the RACK signal are ANDed to enable the device number from the device to the Data Lines (D080 through D150), and to send SYN to the Processor to indicate that the device number is on the lines. The ATN flip-flop is also reset.
4. The Processor gates the device number into General Register R1.
5. The Processor then lowers the ACK line which, in turn, causes the device controller to lower the SYN line.
6. The Processor then addresses the same device and gates its Status Byte to Address A and the Condition Code of the PSW exactly as described in Steps 1 through 5 of the Sense Status Instruction, Section 5.3.5.

The device controller Address flip-flop and the Time Out feature are as described previously in Section 5.3.1 for the RD instruction. The AIR instruction transfers the Status Byte to General Register R2, rather than to memory location A, as in the AI instruction.

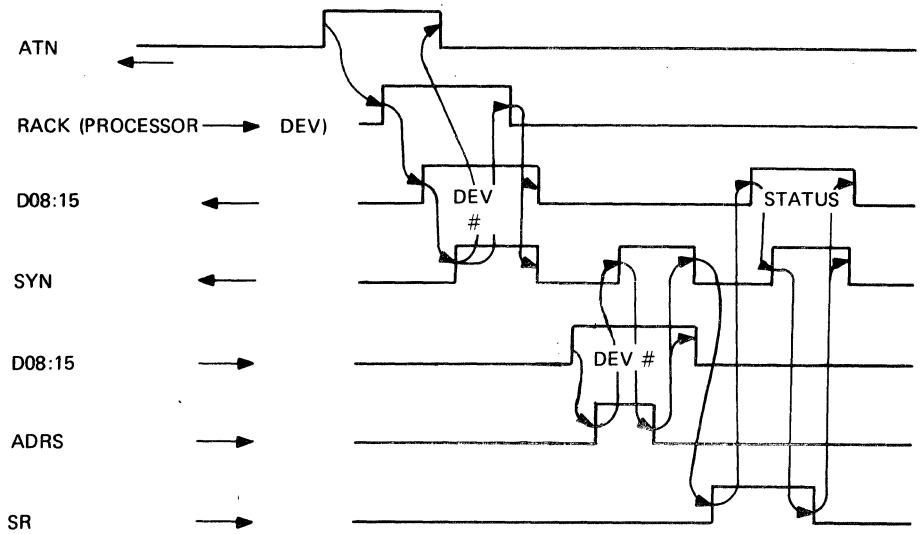


Figure 5-21. Acknowledge Interrupt Instruction Timing

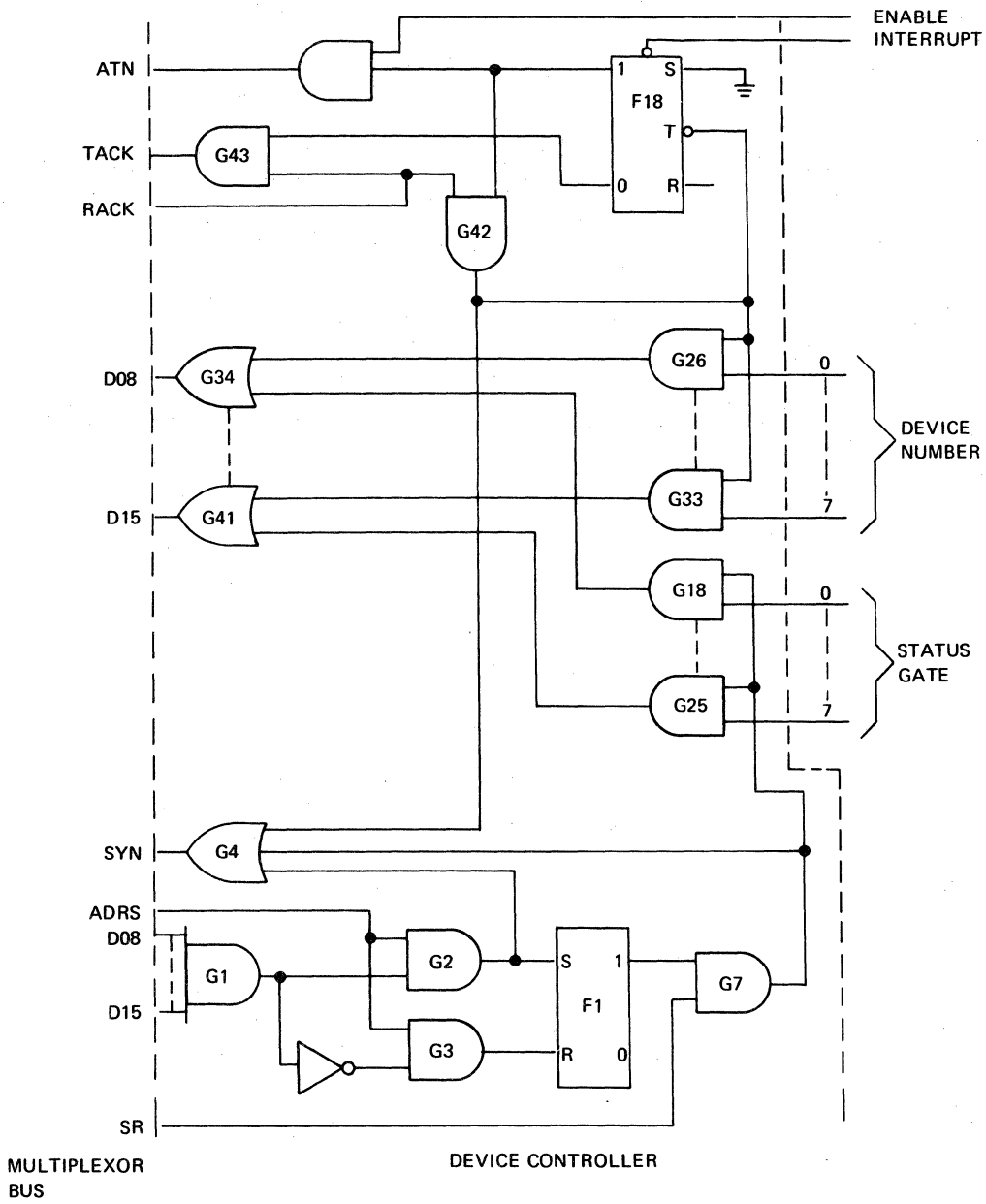
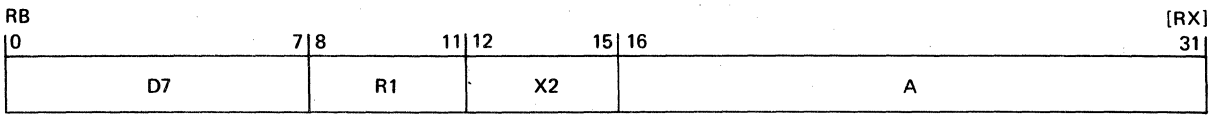
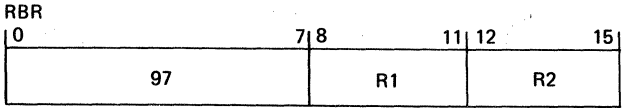


Figure 5-22. Device Controller Logic for Acknowledge Interrupt Instruction

5.3.8 Read Block (RB) Instruction





General Register R1 specifies the device number, the indexed address A contains the starting address for the block transfer. The next sequential halfword contains the ending address. When an RB instruction is executed, a block of data is transferred from an external device to sequential byte locations in memory.

Between each byte transfer the device status is checked. The four least significant bits of the Status Byte are scanned. If the BSY Bit (C flag) is set, the Processor assumes that the transfer is still in progress. The Processor initiates another Status Byte input sequence each time the BSY Bit is set. When the BSY Bit is low (indicating that the byte transfer is complete), the Processor scans the remaining three least significant bits. If any of these bits are set, the transfer sequence is terminated. If all bits are reset, the next byte transfer is initiated. At the end of the transfer sequence (when the present address equals the ending address), the Status Condition Code should contain all zeros. The program may therefore Branch conditionally on the Status Condition Code following the RB (or RBR) instruction. Condition Code assignments are as described in Section 5.3.5. Refer to Figures 5-23 and 5-24 during the following sequence of operation description. The sequence is essentially a combination of RD and SS instructions. However, the device is only addressed once, the DR and SR control lines are then raised alternately until the transfer is terminated.

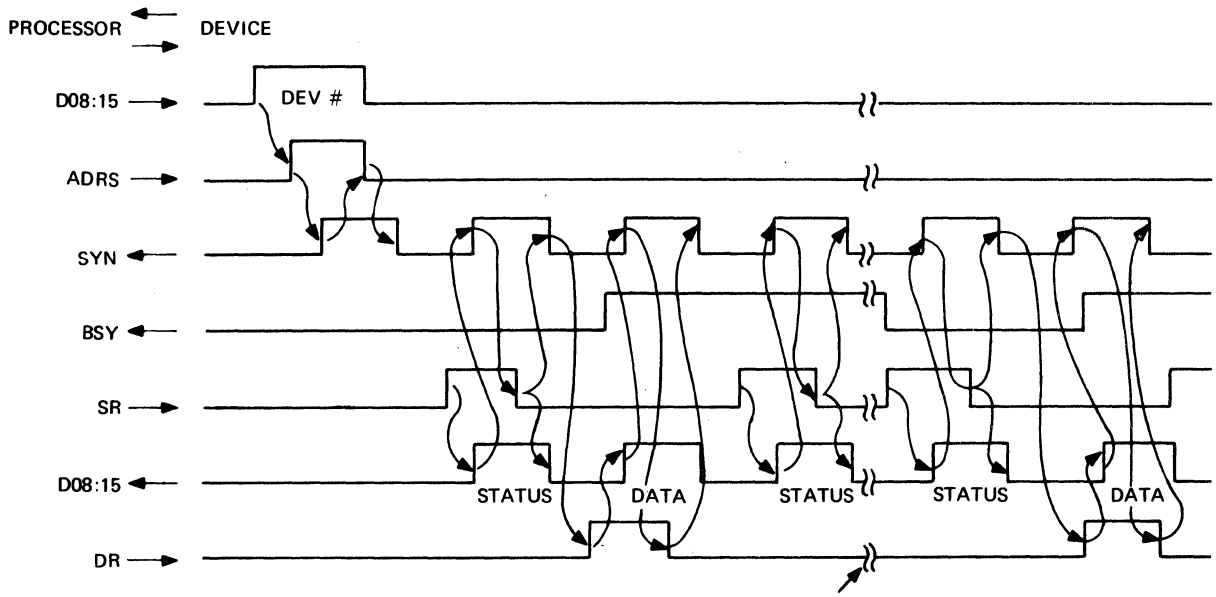


Figure 5-23. Read Block Instruction Timing

1. The device controller is addressed and connected as described in Steps 1 through 5 of the RD sequence, Section 5.3.1.
2. A byte is transferred from the device to the Processor as described in Steps 6 through 9 of the RD sequence, Section 5.3.1.
3. The Status Byte from the device is then transferred to the Processor as described in Steps 2 through 5 of the SS instruction sequence, Section 5.3.5.
4. If the address does not match the final address, and the Condition Code of the PSW is all zeros, the Processor increments the address and repeats Steps 2 through 4 of this sequence.
5. The instruction terminates normally when the address equals the final address. If Bit 5, 6, or 7 of the Condition Code in the PSW is set after a transfer, the instruction is terminated. As described previously, the program may then Branch conditionally on the Condition Code. If Bit 4 (BSY) is set, Step 3 is repeated.

The device controller Address flip-flop and the Time Out feature are as described previously in Section 5.3.1 for the RD instruction. The RBR instruction differs from the RB instruction only in that the starting address is specified by the contents of General Register R2.

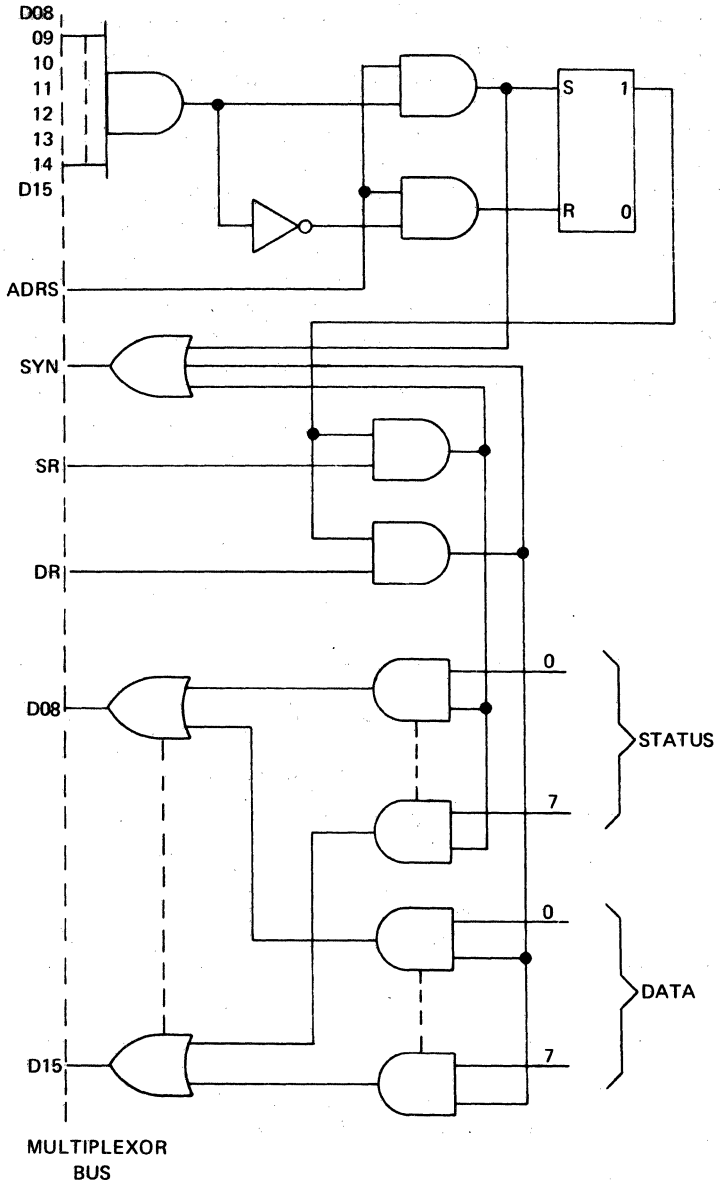
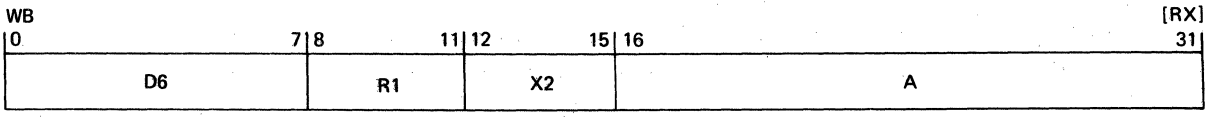
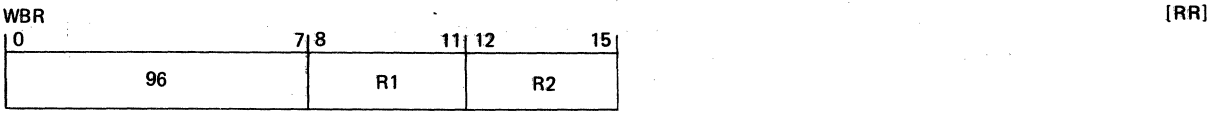


Figure 5-24. Device Controller Logic Read Block Instruction

5.3.9 Write Block (WB) Instruction





The General Register specified by R1 contains the device number. The indexed address A contains the starting address for the block transfer. The next sequential halfword contains the ending address. Execution of this instruction transfers bytes from sequential locations in memory to the specified external device. The Status Byte is checked between each byte transfer. The four least significant bits of the Status Byte are scanned. If the BSY Bit (C flag) is set, the Processor assumes that the transfer is still in progress. The Processor initiates another Status Byte input sequence each time the BSY Bit is set. When the BSY Bit is low (indicating that the byte transfer is complete), the Processor scans the remaining three least significant bits. If any of these bits are set, the transfer sequence is terminated. If all bits are reset, the next byte transfer is initiated. At the end of the transfer sequence (when the present address equals the ending address), the Status Condition Code should contain all zeros. The program may therefore Branch conditionally on the Status Condition Code following the WB (or WBR) instruction. Condition Code assignments are as described for the Sense Status instruction in Section 5.3.5. Refer to Figures 5-25 and 5-26 during the following sequence of operation description.

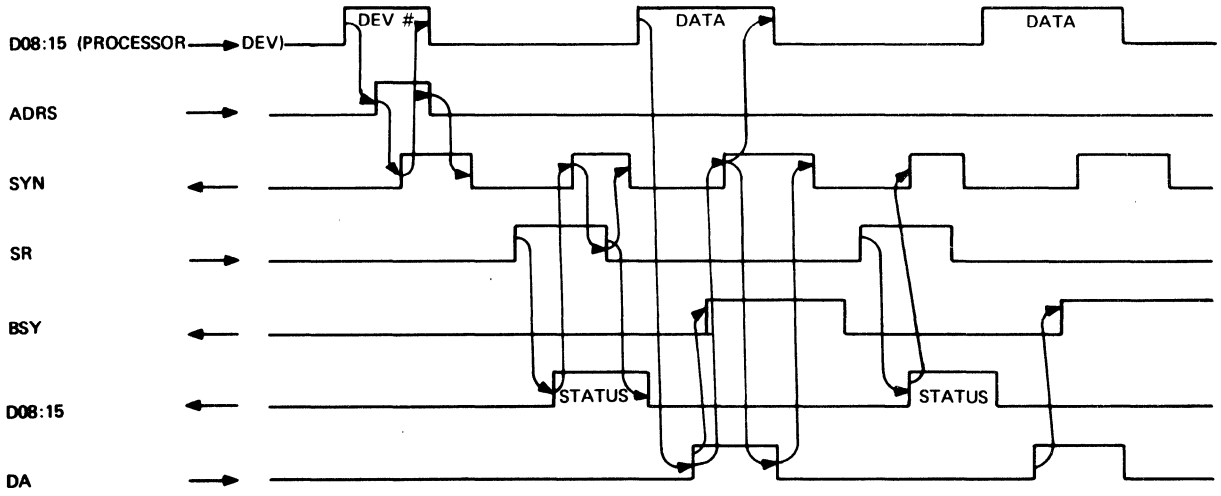


Figure 5-25. Write Block Instruction Timing

1. The device controller is addressed and connected as described in Steps 1 through 5 of the RD sequence, Section 5.3.1.
2. A byte is transferred from the Processor to the device as described in Steps 2 through 7 of the WD sequence, Section 5.3.3.
3. The Processor reads in the Status Byte as described in Steps 2 through 5 of the SS sequence, Section 5.3.5.
4. If the BSY indication is set, Step 3 is repeated to input the Status Byte again.
5. If the BSY indication is low, and any of the three least significant bits are set, the transfer is terminated.
6. If all four least significant Status Byte bits are reset, the Processor compares the memory address with the ending address. If they are equal, the transfer is terminated. If the addresses are not equal, the address if incremented and Steps 2 through 6 of this sequence are repeated.

Again, the device controller Address flip-flop and the Time Out feature are as described previously for the RD instruction, Section 5.3.1. The WBR instruction is the same as the WB instruction except that the starting address is specified by the contents of General Register R2 rather than by the effective address.

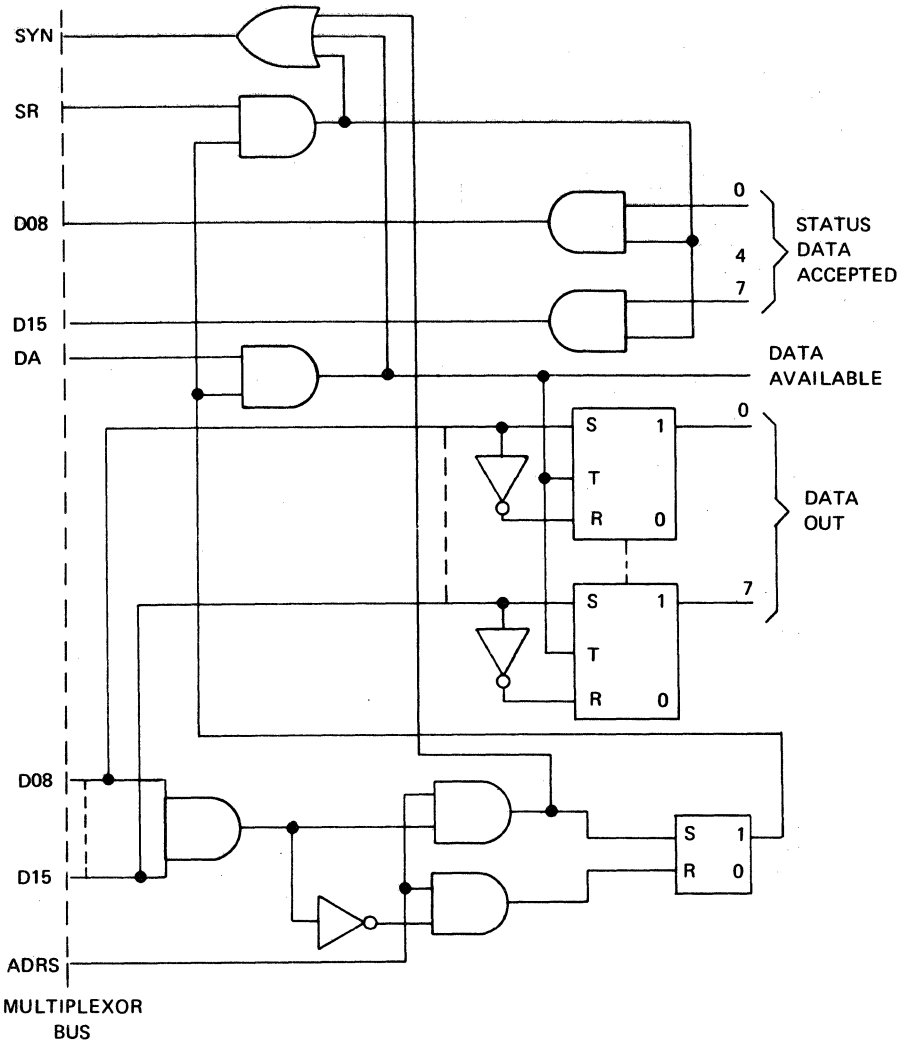
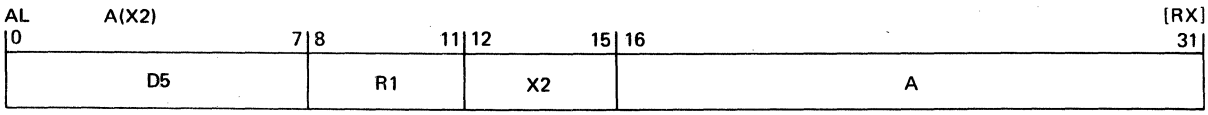


Figure 5-26. Device Controller Logic for Write Block Instruction

5.3.10 Autoload (AL) Instruction



The Autoload instruction loads memory with a block of data from a byte oriented input device (e.g., teletypewriter, photo-electric Paper Tape Reader, etc.). The data is read a byte at a time and stored in successive memory locations starting with location X'80'. The last byte is loaded into the memory location specified by the address of the second operand, A + (X2). Any blank or zero bytes that are input prior to the first non-zero byte are considered to be leader and are therefore ignored; all other zero bytes are stored as data. The input device number is specified by memory location X'78'. The device command code is specified by memory location X'79'. The R1 field has no significance in this instruction.

The AL instruction is a combination of an Output Command (OC) instruction and a Read Block (RB) instruction. Refer to Figure 5-27 during the following sequence of operation description.

1. The Processor places the contents of memory location X'78' onto the Data Lines (D080 through D150).

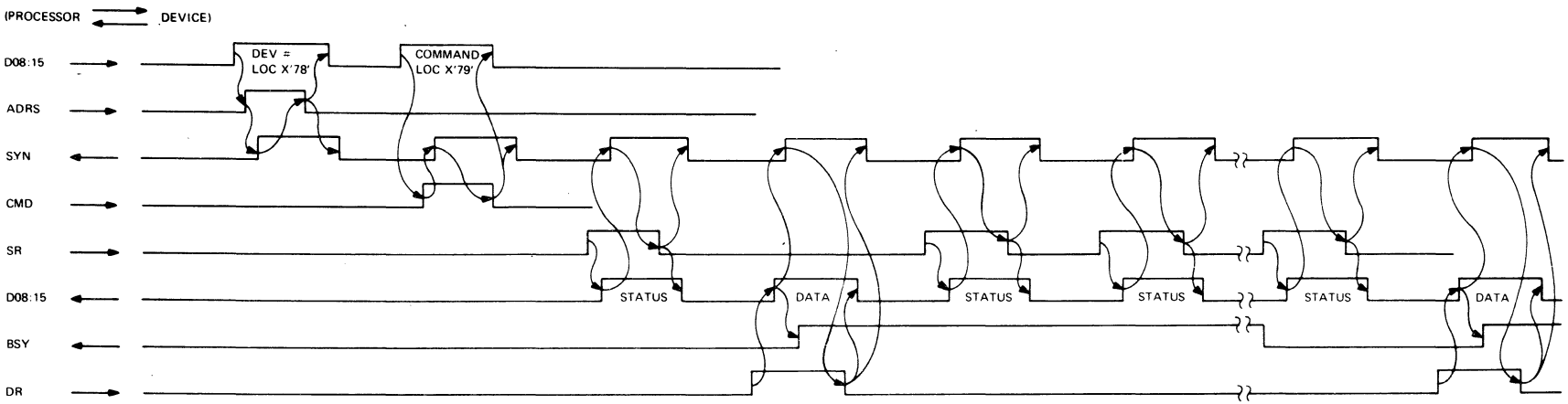


Figure 5-27. Autoload Instruction Timing

2. The device controller is addressed and connected as described in Steps 2 through 5 of the RD Sequence, Section 5.3.1.

3. The Processor next places the contents of memory location X'79' onto the Data Lines (D080 through D150).

4. The device controller is now issued a Command as described in Steps 3 through 6 of the OC sequence, Section 5.3.6.

5. The instruction continues exactly as the Read Block instruction except that the data bytes are loaded to memory beginning at memory address X'80' and that the final byte address is defined by A indexed by X2.

5.4 DEVICE CONTROLLER LOGIC DESIGN

The design of the Multiplexor Channel and Bus and the Multiplexor Bus interface in all controllers on the bus, is described in this subsection, so that the operation of the hardware and the user I/O instructions may be understood. While the actual circuitry used in the various device controller bus interfaces may differ, the operation is the same for all controllers.

5.4.1 Multiplexor Bus

The Multiplexor Channel is a byte or halfword oriented I/O system which communicates with up to 255 peripheral devices. The Multiplexor Bus consists of 30 lines; 16 bi-directional Data Lines, 8 Control Lines, 5 Test Lines, and an Initialize Line.

The lines in the Multiplexor Bus are listed below:

Data Lines	D00:15	(Processor ⇆ Device)	16 Lines
Control Lines	SR	(⟶)	1 Line
	DR	(⟶)	1 Line
	CMD	(⟶)	1 Line
	DA	(⟶)	1 Line
	ADRS	(⟶)	1 Line
	ACK	(⟶)	1 Line
	DACK	(⟶)	1 Line
	CL07	(⟶)	1 Line
Test Lines	ATN	(⟵)	1 Line
	SYN	(⟵)	1 Line
	HW	(⟵)	1 Line
	DC	(⟵)	1 Line
	DCR	(⟵)	1 Line
Initialize Line	SCLRO	(⟶)	1 Line

The following general definitions apply to the lines in the Multiplexor Bus:

Data Lines D00:15

The data lines are used to transfer one 8-bit byte or one 16-bit halfword of data between the Processor and the device. One byte of address or command is transferred from the Processor to the device over Data Lines 8:15 (D08:15) when accompanied by either an Address (ADRS) or a Command (CMD) control line. One byte of data or one halfword of data is transferred from the Processor to the device when accompanied by the Data Available (DA) control line. The device, in response to an Acknowledge (ACK) control line or a Sense Status (SR) control line, sends one byte of address or status information to the Processor over D08:15. In response to a Data Request (DR) control line, the device sends either an 8-bit byte or a 16-bit halfword of data to the Processor. The device always sends a Synchronize (SYN) signal to the Processor to indicate that it has either received the data from the Processor or that it has sent the data to the Processor. The SYN signal is removed immediately after the Processor removes the control line.

Control Lines

SR — Status Request. The device controller must present device status to Data Lines (D08:15), followed by a SYN.

DR — Data Request. The device controller presents data to Data Lines 8:15 or 0:15 (D08:15 or D00:15), followed by a SYN. If a Halfword (HW) of data is presented, the HW test line is also active.

ACK — Acknowledge. The interrupting device controller presents its address on D08:15, followed by a SYN.

DA — Data Available. The Processor presents data on D00:15 for transfer to the device. The device controller accepts the low byte or the entire halfword and responds with a SYN.

CMD — Command. The Processor presents a Command Byte on D08:15. The device controller accepts the Command Byte and responds with a SYN.

ADRS — Address. The Processor presents an Address Byte on D08:15. The device controller accepts the Address byte and responds with a SYN.

DACK — Data Channel Acknowledge. The Processor presents an address of zero on D08:15. The ADRS control line and the DACK control line are simultaneously active. The interrupting Data Channel device controller becomes selected and responds with a SYN. As a result of addressing device zero (a null address), only the selected data channel device controller remains addressed.

CL070 — This control line is activated by the Processor when a Power Fail condition is detected by the Processor if the Power Fail option is equipped. This line is held active until the SCLR0 signal occurs.

Test Lines

ATN — Attention. Any device desiring to interrupt the Processor will activate the ATN line and hold this line until an ACK is received from the Processor.

HW — Halfword. The HW line is activated by a halfword oriented device controller whenever it is communicating normally with the Processor. The HW line is not activated when a device controller is operating in the Interleaved Data Channel mode.

DC — Data Channel Request. Any Data Channel device desiring to interrupt the Processor will activate the DC line and hold this line until a DACK is received from the Processor.

DCR — Data Channel Read. The selected Data Channel device controls the state of the DCR line, high for read, low for write, from the device.

SYN — Synchronize. This signal is generated by the device to inform the Processor that it has properly responded to a control line.

Initialize Line

SCLR — System Clear. This is a metallic contact to ground which is closed during Power Fail, Power Up or Initialize.

NOTE

All control lines, except ACK and DACK, are connected in parallel to all devices. These lines are activated by the Processor in response to an external interrupt. The ACK line is connected in series with all devices. If no interrupt is pending in the first controller when the ACK or DACK signal arrives, the signal is passed on daisy chain fashion to the next controller, and so on until it is captured by the interrupting controller. See definition of ACK and DACK.

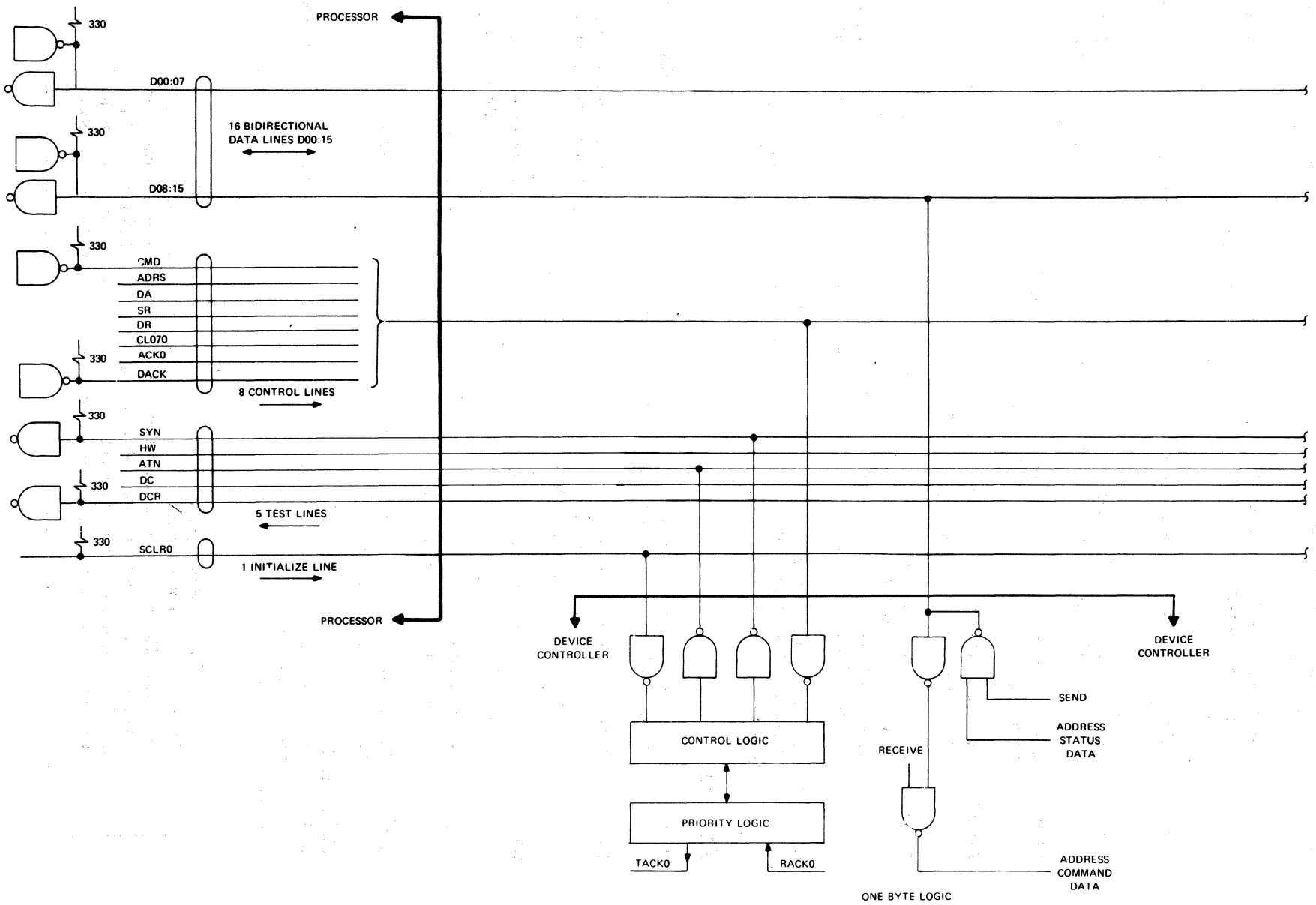


Figure 5-28. Processor/Device Controller Logic Interface (Sheet 1 of 2)

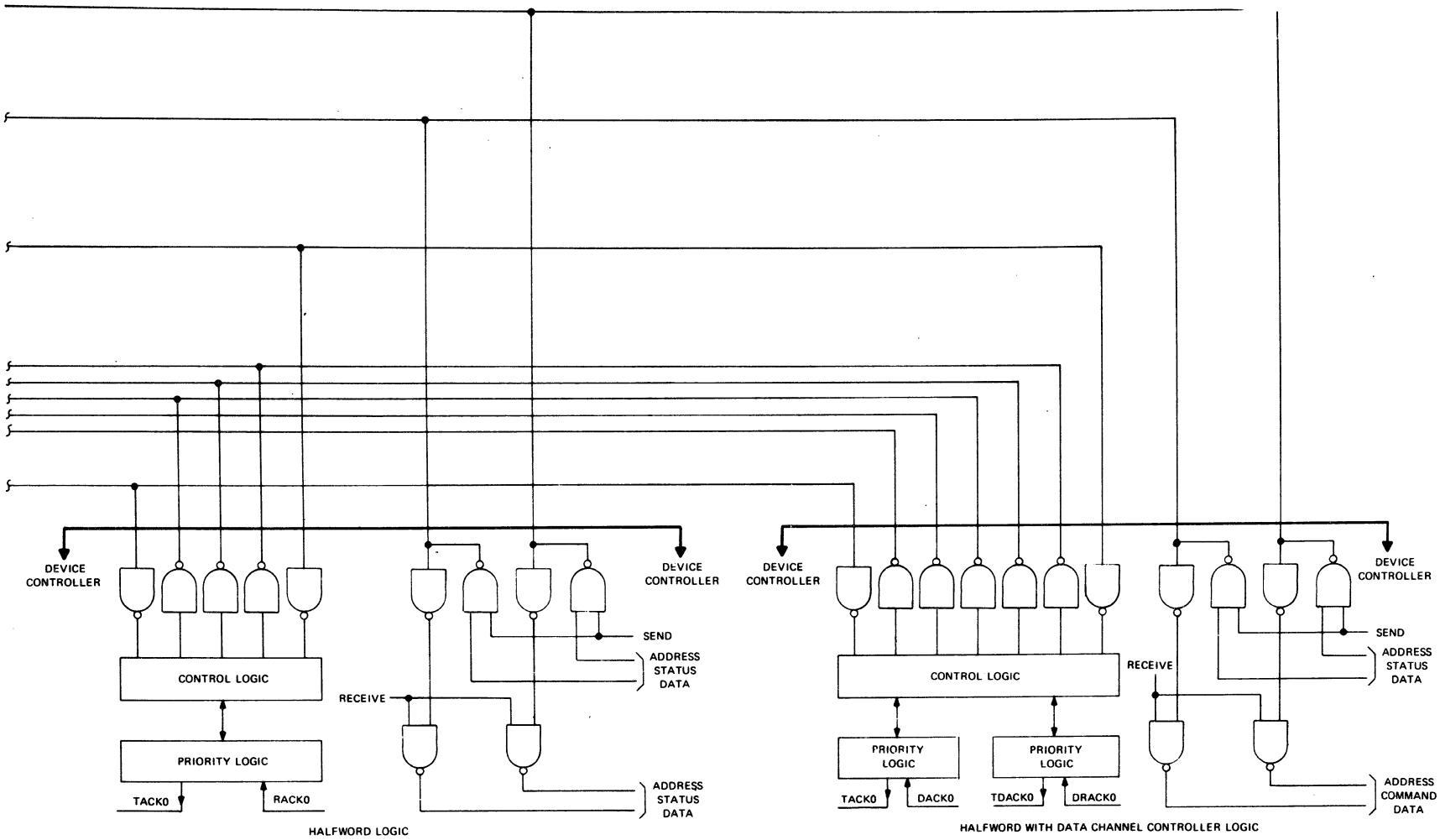


Figure 5-28. Processor/Device Controller Logic Interface (Sheet 2 of 2)

All buses are false type, i.e. low level is active, high level is inactive. The device controller circuits used to communicate with the Multiplexor Bus are shown in Figure 5-28.

In a typical case, a device controller will receive an 8-bit Address Byte, an 8-bit Command Byte, and either an 8-bit data byte or a 16-bit data halfword from the Processor over the 16 bi-directional Data Lines (D00:15). Likewise, a device controller will send an 8-bit Address Byte, an 8-bit Status Byte, and either an 8-bit data byte or a 16-bit data halfword to the Processor over the 16 bi-directional Data Lines (D00:15). When only a byte of data is transferred, that byte is passed over the lower eight Data Lines (D08:15). The load resistors for all lines in the Multiplexor Bus are located in the Processor.

Each device controller is permitted one TTL load on any of the 16 bi-directional Data Lines, the 8 Control Lines, or the single Initialize Line. Each device controller is permitted one high power ope collector TTL OR tie onto each of the 16 bi-directional Data Lines and each of the 5 Test Lines. The Multiplexer Bus section originating in the Processor has a drive capacity of 7 Multiplexer Bus loads i addition to it two built in loads. Several bus extension modules are available, including the Multiplexer Bus Buffer shown on Fig. 5-29. See the 3010/2 Price List and Configurator, PCP-230 and the General Description, GET-6227.

5.4.2 Device Controller Addressing

Refer to Figure 5-30 during the following description. The dotted lines around the groups of logic functions represent standard logic. Further details on the logic packs may be found later in this chapter. A designer of custom equipment could use his own logic modules, provided they are level compatible with standard logic. When a device controller is addressed, the eight-bit address code is placed on the Data Lines (D080 through D150). The two buffers provide the true and false data lines. The Address Decoder circuit is hard-wired on each controller with its assigned address code, and the eight coded outputs are applied to an eight input gate. Thus, the Decoded Device output (DD1) goes true. The Address control line (ADRS1) then strobes the DD1 line into the ADRS flip-flop.

The Synchronize (SYN) signal is returned to the Processor, during the presence of ADRS1, via the Address Sync line, (ADSY0). Notice that an OR gate is used here for returning the other device Command Sync lines. The set output from the Address flip-flop, called Device Enable (DENB1), is used to gate all other I/O control lines to the device controller. When another device is addressed, the Decoded Device line (DD1) is low, causing the ADRS1 strobe line to reset the Address flip-flop and disabling the controller. Thus, only one device controller may be addressed at any time. During the address cycle, only the device that was addressed returns a SYN.

> NOTE
>
> The device controller address logic is designed so that when some other device is addressed, the previously addressed controller will clear its Address flip-flop within 350 nanoseconds. Otherwise the system could have two devices addressed simultaneously.

The device controller logic must delay SYN until it has reacted to the Multiplexor Bus control line, however, unnecessarily long delays serve only to reduce the system input/output operation.

> NOTE
>
> If the device controller is a 16-bit halfword oriented controller it activates the Halfword Enable Line (HW0) immediately when its Address flip-flop is set. The HW0 is used by the Processor to determine if the device is capable of sending or receiving 16-bit halfword data in parallel.

5.4.3 Data and Status Input/Output

5.4.3.1 Data

Figure 5-31 shows how a byte or halfword of data may be read into the Processor. When an 8-bit byte oriented device controller is addressed, DENB1 is high, enabling the Data Request (DR) control line from

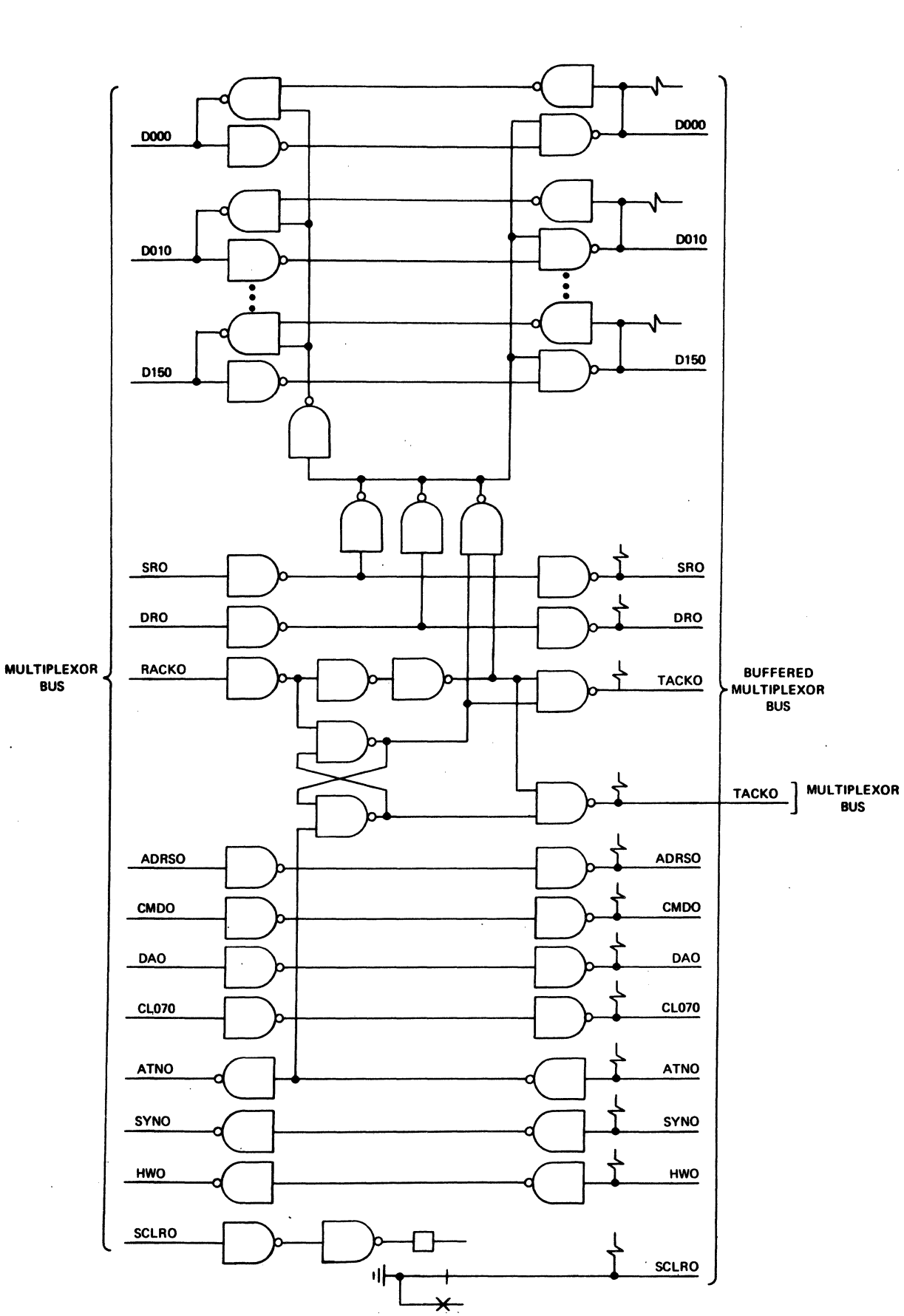


Figure 5-29. Multiplexor Bus Buffer

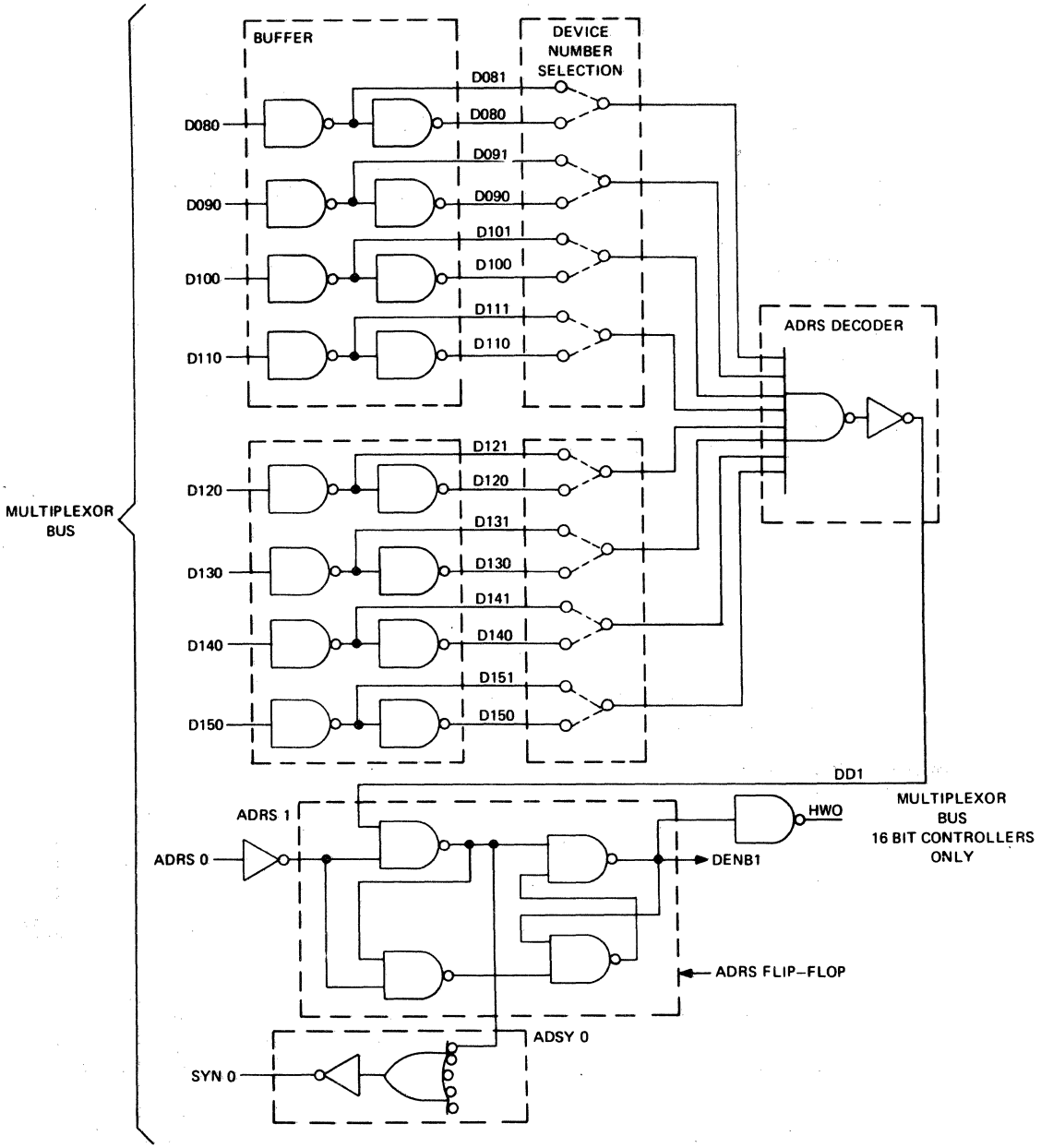


Figure 5-30. Device Addressing, Logic Diagram

the Processor. The DR enables the Data Byte onto the eight bottom Data Lines (D080 through D150). If a 16-bit halfword oriented device controller is addressed, one Data Byte is enabled as described above, and a second Data Byte is enabled onto the eight top Data Lines (D000 through D070) by an active Halfword (HW) signal. A system requirement is that the addressed controller must respond to all control lines (i.e., Data Request) with a SYN.

5.4.3.2 Status

Figure 5-31 shows how a byte of status may be read into the Processor. When the byte or halfword oriented device controller is addressed, DENB1 is high, enabling the Status Request (SR) control line from the Processor. Open collector gates are used for OR tying multiple data and status sources onto the eight Data Lines (D080 through D150). A system requirement is that the addressed controller must respond to all control lines (i.e., Data Request) with a SYN.

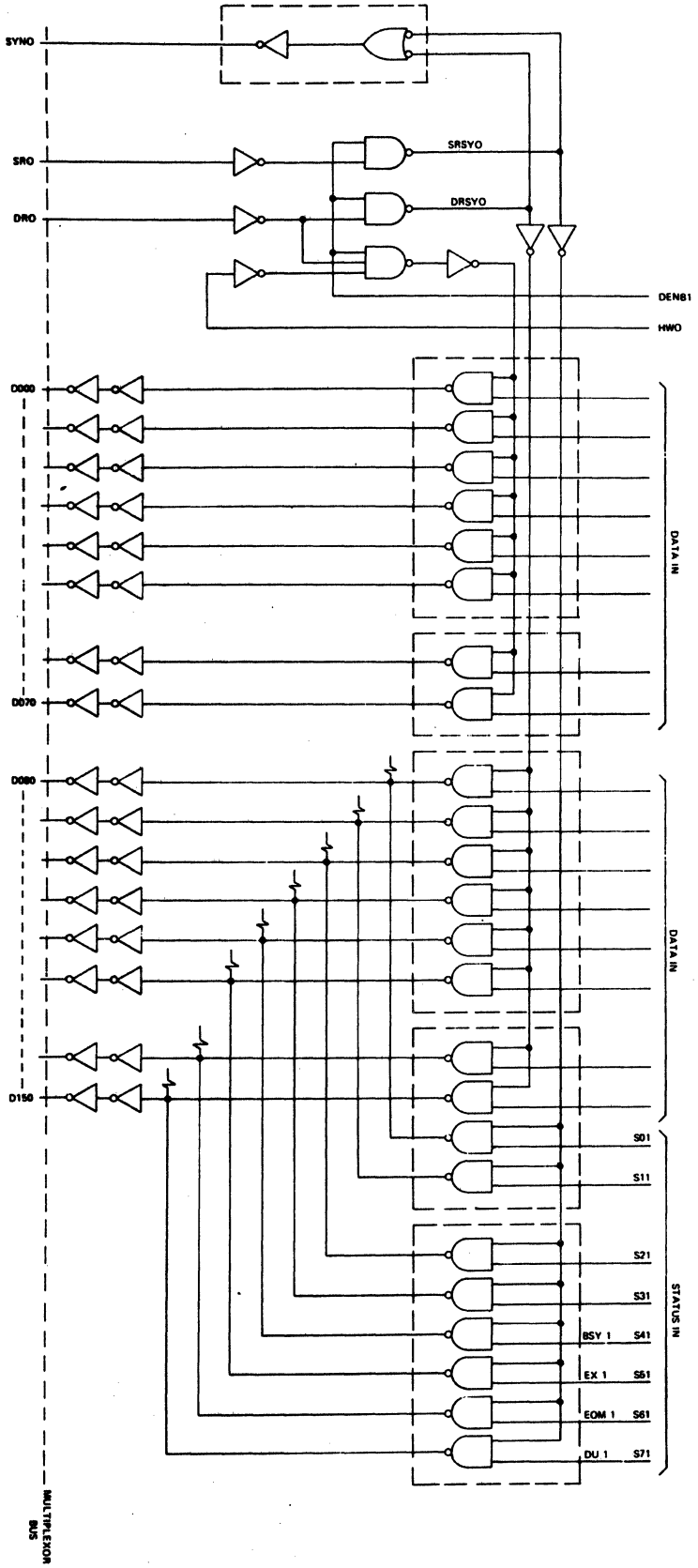


Figure 5-31. Data and Status Input Logic Diagram

The device controller logic should place a high on BSY1 until the data is ready. The Processor may now be synchronized to the device data rate by testing the device status until the Busy Bit is low. Then, when the Busy Bit is low, the program may transfer data. Device synchronization can also be achieved by generating an interrupt when the data is ready.

The End of Medium (EOM) Bit is normally placed high at the termination of the device medium, such as End of Card. The Device Unavailable (DU) Bit typically signifies that device power is not turned on.

The Examine Status (EX) Bit is used to signify other appropriate device conditions. In this case, the user assigns S01 through S31 to appropriate conditions, such as Parity Error, etc.

It is appropriate to note here that the Busy Status is unconditionally defined such that data cannot be transferred unless Busy is false. The remaining status bits are defined as required by the device controller. Not all device controllers require all eight status bits.

Device controllers must be designed such that the Processor or the Selector Channel maintains the Status Request line once the current status of the device is presented. Specifically, if the status changes while the Status Request line is true, the Status Byte returned to the Processor or Selector Channel should also change.

5.4.4 Data and Command Output

5.4.4.1 Data

Figure 5-32 shows how a byte of data may be output from the Processor. The buffered true and false Data Lines (D081 through D151 and D080 through D150) from Figure 5-30 connect to the set and reset inputs of the Data Register.

> NOTE
>
> If the device controller is a 16-bit halfword oriented controller it must typically invert Data Lines D000 through D007 also.

When the device is addressed, DENB1 is high, enabling the control line DAG1 to strobe the data condition into the J-K flip-flop Data Register. The DASY0 line also returns the SYN signal to the Processor.

5.4.4.2 Command

The command lines are shown on Figure 5-32 as being used in the toggle mode. For example, a high on Bit 8 (D081) sets a control relay when CMG1 goes high. A high on Bit 9 (D091) resets the relay. Bits 14 and 15 are shown operating an indicator. Other pairs of bits may be used to enable/disable interrupts, etc.

Again, note that definition of the Command Bits is a function of the device controller only. Not all device controllers require eight separate commands. However, up to 256 commands are possible.

5.4.5 Interrupt Control

Figure 5-33 shows a complete general purpose interrupt and interrupt acknowledge logic system. When an interrupt is generated, the Queue flip-flop is DC set via a differentiated negative going pulse. The output from the Queue flip-flop generates an Attention signal (ATN0) to the Processor. ATN0 is connected to the interrupt line in the Processor. The program responds with an Acknowledge Interrupt signal, which is received by the controller as Receive Acknowledge (RACK). Since the Queue flip-flop was set prior to receiving RACK, the output of Gate G1 disables Gate G9, holding the Gate G9 output high. The high output from Gate G9 stops TACK0 from sending the Acknowledge to the next device. Thus, RACK1 and the Gate G2 output generate ATSY0 via Gate G3. ATSY0 sends a SYN back to the Processor, and also forces the outputs from G18 through G25 high.

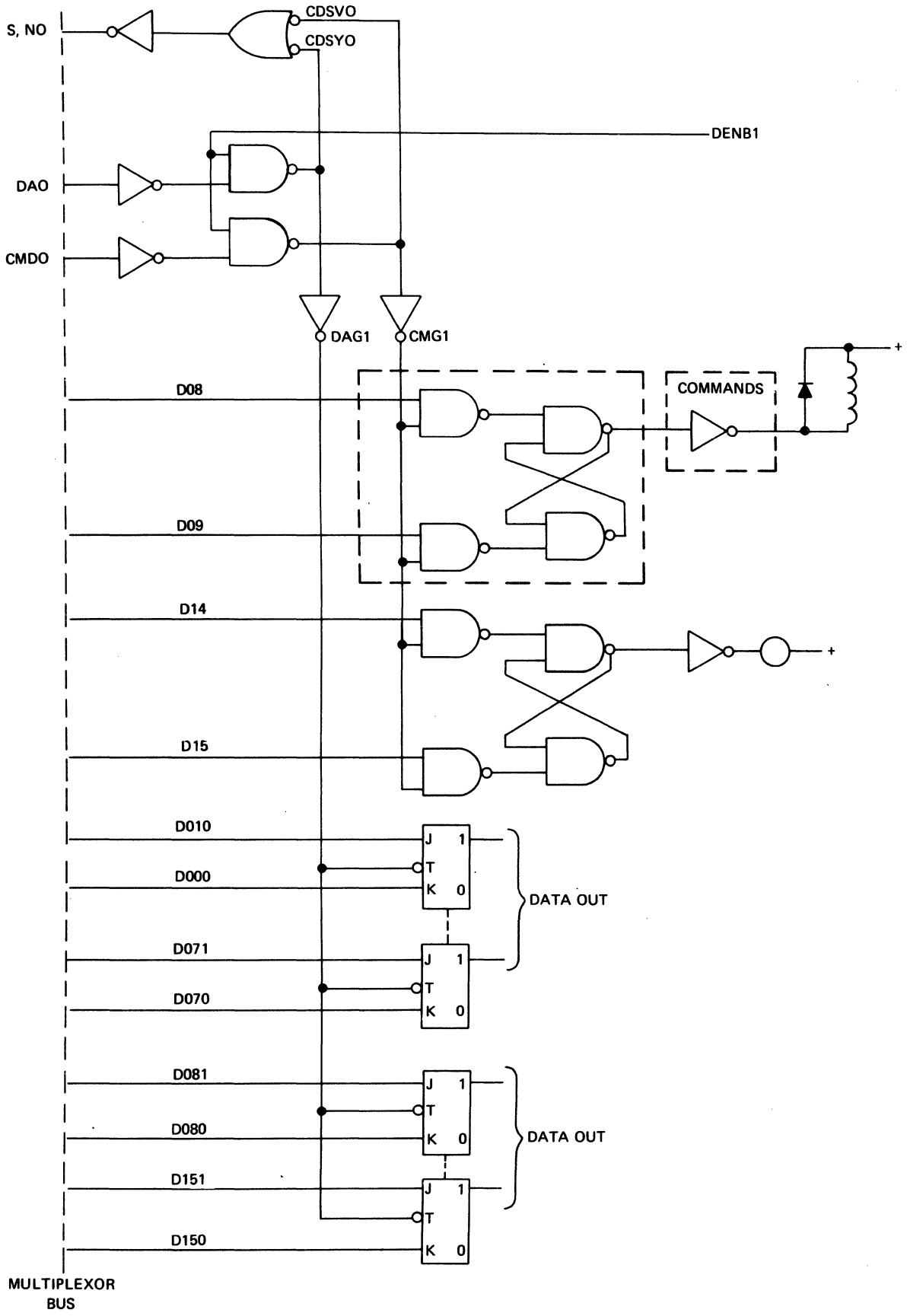


Figure 5-32. Data and Command Output, Logic Diagram

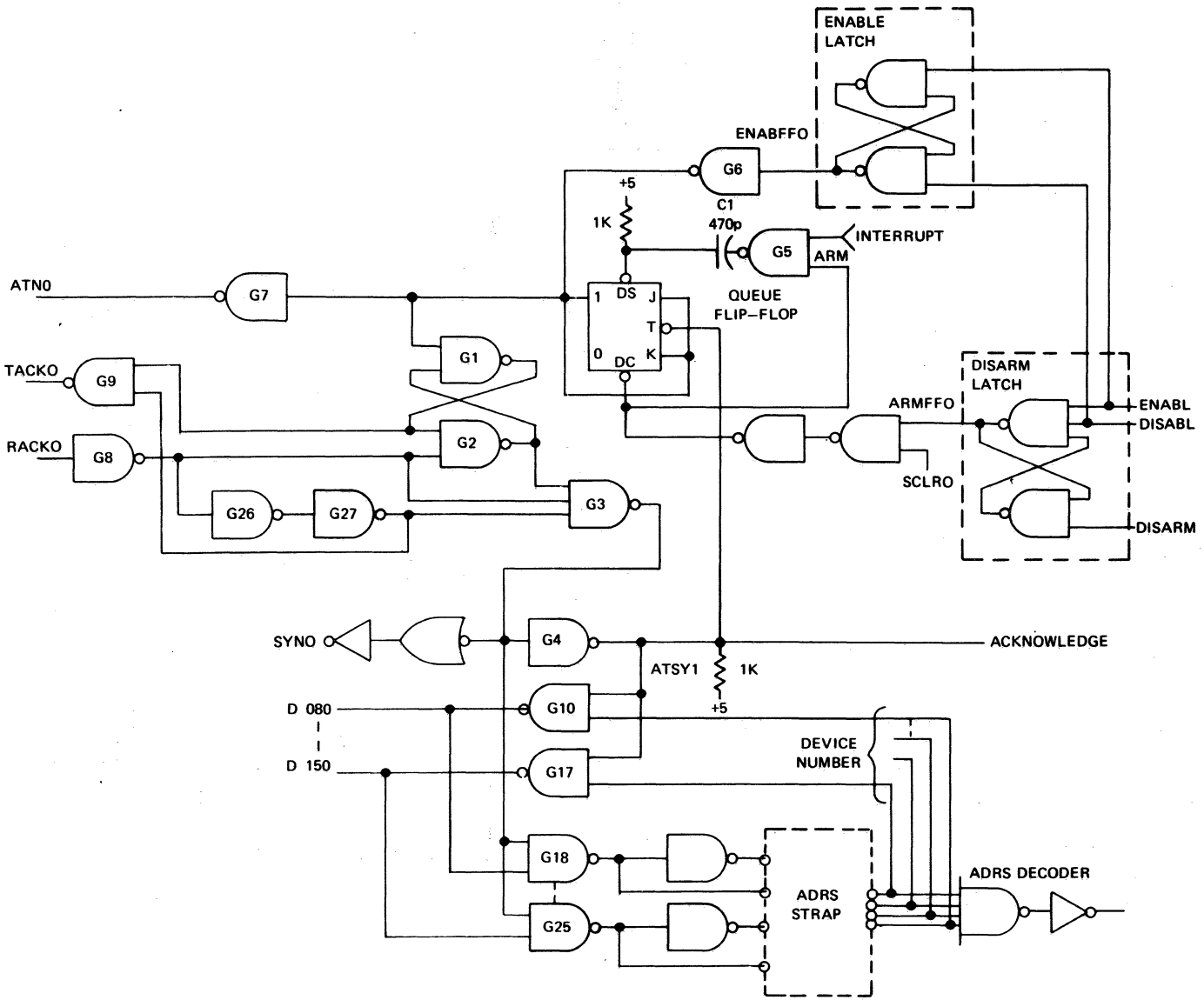


Figure 5-33. Interrupt Control, Logic Diagram

This causes the device number wired in by the address strap board to appear on the inputs of Gates G10 through G17. Thus, the ATSY1 output from Gate G4 enables the device number onto the Data Lines (D080 through D150).

The output from Gate G4 also raises the Acknowledge signal to the device. On receiving the SYN0, the Processor lowers RACK1, causing the output of Gate G4 to drop. This in turn causes the Queue flip-flop to reset.

> NOTE
>
> If the interrupt has not set the Queue flip-flop, the RACK1 signal passes through Gate G2 to TACK0, and on to the next device.

If RACK1 is high in response to another device, the output from Gate G2 is low, thus disabling the interrupt from affecting Gate G1. However, the interrupt remains in the Queue flip-flop, and is serviced after completion of the previous interrupt service.

The ENABFF0 and ARMFF0 lines provide control over the Interrupt Queue flip-flop and the ATN0 line to the Processor. Normally, two bits of a Command Byte (Bits 0 and 1) are decoded such that, with Bit 0 true and Bit 1 false, the Queue flip-flop is disabled. That is, the flip-flop may be set, however, its output is held low. Gate G6, whose input is ENABFF0 from the false side of the ENABLE latch, provides this function. The Command Byte, with Bit 0 false and Bit 1 true, is decoded (ENABL goes false) and sets the ENABLE latch which allows new interrupts or a queued interrupt to be recognized. Bits 0 and 1, both true, are decoded to drive DISARM false which sets the DISARM latch. The false side of the latch is used to clear the Queue flip-flop and to prevent the interrupt line from setting it. The DISARM latch is cleared whenever the ENABLE or DISABLE commands are recognized. Encoded commands ENABLE/DISABLE/DISARM thus provide interrupt masking or inhibiting within the device controller.

As described previously, the Control Line, CL050, from the Processor carries the Interrupt Acknowledge (ACK) signal. This line breaks up into a series of short lines to form the "daisy-chain" priority system. The ACK signal must pass through every controller that is equipped with Interrupt Control circuits. This includes all device controllers except a few special cases.

Back panel wiring for interrupt control is shown in Figure 5-34. At a given position, the Received ACK (RACK0) appears at Pin 122-1 and the Transmitted ACK (TACK0) at Pin 222-1. The daisy-chain bus is formed by a series of isolated lines which connect Terminal 222-1 of a given position to Terminal 122-1 of the next position (lower priority). On unequipped positions, a jumper shorts 122-1 and 222-1 of the same connector to complete the bus. Back panels are wired with jumpers on all positions. Whenever a card chassis position is equipped with a controller, the jumper from 122-1 to 222-1 must be removed from the back panel at that position.

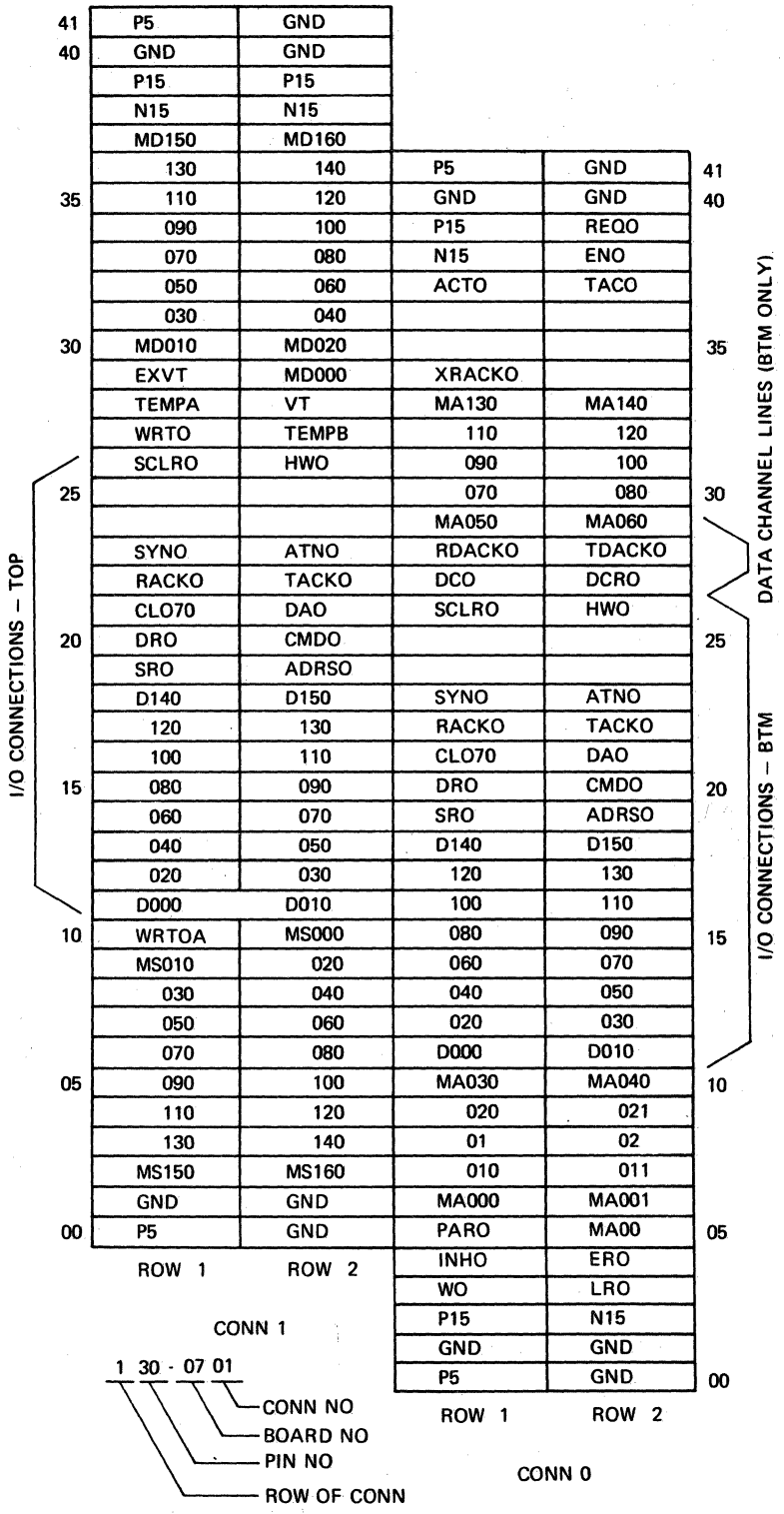
For controllers that occupy several positions, the jumper is removed only at the position where the controller board has ATN/ACK circuits.

5.4.6 Multiplexor Bus Wiring

Wiring for the Multiplexor Bus and for the Selector Channel Bus is identical for 15-inch 3010/2 chassis. Each card position contains two connectors with the Mux Bus wired to each at pin position indicated in the Figure 5-34.

5.4.7 Multiplexor Channel Timing

Both the Input and Output operations on the Multiplexor Channel make use of request/response signaling. This allows the system to run at its maximum speed whenever possible, but permits a graceful slowdown if the characteristics of a particular device controller requires signals of longer duration. Device controller designs should keep Multiplexor Channel usage as fast as possible, consistent with practical circuit margins. Doing this assures the fastest computer input/output operation when a system is configured with a number of peripheral devices.



	ROW 1	ROW 2	ROW 1	ROW 2	
41	P5	GND	P5	GND	41
40	GND	GND	GND	GND	40
	P15	P15	P15	REQO	
	N15	N15	N15	ENO	
	MD150	MD160	ACTO	TACO	
	130	140			
35	110	120			35
	090	100	XRACKO		
	070	080	MA130	MA140	
	050	060	110	120	
	030	040	090	100	
30	MD010	MD020	070	080	30
	EXVT	MD000	MA050	MA060	
	TEMPA	VT	RDACKO	TDACKO	
	WRTO	TEMPB	DCO	DCRO	
	SCLRO	HWO	SCLRO	HWO	
25					25
	SYNO	ATNO	SYNO	ATNO	
	RACKO	TACKO	RACKO	TACKO	
	CLO70	DAO	CLO70	DAO	
20	DRO	CMDO	DRO	CMDO	20
	SRO	ADRSO	SRO	ADRSO	
	D140	D150	D140	D150	
	120	130	120	130	
	100	110	100	110	
15	080	090	080	090	15
	060	070	060	070	
	040	050	040	050	
	020	030	020	030	
	D000	D010	D000	D010	
10	WRTOA	MS000	MA030	MA040	10
	MS010	020	020	021	
	030	040	01	02	
	050	060	010	011	
	070	080	MA000	MA001	
05	090	100	PARO	MA00	05
	110	120	INHO	ERO	
	130	140	WO	LRO	
	MS150	MS160	P15	N15	
	GND	GND	GND	GND	
00	P5	GND	P5	GND	00

Figure 5-34. Typical Universal Expansion Slot Wiring

Timing for typical Input/Output operations are shown on Figure 5-35. On the Output operation, the Processor places a signal on the data lines followed by an appropriate control line signal. This stagger (T1) will vary, but it is guaranteed to be at least 100 nanoseconds. When the device controller has received the Output Byte, the SYN signal is returned to the Processor, which terminates the control line signal. Realizing that T5 is 100 nanoseconds minimum, the SYN delay T2 should be only long enough to guarantee proper reception of the Output Byte. The control line/data line removal time (T3) is important where single-rail to double-rail operation is used - e.g. the ADRS flip-flop on Figure 5-30. A minimum of 100 nanoseconds is guaranteed for T3. For SYN generation as per Figure 5-30 and 5-33, the control line signal is DC coupled through the gates to form the SYN signal. The SYN removal time (T4) should be minimized. This delay should not be unnecessarily extended since the Processor will not begin another Input/Output operation until SYN is removed.

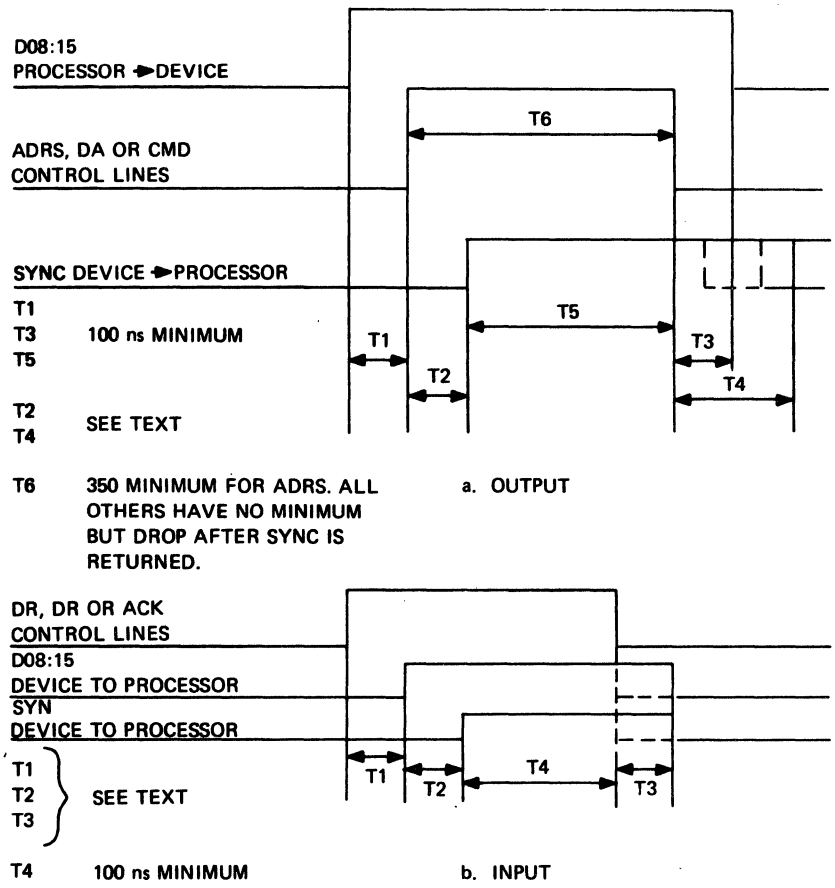


Figure 5-35. Multiplexor Channel Timing

Device controllers must be designed to accept a minimum control pulse width (particularly ADRS) of 350 nanoseconds.

It should be emphasized that the times shown on Figure 5-35 are defined for signals on the Multiplexor Channel. Within a given controller, one signal may flow through more gates than another signal and these delays must be considered.

For the Input operation, the Processor places a signal on a control line. The currently addressed device controller should gate signals to the data lines as soon as possible to keep T1 at a minimum. The SYN delay (T2) must guarantee that the Input Byte is on the data lines considering the slowest data gates and the fastest SYN gates. The Processor will remove the control line signal when SYN is received with a

minimum delay (T4) of 100 nanoseconds. With SYN and the byte gate DC coupled to the control line, the removal delay (T3) will be the sum of the corresponding gate delays. The Processor considers the operation complete when SYN falls.

When the control signal is ACK, the delay T1 will include the cumulative Gate G8/G9/G26/G27 delays (See Figure 5-33) for all the controllers between the responding controller and the Processor. This will be less than the Processor time-out even with the maximum limit of 255 controllers.

> NOTE
>
> It is essential to realize that after the Processor initiates a control line signal, the Processor does nothing until the SYN signal is returned by the device controller; one or more cycles are skipped if necessary and the data transfer rates decreased proportionally. While this may not affect a particular controller, the overall system performance is degraded. Furthermore, if a device controller fails to respond with a SYN in the time out period of about 35 microseconds, the Processor will abort the instruction.

5.4.8 Typical Device Controller Interface

Figure 5-36 illustrates a typical interface to the Multiplexor Bus which might be used in the design of custom device controllers, either 8-bit byte or 16-bit halfword oriented. (If an 8-bit byte oriented interface is being designed, Gates G1 through G8, G17 through G24, G78 through G85, and Gate G45 are not used.) The address straps to Gate G57 can be hardwired by the user for any device number from 03 to 255. Address 01 and 02 are hardwired for the Control Console and Teletypewriter, respectively. The user can use the Gated Status Request (SRG0) or the Gated Data Request (DRG0) control lines to gate status or data from appropriate points in his logic, to the Processor, by OR typing onto the Send Data points SD 000:150. Data from the Processor is available to the user's circuits, double rail, at the points labeled D001:151 and D000:150. The user can use the Gated Data Available (DAG0) and the Gated Command (CMG0) control lines to gate the data from the Processor to appropriate points in his logic. The delay of the SYN0 signal should be arranged such that is is the minimum delay necessary for the custom controllers to function properly, per Section 5.4.7.

5.4.9 Data Channel Interface Design

The Data Channel feature of the GE-PAC 3010/2 Processor provides high speed, autonomous memory access to customer designed device controllers. Halfword data transfer to memory is accomplished over the Multiplexor Bus at a maximum data rate of 440K bytes per second in the Burst Mode.

Interfaces to the Data Channel must follow the same general design rules which apply to regular interfaces to the Multiplexor Bus.

The program initiates a Data Channel operation, usually by issuing an Output Command instruction. Then the Data Channel device completes the transfer without further direction by the Processor.

When a memory access is desired, the controller activates the Data Channel Request Line (DC). This line is separate from and of higher priority than the normal I/O attention line (ATN).

The Data Channel Request is honored on an instruction stealing basis. If the Processor is halted (programmed Wait State), the latency time is 1.5 to 1.75 microseconds. If the Processor is executing instructions, latency time depends upon the instruction being performed, and when during instruction execution, the DC line became active. (This latency time is typically less than 4.0 microseconds and it may be as high as 22 microseconds for worst case Load Multiple and Store Multiple. Read Block, Write Block, and Auto Load can result in a latency time proportional to the amount of data transferred.)

The Processor acknowledges the DC line by addressing device number zero and also activating the DACK control line. (Addressing device number zero resets the Address flip-flop in all devices.)

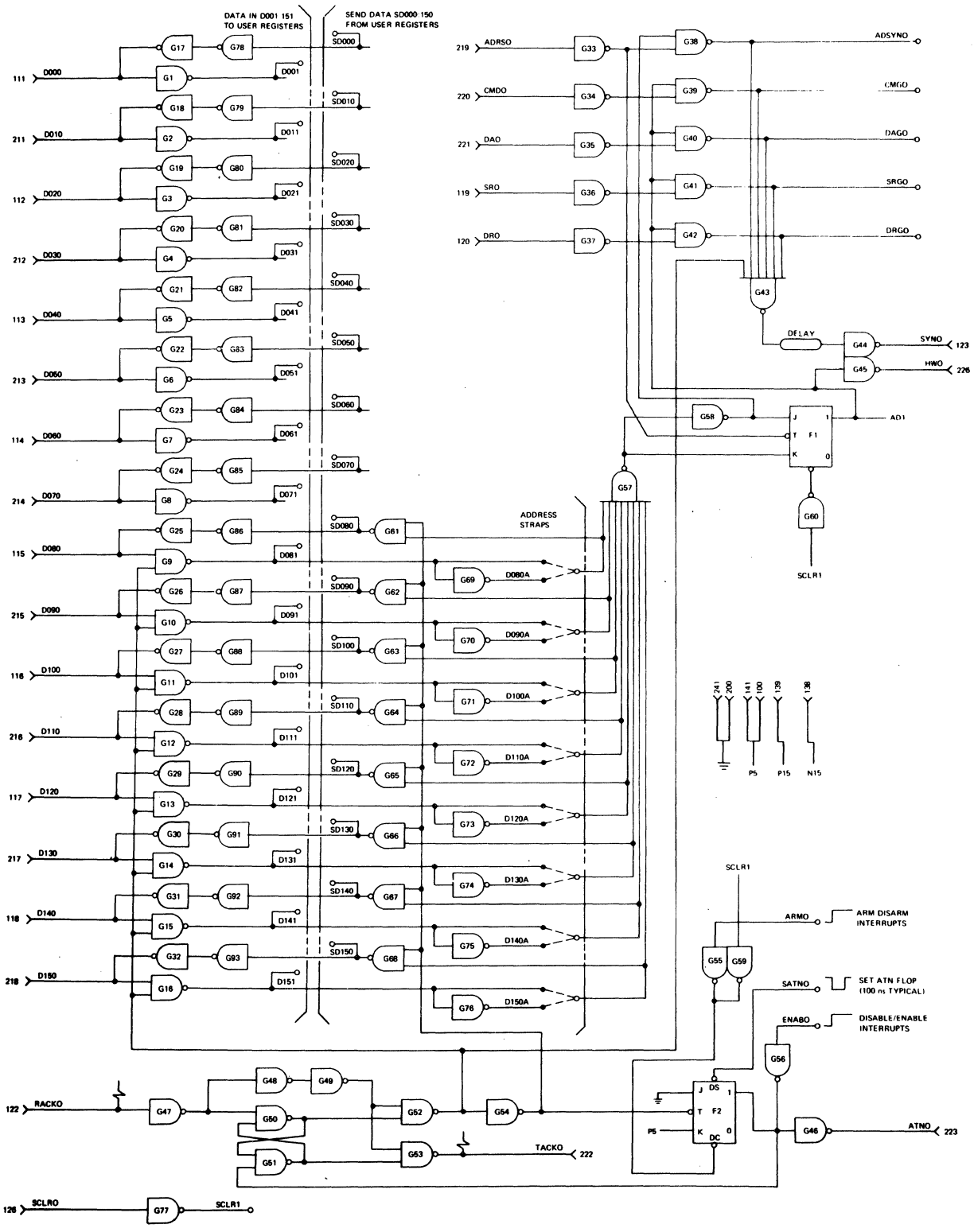


Figure 5-36. General Multiplexor Bus Interface

The Data Channel Acknowledge line (DACK) is "daisy chained" through all device controllers on the Data Channel. The device controller closest to the Processor that originated the Data Channel Interrupt captures the DACK signal. That device becomes the 'on line' device and controls the DCR line. The DCR line tells the micro-program whether a memory Read (DCR=0) or a memory Write (DCR=1) operation is to be performed.

If a memory Read is requested, the micro-program reads the 16-bit memory address from the controller. The selected location is read and the 16-bit data is output to the controller.

If a memory Write is requested, the micro-program reads the 16-bit memory address from the controller, then reads the 16-bit data word. The data word is stored in the selected core location.

If another memory cycle is desired, the device controller re-activates the DC line.

Figure 5-37 shows a block diagram of a typical Data Channel device controller. Figure 5-38 illustrates the timing for a typical Data Channel Read and Write operation.

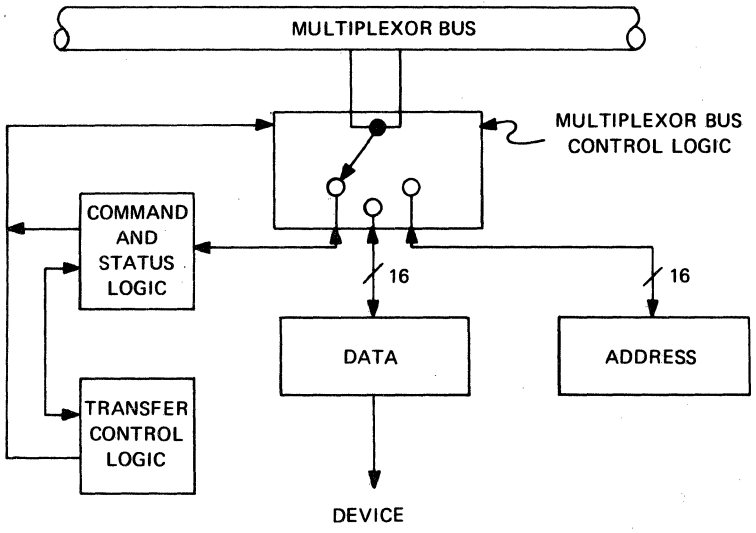


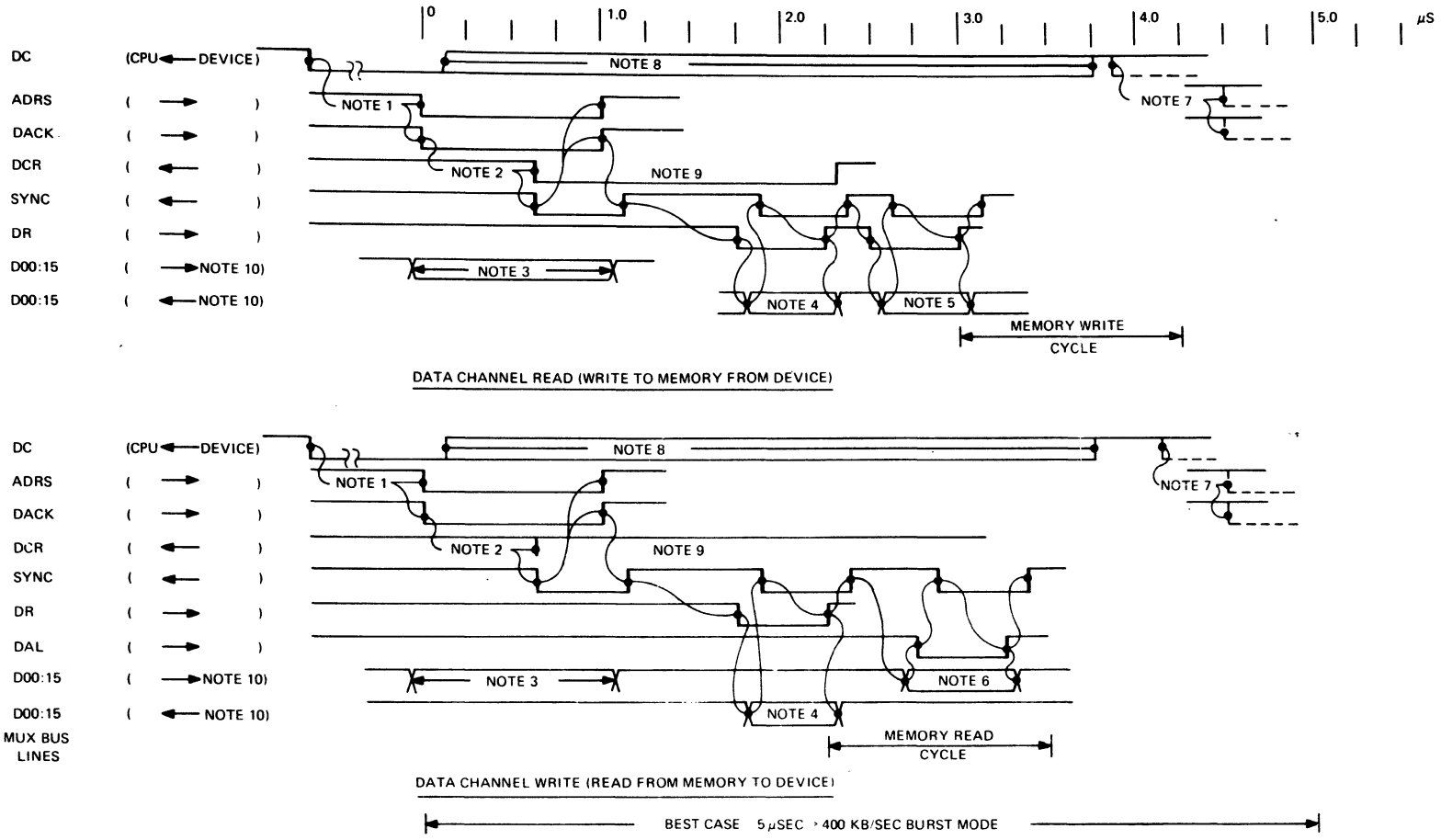
Figure 5-37. Interleaved Data Channel, Block Diagram

Figure 5-39 illustrates a typical interface to the Multiplexor Bus which may be used when designing custom interfaces to the Interleaved Data Channel. Note that the logic circuits in Figure 5-39 are nearly identical to those in Figure 5-37 (Standard Multiplexor Bus Interface). A second daisy chain circuit has been added (Gates G94 through G101) to the data channel interface. This daisy chain captures the RDACK0/TDACK0 pulse, and forces the interface Address flip-flop (F1) set.

The user's circuit must provide a Set Data Channel (SDC0) request pulse to initiate a Data Channel (DC) operation. At the same time the user provides a RD1 level (high for read, low for write) to indicate the type of operation. The Select flip-flop (F5) is set when this interface captures the data channel cycle, and remains set until the cycle is complete.

Note from Figure 5-34 that the Interleaved Data Channel control line DCO, DCRO, and the daisy chain priority line RDACK0/TDACK0 appear only on the 0 level connector on the back panel. Therefore Data Channel devices must connect into the 0 level connectors. Note also that these lines are not reconstituted by the Bus Buffer (Figures 5-29).

Note on Figure 5-39 that when a Data Channel device is selected (i.e., SEL flip-flop set) it must not hold the HW line active even though all data transmitted over the Multiplexor Bus is 16-bit parallel.



NOTES:

1. DC CAN BE ACTIVATED AT ANY TIME. TYPICAL LATENCY TIME IS 4 µS EXCEPT FOR BLOCK I/O, LOAD/STORE MULTIPLE OR AUTOMATIC I/O CHANNEL.
2. SYNC SHOULD BE DELAYED LONG ENOUGH TO ASSURE THAT THE ADDRESS FLOPS OF ALL CONTROLLERS ARE RESET, 600 ns TYPICAL. SEE NOTE 3.
3. D00:15 ZEROS THIS INTERVAL (A NULL ADDRESS).
4. THE DEVICE SENDS A 16-BIT MEMORY ADDRESS
5. THE DEVICE SENDS A 16-BIT DATA WORD WHICH IS WRITTEN TO MEMORY.
6. THE PROCESSOR OUTPUTS A 16-BIT READOUT FROM MEMORY.
7. THE NE/T DATA CHANNEL CYCLE BEGINS IN BURST MODE.
8. DC MAY BE RELEASED ANY TIME IN THIS INTERVAL.
9. DCR LOW › WRITE TO MEMORY, HIGH › READ FROM MEMORY.
10. D00:15 IS A COMMON BI-DIRECTIONAL BUS SHOWN TWICE FOR CLARITY.

Figure 5-38. Data Channel Timing Chart

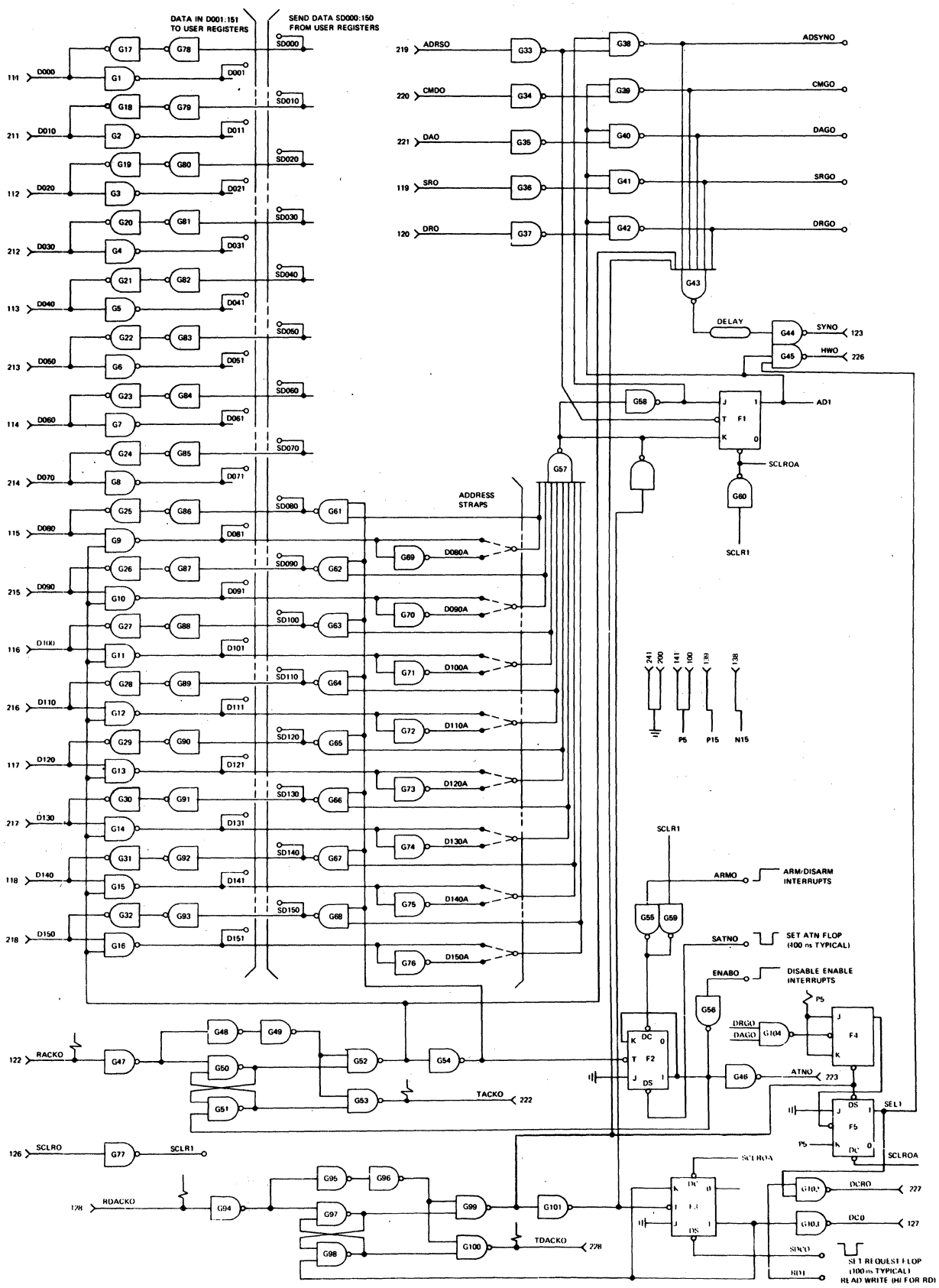


Figure 5-39. Data Channel/Multiplexor Bus Interface

5.5 AUTOMATIC I/O

The GE-PAC 3010/2 Automatic I/O feature allows external I/O interrupts to be serviced and data exchanged between the Central Processor and I/O controllers and devices. Automatic I/O allows interrupt driven data exchanges to take place with a minimal effect on the running program. Once the appropriate Automatic I/O pointers and table are in core, the hardware and the micro-program take care of the Automatic I/O operations, and each I/O interrupt merely delays program execution, rather than interrupting the program.

Bits 1 and 4 of the current PSW control Automatic I/O operations. Both of these bits must be set to enable the external I/O interrupts and permit Automatic I/O. Automatic I/O depends upon a properly configured Automatic I/O Service Pointer Table, and an Interrupt Service Block with an appropriate Function Word. The Automatic I/O firmware may generate an interrupt itself, because of abnormal conditions, or because of the occurrence of an event for which the program had requested an interrupt.

5.5.1 Automatic I/O Service Pointer Table

The Automatic I/O Service Table starts at location X'00D0' (Figure 5-40). It contains a halfword entry for each of the 256 possible peripheral device addresses. If Bit 15 of the entry in this table is reset, then the entry is the address of an Immediate Interrupt PSW exchange location. If Bit 15 of the entry is set, then the entry minus one is the address of a Channel Command Word.

5.5.2 Interrupt Service Block

The Interrupt Service Block contains the Function Word plus the storage locations and data required by the operation. The Function Word is a bit encoded command that completely describes the Automatic I/O Operation. Note that it is the address of the Function Word plus one that is entered in the Automatic I/O Service Table. A complete Interrupt Service Block is shown in Figure 5-41.

5.5.3 Automatic I/O Termination Queue

The Automatic I/O Termination Queue is a circular list identical to those described under "list processing instructions". The queue may be set up at any convenient core location. The maximum size of the queue allows for 255 entries, but any convenient length may be used. The address of the queue must be stored at location X'0080' prior to starting any channel program. Automatic I/O uses the queue to indicate termination of an automatic operation.

5.5.4 General Operation

When the Processor detects the presence of an interrupt signal from a peripheral device, it automatically acknowledges the signal and obtains the address of the device. It uses the device address times two to index into the Automatic I/O Service Table to the entry reserved for the device. If Bit 15 of the entry is reset, the Processor takes an Immediate Interrupt. If Bit 15 is set, the Processor activates the Automatic I/O. Automatic I/O uses the entry minus one to locate the Function Word. It decodes the Function Word and performs the required service, using the data entries in the Interrupt Service Block as necessary. If the operation for this device is not yet complete, the channel returns control to the Processor. The Processor now checks for pending interrupt signals. If none are present, it continues program execution. If any are present, it services them before returning to program execution.

If the channel determines that the operation for this device is complete, it terminates the Automatic I/O by storing the device address and final status in the CCB, and, for data transfers, changes the Function Word to a "no operation". This causes subsequent interrupt signals from the device coming to this Function Word to be ignored. At this point, the channel can take any or all of the following actions:

1. Make an entry in the Termination Queue.
2. Chain to another Function Word.
3. Generate an Immediate Interrupt.

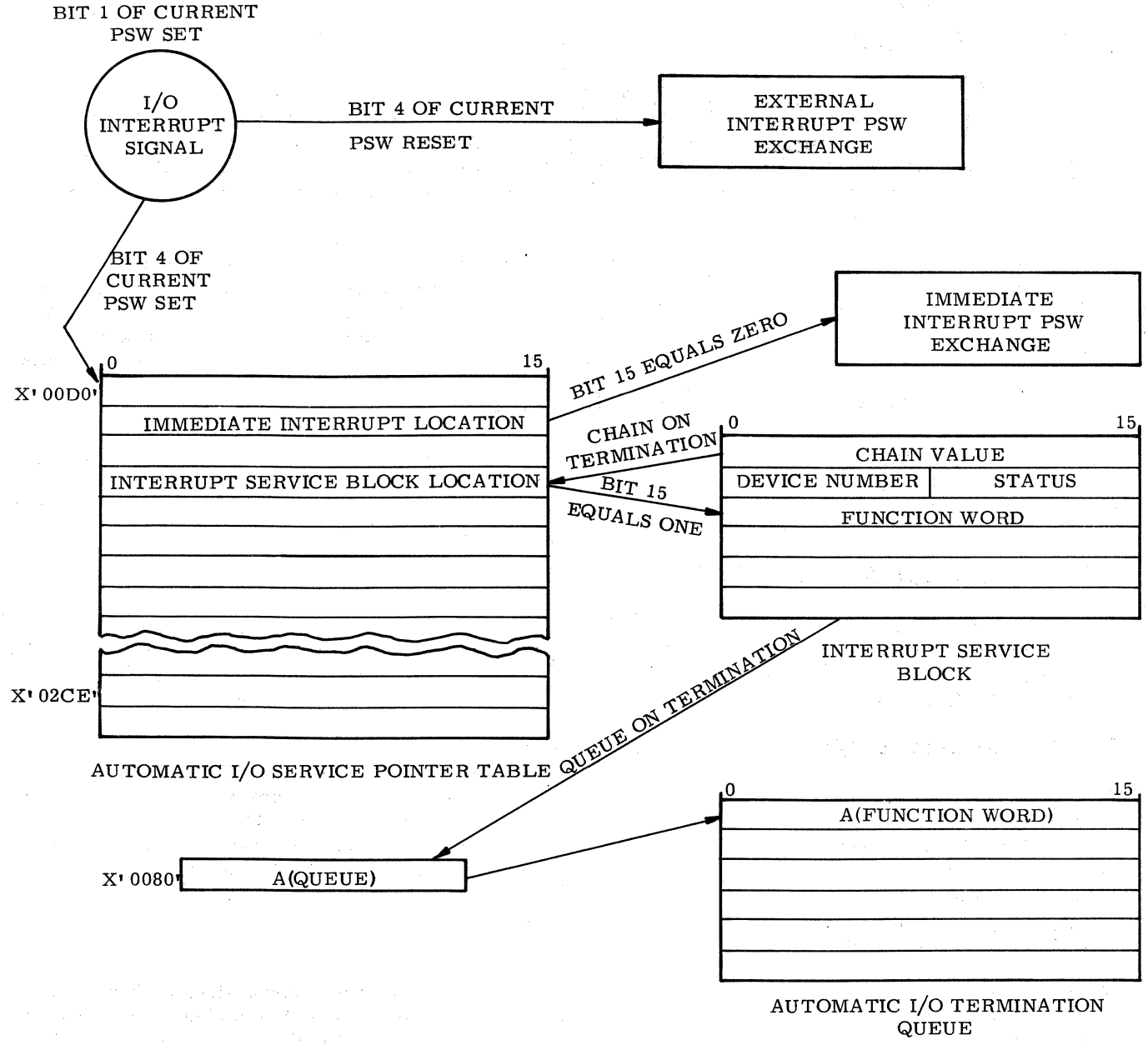


Figure 5-40. Automatic I/O Sequence

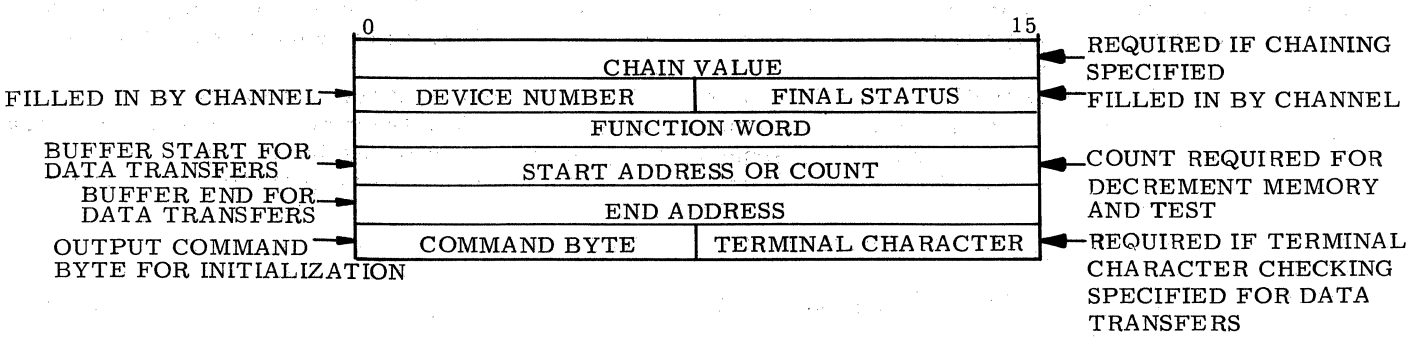


Figure 5-41. Interrupt Service Block

The action taken by the channel depends on the bit configuration of the Function Word.

In the queuing operation, the channel generates a Queue Overflow Interrupt if the queue is full when it attempts to make an entry.

5.5.5 Function Words

There are three phases involved in Automatic I/O operations. These are:

1. Initialization
2. I/O Operation
3. Termination

All three phases are controlled by the bit configuration of the Function Word. A single Function Word can be encoded to perform all three types of operation. The bit assignments for Function Words are shown in Figure 5-42.

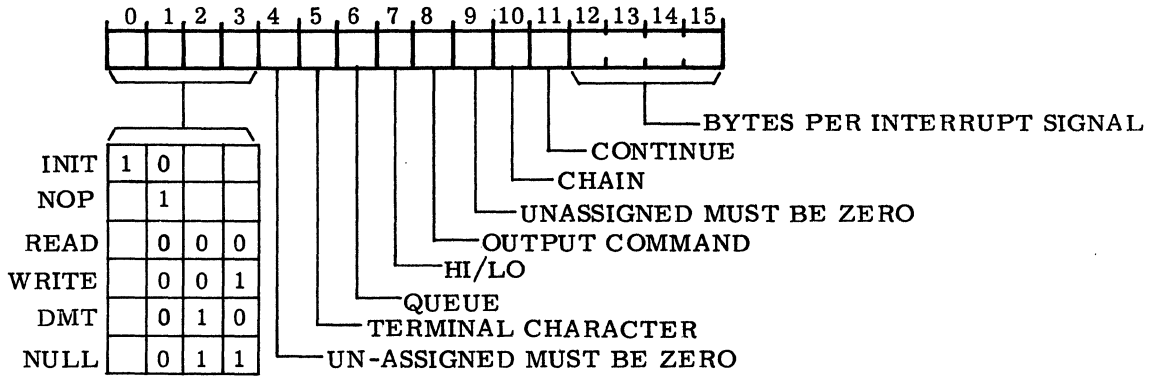
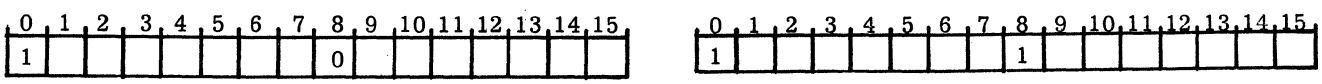


Figure 5-42. Bit Configuration For Function Word

5.5.6 Initialization

Bits 0 (INIT) and 8 (Output Command) of the Function Word control the initialize phase of Automatic I/O operations. If Bit 0 (INIT) is set when the channel decoded the command word, it resets Bit 0 (INIT) and checks Bit 8 (Output Command). If Bit 8 is set, the channel issues the output command located at the start of the Interrupt Service Block plus ten and returns control to the Processor. Automatic I/O operations with the device resume when an interrupt signal occurs from the device. Since the channel resets bit zero, it can pass through the initialize phase only once. This phase is optional. The software may initialize the device by Output Command instructions prior to starting the Automatic I/O operation. The bit configurations of the Function Word for the Initialization phase are illustrated in Figure 5-43.



FUNCTION WORD FOR INITIALIZE WITHOUT OUTPUT COMMAND

FUNCTION WORD FOR INITIALIZE AND OUTPUT COMMAND

Figure 5-43. Function for Initialize and Output Command

5.5.7 I/O Operation

There are five distinct types of I/O operations the Automatic I/O Channel can perform. These are:

1. Read
2. Write
3. Decrement Memory and Test
4. No Operation
5. Null

The Function Word configurations for these operations are illustrated in Figure 5-44.

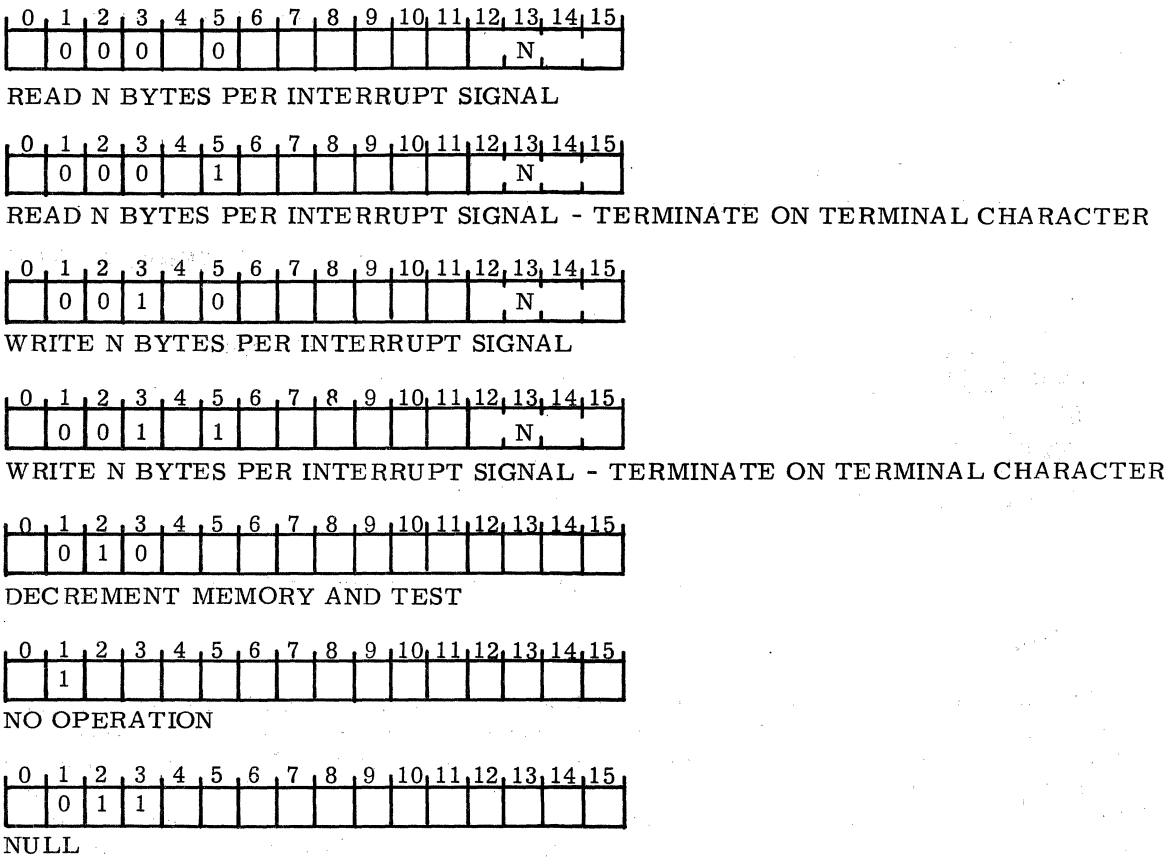


Figure 5-44. Function Words for I/O Operation

For all Read/Write operations, Bits 12 through 15 must contain the number of bytes to be transferred on each interrupt signal. All zeroes in these bit positions indicates that sixteen bytes are to be transferred on each interrupt signal. The two halfwords following the Function Word in the Interrupt Service Block must contain the starting address of the I/O Buffer and the ending address of the I/O Buffer. After the number of bytes specified for each interrupt signal has been transferred, the starting address is incremented by the appropriate amount and compared to the ending address. If it is greater, the channel enters the termination phase. If it is not greater, the channel returns control to the Processor for program execution. Bit 5 of the Function Word controls the optional terminal character data transfer. When this bit is set, the transfer proceeds as described above with the exception that the last byte

transferred on each interrupt signal is compared with the terminal character byte located at Interrupt Service Block plus eleven. If these two bytes match, the channel enters the termination phase. In this way, an Automatic I/O can terminate because the buffer is exhausted or a terminal character has been found in the data stream.

Before starting a data transfer, the Automatic I/O Channel checks the device status. Any non-zero status condition will stop the transfer and cause the channel to enter the termination phase. Before entering the termination phase, the Initial (INIT) Bit and No Operation Bit are set in the Function Word, the Queue bit is set to force an entry in the Termination Queue, and the Chain bit and Continue bit are reset to prevent chaining.

The Decrement Memory and Test Operation causes the value contained in the halfward immediately following the Function Word to be decremented by one for each interrupt signal. The new value is compared to zero. If greater than zero, the channel returns control to the Processor for program execution. If equal to zero, the channel enters the termination phase without changing the Function Word to a "no operation". Subsequent interrupt signals from the device will cause the count field to increase negatively. The No Operation code in the Function Word indicates that the channel is to ignore any interrupt signal from the associated device. The channel itself sets this code in the Function Word on completion of data transfers. The software can use this code to ignore unsolicited interrupt signals. The Automatic I/O Service Pointer Table should contain pointers to "no operation" control words for all nonexistent devices.

The Null Operation differs from the No Operation differs from the No Operation in that while no I/O function is performed, the channel enters the termination phase without setting the No Operation code.

5.5.8 Termination

The Automatic I/O Channel enters the termination phase upon completion of a data transfer, when the count field of a Decrement Memory and Test operation has reached zero, or when the Null Operation is decoded. All of the operations in the termination phase are optional. If none are specified, the channel returns control to the Processor. The two termination functions are Queue and Chain. The Function Word bit configuration for Queuing and chaining is shown in Figure 5-45. Bit 6 of the Function Word controls queuing. If this bit is set, the channel, on entering the termination phase, stores the address of the Function Word in the Termination Queue. The condition of Bit 7 of the Function Word controls positioning with the queue. If Bit 7 is set, the entry is made at the bottom of the queue. If Bit 7 is reset, the entry is made at the top of the queue.

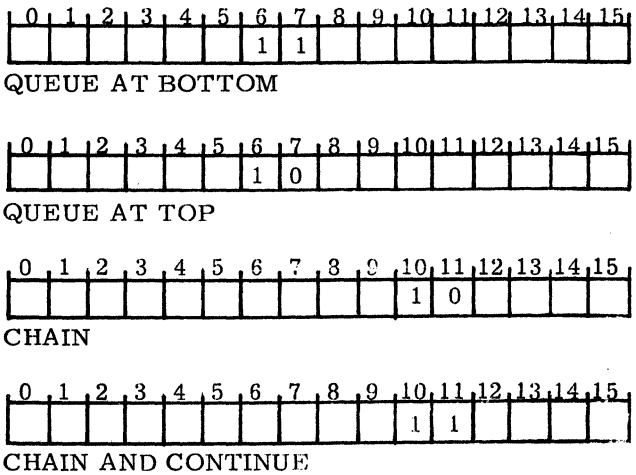


Figure 5-45. Function Words for Termination

Bit 10 of the Function Word controls chaining. In this operation, the channel stores the first halfword of the Interrupt Service Block in the appropriate location in the Automatic I/O Service Table for this device. This chain value may be either the address of another Function Word or the address of a PSW exchange location for the Immediate Interrupt. Subsequent interrupt signals will be handled as indicated by this value. If the chain bit and the Continue bit (Bit 11) are both set, the channel checks the new value placed in the Service Pointer Table and takes appropriate action before returning control to the Processor. In this way, depending on the new value stored in the Service Pointer Table, the channel can either generate an Immediate Interrupt or start another Automatic I/O Operation.

5.5.9 Example of Automatic I/O Programming

This example of Automatic I/O Programming assumes a teletypewriter located at physical address X'02'. The program is set up to:

1. Issue an Output Command to start the device.

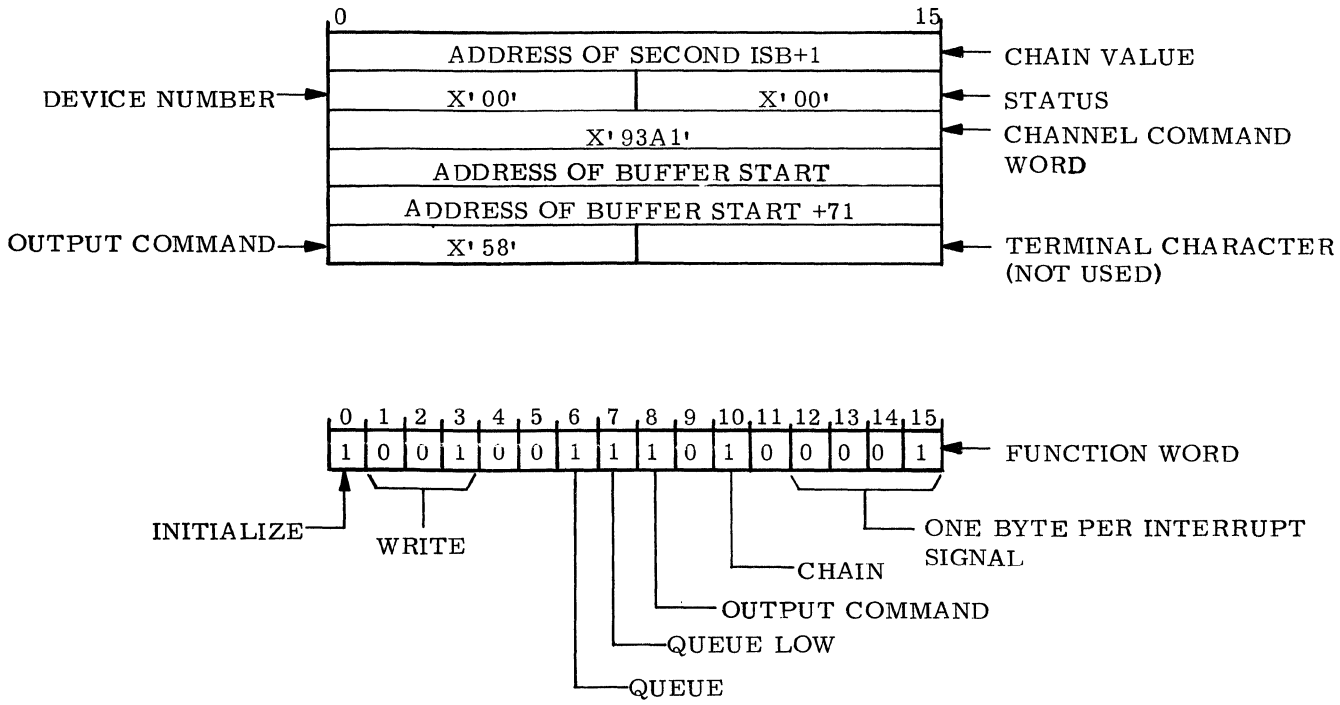
2. Write 72 bytes from core memory to the device, one byte per interrupt signal.

3. On completion of the transfer, make an entry at the bottom of the Termination Queue and chain to a second Interrupt Service Block without specifying Continue.

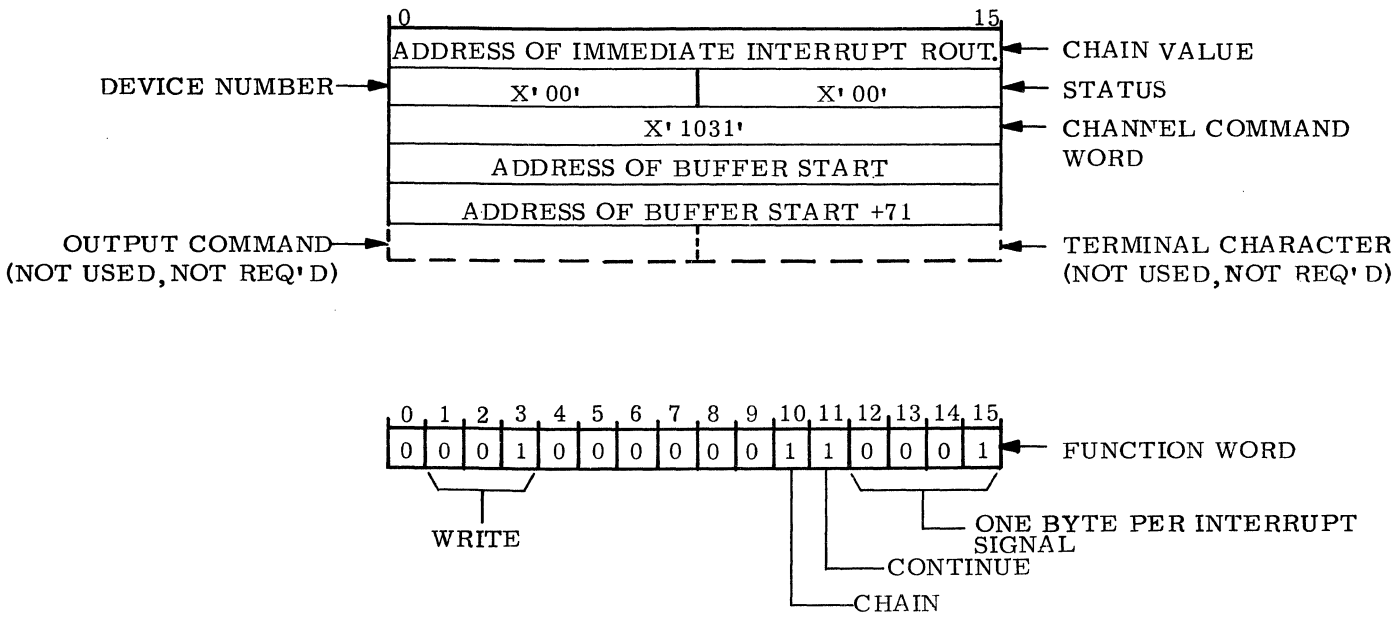
4. The second Interrupt Service Block writes an additional 72 bytes to the device and terminates by chaining to an Immediate Interrupt and causing the interrupt to occur.

The first Interrupt Service Block is shown in Figure 5-46A. The Chain value is set to point to the second Interrupt Service Block. The Status Byte and Device Number are set to zero. The Function Word is set for Initialize, Write, Queue, Queue Low, Output Command, Chain, and transfer one byte per interrupt signal. The next two halfwords point to the beginning and end of the 72 byte buffer. The Output Command byte is set to enable and write. The user program stores the address of this Interrupt Service Block in location X'00D4', the Service Pointer Table entry for device X'02'. It issues a Simulate Interrupt instruction specifying device X'02' to get the operation started. On execution of the Simulate Interrupt instruction, the channel issues the Output Command and resets the Initialize bit. It gives control to the Processor for the execution of normal instructions. As each subsequent interrupt signal is received from the device, the channel outputs one byte until it has output the entire buffer of 72 characters. In between each interrupt signal it returns control to the Processor. After the last byte has been transferred, it sets the No Operation Bit in the Function Word, and puts the address of the Function Word at the bottom of the Channel Termination Queue located at the address specified by the contents of X'0080' (Termination Queue Pointer). It stores the chain value (the address of the second Function Word) in location X'00D4', Service Pointer Table entry for device X'02'.

On the next and on each subsequent interrupt signal, the channel is directed to the second Interrupt Service Block. This block is illustrated in Figure 5-46B. The Chain value points to an Immediate Interrupt location. The status and Device Number are set to zero. The Function Word specifies Write, Chain, and Continue. The channel outputs the data as described above until the last byte is written. It then sets the No Operation bit, stores the Immediate Interrupt address in the Service Pointer Table location for device X'02', and generates an interrupt allowing software to take over servicing this device. If, during the data transfers, the channel had received an unsatisfactory status from the device, it would have terminated the operation by setting the Initialize and No Operation bits (bad status indicators) in the Function Word, suppressed Chaining, and forced an entry at the top of the I/O Termination Queue. Setting the Continue bit in the second Function Word causes the Channel to generate the Immediate Interrupt on the same interrupt signal that causes output of the last data byte. If this bit were reset, the next interrupt signal from the device would generate the Immediate Interrupt.



(A) First Interrupt Service Block



(B) Second Interrupt Service Block

Figure 5-46. First/Second Interrupt Service Blocks

CHAPTER 6
CORE MEMORY

6.1 INTRODUCTION

The Core Memory, Figure 6-1, features highly reliable 4,096 by 16-bit halfword (8,192 byte), 1.0 microsecond memory modules. The system is modularly expandable to 65,536 bytes of storage (eight modules).

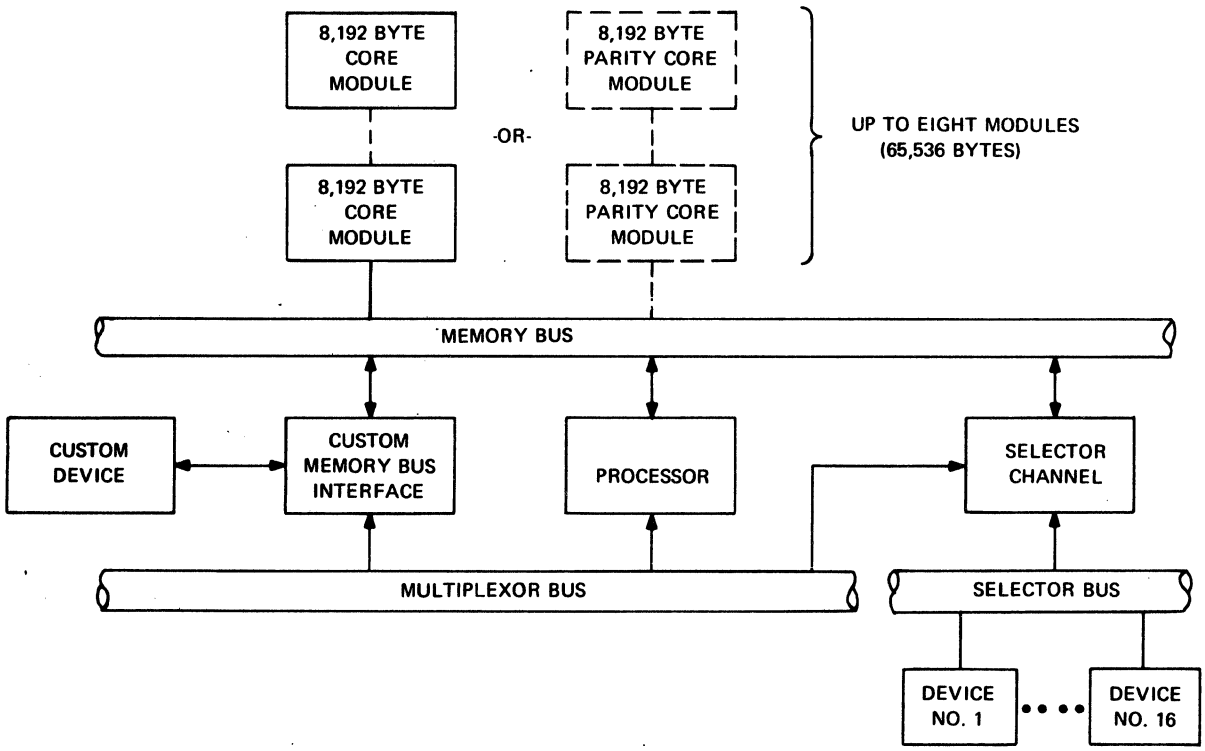


Figure 6-1. Memory System

Memory modules use field proven, 3D, three-wire technology. The memory core plane uses 23 mil ferrite cores, and is mounted complete with access and readout circuits, on a single GE-PAC 3010/2 inch circuit card. The memory modules are arranged as 4,096 by 16-bit (or 17-bit for parity memory) modules. Data is transferred to/from memory on a 16-bit halfword basis. The Processor can address memory to the byte level.

Up to four Direct Memory Access Channels (DMACs) can be added to the Core Memory. The DMACs operate over the common Memory Bus, on a cycle stealing basis, through a Direct Memory Access Port which is built into the Memory System. Data rates through a DMAC of up to 2,000,000 bytes per second can be achieved.

Two types of Direct Memory Access Channels can be used with the GE-PAC 3010/2. The Selector Channel (SELCH) permits direct data transfer between any standard 3010/2 device controller and memory. Up to 16 controllers can be connected to a single SELCH. The program sets up the SELCH by sending it a starting address, an ending address, and a GO command. The SELCH then proceeds with the data

transfer direct to memory, on a cycle stealing basis, without further direction by the program. The SELCH interrupts the Processor when the transfer terminates. A more detailed description of the SELCH is provided in Chapter 5.

The second type of Direct Memory Access Channel is custom designed by the user for special applications. Those customers wishing to design their own Direct Memory Access Channel (DMAC) interfaces directly to the Memory Bus can do so with data transfer rates of up to 2,000,000 bytes per second in the Burst Mode through a custom-built DMAC. This is accomplished by using a general purpose wire wrapped circuit board available from General Electric. In addition to the circuits required for the user's device, his DMAC must include a 16-bit Memory Address Register and a 16-bit Memory Data Register, which are switched onto the Memory Bus when the DMAC captures the daisy-chain response for a request for service.

6.2 CORE MEMORY MODULES

GE-PAC 3010/2 core memory modules are three-wire, 3D, ferrite core memories which use 23 mil cores. Each core memory module contains 8,192 bytes of storage. Parity core memory modules contain a seventeenth bit plane in the core stack corresponding to the Parity Bit. There are 64 X-axis wires and 64 Y-axis wires common to all bit planes. Each bit plane has its own sense/inhibit wire. In a Write operation, inhibit current is passed over this wire to write a ZERO (inhibit current is not passed over this wire to write a ONE). In a Read operation, the bit readout is sensed over the sense/inhibit wire. The block diagram in Figure 6-2 illustrates a core memory module.

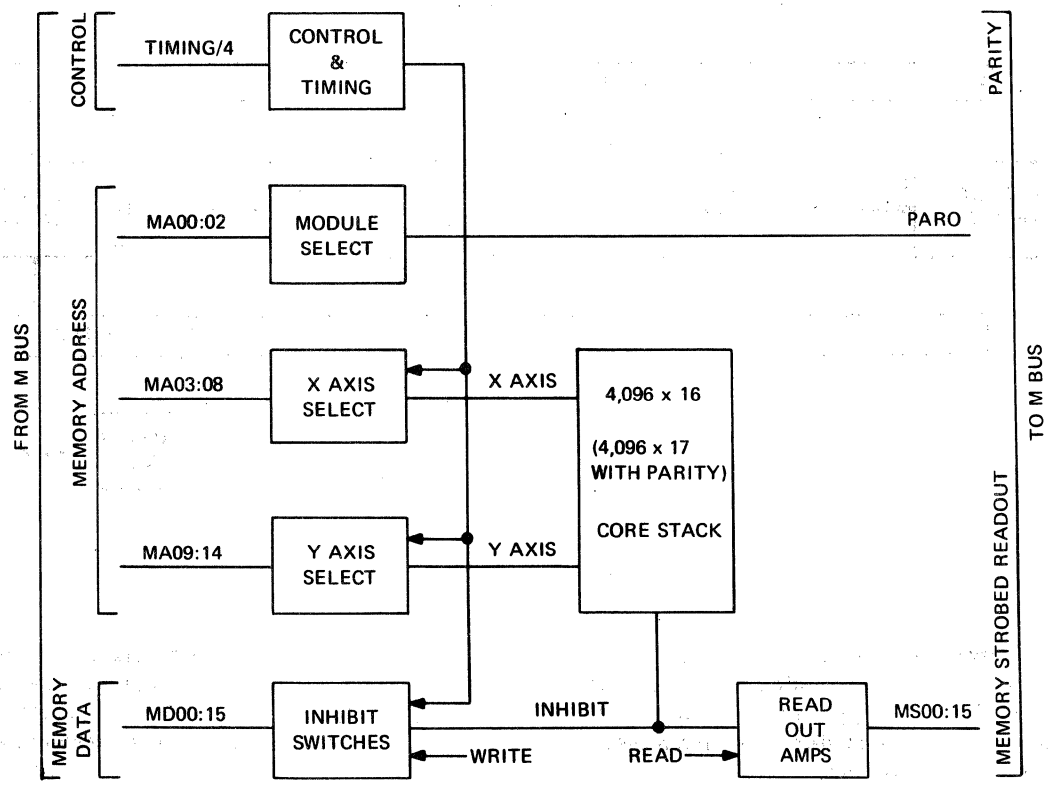


Figure 6-2. Core Memory Module

6.3 PARITY OPTION

The parity control logic is built into the Processor, and is enabled/disabled by the currently addressed memory module, via a Parity Enable (PAR0) line in the Memory Bus. The Processor parity circuit generates parity whether the current memory operation is from a Direct Memory Access Channel or the Processor. The Processor does not check parity on a Read operation unless the addressed memory module has the Parity option feature. This allows both Parity and non-parity memory

modules to be mixed in a given GE-PAC 3010/2 Memory System. The output from the parity check circuit is connected to the Machine Malfunction Interrupt when a Parity memory module is installed in the system. If a parity error is detected on a Processor Read operation, and if the Machine Malfunction Interrupt is enabled (PSW 02), the Processor is interrupted and performs a PSW swap with address X'3C'.

6.4 MEMORY BUS

The Memory Bus provides the communication path between the memory modules, the Processor, and the Direct Memory Access Channels. The Memory Bus totals 59 lines. See Figure 6-3.

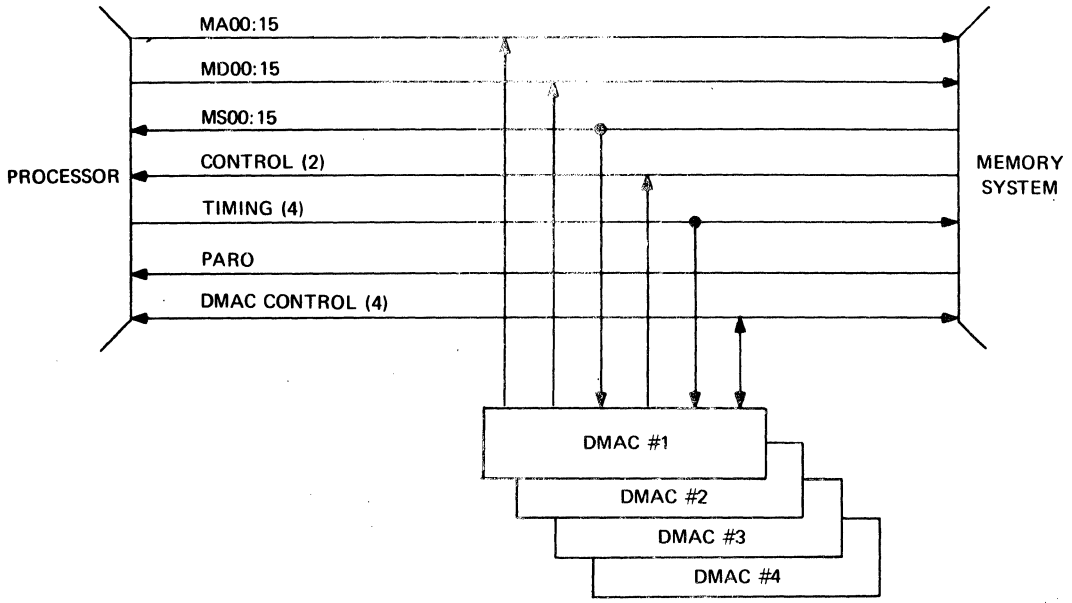


Figure 6-3. Memory System Diagram

NOTE

MD16 and MS16 correspond to the Parity Bit. The Processor always controls these lines. The DMAC should not OR tie onto MD16 or MS16.

The four DMAC Control lines are used to establish whether the Processor or one of the DMA Channels will be selected for the next memory cycle. The memory service is granted on a priority basis, with the Processor always lowest in priority. The four DMA Channels are assigned priority on a parallel daisy-chain basis where the device "closest" to the Processor is highest in priority.

The Parity Enable (PAR0) line enables the parity control logic whenever a Parity memory module is addressed. The parity logic is disabled when a memory module without the Parity option is addressed.

The four Timing lines required by the memory modules are generated by the Processor whether it is selected for the current memory cycle or not.

The total memory cycle time is 1.0 microsecond. The memory access time is 300 nanoseconds. See Figure 6-4.

The Memory Strobed Readout lines (MS00:16), carry the strobed readout from the addressed memory location approximately 300 nanoseconds after the beginning of the memory cycle. (MS16 corresponds to the Parity Bit.)

The Memory Data lines (MD00:16) carry the data to be written into the addressed memory location, or the data to be restored to the addressed memory location following a Read operation. The MD lines must remain settled for the entire write portion of the memory cycle. (MD16 corresponds to the Parity Bit.)

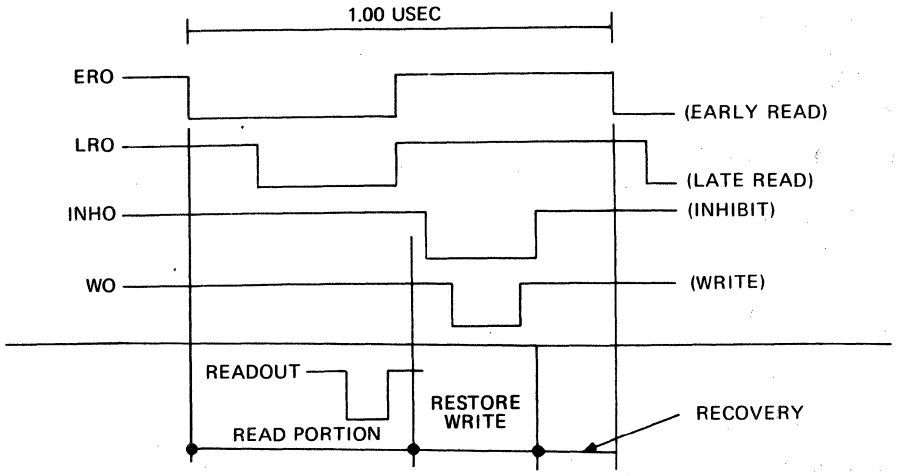


Figure 6-4. Typical Memory Cycle

The Memory Address lines (MA00:15) define a unique memory location for the current memory cycle. Since the memory is organized in 16-bit halfwords, MA15 must always be zero.

The two Control lines are used to provide externally generated memory busy and memory access timing for future product expansion.

Only one device may communicate with the Memory Bus at a time. When a DMA Channel requests memory, the Processor disconnects itself from the Memory Bus. At the same time, the Processor sends an enable signal which allows the highest priority DMA Channel to connect itself to the Memory Bus. The Processor generates all Memory Timing signals, and generates Memory Parity if a Parity memory module is addressed. The DMA Channel must provide Memory Address (MA00:15) and Memory Data (MD00:15).

6.4.1 Memory Bus Priority

The Processor is assigned the lowest priority on the Memory Bus. It will always stop and give the next available memory cycle to the DMA Channels whenever any one of them requests service. Priority between the four DMA Channels is established on a parallel daisy-chain basis. See Figure 6-5.

Any DMA Channel may request memory service at any time. The Processor scans the REQ0 line during every memory cycle. If the REQ0 line is active, the Processor responds by generating the Enable (EN0) line. The EN0 line is applied to the highest priority DMA Channel. The rising edge of EN0 marks the time the selected DMA Channel may switch itself onto the Memory Bus. See Figure 6-5.

If the EN0 signal is not "captured" by the highest priority DMA channel, the EN0 signal is passed daisy-chain fashion (ACT0/TAC0) from the top priority device, to the next highest priority device, and so on, until it is captured by the DMA Channel that requested service. The "captured" daisy-chain pulse ANDed with the rising edge of EN0, selects the DMA Channel. At this time, the selected DMA Channel must be prepared to execute the memory operation.

6.4.2 Memory Bus Interfacing

Interfaces to the Memory Bus are made directly according to the rules in this subsection.

All signals on the Memory Bus are DTL/TTL level compatible. A logical ONE or a true condition corresponds to a low level, V = 0.4VDC. A logical ZERO or a false condition corresponds to a high level, V = 2.5VDC. A user interface is permitted one DTL/TTL standard

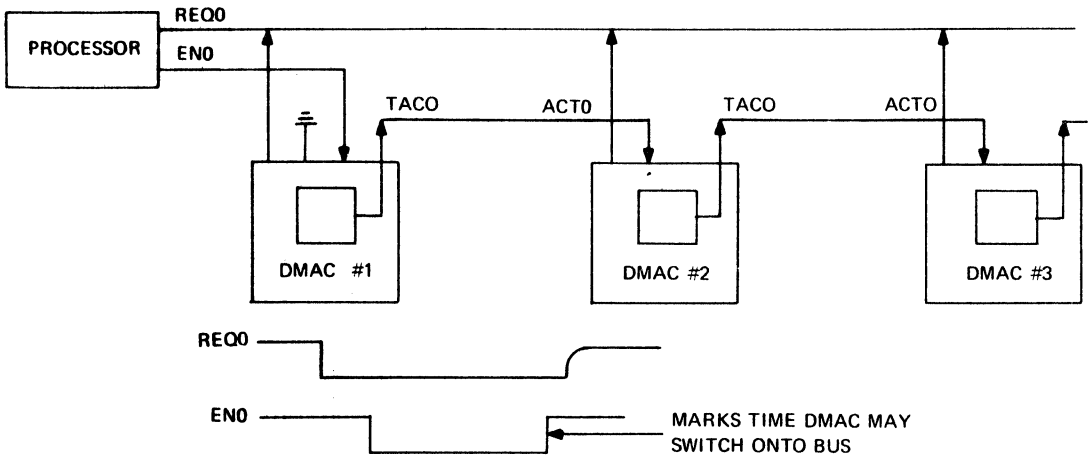


Figure 6-5. Example of Memory Bus Priorities

load and two TTL open collector power gate OR ties (48ma sink) onto lines on the Memory Bus. A user may not OR tie onto any of the four Timing lines, the Parity Enable (PAR0) line, the Memory Strobed Readout lines (MS00:16), or the two Control lines.

> NOTE
>
> **The Memory Bus may not be physically extended beyond the Processor chassis and one expansion chassis.**

Refer to Figure 6-6, Memory Bus Timing, and to Figure 6-3, Memory System Diagram. Figure 6-6 illustrates the activity on the Memory Bus for both the Read and write operations. Back panel pin assignments for an expansion slot in a memory expansion chassis are shown in Figure 6-6. An expansion slot will accommodate either I/O or memory.

The following brief descriptions of each interface signal are provided to aid the custom interface designer.

REQUEST (REQ0)

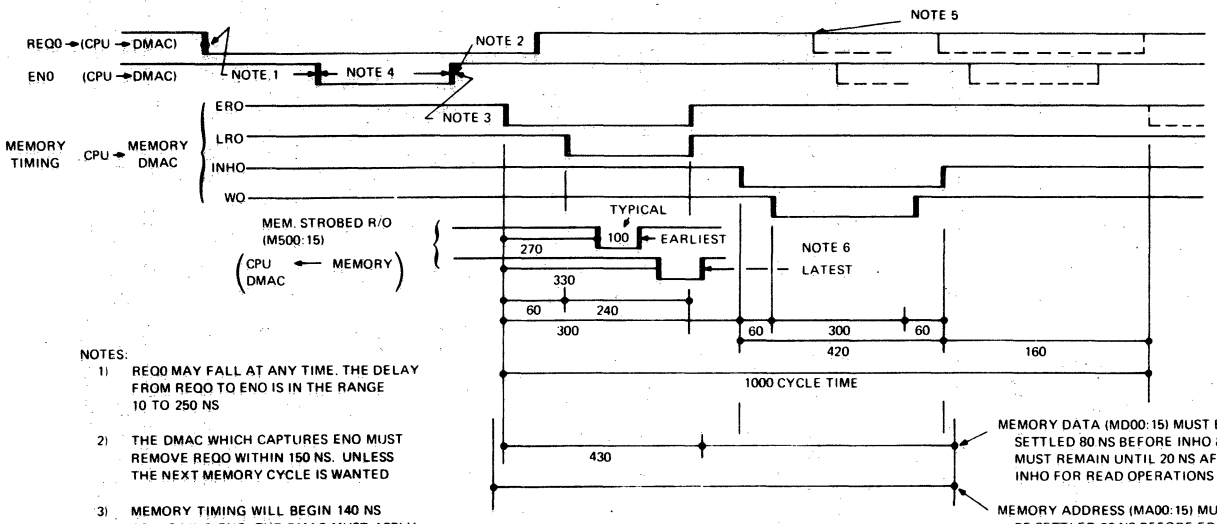
Up to four DMACs OR tie onto the REQ0 line. Any DMAC may activate REQ0 at any time. The system will not tolerate temporary false outputs on REQ0. The REQ0 line must be released 150 nanoseconds after the rising edge of EN0, unless the DMAC wants two consecutive memory cycles.

ENABLE (EN0)

The Processor responds to a REQ0 by activating EN0. The delay between REQ0 and EN0 will be 10 to 250 nanoseconds depending on when, during the current memory cycle, REQ0 is activated. The width of EN0 will vary from 250 nanoseconds minimum to 750 nanoseconds maximum depending on whether the memory is currently busy.

EN0 is used to generate the daisy-chain priority loop through all DMACs in the system. The daisy-chain loop begins at the highest priority DMAC and propogates to the lower priority DMACs until it is "captured" by the DMA which requested service.

The "captured" daisy-chain pulse is ANDed with the rising edge of EN0 to set the Select flip-flop in the DMAC. Figure 6-7 illustrates a suitable circuit for the daisy-chain select request circuits.



NOTES:

1) REQ0 MAY FALL AT ANY TIME. THE DELAY FROM REQ0 TO ENO IS IN THE RANGE 10 TO 250 NS

2) THE DMAC WHICH CAPTURES ENO MUST REMOVE REQ0 WITHIN 150 NS. UNLESS THE NEXT MEMORY CYCLE IS WANTED

3) MEMORY TIMING WILL BEGIN 140 NS FOLLOWING ENO. THE DMAC MUST APPLY MA00:15 WITHIN 80 NS OF ENO.

4) ENO WIDTH WILL BE IN THE RANGE OF 250 TO 750 NS, DEPENDING ON WHETHER OR NOT THE MEMORY IS CURRENTLY BUSY

5) THE DMAC MUST ACTIVATE REQ0 WITHIN 650 NS OF ENO TO INSURE GETTING THE NEXT MEMORY CYCLE FOR BURST MODE OPERATION.

6) THE MS BUS SHOULD BE IGNORED IF THE OPERATION IS WRITE TO MEMORY.

CONN 0		
ROW 1	ROW 2	
P5	GND	41
GND	GND	40
P15	REQ0	
N15	EN0	
ACT0	TAC0	
		35
XRACK0		
MA130	MA140	
110	120	
090	100	
070	080	30
MA050	MA060	
RDACK0	TDACK0	
DC0	DCR0	
SCLR0	HW0	
		25
SYN0	ATN0	
RACK0	TACK0	
CL070	DA0	
DR0	CMD0	20
SR0	ADRS0	
D140	D150	
120	130	
100	110	
080	090	15
060	070	
040	050	
020	030	
D000	D010	
MA030	MA040	10
020	021	
01	02	
010	011	
MA000	MA001	
PAR0	MA00	05
INH0	ER0	
W0	LR0	
P15	N15	
GND	GND	
P5	GND	00

DATA CHANNEL LINES (BTM ONLY)

I/O CONNECTIONS - BTM

CONN 1		
ROW 1	ROW 2	
P5	GND	41
GND	GND	40
P15	P15	
N15	N15	
MD150	MD160	
130	140	
110	120	35
090	100	
070	080	
050	060	
030	040	
MD010	MD020	30
EXVT	MD000	
TEMPA	VT	
WRT0	TEMPB	
SCLR0	HW0	
		25
SYN0	ATN0	
RACK0	TACK0	
CL070	DA0	
DR0	CMD0	20
SR0	ADRS0	
D140	D150	
120	130	
100	110	
080	090	15
060	070	
040	050	
020	030	
D000	D010	
	MS000	10
MS010	020	
030	040	
060	060	
070	080	
090	100	05
110	120	
130	140	
MS150	MS160	
GND	GND	
P5	GND	00

I/O CONNECTIONS - TOP

150 0701

CONN NO
BOARD NO
PIN NO
ROW OF CONN

Figure 6-6. Memory Bus Timing

MEMORY TIMING LINES

EARLY READ (ER0) LATE READ (LR0)	These two lines control the Read Current Switching in the addressed memory module.
INHIBIT (INH0) WRITE (W0)	These two lines control the Restore/Write Current Switching in the addressed memory module.
	The Memory Timing lines are available to the custom built DMAC. The rising edge of INH0 marks the end of the effective memory cycle. The DMAC should disconnect itself from the Memory Bus within 150 nanoseconds after the rising edge of INH0.
MEMORY ADDRESS LINES (MA00:15)	The selected DMAC may activate the MA lines any time following the rising edge of EN0. However, it must guarantee the MA lines be settled, over the interval, beginning 80 nanoseconds after EN0 until 20 nanoseconds following INH0. Further, the DMAC must release the MA lines within 150 nanoseconds after the rising edge of INH0. (MA15 must be zero.)
MEMORY STROBED READ-OUT LINES (MS00:16)	The MS lines carry the pulsed single-rail Memory Readout on all memory operations. The MS lines are inactive on Write operations. The pulsed readout varies in width from 60 to 100 nanoseconds, and in position from 410 to 470 nanoseconds following the rising edge of EN0. The selected DMAC is required to register the Memory Readout and return the data on the MD lines for the restore portion of the operation. The DMAC should ignore the MS lines on all Write operations.
MEMORY DATA LINES (MD00:16)	The MD lines carry data to be written to the addressed memory module. They also carry data to be restored to the addressed memory module. If the operation is Write, the MD lines must be settled, for the interval, beginning 80 nanoseconds before INH0 to 20 nanoseconds after INH0. If the operation is Read, the Strobed Readout from the MS lines must be returned to the MD line by the falling edge of INH0 and must remain there until INH0 rises. Regardless of the operation (Read or Write), the DMAC must release the MD lines within 180 nanoseconds after the rising edge of INH0.
	Note that MD16 and MS16 correspond to the Parity Bit. The Processor always controls these lines. The DMAC should not OR tie onto MD16 or MS16.

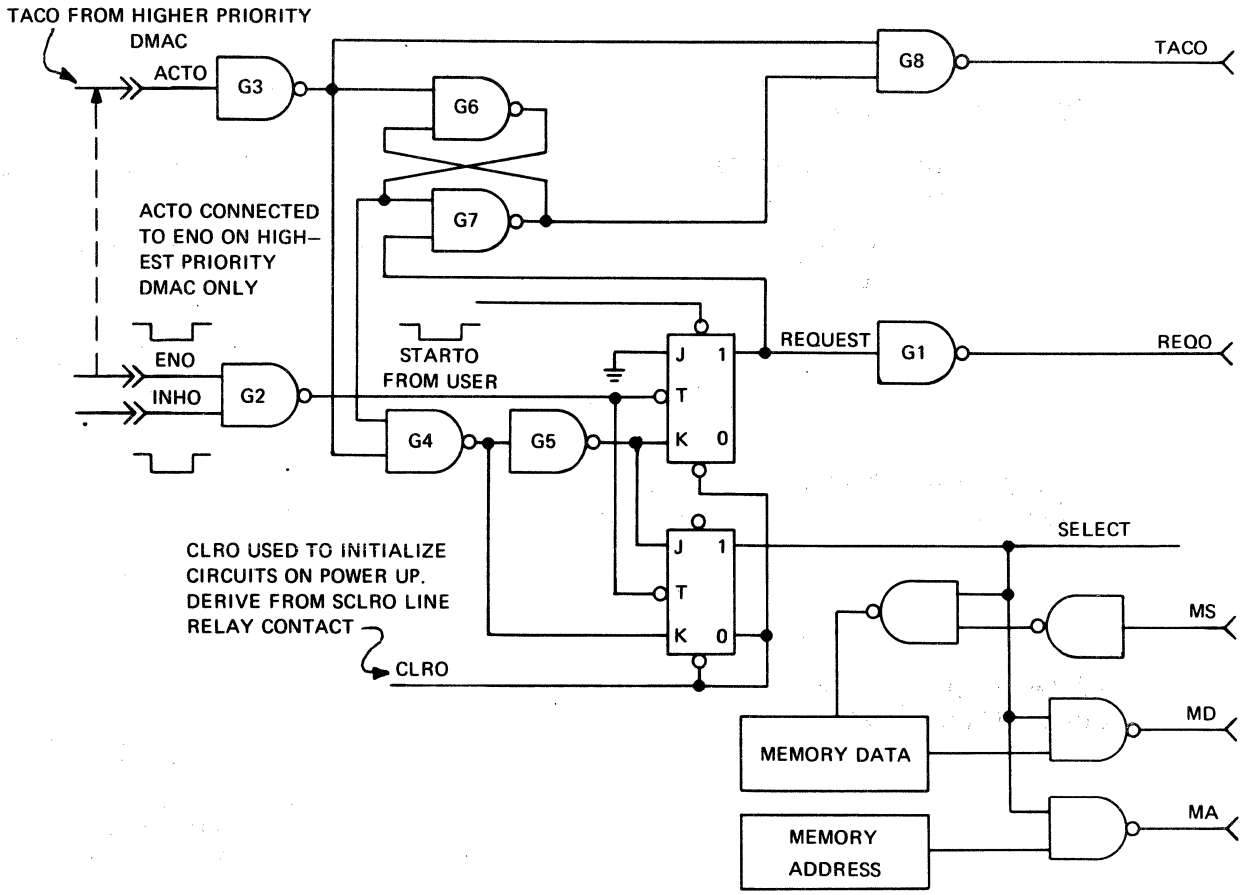
6.4.3 Memory Bus Timing

The Request flip-flop is set by START0 when the user's circuits require memory service. (START0 should be between 30 nanoseconds and 100 nanoseconds wide.) The REQ0 line is driven active through Gate G1 when the Request flip-flop is set.

Gates G2 through G7 form the daisy-chain capture circuit which establishes DMAC priority. Gates G6 and G7 are a contention circuit which will capture the ACT0 pulse if the Request flip-flop is set, or it passes ACT0 onto the next DMAC (as TAC0) through G8 if the Request flip-flop is not set. The contention circuit will ignore a change in the Request flip-flop if it changes during the ACT0 period.

The daisy-chain delay through one DMAC is 48 nanoseconds maximum, assuming TTL logic, 12 nanoseconds maximum delay per stage.

If the circuit captures the daisy chain pulse, the Request flip-flop is toggled reset on the rising edge of EN0 and the Select flip-flop is toggled set at the same time. The Select flip-flop is used to switch the



STARTO (30 TO 100 NS) FROM USER CKT SETS REQUEST FLOP WHEN USER'S MEM. ADRS & DATA REGISTERS ARE READY. CPU RESPONDS TO REQO WITH ENO. IF THIS DMAC DID NOT REQUEST SERVICE IT PASSES THE DAISY CHAIN PULSE (ACTO) ON TO THE NEXT DMAC AS TACO. IF THIS DMAC DID REQUEST SERVICE IT CAPTURES THE ACTO PULSE (A 30 NS GLITCH ON TACO IS PERMISSIBLE). ON THE RISING EDGE OF ENO THE REQUEST FLOP IS RESET & THE SELECT FLOP IS SET. THE SELECT FLOP ENABLES THE DMAC TO THE MEMORY BUS (MS, MD, & MA). THE SELECT FLOP IS RESET AT THE END OF THE MEMORY CYCLE BY THE RISING EDGE OF INHO.

Figure 6-7. Typical Control Logic For Interfacing To The Memory Bus

DMAC onto the Memory Bus. The Select flip-flop is cleared on the rising edge of INH0 by Gate G2 which marks the end of the memory cycle.

The ACT0 line is grounded on the top priority DMAC. On all other DMACs, the ACT0 input is driven by TAC0 from the next higher priority DMAC.

The two flip-flops, Request and Select, should be initialized by CLR to a reset condition on Power Up by the user's circuits. (The SCLR0 line is useful for this function.)

CHAPTER 7

CONTROL CONSOLE

7.1 INTRODUCTION

This chapter describes the Control Console which is supplied as part of the GE-PAC 3010/2 Processor. The Control Console provides the system operator with visual indications of the state of the Processor, as well as manual control over the GE-PAC 3010/2

7.2 CONTROL CONSOLE DESCRIPTION

The Control Console, shown in Figure 7-1, is a RETMA standard 5 1/4" X 19" panel which is plug removable from the Processor. It displays the current state of the Processor and provides all necessary manual controls for the GE-PAC 3010/2. The Control Console includes the following control and display elements:

Indicators	- Power ON Lamp
	- Wait Lamp
	- Display Register D1 - 16 Bits (top)
	- Display Register D2 - 16 Bits (bottom)
Data Switches	- Sixteen Data/Address Switches
Control Switches	- Key operated OFF-ON-LOCK, Security Lock Switch
	- Initialize Switch (INT)
	- Execute Switch (EXE)
Function Switches	- Single Switch (SGL)
	- Run Switch (RUN)
	- Twelve Position Rotary Function Switch

A functional description of each of these control and display elements is given in this section.

7.2.1 Key Operated Security Lock

This is a three-position, OFF-ON-LOCK, key-operated locking switch, which controls the primary power to the GE-PAC 3010/2. This switch can also disable the Control Console, thereby preventing any accidental manual input to the system. The POWER indicator lamp associated with the key lock is located in the upper left corner of the Control Console. The POWER lamp is lit when the key lock is in the ON or LOCK position. The relationship between the key lock switch positions, primary power, and the Control switches is:

OFF	The primary power is Off.
ON	The primary power is On and the Control switches are enabled.
LOCK	The primary power is On and the Control switches are disabled.

7.2.2 Control Switches

The Control switches on the Control Console are active only when the key-operated locking switch is in the ON position. The function of each of these switches is as follows:

INITIALIZE (INT)	The momentary Initialize (INT) switch causes the system hardware to be initialized. After this initialize operation, all device controllers on the system Multiplexor Bus are cleared and certain other functions in the Processor are reset.

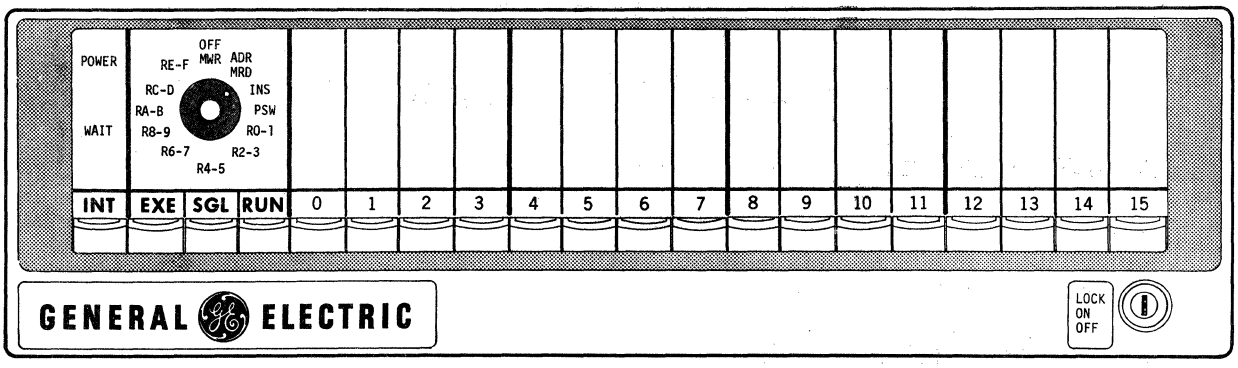


Figure 7-1. Control Console

EXECUTE (EXE) The momentary Execute (EXE) switch causes the Processor to perform the Control Console operation selected by the Function switches, as discussed in Section 7.2.3. Note that only the Initialize (INT) Control switch acts independently of the Execute switch.

7.2.3 Function Switches

The three Function switches, SINGLE (SGL), RUN (RUN), and the 12 position rotary Function switch are used to place the Processor in various operating modes. The Processor is controlled by setting the function switches in the proper positions and then depressing the Execute (EXE) switch to activate the function. The various Processor modes, and the method of entering each mode, are as follows:

ADDRESS - To enter the ADDRESS Mode
a) the 12 position Function switch must be in the ADR/MRD position
b) the SGL switch must be UP
c) the RUN switch must be UP
The desired address is specified on the 16 Data/Address switches. When the Execute (EXE) switch is depressed, the specified address is placed into the address portion of the Current Program Status Word PSW (16:31). (When the address is transferred from the Data/Address switches, the least significant bit (PSW31) is cleared so that the resulting address is always even.) The complete PSW is displayed on Display Register D1 (top) and Display Register D2 (bottom). This address can be used to read data from memory, to write data to memory, or to start the execution of a program.

MEMORY READ - To enter the MEMORY READ Mode
a) the 12 position Function switch must be in the ADR/MRD position
b) the SGL switch must be DOWN
c) the RUN switch must be UP
When EXE is depressed, the data read from memory is displayed on the Display Register D2 (bottom) indicator lamps and the address incremented by 2 of that data is displayed on the Register Display D1 (top) lamps. The address portion of the PSW is also incremented by 2. Depressing EXE repeatedly displays consecutive locations from memory.

MEMORY WRITE - To enter the MEMORY WRITE Mode
a) the 12 position Function switch must be in the OFF/MWR position
b) the SGL switch must be DOWN
c) the RUN switch must be UP
The desired word to be written is specified on the 16 Data/Address switches. When EXE is depressed, the data written to memory is displayed on the Display Register D2 (bottom) lamps and the address incremented by 2 of that data is displayed on the Register Display D1 (top) lamps. The address portion of the PSW is also incremented by 2. Depressing EXE repeatedly writes data from the Data/Address switches into consecutive memory locations.

RUN - To enter the RUN Mode
a) the 12 position Function switch may be in any position except OFF/MWR or ADR/MRD
b) the SGL switch must be UP
c) the RUN switch must be DOWN
When EXE is depressed, the Processor begins program execution. In the RUN Mode, the Display Registers are not activated or controlled by the Processor. Rather, the program can use the Control Console as an I/O device. If the program does not output to the Display Registers, these registers retain the last value displayed prior to entering the RUN Mode.

SINGLE - To enter the SINGLE Mode
a) the 12 position Function switch may be in any position except OFF/MWR or ADR/MRD
b) the SGL switch must be DOWN
c) the RUN switch must be DOWN
This mode allows programs to be executed, one instruction at a time, each time EXE is depressed. The Display Registers 1 and 2 contain various information specified by the 12 position Function switch after each EXE as follows:

INS - the next instruction to be executed is displayed. The first halfword from memory is shown in Display Register D1 (top) and the second halfword from memory is shown in Display Register D2 (bottom).

PSW- the Current Program Status Word is displayed. The program status and the condition code is shown in Display Register D1 and the location counter is shown in Display Register D2.

R0:1
R2:3
R4:5
R6:7
R8:9
RA:B
RC:D
RE:F

the pair of general registers specified by the switch is displayed. For example: if the switch is in the R8:9 position, General Register 8 is displayed on Display Register D1 and General Register 9 is displayed on Display Register D2.

HALT - To enter the HALT Mode
a) the 12 position Function switch may be in any position except OFF/MWR or ADR/MRD.
b) the SGL switch may be UP or DOWN (don't care)
c) the RUN switch must be UP
When EXE is depressed, the Processor enters a HALT Mode which is non-interruptable. Any of the information listed above under the SINGLE Mode can be displayed on Display Registers D1 and D2 by placing the 12 position Function switch in the proper position and depressing EXE.

When the Processor is in the HALT Mode, the WAIT indicator on the Control Console is illuminated. The WAIT indicator is also on between instructions when the Processor is in the SINGLE Mode. Finally, the WAIT indicator is lighted when an executing program enters the WAIT state by setting Bit 0 of the PSW. Note that the program initiated Wait state is interruptable, while the console initiated HALT Mode is not interruptable.

7.3 CONTROL CONSOLE OPERATING PROCEDURES

7.3.1 Power Up

To power up and initialize (clear) the system:

1. Turn the key-operated security lock clockwise from the OFF position to the ON position.
2. Depress the momentary Initialize (INT) Control switch, to initialize the system.

This action provides electrical power to the system, and leaves the Processor in the HALT Mode. It is recommended that before the system is used, a few important pointers and New PSWs in core memory be initialized. The locations in memory to be adjusted are shown in Table 7-1.

TABLE 7-1
CORE MEMORY INITIALIZATION

Location (hex)	Function	Suggested Setting	Comment
0022	Pointer to register save area	0058	This pointer should contain the address of a block of 32 bytes which are available for register save and restore operations.
0034 0036	New PSW for Illegal Instruction Interrupts	8000 0050	If an Illegal instruction occurs, this New PSW clears all interrupts and puts the Processor into Wait state with Location Counter = 0050.
003C 003E	New PSW for Machine Malfunction Interrupts	8000 0050	This New PSW treats Machine Malfunction Interrupts the same as Illegal instructions for purposes of initialization.
0050 0052 0054 0056 0078	Auto-Load sequence for loading programs	D500 00CF 4300 0080 XXYY	This sequence uses the Auto-Load instruction at 50 followed by an Unconditional Branch to 80 to perform initial program loads. With this sequence location 78 should be loaded with Device Number XX and Command Byte YY. Refer to Appendix 8 for information on I/O devices.

The memory locations mentioned previously can be set using Memory Write operations as follows:

1. Enter 0022 (0000 0000 0010 0010) into the Data/Address switches, release the SGL switch, set the rotary Function switch to ADR, and depress EXE.
2. Enter 0058 (0000 0000 0101 1000) into the Data/Address switches, depress the SGL switch, set the rotary Function switch to MWR, and depress EXE. This enters value 0058 into location 0022.
3. Enter 0034 into the Data/Address switches, release the SGL switch, set the rotary Function switch to ADR, and depress EXE.
4. Enter 8000 into the Data/Address switches, depress the SGL switch, set the rotary Function switch to MWR, and depress EXE.
5. Enter 0050 into the Data/Address switches, and depress EXE. These steps enter values 8000 and 0050 into memory starting at 0034.
6. Follow similar steps until all specified locations in memory have been set properly.

Once these locations are set, their contents can be verified using Memory Read operations as follows:

1. Enter 0022 into the Data/Address switches, release the SGL switch, set the rotary Function to switch ADR, and depress EXE.
2. Depress the SGL switch, set the rotary Function switch to MRD, and depress EXE. At this point, the address (0024) should be displayed in Register Display D1, and the contents of 0022 (0050) should be displayed in Register Display D2.

3. Enter 0034 into the Data/Address switches, release the SGL switch, set the rotary Function switch to ADR, and depress EXE.

4. Depress the SGL switch, set the rotary Function switch to MRD, and depress EXE. At this point, Register Display D1 should show the address (0036) and Register Display D2 should show the contents of 0034 (8000).

5. To examine the next location, depress EXE. At that time, Register Display D1 should show the address (0038), and Register Display D2 should show the data (0050).

6. Follow similar steps until all appropriate locations have been verified.

Once memory locations are set and verified, it is still necessary to initialize the Current PSW. Note that while the Location Counter [PSW (16:31)] can be set using the ADR/MRD position of the rotary Function switch, there is no way to directly adjust the program status [PSW (0:15)] from the Control Panel. When power is turned on, the program status is loaded from memory location 0024, the PSW save area. Following a cold start, this initial setting is arbitrary. The program status can be set in two ways: either by executing an LPSW or EPSR instruction, or by servicing an interrupt with a PSW swap. For system initialization, the recommended procedure is to execute an Illegal instruction, which forces the Illegal instruction PSW swap. Using core memory settings suggested above, this PSW initialization can be performed by starting program execution at location 0034, which is an Illegal instruction. The specific steps are:

1. Enter 0034 into the Data/Address switches, release the SGL switch, set the rotary Function switch to ADR, and depress EXE.

2. Depress the RUN switch, set the rotary Function switch to a position other than OFF/MWR or ADR/MRD, and depress EXE.

The result of performing these two steps is that the Processor attempts to execute the contents of location 0034 (8000) which is an Illegal instruction. An Illegal instruction PSW swap occurs, which loads the program status with 8000, loads the Location Counter with 50, and leaves the Processor in the Wait State. At this point, if EXE is depressed, the Processor executes the Auto-Load sequence at 0050.

7.3.2 Power Down

To shut down power to the system,

1. Place the Processor in the HALT Mode as described in Section 7.2.3.

2. Turn the key-operated security lock to the OFF position.

This removes AC power from the system and forces a Primary Power Fail Interrupt to the Processor. Refer to 7.4.4 for further details.

7.3.3 Program Loading

There are many ways to load the memory with programs and/or data. Most programs are loaded using one of the program loaders associated with the 3010/2 software. Refer to GET-6171 for details on loaders and other software programs.

The Auto-Load sequence, referred to in the previous section, is useful for loading programs when the system is being initially loaded, or when no other program loaders are in memory. The sequence recommended in the previous section is described below:

TABLE 7-2
AUTO LOAD SEQUENCE

Location	Contents	Instruction		
50	D500 00CF	AL	X'CF'	AUTO LOAD
54	4300 0080	B	X'80'	BRANCH TO 80
78	XXYY	DC	X'XXYY'	DEV NO AND CMND

This sequence is based on the Auto-Load instruction, which is described in Section 5.3.10. This instruction reads eight-bit data bytes from Device XX into memory, starting at location 0080. The load operation proceeds until the device indicates a termination status, or until a specified upper limit is reached. In the sequence above, the limit is defined as location 00CF, which allows 80 bytes to be read. With the Auto-Load instruction, the device address to be used is specified in byte location 78, and the command byte which starts the device is specified in byte location 79. Leading zero data bytes are skipped and not loaded. This sequence is appropriate with Teletypewriter or a paper tape reader which transfers eight-bit data bytes. When the Auto-Load instruction terminates, the Branch instruction transfers control to location 0080. This Auto-Load sequence can be easily changed to meet other requirements. The upper limit (00CF), at 0052, can be changed to load programs of different length. The transfer address (0080), at 0056, can be changed to branch to a different location. Following the Auto-Load instruction, it is possible to test the Condition Code to determine exactly how the Auto-Load operation terminated. An all zero Condition Code implies that the specified program length was loaded. A non-zero Condition Code implies that the device terminated the load sequence before the program length was satisfied.

7.3.4 Program Execution

To begin the execution of a program, the system must be in the Halt Mode.

1. Set the rotary Function switch to ADR/MRD.
2. Release the SGL switch.
3. Enter the program starting address in the 16 Data/Address switches.
4. Depress the momentary EXE switch.
5. Set the rotary Function switch to a position other than OFF/MWR or ADR/MRD.
6. Depress the RUN switch.
7. Depress the momentary EXE switch. The Processor is now in the RUN Mode.

To execute a program in the Single Step Mode-one instruction at a time-the system must be in the HALT Mode.

1. Set the rotary Function switch to ADR/MRD.
2. Release the SGL switch.
3. Enter the program starting address in the 16 Data/Address switches.
4. Depress the EXE switch.
5. Depress the SGL switch and the RUN switch.
6. Set the rotary Function switch to select the register(s) desired for display. (Must be a position other than OFF/MWR or ADR/MRD).

7. Depress the EXE switch to execute one instruction.
8. Repeat Step 7. to execute successive instructions.

At any time during the single-step sequence, the rotary Function switch can be adjusted to change the selection of registers to be displayed. To examine more than one register pair without executing more instructions, release (raise) the RUN switch. This leaves the Processor in the HALT Mode. Then select the registers to be displayed, with the rotary Function Switch, and depress EXE to observe those registers. Depress the RUN switch to resume single-step execution.

7.3.5 Program Termination

To manually halt the execution of a program,

1. Set the rotary Function switch to a position other than OFF/MWR or ADR/MRD.
2. Release the RUN Switch.
3. Depress the momentary EXE switch.

When the Processor enters the HALT Mode, the Register Displays are updated as specified by the rotary Function switch. For example; if EXE is depressed with the rotary Function switch set at the PSW position, the execution is halted and the Current PSW is displayed on the Register Displays. Each time EXE is depressed, while in the HALT Mode, the Register Displays are updated as specified by the rotary switch. Therefore, to display different register pairs once the Processor is in the HALT Mode, change the rotary Function switch setting and depress EXE.

7.3.6 Manually Initiated Memory Operations

7.3.6.1 Memory Read

To display the contents of memory locations on the Register Displays the Processor must be in the HALT Mode.

1. Set the rotary Function switch to the ADR/MRD position.
2. Release the SGL switch.
3. Enter the memory read starting address in the 16 Data/Address switches.
4. Depress the EXE switch.
5. Depress the SGL switch.
6. Depress the EXE switch.
7. The address read from, plus two, appears in Register Display D1. The data read from memory appears in Register Display D2.
8. Repeat from Step 6. to read successive memory locations. The Location Counter is automatically incremented each time EXE is depressed.

7.3.6.2 Memory Write

To write data from the Data/Address switches into memory the Processor must be in the HALT Mode.

1. Set the rotary Function switch to the ADR/MRD position.
2. Release the SGL switch.
3. Enter the memory write starting address in the 16 Data/Address switches.
4. Depress the EXE switch.
5. Set the rotary Function Switch to the OFF/MWR position.
6. Depress the SGL switch.

7. Enter the data to be written in the 16 Data/Address switches.

8. Depress the EXE switch.

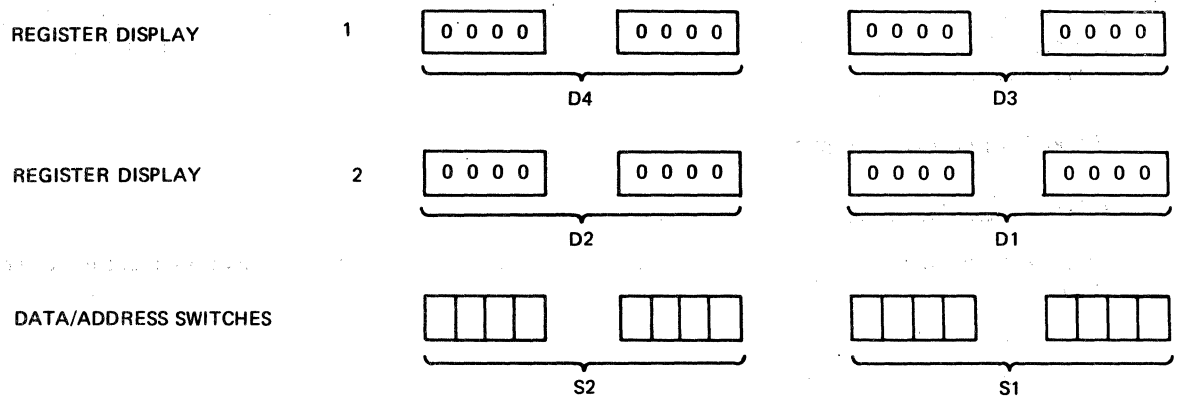
9. The address written into plus two appears in Register Display D1. The data written into memory appears in Register Display D2.

10. Repeat from Step 7. to write into successive memory locations. The Location Counter is automatically incremented each time EXE is depressed.

7.4 PROGRAMMING CONSIDERATIONS

7.4.1 Control Console I/O

The Control Console is available to any running program as an I/O device with Device Address 01. The status and command bytes for the Control Console are summarized in Appendix 8. The status byte indicates the setting of the Function switches. The command byte specifies either Normal or Incremental Mode, which pertains to data transfers. In the Normal Mode, the selection logic which determines which half of the Data/Address switches and which byte of the Register Displays is transferred is reset every time the Control Console is addressed on the Multiplexor Bus. The Control Console is addressed by every I/O instruction using Device Address 01. Subsequent Read or Write instructions transfer subsequent bytes as shown in Figure 7-2. Normal I/O instructions, therefore, can be used to input data from the Data/Address switches, and output data to the Register Displays.



Instructions Executed (RR or RX)	Data Transferred: Normal Mode	Data Transferred: Incremental Mode
RD	S1	S1
RD	S1	S2
RD	S1	S1
RD	S1	S2
RH	S1, S2	S1, S2
RH	S1, S2	S1, S2
RB*	S1, S2, S1, S2	S1, S2, S1, S2
WD	D1	D1
WD	D1	D2
WD	D1	D3
WD	D1	D4
WH	D1, D2	D1, D2
WH	D1, D2	D3, D4
WB*	D1, D2, D3, D4	D1, D2, D3, D4

*Block Length = 4 bytes

Figure 7-2. Control Console Data Transfers

7.4.2 Console Interrupt

In the GE-PAC 3010/2, an interrupt can be generated from the Control Console as follows:

1. The program must have Bit 4 of the Current PSW set, which specifies Automatic I/O Service Mode.
2. The operator must have the rotary Function switch in the OFF/MWR position, the SGL switch released, the RUN switch depressed, and then depress EXE.

This feature enables an operator to inform the running program that some operator service or function is needed. No acknowledgement of the interrupt is needed by the running program. If the Automatic I/O Service Mode is not enabled, console interrupts are not generated and are not queued.

7.4.3 Wait State

The running program can put the Processor into the Wait State by setting Bit 0 of the Current PSW. The operator is informed of this action by the WAIT indicator lamp being illuminated. The Processor can leave the Wait State and resume execution in two ways:

1. An interrupt can occur, causing a PSW swap and the execution of a routine to service the interrupt. When the routing restores the original PSW, the Wait State will be re-established.
2. The operator places the Processor into the RUN Mode as described in Section 7.2.3, which causes the execution to resume at the address specified by the Location Counter [PSW (16:31)] .

Note that the use of the programmed Wait State must be considered carefully when using Single-Step Mode. That is, with single-step execution it is possible to inadvertently "step through" the Wait State, through rapid or continuous use of the EXE switch.

7.4.4 Power Fail

Depressing the INT switch of the Control Console deactivates the system clear relay (SCLR0) in the system for a brief interval (approximately one-second). When this happens, the Processor saves the Current PSW in memory locations 0024 and 0026, and the 16 General Registers in the block indicated by the contents of 0022, and then follows an orderly shut-down sequence, which preserves the data within memory. When the relay is reactivated, the PSW and General Registers are reloaded from memory, and the device controllers are all initialized. At this point, the Processor interrogates the Control Console. If the Function switches are not set up to enter the RUN Mode, the Processor enters the HALT Mode and the WAIT lamp is illuminated. If the Function switches are set up to enter the RUN Mode, the Processor examines Bit 2 of the restored PSW. If PSW Bit 2 (the Machine Malfunction Interrupt Enable) is set, the Processor then performs the appropriate PSW swap for Machine Malfunction Interrupts. The purpose of this interrupt is to inform the running program that a power failure/restore has occurred. If Bit 2 of the PSW is not set, program execution resumes where it left off, with no Machine Malfunction Interrupt.

The use of the INT switch should be considered carefully when Bit 2 of the Current PSW is enabled.

APPENDIX 1
INSTRUCTION SUMMARY - ALPHABETICAL

INSTRUCTION	OP CODE	MNEMONIC	PAGE NO.
Acknowledge Interrupt	DF	AI	4-47
Acknowledge Interrupt RR	9F	AIR	4-47
Add Halfword	4A	AH	4-8
Add Halfword Immediate	CA	AHI	4-8
Add Halfword RR	0A	AHR	4-8
Add Halfword Memory	61	AHM	4-8
Add Immediate Short	26	AIS	4-8
Add to Bottom of List	65	ABL	4-44
Add to Top of List	64	ATL	4-44
Add with Carry Halfword	4E	ACH	4-9
Add with Carry Halfword RR	0E	ACHR	4-9
AND Halfword	44	NH	4-17
AND Halfword Immediate	C4	NHI	4-17
AND Halfword RR	04	NHR	4-17
Autoload	D5	AL	4-55
Branch and Link	41	BAL	4-42
Branch and Link RR	01	BALR	4-42
*Branch on False Condition	43	BFC	4-40
*Branch on False Condition RR	03	BFCR	4-40
*Branch on True Condition	42	BTC	4-39
*Branch on True Condition RR	02	BTCR	4-39
*Branch on True Backward Short	20	BTBS	4-39
*Branch on True Forward Short	21	BTFS	4-39
*Branch on False Backward Short	22	BFBS	4-40
*Branch on False Forward Short	23	BFFS	4-40
Branch on Index High	C0	BXH	4-41
Branch on Index Low or Equal	C1	BXLE	4-41
Compare Halfword	49	CH	4-13
Compare Halfword Immediate	C9	CHI	4-13
Compare Halfword RR	09	CHR	4-13
Compare Logical Byte	D4	CLB	4-23
Compare Logical Halfword	45	CLH	4-12
Compare Logical Halfword Immediate	C5	CLHI	4-12
Compare Logical Halfword RR	05	CLHR	4-12
Divide Halfword	4D	DH	4-15
Divide Halfword RR	0D	DHR	4-15

*See Extended Branch Mnemonics in Appendix 3 for forty-four (44) additional symbolic instructions.

INSTRUCTION	OP CODE	MNEMONIC	PAGE NO.
Exchange Byte RR	94	EXBR	4-23
Exchange Program Status RR	95	EPSR	4-57
Exclusive OR Halfword	47	XH	4-19
Exclusive OR Halfword Immediate	C7	XHI	4-19
Exclusive OR Halfword RR	07	XHR	4-19
Floating - Point Add	6A	AE	4-26
Floating - Point Add RR	2A	AER	4-26
Floating - Point Compare	69	CE	4-28
Floating - Point Compare RR	29	CER	4-28
Floating - Point Divide	6D	DE	4-30
Floating - Point Divide RR	2D	DER	4-30
Floating - Point Load	68	LE	4-25
Floating - Point Load RR	28	LER	4-25
Floating - Point Multiply	6C	ME	4-29
Floating - Point Multiply RR	2C	MER	4-29
Floating - Point Store	60	STE	4-25
Floating - Point Subtract	6B	SE	4-27
Floating - Point Subtract RR	2B	SER	4-27
Load Byte	D3	LB	4-22
Load Byte RR	93	LBR	4-22
Load Complement Short	25	LCS	4-4
Load Halfword	48	LH	4-4
Load Halfword Immediate	C8	LHI	4-4
Load Halfword RR	08	LHR	4-4
Load Immediate Short	24	LIS	4-4
Load Multiple	D1	LM	4-5
Load Program Status Word	C2	LPSW	4-56
Multiply Halfword	4C	MH	4-14
Multiply Halfword RR	0C	MHR	4-14
Multiply Halfword Unsigned	DC	MHU	4-14
Multiply Halfword Unsigned RR	9C	MHUR	4-14
OR Halfword	46	OH	4-18
OR Halfword Immediate	C6	OHI	4-18
OR Halfword RR	06	OHR	4-18
Output Command	DE	OC	4-49
Output Command RR	9E	OCR	4-49

INSTRUCTION	OP CODE	MNEMONIC	PAGE NO.
Read Block	D7	RB	4-51
Read Block RR	97	RBR	4-51
Read Data	DB	RD	4-49
Read Data RR	9B	RDR	4-49
Read Halfword	D9	RH	4-53
Read Halfword RR	99	RHR	4-53
Rotate Left Logical	EB	RLL	4-34
Rotate Right Logical	EA	RRL	4-35
Remove from Bottom of List	67	RBL	4-45
Remove from Top of List	66	RTL	4-45
Sense Status	DD	SS	4-48
Sense Status RR	9D	SSR	4-48
Shift Left (Fullword) Arithmetic	EF	SLA	4-36
Shift Left (Fullword) Logical	ED	SLL	4-32
Shift Left (Halfword) Arithmetic	CF	SLHA	4-36
Shift Left (Halfword) Logical	CD	SLHL	4-32
Shift Left Logical Short	91	SLLS	4-32
Shift Right (Fullword) Arithmetic	EE	SRA	4-37
Shift Right (Fullword) Logical	EC	SRL	4-33
Shift Right (Halfword) Arithmetic	CE	SRHA	4-37
Shift Right (Halfword) Logical	CC	SRHL	4-33
Shift Right Logical Short	90	SRLS	4-33
Simulate Interrupt	E2	SINT	4-57
Store Byte	D2	STB	4-22
Store Byte RR	92	STBR	4-22
Store Halfword	40	STH	4-5
Store Multiple	D0	STM	4-6
Subtract Halfword	4B	SH	4-10
Subtract Halfword Immediate	CB	SHI	4-10
Subtract Halfword RR	0B	SHR	4-10
Subtract Immediate Short	27	SIS	4-10
Subtract with Carry Halfword	4F	SCH	4-11
Subtract with Carry Halfword RR	0F	SCHR	4-11
Supervisor Call	E1	SVC	4-58
Test Halfword Immediate	C3	THI	4-20
Write Block	D6	WB	4-52
Write Block RR	96	WBR	4-52
Write Data	DA	WD	4-50
Write Data RR	9A	WDR	4-50
Write Halfword	D8	WH	4-54
Write Halfword RR	98	WHR	4-54

APPENDIX 2
INSTRUCTION SUMMARY - NUMERICAL

OP CODE	MNEMONIC	INSTRUCTION	PAGE NO.
01	BALR	Branch and Link RR	4-42
02	BTCR	Branch on True Condition RR	4-39
03	BFCR	Branch on False Condition RR	4-40
04	NHR	AND Halfword RR	4-17
05	CLHR	Compare Logical Halfword RR	4-12
06	OHR	OR Halfword RR	4-18
07	XHR	Exclusive OR Halfword RR	4-19
08	LHR	Load Halfword RR	4-4
09	CHR	Compare Halfword RR	4-13
0A	AHR	Add Halfword RR	4-8
0B	SHR	Subtract Halfword RR	4-10
0C	MHR	Multiply Halfword RR	4-14
0D	DHR	Divide Halfword RR	4-15
0E	ACHR	Add with Carry Halfword RR	4-9
0F	SCHR	Subtract with Carry Halfword RR	4-11
20	BTBS	Branch on True Backward Short	4-39
21	BTFS	Branch on True Forward Short	4-39
22	BFBS	Branch on False Backward Short	4-40
23	BFFS	Branch on False Forward Short	4-40
24	LIS	Load Immediate Short	4-4
25	LCS	Load Complement Short	4-4
26	AIS	Add Immediate Short	4-8
27	SIS	Subtract Immediate Short	4-10
28	LER	Floating-Point Load RR	4-25
29	CER	Floating-Point Compare RR	4-28
2A	AER	Floating-Point Add RR	4-26
2B	SER	Floating-Point Subtract RR	4-27
2C	MER	Floating-Point Multiply RR	4-29
2D	DER	Floating-Point Divide RR	4-30
40	STH	Store Halfword	4-5
41	BAL	Branch and Link	4-42
42	BTC	Branch on True Condition	4-39
43	BFC	Branch on False Condition	4-40
44	NH	AND Halfword	4-17
45	CLH	Compare Logical Halfword	4-12
46	OH	OR Halfword	4-18
47	XH	Exclusive OR Halfword	4-19
48	LH	Load Halfword	4-4
49	CH	Compare Halfword	4-13
4A	AH	Add Halfword	4-8
4B	SH	Subtract Halfword	4-10
4C	MH	Multiply Halfword	4-14
4D	DH	Divide Halfword	4-15
4E	ACH	Add with Carry Halfword	4-9
4F	SCH	Subtract with Carry Halfword	4-11
60	STE	Floating-Point Store	4-25
61	AHM	Add Halfword Memory	4-8
64	ATL	Add to Top of List	4-44
65	ABL	Add to Bottom of List	4-44

OP CODE	MNEMONIC	INSTRUCTION	PAGE NO.
66	RTL	Remove from Top of List	4-45
67	RBL	Remove from Bottom of List	4-45
68	LE	Floating-Point Load	4-25
69	CE	Floating-Point Compare	4-28
6A	AE	Floating-Point Add	4-26
6B	SE	Floating-Point Subtract	4-27
6C	ME	Floating-Point Multiply	4-29
6D	DE	Floating-Point Divide	4-30
90	SRLS	Shift Right Logical Short	4-33
91	SLLS	Shift Left Logical Short	4-32
92	STBR	Store Byte RR	4-22
93	LBR	Load Byte RR	4-22
94	EXBR	Exchange Byte RR	4-23
95	EPSR	Exchange Program Status RR	4-57
96	WBR	Write Block RR	4-52
97	RBR	Read Block RR	4-51
98	WHR	Write Halfword RR	4-54
99	RHR	Read Halfword RR	4-53
9A	WDR	Write Data RR	4-50
9B	RDR	Read Data RR	4-49
9C	MHUR	Multiply Halfword Unsigned RR	4-14
9D	SSR	Sense Status RR	4-48
9E	OCR	Output Command RR	4-49
9F	AIR	Acknowledge Interrupt RR	4-47
C0	BXH	Branch on Index High	4-41
C1	BXLE	Branch on Index Low or Equal	4-41
C2	LPSW	Load Program Status Word	4-56
C3	THI	Test Halfword Immediate	4-20
C4	NHI	AND Halfword Immediate	4-17
C5	CLHI	Compare Logical Halfword Immediate	4-12
C6	OHI	OR Halfword Immediate	4-18
C7	XHI	Exclusive OR Halfword Immediate	4-19
C8	LHI	Load Halfword Immediate	4-4
C9	CHI	Compare Halfword Immediate	4-13
CA	AHI	Add Halfword Immediate	4-8
CB	SHI	Subtract Halfword Immediate	4-10
CC	SRHL	Shift Right (Halfword) Logical	4-33
CD	SLHL	Shift Left (Halfword) Logical	4-32
CE	SRHA	Shift Right (Halfword) Arithmetic	4-37
CF	SLHA	Shift Left (Halfword) Arithmetic	4-36
D0	STM	Store Multiple	4-6
D1	LM	Load Multiple	4-5
D2	STB	Store Byte	4-22
D3	LB	Load Byte	4-22
D4	CLB	Compare Logical Byte	4-23
D5	AL	Auto Load	4-55
D6	WB	Write Block	4-52
D7	RB	Read Block	4-51
D8	WH	Write Halfword	4-54
D9	RH	Read Halfword	4-53
DA	WD	Write Data	4-50
DB	RD	Read Data	4-49
DC	MHU	Multiply Halfword Unsigned	4-14
DD	SS	Sense Status	4-48

OP CODE	MNEMONIC	INSTRUCTION	PAGE NO.
DE	OC	Output Command	4-49
DF	AI	Acknowledge Interrupt	4-47
E1	SVC	Supervisor Call	4-58
E2	SINT	Simulate Interrupt	4-57
EA	RRL	Rotate Right Logical	4-35
EB	RLL	Rotate Left Logical	4-34
EC	SRL	Shift Right (Fullword) Logical	4-33
ED	SLL	Shift Left (Fullword) Logical	4-32
EE	SRA	Shift Right (Fullword) Arithmetic	4-37
EF	SLA	Shift Left (Fullword) Arithmetic	4-36

APPENDIX 3
EXTENDED BRANCH MNEMONICS

INSTRUCTION	OP CODE (M1)	MNEMONIC	OPERANDS
Branch on Carry	428	BC	A(X2)
Branch on Carry RR	028	BCR	R2
Branch on No Carry	438	BNC	A(X2)
Branch on No Carry RR	038	BNCR	R2
Branch on Equal	433	BE	A(X2)
Branch on Equal RR	033	BER	R2
Branch on Not Equal	423	BNE	A(X2)
Branch on Not Equal RR	023	BNER	R2
Branch on Low	428	BL	A(X2)
Branch on Low RR	028	BLR	R2
Branch on Not Low	438	BNL	A(X2)
Branch on Not Low RR	038	BNLR	R2
Branch on Minus	421	BM	A(X2)
Branch on Minus RR	021	BMR	R2
Branch on Not Minus	431	BNM	A(X2)
Branch on Not Minus	031	BNMR	R2
Branch on Plus	422	BP	A(X2)
Branch on Plus RR	022	BPR	R2
Branch on Not Plus	432	BNP	A(X2)
Branch on Not Plus RR	032	BNPR	R2
Branch on Overflow	424	BO	A(X2)
Branch on Overflow RR	024	BOR	R2
Branch Unconditional	430	B	A(X2)
Branch Unconditional RR	030	BR	R2
Branch on Zero	433	BZ	A(X2)
Branch on Zero RR	033	BZR	R2
Branch on Not Zero	423	BNZ	A(X2)
Branch on Not Zero RR	023	BNZR	R2
No Operation	420	NOP	
No Operation RR	020	NOPR	
Branch on Carry Short	208	BCS	A (Backward Reference)
	218	BCS	A (Forward Reference)
Branch on No Carry Short	228	BNCS	A (Backward Reference)
	238	BNCS	A (Forward Reference)
Branch on Equal Short	223	BES	A (Backward Reference)
	233	BES	A (Forward Reference)
Branch on Not Equal Short	203	BNES	A (Backward Reference)
	213	BNES	A (Forward Reference)

INSTRUCTION	OP CODE (M1)	MNEMONIC	OPERANDS
Branch on Low Short	208	BLS	A (Backward Reference)
	218	BLS	A (Forward Reference)
Branch on Not Low Short	228	BNLS	A (Backward Reference)
	238	BNLS	A (Forward Reference)
Branch on Minus Short	201	BMS	A (Backward Reference)
	211	BMS	A (Forward Reference)
Branch on Not Minus Short	221	BNMS	A (Backward Reference)
	231	BNMS	A (Forward Reference)
Branch on Plus Short	202	BPS	A (Backward Reference)
	212	BPS	A (Forward Reference)
Branch on Not Plus Short	222	BNPS	A (Backward Reference)
	232	BNPS	A (Forward Reference)
Branch on Overflow Short	204	BOS	A (Backward Reference)
	214	BOS	A (Forward Reference)
Branch Unconditional Short	220	BS	A (Backward Reference)
	230	BS	A (Forward Reference)
Branch on Zero Short	223	BZS	A (Backward Reference)
	233	BZS	A (Forward Reference)
Branch on Not Zero Short	203	BNZS	A (Backward Reference)
	213	BNZS	A (Forward Reference)

APPENDIX 4
OP CODE MAP

MSD → / LSD ↓	0	2	4	6	9	C	D	E
0		BTBS	STH	STE	SRLS	BXH	STM	
1	BALR	BTFS	BAL	AHM	SLLS	BXLE	LM	SVC
2	BTCR	BFBS	BTC		STBR	LPSW[P]	STB	SINT[P]
3	BFCR	BFFS	BFC		LBR	THI	LB	
4	NHR	LIS	NH	ATL	EXBR	NHI	CLB	
5	CLHR	LCS	CLH	ABL	EPSR [P]	CLHI	AL [P]	
6	OHR	AIS	OH	RTL	WBR [P]	OHI	WB [P]	
7	XHR	SIS	XH	RBL	RBR [P]	XHI	RB [P]	
8	LHR	LER	LH	LE	WHR [P]	LHI	WH [P]	
9	CHR	CER	CH	CE	RHR [P]	CHI	RH [P]	
A	AHR	AER	AH	AE	WDR [P]	AHI	WD [P]	RRL
B	SHR	SER	SH	SE	RDR [P]	SHI	RD [P]	RLL
C	MHR	MER	MH	ME	MHUR	SRHL	MHU	SRL
D	DHR	DER	DH	DE	SSR [P]	SLHL	SS [P]	SLL
E	ACHR		ACH		OCR [P]	SRHA	OC [P]	SRA
F	SCHR		SCH		AIR [P]	SLHA	AI [P]	SLA
	RR	RR	RX	RX	RR	RS	RX	RS

P = Privileged Instructions

APPENDIX 5
INSTRUCTION EXECUTION TIMES

Instr.	RR or SF	RS	RS Indexed	RX	Comments
ABL				5/13/13	OVF/NORM/WRAP
ACH	1.25			3.25	
AE	21.25/25.25/34			22/26/34.75	MIN/AVE/MAX
AH	1.0	2.25	3.25	3.25	
AHM				4.0	
AI	2.75			5.0	
AIS	1.5				
AL				6.5 + 4.5n	n=no. of bytes
ATL				5/11.5/11.75	OVF/NORM/WRAP
BAL	1.5			2.5	
BFBS	1.5/3.0				No BR/BR
BFC	1.5			2.75/3.0	No BR/BR
BFFS	1.5/3.25				No BR/BR
BTBS	1.25/3.25				No BR/BR
BTC	1.5			2.75/3.0	No BR/BR
BTFS	1.25/3.5				No BR/BR
BXH		5.0/4.5	6.0/5.5		No BR/BR
BXLE		5.0/5.0	6.0/6.0		No BR/BR
CE	10.75/12.75/9.75			11.5/13.5/10.5	+,+/-,-/+,-
CH	2.0/2.25	3.0/3.25	4.0/4.25	4.0/4.25	Signs alike/Signs differ
CLB				3.75	
CLH	1.0	2.25	3.25	3.25	
DE	105.5/108.25/116.25			106.25/109/117	MIN/AVE/MAX
DH	10.25/12/10.25/10.5			12.25/14/12.25/12.5	++/+-/-+/--
EPSR	3.25				
EXBR	1.0				
LB	1.0			3.25	
LCS	1.5				
LE	12.5/13.25/19.25/27.5			13.25/14/20/28.25	0/BEST/AVE/WORST
LH	1.0	2.0	3.0	3.0	
LIS	1.25				
LM				4.5 + 1.5n	n=no. of regs.
LPSW		5.25	6.25		
ME	60.5/74.25/91.25			61/75/92	MIN/AVE/MAX
MH	8/9/8.75/8.25			10/11/10.75/10.25	++/+-/-+/--
MHU	6.0			8.0	
NH	1.0	2.25	3.25	3.25	
OC	2.25			4.0	
OH	1.0	2.25	3.25	3.25	
RB	5.5+3n			6.5 + 3n	n=no. of bytes
RBL				4.25/11.75/12	EMPTY/NORM/WRAP
RD	2.0			4.5	
RH	2.75/2			5.5/4.75	BYTE/HALFWORD
RLL		3.5+1.0(n-1)	4.5+1.0(n-1)		n=no. of shifts
RRL		3.5+1.0(n-1)	4.5+1.0(n-1)		n=no. of shifts
RTL				4.25/13/13	EMPTY/NORM/WRAP

Instr.	RR or SF	RS	RS Indexed	RX	Comments
SCH	1.25			3.25	
SE	22/30.5/29.25			22.5/31.5/39.75	MIN/AVE/MAX
SH	1.0	2.25	3.25	3.25	
SINT	See I/O Channel Timing				
SIS	1.5				
SLA		3.5+.25(n-1)	4.5+.25(n-1)		n=no. of shifts
SLHA		3.5+.25(n-1)	4.5+.25(n-1)		n=no. of shifts
SLHL		2.75+.25(n-1)	3.75+.25(n-1)		n=no. of shifts
SLL		3.0+.25(n-1)	4.0+.25(n-1)		n=no. of shifts
SLLS	2.0+.25(n-1)				n=no. of shifts
SRA		3.25+.25(n-1)	4.25+.25(n-1)		n=no. of shifts
SRHA		3.25+.25(n-1)	4.25+.25(n-1)		n=no. of shifts
SRHL		2.75+.25(n-1)	3.75+.25(n-1)		n=no. of shifts
SRL		3.0+.25(n-1)	4.0+.25(n-1)		n=no. of shifts
SRLS	2.0+.25(n-1)				n=no. of shifts
SS	2.25			4.5	
STB	2.0			4.0	
STE				7.0	
STH				3.25	
STM				4.5+1.25n	n=no. of regs.
SVC		7.0	8.0		
THI		2.25	3.25		
WB	5.0+3n			5.5+3n	n=no. of bytes
WD	2.25			4.25	
WH	3.5/2.75			4.75/4.0	BYTE/HALFWORD
XH	1.0	2.25	3.25	3.25	

APPENDIX 6
ARITHMETIC REFERENCES

TABLE OF POWERS OF TWO

2^n	n	2^{-n}
1	0	1.0
2	1	0.5
4	2	0.25
8	3	0.125
16	4	0.062 5
32	5	0.031 25
64	6	0.015 625
128	7	0.007 812 5
256	8	0.003 906 25
512	9	0.001 953 125
1 024	10	0.000 976 562 5
2 048	11	0.000 488 281 25
4 096	12	0.000 244 140 625
8 192	13	0.000 122 070 312 5
16 384	14	0.000 061 035 156 25
32 768	15	0.000 030 517 578 125
65 536	16	0.000 015 258 789 062 5
131 072	17	0.000 007 629 394 531 25
262 144	18	0.000 003 814 697 265 625
524 288	19	0.000 001 907 348 632 812 5
1 048 576	20	0.000 000 953 674 316 406 25
2 097 152	21	0.000 000 476 837 158 203 125
4 194 304	22	0.000 000 238 418 579 101 562 5
8 388 608	23	0.000 000 119 209 289 550 781 25
16 777 216	24	0.000 000 059 604 644 775 390 625
33 554 432	25	0.000 000 029 802 322 387 695 312 5
67 108 864	26	0.000 000 014 901 161 193 847 656 25
134 217 728	27	0.000 000 007 450 580 596 923 828 125
268 435 456	28	0.000 000 003 725 290 298 461 914 062 5
536 870 912	29	0.000 000 001 862 645 149 230 957 031 25
1 073 741 824	30	0.000 000 000 931 322 574 615 478 515 625
2 147 483 648	31	0.000 000 000 465 661 287 307 739 257 812 5
4 294 967 296	32	0.000 000 000 232 830 643 653 869 628 906 25
8 589 934 592	33	0.000 000 000 116 415 321 826 934 814 453 125
17 179 869 184	34	0.000 000 000 058 207 660 913 467 407 226 562 5
34 359 738 368	35	0.000 000 000 029 103 830 456 733 703 613 281 25
68 719 476 736	36	0.000 000 000 014 551 915 228 366 851 806 640 625
137 438 953 472	37	0.000 000 000 007 275 957 614 183 425 903 320 312 5
274 877 906 944	38	0.000 000 000 003 637 978 807 091 712 951 660 156 25
549 755 813 888	39	0.000 000 000 001 818 989 403 545 856 475 830 078 125
1 099 511 627 776	40	0.000 000 000 000 909 494 701 772 928 237 915 039 062 5

TABLE OF POWERS OF SIXTEEN

16^n							n
						1	0
						16	1
						256	2
					4	096	3
					65	536	4
				1	048	576	5
				16	777	216	6
				268	435	456	7
			4	294	967	296	8
			68	719	476	736	9
		1	099	511	627	776	10
		17	592	186	044	416	11
		281	474	976	710	656	12
	4	503	599	627	370	496	13
	72	057	594	037	927	936	14
1	152	921	504	606	846	976	15

Decimal Values

HEXADECIMAL TO DECIMAL CONVERSION TABLE

BYTE				BYTE			
HEX	DEC	HEX	DEC	HEX	DEC	HEX	DEC
0	0	0	0	0	0	0	0
1	4,096	1	256	1	16	1	1
2	8,192	2	512	2	32	2	2
3	12,288	3	768	3	48	3	3
4	16,384	4	1,024	4	64	4	4
5	20,480	5	1,280	5	80	5	5
6	24,576	6	1,536	6	96	6	6
7	28,672	7	1,792	7	112	7	7
8	32,768	8	2,048	8	128	8	8
9	36,864	9	2,304	9	144	9	9
A	40,960	A	2,560	A	160	A	10
B	45,056	B	2,816	B	176	B	11
C	49,152	C	3,072	C	192	C	12
D	53,248	D	3,328	D	208	D	13
E	57,344	E	3,584	E	224	E	14
F	61,440	F	3,840	F	240	F	15

HEXADECIMAL ADDITION TABLE

	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	
1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	10	1
2	3	4	5	6	7	8	9	A	B	C	D	E	F	10	11	2
3	4	5	6	7	8	9	A	B	C	D	E	F	10	11	12	3
4	5	6	7	8	9	A	B	C	D	E	F	10	11	12	13	4
5	6	7	8	9	A	B	C	D	E	F	10	11	12	13	14	5
6	7	8	9	A	B	C	D	E	F	10	11	12	13	14	15	6
7	8	9	A	B	C	D	E	F	10	11	12	13	14	15	16	7
8	9	A	B	C	D	E	F	10	11	12	13	14	15	16	17	8
9	A	B	C	D	E	F	10	11	12	13	14	15	16	17	18	9
A	B	C	D	E	F	10	11	12	13	14	15	16	17	18	19	A
B	C	D	E	F	10	11	12	13	14	15	16	17	18	19	1A	B
C	D	E	F	10	11	12	13	14	15	16	17	18	19	1A	1B	C
D	E	F	10	11	12	13	14	15	16	17	18	19	1A	1B	1C	D
E	F	10	11	12	13	14	15	16	17	18	19	1A	1B	1C	1D	E
F	10	11	12	13	14	15	16	17	18	19	1A	1B	1C	1D	1E	F
	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	

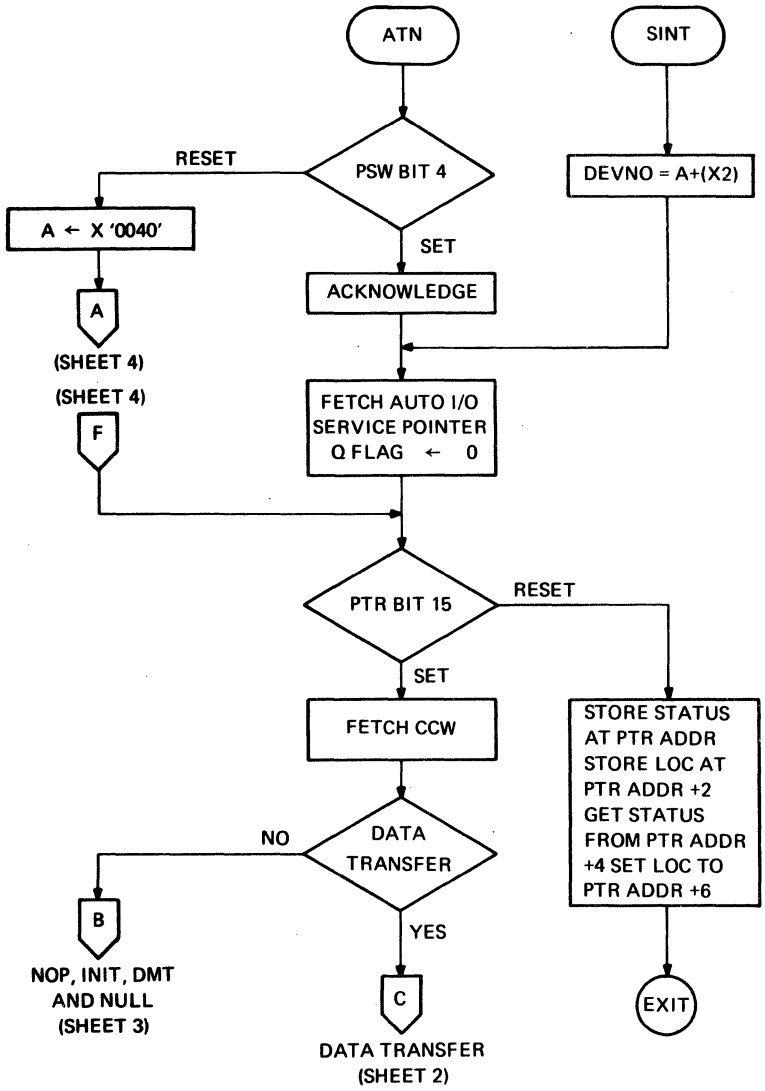
HEXADECIMAL MULTIPLICATION TABLE

	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	
1	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	1
2	2	4	6	8	A	C	E	10	12	14	16	18	1A	1C	1E	2
3	3	6	9	C	F	12	15	18	1B	1E	21	24	27	2A	2D	3
4	4	8	C	10	14	18	1C	20	24	28	2C	30	34	38	3C	4
5	5	A	F	14	19	1E	23	28	2D	32	37	3C	41	46	4B	5
6	6	C	12	18	1E	24	2A	30	36	3C	42	48	4E	54	5A	6
7	7	E	15	1C	23	2A	31	38	3F	46	4D	54	5B	62	69	7
8	8	10	18	20	28	30	38	40	48	50	58	60	68	70	78	8
9	9	12	1B	24	2D	36	3F	48	51	5A	63	6C	75	7E	87	9
A	A	14	1E	28	32	3C	46	50	5A	64	6E	78	82	8C	96	A
B	B	16	21	2C	37	42	4D	58	63	6E	79	84	8F	9A	A5	B
C	C	18	24	30	3C	48	54	60	6C	78	84	90	9C	A8	B4	C
D	D	1A	27	34	41	4E	5B	68	75	82	8F	9C	A9	B6	C3	D
E	E	1C	2A	38	46	54	62	70	7E	8C	9A	A8	B6	C4	D2	E
F	F	1E	2D	3C	4B	5A	69	78	87	96	A5	B4	C3	D2	E1	F
	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	

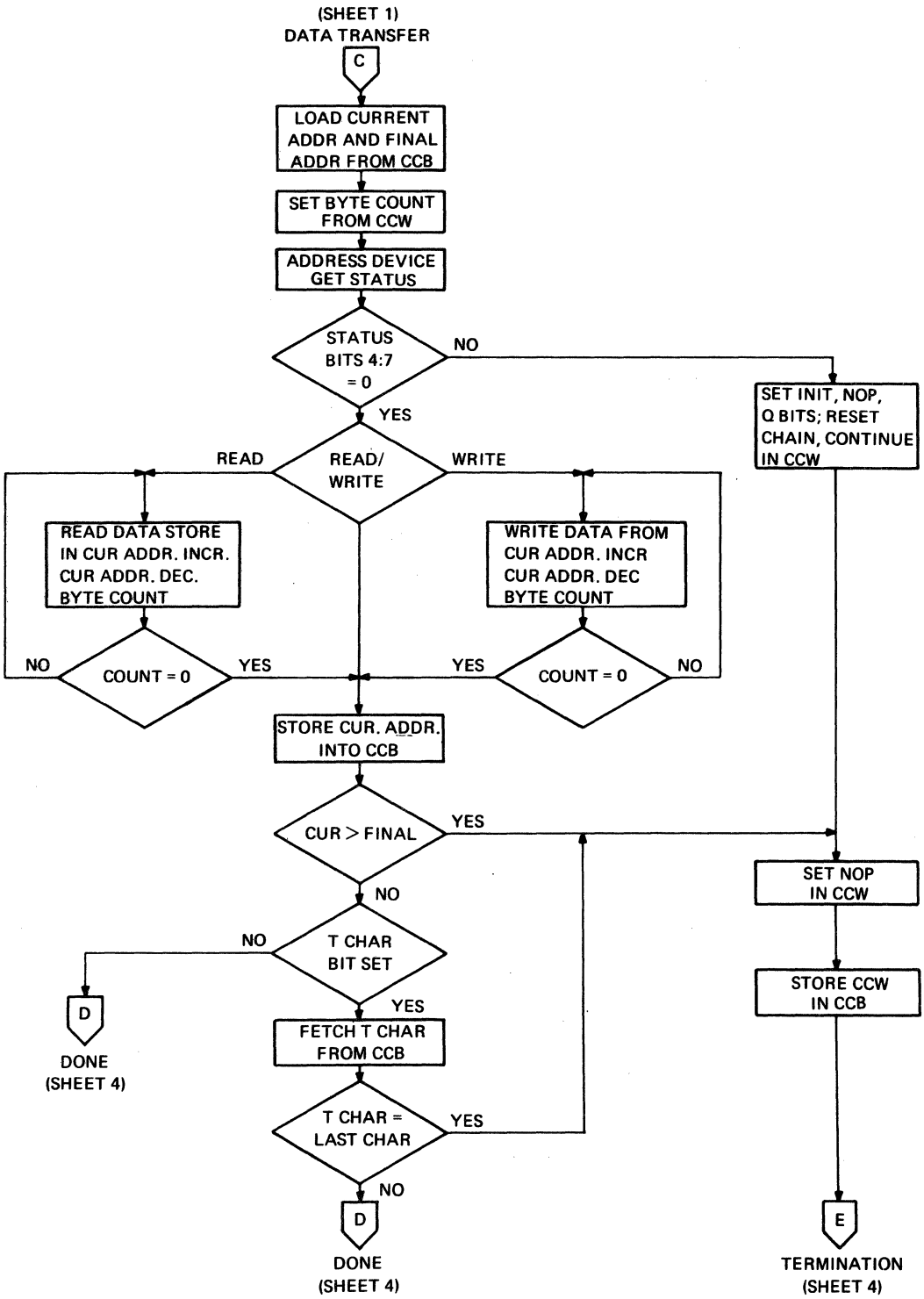
TABLE OF MATHEMATICAL CONSTANTS

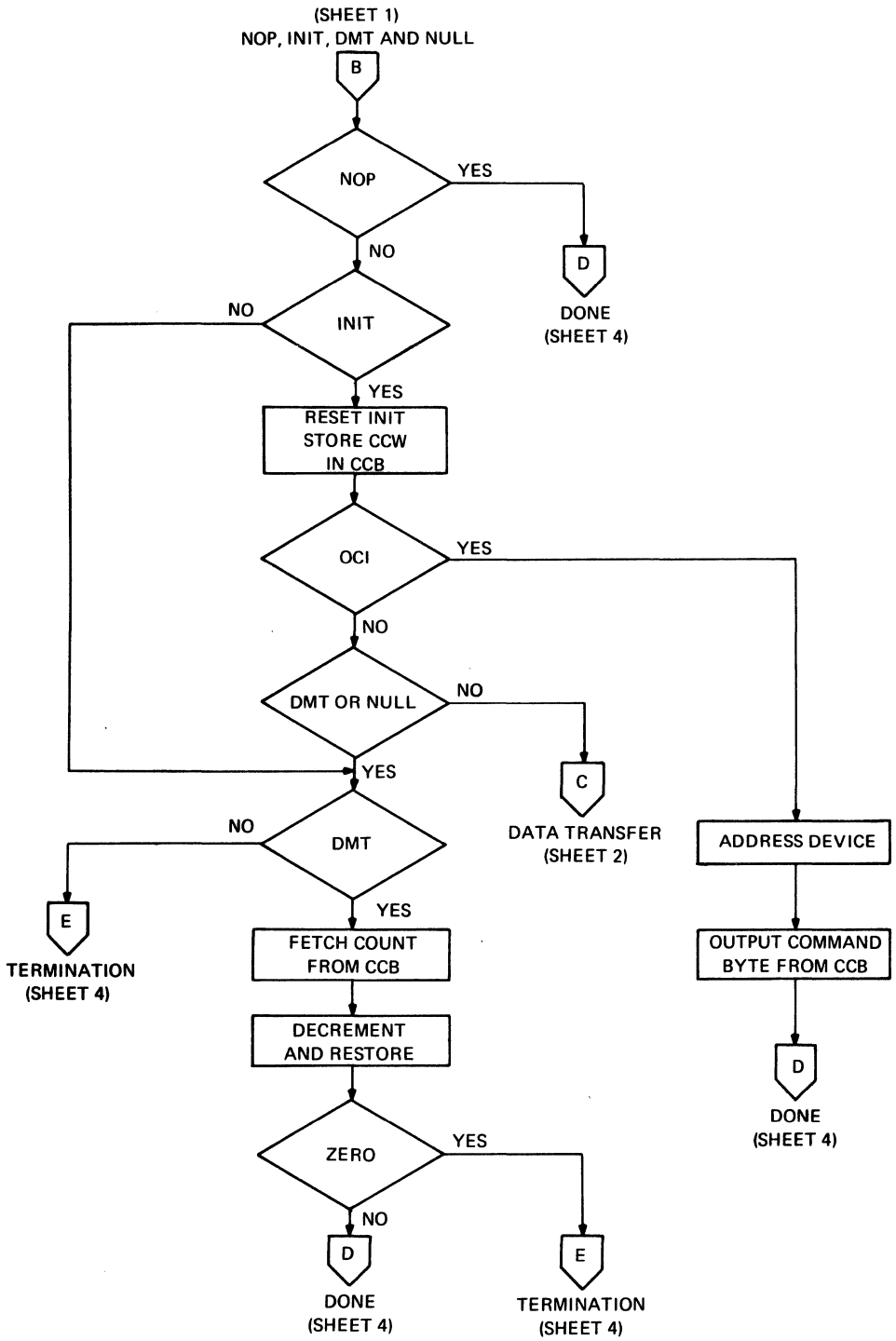
Constant	Decimal Value	Hexadecimal Value
π	3.14159 26535 89793	3.243F 6A89
π^{-1}	0.31830 98861 83790	0.517C C1B7
$\sqrt{\pi}$	1.77245 38509 05516	1.C5BF 891C
$\mathrm{Ln}\,\pi$	1.14472 98858 49400	1.250D 048F
e	2.71828 18284 59045	2.B7E1 5163
e^{-1}	0.36787 94411 71442	0.5E2D 58D9
$\sqrt{e}$	1.64872 12707 00128	1.A612 98E2
$\log_{10}e$	0.43429 44819 03252	0.6F2D EC55
$\log_{2}e$	1.44269 50408 88963	1.7154 7653
γ	0.57721 56649 01533	0.93C4 67E4
$\mathrm{Ln}\,\gamma$	-0.54953 93129 81645	-0.8CAE 9BC1
$\sqrt{2}$	1.41421 35623 73095	1.6A09 E668
Ln2	0.69314 71805 59945	0.B172 17F8
$\log_{10}2$	0.30102 99956 63981	0.4D10 4D42
$\sqrt{10}$	3.16227 76601 68379	3.298B 075C
Ln10	2.30258 50929 94046	2.4D76 3777

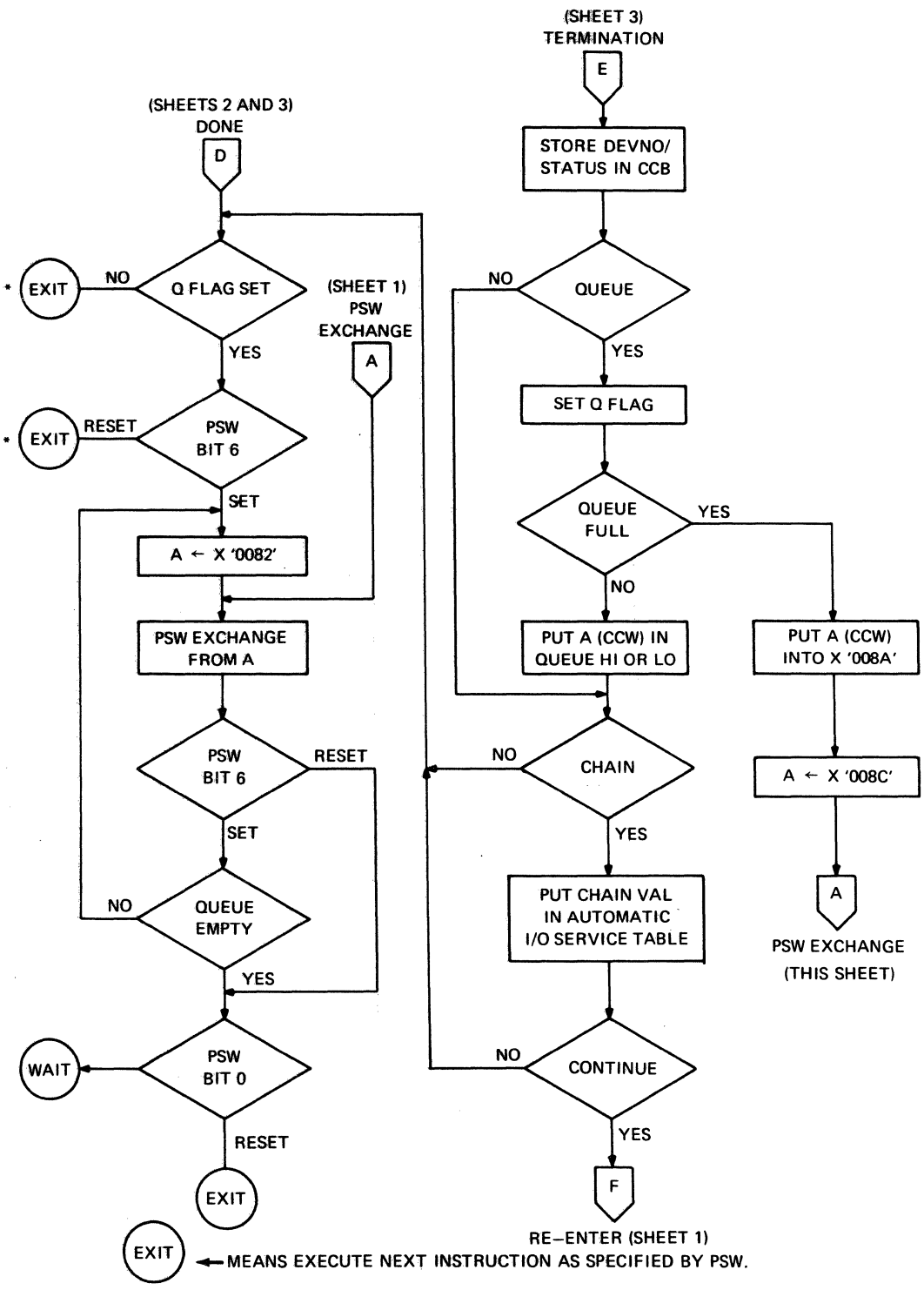
APPENDIX 7
I/O AUTOMATIC SERVICE OPERATION AND TIMING DATA



Sheet 1 of 4


(SHEET 1)
DATA TRANSFER
C
LOAD CURRENT ADDR AND FINAL ADDR FROM CCB
SET BYTE COUNT FROM CCW
ADDRESS DEVICE GET STATUS
STATUS BITS 4:7 = 0
NO
YES
SET INIT, NOP, Q BITS; RESET CHAIN, CONTINUE IN CCW
READ/ WRITE
READ
WRITE
READ DATA STORE IN CUR ADDR. INCR. CUR ADDR. DEC. BYTE COUNT
WRITE DATA FROM CUR ADDR. INCR CUR ADDR. DEC BYTE COUNT
NO
COUNT = 0
YES
YES
COUNT = 0
NO
STORE CUR. ADDR. INTO CCB
CUR > FINAL
YES
NO
SET NOP IN CCW
NO
T CHAR BIT SET
YES
STORE CCW IN CCB
D
DONE
(SHEET 4)
FETCH T CHAR FROM CCB
T CHAR = LAST CHAR
YES
NO
D
DONE
(SHEET 4)
E
TERMINATION
(SHEET 4)


(SHEET 1)
NOP, INIT, DMT AND NULL
B
NOP
YES
NO
D
DONE
(SHEET 4)
NO
INIT
YES
RESET INIT
STORE CCW
IN CCB
OCI
YES
NO
DMT OR NULL
NO
YES
C
DATA TRANSFER
(SHEET 2)
ADDRESS DEVICE
NO
DMT
E
TERMINATION
(SHEET 4)
YES
FETCH COUNT
FROM CCB
OUTPUT COMMAND
BYTE FROM CCB
DECREMENT
AND RESTORE
D
DONE
(SHEET 4)
ZERO
YES
NO
D
DONE
(SHEET 4)
E
TERMINATION
(SHEET 4)



* - IF INTERRUPT SIGNAL IS PRESENT, FIRMWARE WILL SERVICE IT BEFORE EXECUTION OF THE NEXT INSTRUCTION.

AUTOMATIC INPUT/OUTPUT
INTERRUPT SERVICE TIMES

		NOP	NULL	DMT	OCI	READ	WRITE	BAD STATUS
	BASE	12.00	15.25	17.00	17.25	18.00+2.5n	18.75+2.25n	20.00
	INIT	—	3.50	3.50	—	4.75	4.75	4.75
	TCHAR no match	—	—	—	—	2.75	2.75	—
1	TCHAR match	—	—	—	—	6.5	6.5	—
1	COUNT=0 or CUR=final	—	—	1.75	—	3.75	3.75	—
2	QUEUE high	—	11.00	11.00	—	11.00	11.00	—
2	QUEUE low	—	12.75	12.75	—	12.75	12.75	—
2	CHAIN	—	3.0	3.0	—	3.0	3.0	—
2	CONTINUE	—	1.25+	1.25+	—	1.25+	1.25+	—
2	QSVC INT.	3.5	3.5	3.5	3.5	3.5	3.5	3.5

1. Reason for termination
2. Termination Procedure

All times are given in microseconds. To determine the execution time of a particular interrupt, add to the base time, the time for each pertinent option. For example: a Write of one character using a Termination Character (TCHAR) with no match takes 21.00 (BASE)
plus 2.75 (TCHAR no match)

23.75 microseconds

SINT Execution time is the same. Add 1.6 microseconds if indexed.
On Read and Write times, n is the number of bytes transferred per interrupt.

Normal Interrupt Latency Time: 4 microseconds

Non-Interruptable Instructions: Load/ Store Multiple, Read/Write Block, Autoload, Automatic I/O Service, Supervisor Call, Remove from Top/Bottom of List, Add to Top/Bottom of List

Machine Malfunction Interrupt: 7.75us

Normal I/O Interrupt: 7.25us

Immediate I/O Interrupt: 7.75us

	One HW	Burst
Data Channel Read	7.5us	4.5n
Data Channel Write	7.75us	4.75n

n = halfwords

These times assume the standard 1.00 microsecond core memory with no interference from a Selector Channel or any other device on the memory bus.

APPENDIX 8
I/O REFERENCES

CONTROL CONSOLE STATUS AND COMMAND BYTE DATA
(HEX ADDRESS 01)

BIT NUMBER	0	1	2	3	4	5	6	7
STATUS BYTE	MODE				REGISTER DISPLAY			
COMMAND BYTE	NORM	INC						

STATUS:

MODE	SINGLE STEP	0	1	0	0	x	x	x	x
	RUN	1	0	0	0	x	x	x	x
	HALT	1	1	0	0	x	x	x	x
	MEM WRITE	0	0	0	1	0	0	0	0
	MEM READ	0	0	1	0	0	0	0	0
	ADRS	0	0	1	1	0	0	0	0
	OFF					0	0	0	0
REGISTER DISPLAY	INST					0	0	1	0
	PSW					0	1	0	0
	R0, R1					1	0	0	0
	R2, R3					1	0	0	1
	R4, R5					1	0	1	0
	R6, R7					1	0	1	1
	R8, R9					1	1	0	0
	RA, RB					1	1	0	1
	RC, RD					1	1	1	0
	RE, RF					1	1	1	1

COMMAND:

NORM — In the Normal Mode, Byte 0 of the Register Display or Data/Address switches is accessed each time an I/O operation is directed to the Control Console.

INC — In the Incremental Mode, subsequent I/O operations access subsequent bytes of the Register Display or Data/Address switches.

TELETYPEWRITER/ASC11/HEX CONVERSION TABLE

HEX (MSD) →					0	1	2	3	4	5	6	7
(LSD) ↓	Teletypewriter Tape Channels →			8	DEPENDS UPON PARITY*							
				7	0	0	0	0	1	1	1	1
				6	0	0	1	1	0	0	1	1
				5	0	1	0	1	0	1	0	1
	4	3	2	1								
0	0	0	0	0	NULL	DC_o	SPACE	0	@	P		
1	0	0	0	1	SOM	X-ON	!	1	A	Q		
2	0	0	1	0	EOA	TAPE ON	"	2	B	R		
3	0	0	1	1	EOM	X-OFF	#	3	C	S		
4	0	1	0	0	EOT	TAPE OFF	$	4	D	T		
5	0	1	0	1	WRU	ERR	%	5	E	U		
6	0	1	1	0	RU	SYNC	&	6	F	V		
7	0	1	1	1	BELL	LEM	'	7	G	W		
8	1	0	0	0	FE_o	S_0	(	8	H	X		
9	1	0	0	1	HT/SK	S_1	)	9	I	Y		
A	1	0	1	0	LF	S_2	*	:	J	Z		
B	1	0	1	1	VT	S_3	+	;	K	[		
C	1	1	0	0	FF	S_4	,	<	L	\		ACK
D	1	1	0	1	CR	S_5	-	=	M	]		ALT. MODE
E	1	1	1	0	SO	S_6	.	>	N	↑		ESC
F	1	1	1	1	SI	S_7	/	?	O	←		DEL

*Parity bit adjusted for even parity (even number of 1's) on input from keyboard. Parity bit is ignored on output to printer.

ASCII CARD CODE CONVERSION TABLE

GRAPHIC	7-BIT ASCII CODE	CARD CODE	GRAPHIC	7-BIT ASCII CODE	CARD CODE
SPACE	20	BLANK	@	40	8-4
!	21	12-8-7	A	41	12-1
"	22	8-7	B	42	12-2
#	23	8-3	C	43	12-3
$	24	11-8-3	D	44	12-4
%	25	0-8-4	E	45	12-5
&	26	12	F	46	12-6
'	27	8-5	G	47	12-7
(	28	12-8-5	H	48	12-8
)	29	11-8-5	I	49	12-9
*	2A	11-8-4	J	4A	11-1
+	2B	12-8-6	K	4B	11-2
,	2C	0-8-3	L	4C	11-3
-	2D	11	M	4D	11-4
.	2E	12-8-3	N	4E	11-5
/	2F	0-1	O	4F	11-6
0	30	0	P	50	11-7
1	31	1	Q	51	11-8
2	32	2	R	52	11-9
3	33	3	S	53	0-2
4	34	4	T	54	0-3
5	35	5	U	55	0-4
6	36	6	V	56	0-5
7	37	7	W	57	0-6
8	38	8	X	58	0-7
9	39	9	Y	59	0-8
:	3A	8-2	Z	5A	0-9
;	3B	11-8-6	[	5B	12-8-2
<	3C	12-8-4	\	5C	11-8-1
=	3D	8-6	]	5D	11-8-2
>	3E	0-8-6	↑	5E	11-8-7
?	3F	0-8-7	←	5F	0-8-5

EIGHT-LINE INTERRUPT MODULE STATUS
AND COMMAND BYTE DATA
(HEX ADDRESS 20-27)

BIT NUMBER	0	1	2	3	4	5	6	7
STATUS BYTE	0	0	0	0	0	0	0	0
COMMAND BYTE	DISABLE	ENABLE	RESET	SET	CLEAR	GCMD0	GCMD1	*

*Bit not used.

The status byte is always zero.

DIS - Disable DEVICE INTERRUPT (but allow queueing)

ENAB - Enable DEVICE INTERRUPT

RESET - Establish Reset Mode, one Write Data selectively reset interrupt lines.

SET - Establish Set Mode, one Write Data selectively set interrupt lines.

CLEAR - Clear all pending interrupts

GCMDO, GCMD1 - These Command bits may be optionally gated-out to a user's own equipment. Their function, if used, is dependent upon this external equipment.

Any command in which Bit 2 and 3 = 0 places the module in the Load Mask Mode. In the Load Mask Mode, one Write Data is required to set the mask.

INITIALIZATION - Disables, interrupts, clears all commands, clears all pending interrupts, and places the module in the Load Mask Mode.

AUTOMATIC MEMORY PROTECT
STATUS AND COMMAND BYTE DATA
(HEX ADDRESS AE)

BIT NUMBER	0	1	2	3	4	5	6	7
STATUS BYTE	*	*	P ON	PWF	*	EX	*	*
COMMAND BYTE	DISARM	ARM	P ON	P OFF	*	*	*	*

* Bit not used.

STATUS

P ON - This Status bit is set when memory protection is enabled.

PWF - The Protected Write Flag is set when an attempt has been made to write into a protected area of memory. PWF is reset only by an Output Command (OC or OCR), an Acknowledge Interrupt (AI or AIR), or the INITIALIZE pushbutton.

EX - The Examine bit is set whenever the PWF bit is set.

COMMAND

DISARM - This Command bit disables the device interrupt feature and prevents interrupts from being queued.

ARM - This Command bit enables a device interrupt to occur when an attempt is made to write into a protected area of memory.

P ON - This Command bit enables memory to be protected as per the protection pattern.

P OFF - This Command bit overrides all memory protection.

PROTECT PATTERN - After an output command (any output command), the protect pattern may be set up with consecutive Write Data instructions. If more than eight Write Data instructions are issued, then the protect pattern will wrap around. The ninth WD instruction will change Blocks 0-7 etc.

INITIALI-ZATION - Dis-arm interrupts, clears PON and PWF flip-flops, and leaves the protect pattern un-changed.

SELECTOR CHANNEL STATUS AND
COMMAND BYTE DATA
(HEX ADDRESS F0)

BIT NUMBER	0	1	2	3	4	5	6	7
STATUS BYTE					BSY			
COMMAND BYTE			READ	GO	STOP			

BSY This bit is set when the Selector Channel is in the process of transferring data.

READ This command changes the mode of the Selector Channel from Write to Read. In the Read Mode, data is transmitted from the active device on the Selector Channel and written into core memory. Whenever a Read Data or a Write Data Instruction is issued to the Selector Channel, the Selector Channel is placed in the Write Mode. Each time a READ operation is required, a Read Command must be issued.

GO This command initiates a data transmission. This command can be issued at the same time the Read/Write Mode is established.

STOP This command halts any data transmission in process, and initializes the Selector Channel for starting a new operation. It should be given when the Selector Channel Terminates.

DEVICE NUMBER

The Selector Channel is normally assigned device number X'F0', but may easily be changed by a minor wiring modification on the Selector Channel device controller board. Refer to the maintenance manual for specific details.

INITIALIZATION

Whenever the Initialize switch on the Processor is depressed, or a Stop Command is issued, the following actions occur:

1. Any data transmission in process is halted and the Stop Mode is effected.
2. The Selector Channel is placed in the Write Mode.
3. The Selector Channel is made idle.
4. The Selector Channel interrupt is reset.

READER COMMENTS

The General Electric Company solicits your comments on publications covering Process Computer equipment. Please explain any "no" responses in the COMMENTS section. Your comments and suggestions become the property of the General Electric Company.

- Name of Manual: _____
- What is your computer application: _____

  _____
- How is this publication used:

  Familiarization ☐ Reference ☐

  Training ☐ Maintenance ☐

  Other (Explain) _____

	YES	NO
• Does this publication meet your requirements	☐	☐
• Is the material:		
1) Presented in clear text	☐	☐
2) Conveniently organized	☐	☐
3) Adequately detailed	☐	☐
4) Adequately illustrated	☐	☐
5) Presented at appropriate technical level	☐	☐

- Please provide specific text references (page number, line, etc.) with your comments.

NAME _____ DATE _____

TITLE _____

COMPANY NAME _____

AND ADDRESS _____

COMMENTS:

No postage necessary if mailed in the U. S. A.

Staple

Communications concerning Technical Publications should be directed to:

Manager, Technical Publications
Utility and Process Automation Systems Operation
2255 West Desert Cove Road
Phoenix, Arizona 85029

Fold Fold

FIRST CLASS
Permit No. 4091
Phoenix, Arizona

BUSINESS REPLY MAIL
NO POSTAGE NECESSARY IF MAILED IN THE UNITED STATES

POSTAGE WILL BE PAID BY...

GENERAL ELECTRIC COMPANY
UTILITY and PROCESS AUTOMATION
SYSTEMS OPERATION
2255 West Desert Cove Road
Phoenix, Arizona 85029

Attention: Technical Publications

Fold Fold

Cut Along Line

Additional Comments: