

**GE-PAC<sup>®</sup> 30/3010**  
COMPUTER SYSTEMS

# REFERENCE MANUAL

GENERAL  ELECTRIC



**GE-PAC<sup>®</sup> 30/3010**  
COMPUTER SYSTEMS

**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.



The GE-PAC 30/3010 Reference Manual, GET-6047,  
obsoletes and replaces the GE-PAC 30-2E Reference  
Manual, PCP-207.





# TABLE OF CONTENTS

## SECTION 1 SYSTEM ARCHITECTURE

1.1 INTRODUCTION .....	1-1
1.2 ELEMENTS OF THE SYSTEM .....	1-2
1.2.1 Processor .....	1-2
1.2.2 Memory .....	1-3
1.2.3 Input/Output .....	1-3
1.2.4 Hexadecimal Notation .....	1-4
1.3 PROCESSOR OPERATION .....	1-4
1.3.1 Program Status Words .....	1-4
1.3.2 Instruction Execution .....	1-5
1.3.3 Core Memory Allocation .....	1-5
1.4 INSTRUCTIONS .....	1-5
1.4.1 Instruction Format .....	1-5
1.4.2 General Register Usage .....	1-9
1.4.3 Storage Addressing .....	1-10
1.5 DATA .....	1-11
1.5.1 Fixed-Point Data .....	1-11
1.5.2 Floating-Point Data .....	1-11
1.5.3 Logical Data .....	1-12

## SECTION 2 INTERRUPT SYSTEM

2.1 INTERRUPT PROCEDURE .....	2-1
2.2 INTERNAL INTERRUPTS .....	2-1
2.2.1 Fixed-Point Divide Fault Interrupt .....	2-1
2.2.2 Floating-Point Arithmetic Fault Interrupt .....	2-1
2.2.3 Machine Malfunction Interrupt .....	2-1
2.2.4 Illegal Instruction .....	2-2
2.2.5 Protect Mode violation .....	2-2
2.2.6 Supervisor Call (SVC) .....	2-2
2.3 INPUT/OUTPUT CONTROL INTERRUPTS .....	2-2
2.3.1 External Interrupt .....	2-2
2.3.2 Immediate Interrupt .....	2-2
2.3.3 I/O Termination Interrupt .....	2-3
2.3.4 Termination Queue Overflow Interrupt .....	2-3
2.4 SPECIAL INTERRUPTS .....	2-3
2.4.1 Console Interrupt .....	2-3
2.4.2 Memory Protect Interrupt .....	2-3

## SECTION 3 INPUT/OUTPUT

3.1 INTRODUCTION .....	3-1
3.2 PROGRAM-CONTROLLED I/O .....	3-1
3.3 INTERRUPT-DRIVEN I/O .....	3-1
3.4 BLOCK I/O .....	3-2

3.5	AUTOMATIC I/O PROGRAMMING .....	3-2
3.5.1	Service Pointer Table .....	3-2
3.5.2	Interrupt Service Block .....	3-2
3.5.3	I/O Termination Queue .....	3-2
3.5.4	General Operation .....	3-2
3.6	ISB FUNCTION WORD .....	3-4
3.6.1	Initialization .....	3-4
3.6.2	I/O Operations .....	3-4
3.6.3	Termination .....	3-5
3.7	EXAMPLE OF AUTOMATIC I/O PROGRAMMING .....	3-6
3.8	SELECTOR CHANNEL I/O .....	3-6
3.8.1	Introduction .....	3-6
3.8.2	Operation .....	3-7
3.8.3	Address Set-Up .....	3-9
3.8.4	Termination .....	3-9
3.8.5	Reading the Final Address .....	3-10

## SECTION 4 INSTRUCTION REPERTOIRE

4.1	INTRODUCTION .....	4-1
4.2	FIXED-POINT LOAD/STORE INSTRUCTIONS .....	4-2
4.2.1	Load Halfword .....	4-2
4.2.2	Load Multiple .....	4-3
4.2.3	Store Halfword .....	4-3
4.2.4	Store Multiple .....	4-3
4.3	FIXED-POINT ARITHMETIC INSTRUCTIONS .....	4-3
4.3.1	Add Halfword .....	4-4
4.3.2	Add with Carry Halfword .....	4-4
4.3.3	Subtract Halfword .....	4-5
4.3.4	Subtract with Carry Halfword .....	4-5
4.3.5	Compare Logical Halfword .....	4-6
4.3.6	Compare Halfword .....	4-6
4.3.7	Multiply Halfword .....	4-7
4.3.8	Multiply Halfword Unsigned .....	4-7
4.3.9	Divide Halfword .....	4-7
4.4	LOGICAL INSTRUCTIONS .....	4-7
4.4.1	AND Halfword .....	4-8
4.4.2	OR Halfword .....	4-8
4.4.3	Exclusive OR Halfword .....	4-9
4.4.4	Test Halfword Immediate .....	4-9
4.5	BYTE HANDLING INSTRUCTIONS .....	4-9
4.5.1	Load Byte .....	4-9
4.5.2	Store Byte .....	4-10
4.5.3	Exchange Byte .....	4-10
4.5.4	Compare Logical Byte .....	4-10



4.6	FLOATING-POINT INSTRUCTIONS .....	4-10
4.6.1	Floating-Point Load .....	4-11
4.6.2	Floating-Point Store .....	4-11
4.6.3	Floating-Point Add .....	4-11
4.6.4	Floating-Point Subtract .....	4-12
4.6.5	Floating-Point Compare .....	4-12
4.6.6	Floating-Point Multiply .....	4-13
4.6.7	Floating-Point Divide .....	4-13
4.7	SHIFT/ROTATE INSTRUCTIONS .....	4-14
4.7.1	Shift Left Logical .....	4-14
4.7.2	Shift Right Logical .....	4-14
4.7.3	Rotate Left Logical .....	4-15
4.7.4	Rotate Right Logical .....	4-15
4.7.5	Shift Left Arithmetic .....	4-16
4.7.6	Shift Right Arithmetic .....	4-16
4.8	BRANCH INSTRUCTIONS .....	4-17
4.8.1	Branch on True Condition .....	4-17
4.8.2	Branch on False Condition .....	4-18
4.8.3	Branch on Index .....	4-18
4.8.4	Branch and Link .....	4-19
4.9	LIST INSTRUCTIONS .....	4-19
4.9.1	Add to Top/Bottom of List .....	4-20
4.9.2	Remove From Top/Bottom of List .....	4-20
4.10	INPUT/OUTPUT INSTRUCTIONS .....	4-21
4.10.1	Acknowledge Interrupt .....	4-21
4.10.2	Sense Status .....	4-22
4.10.3	Output Command .....	4-22
4.10.4	Read Data .....	4-22
4.10.5	Write Data .....	4-23
4.10.6	Read Block .....	4-23
4.10.7	Write Block .....	4-23
4.10.8	Read Halfword .....	4-24
4.10.9	Write Halfword .....	4-24
4.10.10	Autoload .....	4-24
4.11	SYSTEM CONTROL INSTRUCTIONS .....	4-25
4.11.1	Load Program Status Word .....	4-25
4.11.2	Exchange Program Status .....	4-25
4.11.3	Simulate Interrupt .....	4-25
4.11.4	Supervisor Call .....	4-26

## SECTION 5 CONSOLE OPERATING PROCEDURES

5.1	INTRODUCTION .....	5-1
5.2	CONTROL SWITCHES .....	5-1

5.3	MODE CONTROL .....	5-2
5.4	DISPLAY REGISTERS .....	5-2
5.5	OPERATING PROCEDURES .....	5-3
5.5.1	Initialization .....	5-3
5.5.2	Program Loading .....	5-4
5.5.3	Program Execution .....	5-5
5.6	PROGRAMMING CONSIDERATIONS .....	5-6
5.6.1	Display Panel I/O .....	5-6
5.6.2	Console Interrupt .....	5-7
5.6.3	Wait State .....	5-7
5.6.4	Power Fail .....	5-7

APPENDIX 1 INSTRUCTION SUMMARY - ALPHABETICAL

APPENDIX 2 INSTRUCTION SUMMARY - NUMERICAL

APPENDIX 3 EXTENDED BRANCH MNEMONICS

APPENDIX 4 OP CODE MAP

APPENDIX 5 INSTRUCTION EXECUTION TIMES

APPENDIX 6 ARITHMETIC REFERENCES

APPENDIX 7 AUTOMATIC I/O OPERATION AND TIMING DATA

APPENDIX 8 I/O REFERENCES

# SECTION 1 SYSTEM ARCHITECTURE

## 1.1 INTRODUCTION

The elements of the GE-PAC\* 30-2E are the Processor, Memory, I/O Facilities, and Peripheral Devices, as shown in Fig. 1.1.

The GE-PAC 30-2E offers a comprehensive set of 113 instructions as standard equipment which makes the system both easy to program and efficient to operate. The instruction set includes Floating Point and Fixed Point Arithmetic, Logical, Byte Handling, Status Control, and List Handling instructions.

Memory is addressable and alterable to the 8-bit byte level. Memory is expandable from the basic 8,192 bytes to 65,536 bytes. All memory is directly addressable with the primary instructions, no paging or indirect addressing is required.

Sixteen 16-bit General Registers can be used as accumulators or Index Registers. Register-to-Register instructions permit operations between any two of the sixteen General Registers, eliminating redundant loads and stores.

The Protect Mode of the GE-PAC 30-2E enables Memory Protection and detection of Privileged instructions, and can be activated under program control. The protect mode is invaluable in process

control, data communication, and time-sharing operations to guarantee that a running program cannot interfere, for any reason, with the integrity of the system.

The GE-PAC 30-2E also provides a flexible input/output system in addition to conventional means of programmed I/O. In the Automatic I/O Service Mode, the Processor acknowledges all I/O Interrupts and automatically performs much of the overhead prior to activating the Interrupt Service Routine. In conjunction with the Automatic I/O Service, an Interrupt Service Block can perform data transfers and signal counting without interrupting the running program at all until the specified sequence is completed.

### GE-PAC 30-2E System Summary

Instruction Set: 113 instructions, including

A complete set of Floating-Point Instructions

A complete set of Arithmetic and Logical Instructions

Byte Processing Instructions

Both Single-Word and Double-Word Shift Instructions.

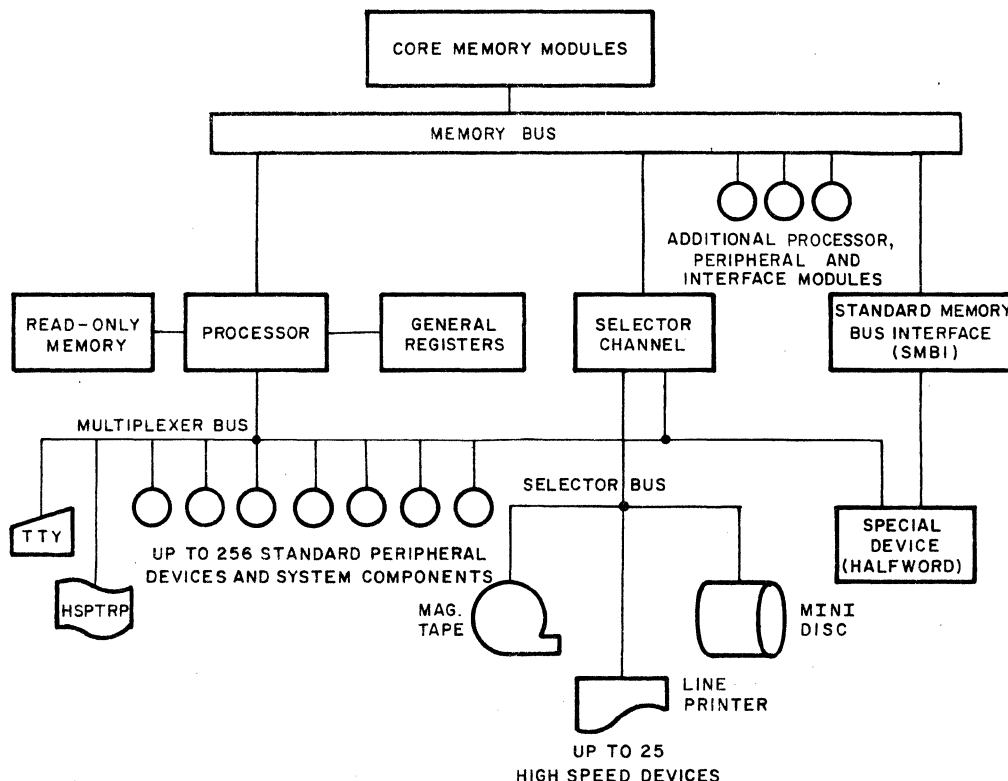


Fig. 1.1 GE-PAC 30-2E Block Diagram

\*Registered Trademark of General Electric Company



List Processing Instructions.

Short Branch Instructions for local branches in addition to Normal Branch Instructions which address any location in memory.

Single-Byte, Double-Byte, and Block I/O Instructions.

Simulate Interrupt Instruction, for sophisticated I/O control.

Supervisor Call Instruction, for communication with an operating system.

Instruction Compatibility;  
GE-PAC 30-1 and 30-2

Instruction Word Lengths;  
16-bit, 32-bit

Data Word Lengths;  
8-bit, 16-bit, 32-bit

User Registers;  
Sixteen 16-bit General Registers. Fifteen General Registers can be used as Index Registers. Eight 32-bit Floating-Point Registers.

Memory Addressing;  
Direct addressing to 64K bytes.

Protect Mode;  
Provides memory protection, plus Privileged instruction detection.

Core Memory;  
1 microsecond cycle time. Basic size 8,192 bytes. Expandable to 65,536 bytes.

Interrupts;  
Identification of up to 256 levels with Automatic Service Mode for rapid interrupt response and minimum program overhead.

Input/Output;  
Multiplexor Bus for up to 256 devices included in the Processor with request/response I/O Logic. Selector Channels for high speed direct-to-memory transfers operating on a cycle-stealing basis (optional). Automatic I/O for signal counting and data transfers without program interrupts.

Display panel programmable as an I/O Device. A full range of peripheral devices and interface modules available for inclusion in the system.

1. Sequencing of instructions in the required order.
2. Arithmetic and logical processing of data.
3. Initiating or controlling communications with external devices.
4. Changing states in response to 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 Bus, See Fig. 1.2.

The Processor operates under the direction of the Control unit which has a pre-wired 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. In GE-PAC 30 publications, such micro-programs are often referred to as Firmware.

The instruction set of the GE-PAC 30-2E is described in Section 4. All memory is directly addressable with the primary instruction word; no paging or indirect addressing is required. The sixteen-bit General Registers can be used as fixed-point accumulators, link registers for subroutine returns, or pointers for program branching. Fifteen of the 16 General Registers 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 Privileged instructions are detected and their execution is prevented. Privileged instructions are I/O instructions and any Control instructions whose execution could change the status of the system. Privileged instructions are discussed in detail in Chapter 4. In the Protect Mode, the execution of any Privileged instruction causes an Illegal Instruction Interrupt.

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

The GE-PAC 30 format for single-precision, floating-point data is identical to that used in the IBM System/360. This format represents numbers in the range from  $5.4 \times 10^{-79}$  to  $7.2 \times 10^{75}$ , with six digits of precision.

## 1.2 ELEMENTS OF THE SYSTEM

### 1.2.1 Processor

The various elements of the system are organized around the Processor, See Fig. 1.1. The Processor contains facilities for:

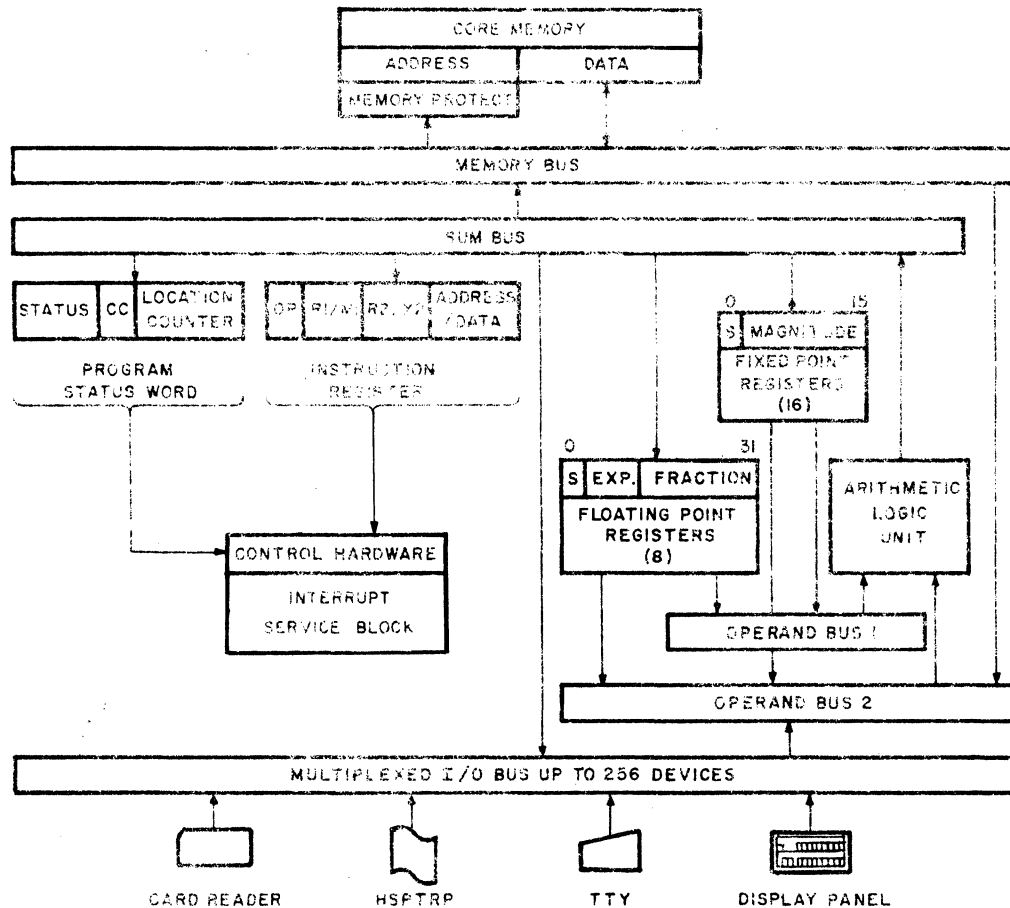


Fig. 1.2 System Block Diagram

## 1.2.2 Memory

The GE-PAC 30-2E system can have from 8K to 64K bytes of core memory, where a byte refers to 8 bits. The byte designation is used to describe GE-PAC 30 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 with 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 is 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 subdivided into two types. The 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 standard instruc-

tions 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 circuitry. 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 Fail. Other privileged write operations are noted where applicable in this manual.

With systems that are equipped with less than 64K bytes of memory, it is possible to address memory at an address greater than that of 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.

## 1.2.3 Input/Output

The principal means of input/output with the GE-PAC 30-2E is the Multiplexor Bus, which connects to the Processor. This Multiplexor Bus, has facility for connecting and addressing up to 256 devices. All communications between the Processor and the Multiplexor

Bus are on a request/response basis. Using the Multiplexor Bus, programs can initiate and control I/O operations in a number of ways. Using normal programmed I/O, a program can interrogate the status of any device, and control the transfer of data to or from a device when the device indicates it is ready.

The Multiplexor Bus provides, in addition, an interrupt facility with which any device can indicate device conditions 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.

Interrupt programming is simplified in the GE-PAC 30-2E 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 then makes the GE-PAC 30-2E I/O compatible with the GE-PAC 30-1 and -2. With the Automatic I/O Service in use, the Interrupt Service Block (ISB) capability of the GE-PAC 30-2E also can be used. The ISB, performs signal counting and data transfers without interrupting the running program. When the ISB completes a specified sequence, a variety of command chaining and interrupt queuing operations can take place.

The Selector Channel is an optional hardware expansion for the GE-PAC 30-2E. Using this option, devices can perform high speed byte-oriented data transfers directly into the GE-PAC 30-2E memory. Once a Selector Channel transfer is initiated, the data is transferred into or out of the memory on a cycle-stealing basis, but the Processor itself is not involved in the operation. Such transfers are initiated under program control by setting up the Selector Channel with starting and final memory addresses over the Multiplexor Bus, and commanding it to start. The transfer then proceeds independently until the block is

completed, or the device terminates prematurely. An interrupt is generated on termination.

Details on all I/O operations with the GE-PAC 30-2E are found in Section 3.

1.2.4 Hexadecimal Notation

In this manual, binary information is expressed using Hexadecimal Notation (base 16). 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 hex digits, and fullword information by eight hexadecimal digits. Table 1.1 lists hexadecimal, binary and decimal equivalents.

1.3 PROCESSOR OPERATION

1.3.1 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 Fig. 1.3.

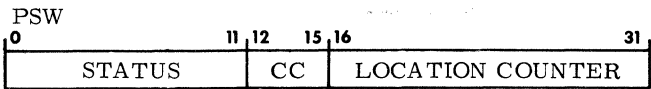


Fig. 1.3 Program Status Word Format

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

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

Hexadecimal	Binary	Decimal
8	1000	8
9	1001	9
A	1010	10
B	1011	11
C	1100	12
D	1101	13
E	1110	14
F	1111	15

Table 1.1 Hexadecimal Notation



The interrupt mechanism of the GE-PAC 30-2E also involves the PSW. A program interrupt is achieved by storing the Current PSW into a reserved area of core memory, referred to as an Old PSW. The Current PSW is then replaced by a new 32-bit quantity appropriate for that interrupt called a New PSW. For example, when an illegal instruction is executed by the GE-PAC 30-2E, an Illegal Instruction Interrupt occurs by storing the Current PSW into memory at location X'0030', and loading the Current PSW with a new value from location X'0034'. The reserved core locations for Old and New PSWs for all interrupts are defined in Section 1.3.3.

The meaning of each bit in the left half of the PSW is explained in Table 1.2, and shown in Fig. 1.4. The particular meaning or function of each bit applies when the bit is a "1".

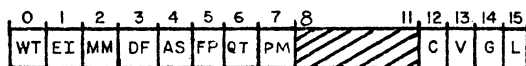


Fig. 1.4 Program Status Bit Format

### 1.3.2 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 sixteen bit address field has the capability of directly addressing the maximum core memory of 64K 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 normal processing of a program, instructions are fetched from the location specified by the Location Counter, the instruction is executed, the Location Counter, is adjusted, and another fetch and execute cycle begins. After instruction execution, (except for Branch or Control instructions) the Location Counter is incremented by 2 if the executed instruction is of the halfword (RR or SF) format, or by 4 if the executed instruction is of the fullword (RX or RS) 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.

### 1.3.3 Core Memory Allocation

The GE-PAC 30-2E Processor requires certain locations in core memory for Floating-Point Registers, register save areas, and interrupt processing. These locations are defined in Table 1.3 and discussed below.

Floating-Point Registers - These registers are used by the Floating-Point Instructions.

Power - Fail Locations - The Register Save Pointer at 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 24-27.

Interrupt PSWs - These locations are reserved for the Old and New PSWs for the various internal and external interrupts. These are discussed further in Section 2.

I/O Termination Parameters - The locations are used in conjunction with termination interrupts from an ISB operation. Refer to Section 3 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.

Service Pointer Table - The 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 3.

## 1.4 INSTRUCTIONS

### 1.4.1 Instruction Format

GE-PAC 30-2E instructions have four formats:

1. Register to Register - RR
2. Short Format - SF
3. Register to Indexed Memory - RX
4. Register to Storage - RS

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

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.
2	MM	Machine Malfunction Interrupt Enable	The Machine Malfunction Enable bit allows an interrupt to occur if the machine is equipped with the Memory Parity Option and a Memory Parity fail occurs. See Section 2.
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	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 3.
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.
6	QT	Queue Termination Interrupt Enable	Queue Termination Interrupt Enable pertains to the ISB, which can be used in conjunction with the Automatic I/O Service as described in Section 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 in the Supervisor Mode.
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 Section 4 for details.
13	V	Overflow	
14	G	Greater than Zero	
15	L	Less than Zero	

Table 1.2 Program Status

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

Table 1.3 Core Memory Allocation



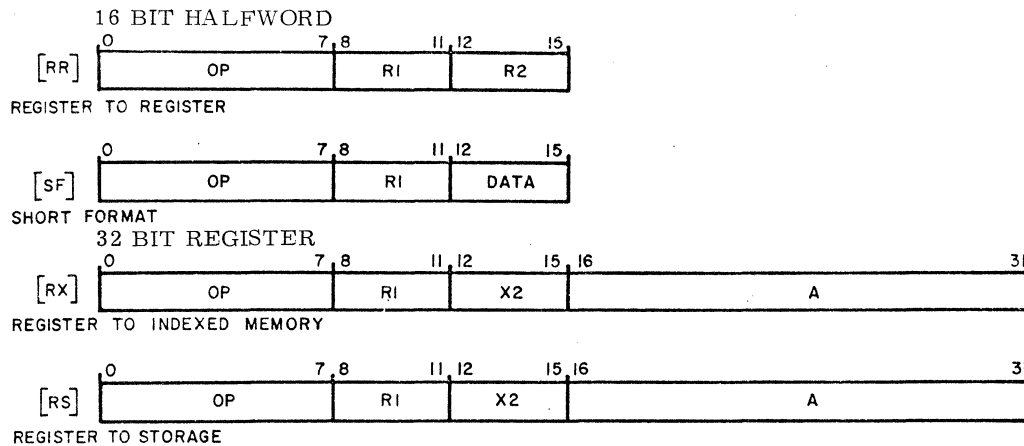


Fig. 1.5 Instruction Word Formats

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 on Fig. 1.5.

The 8-bit OP field in all formats specifies the machine operation to be performed. Operation codes are represented as two hexadecimal characters.

The 4-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 4-bit R2 field in the RR instruction format specifies the address of the second operand, which is normally a register address.

The 4-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, and General Register 0 can never be used for indexing.

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

The RR instructions are used for operations between two 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 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.

Table 1.4 summarizes the first and second operand designations for each instruction format.

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.

The SF instructions are used for short immediates, in which the data field specifies a 4-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.

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 4-bit mask 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. These instructions are described in Section 4.10. 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.

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

Table 1.4 Designations for First and Second Operands

### 1.4.2 General Register Usage

The Sixteen General Registers function as accumulators or Index Registers in all arithmetic and logical operations. Each General Register is a 16-bit half-word 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), 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 zero 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 full-word 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.
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.

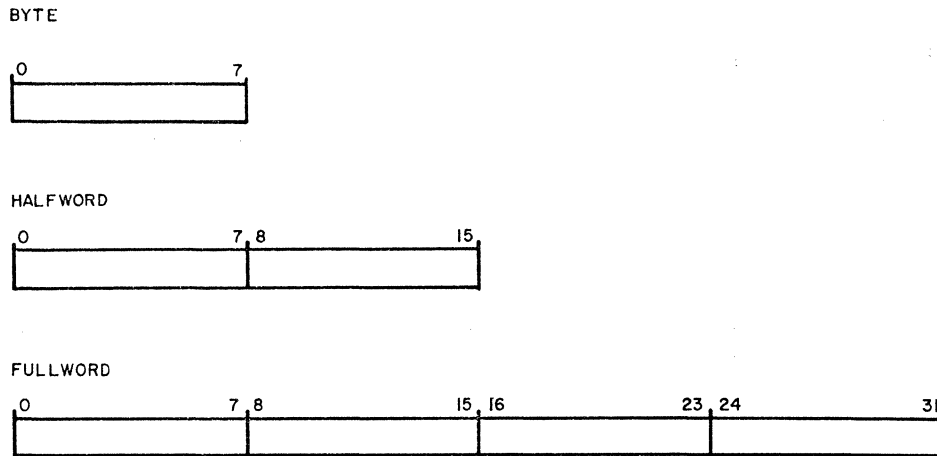


Fig. 1.6 Data Word Formats

### 1.4.3 Storage Addressing

The GE-PAC 30 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 Fig. 1.6.

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 RX instruction is 16 bits wide, it is possible to directly address 65,535 bytes.

The GE-PAC 30-2E 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.

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 1.5 illustrates the addressing scheme.

For example, if the address referenced 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.

Note that if an instruction specified a halfword or fullword operand whose address is not an even number the next lower address is used. For example, if the address referenced is  $0055_{16}$ , then:

1. A Byte-Oriented instruction would extract the value  $AB_{16}$  as an operand.
2. A Halfword-Oriented instruction would extract the value  $89AB_{16}$  as an operand.
3. A Fullword-Oriented instruction would extract the value  $89ABCDEF_{16}$  as an operand.

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

Table 1.5 Memory Address Data

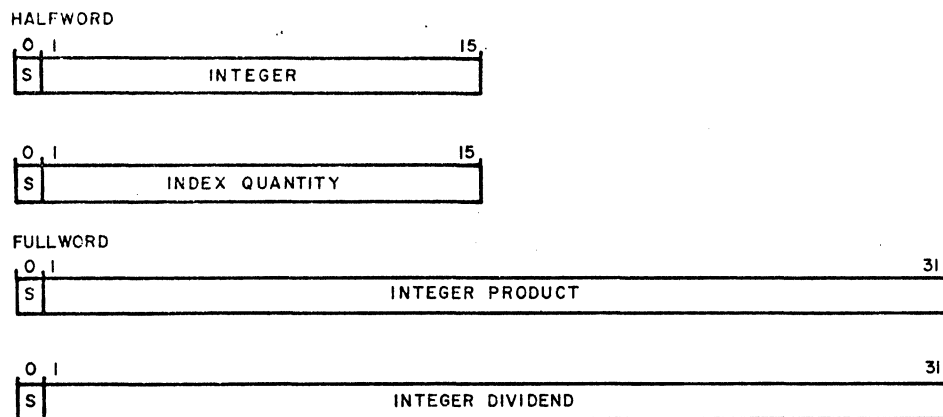


Fig. 1.7 Fixed-Point Word Formats

## 1.5 DATA

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

### 1.5.1 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 Fig. 1.7.

Fixed-point data is treated as signed, 15-bit integers in the halfword format, or 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. The numeric value of zero is always represented with all bits zero. Table 1.6 shows several examples of the fixed-point number representation used in GE-PAC 30 Systems.

The Halfword Arithmetic operand matches the address field of an instruction, permitting Fixed-Point Arithmetic instructions to be used for address arithmetic. Logical, and shift instructions can also be used for address manipulation or computation.

### 1.5.2 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 Fig. 1.8.

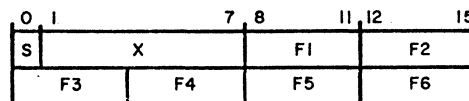


Fig. 1.8 Floating-Point Word Format

Sign and magnitude representation is used, in which the sign bit S is zero for positive values, and one for

Number	Decimal	Binary			
$2^{15}-1$	32767	0111	1111	1111	1111
$2^0$	1	0000	0000	0000	0001
0	0	0000	0000	0000	0000
$-2^0$	-1	1111	1111	1111	1111
$-2^{15}$	-32768	1000	0000	0000	0000

Table 1.6 Examples of Fixed-Point Representation

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:

$$F_1 \cdot 16^{-1} + F_2 \cdot 16^{-2} + F_3 \cdot 16^{-3} + \dots + F_6 \cdot 16^{-6}.$$

A normalized floating-point number has a non-zero high-order hexadecimal fraction digit (F<sub>1</sub>). If the high-order hexadecimal fraction digit (F<sub>1</sub>) 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

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

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. If overflow, the exponent and fraction of the result are set to all ones. The sign of the result is not affected by the overflow. If underflow, the sign, exponent and fraction of the sum are set to zero.

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 result 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 separate from the General Registers and are addressable only by the Floating-Point instructions, which are described in Section 4.6.

### 1.5.3 Logical Data

Logical operations in the GE-PAC 30-2E, 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 Fig. 1.9.

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

Value	Binary			
1.0	0100	0001	0001	0000
	0000	0000	0000	0000
-1.0	1100	0001	0001	0000
	0000	0000	0000	0000
9.5	0100	0001	1001	1000
	0000	0000	0000	0000
-0.5	1100	0000	1000	0000
	0000	0000	0000	0000
$-(1 \cdot 16^{-6}) \cdot 16^{63}$	1111	1111	1111	1111
	1111	1111	1111	1111
$-16^{-65}$	1000	0000	0001	0000
	0000	0000	0000	0000
$0.1 + 16^{-6}$	0100	0000	0001	1001
	1001	1001	1001	1010

Table 1.7 Examples of Floating-Point Representation

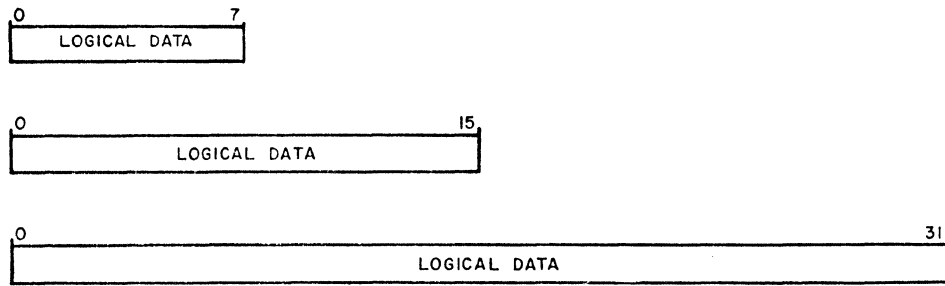


Fig. 1.9 Logical Data Word Formats





# SECTION 2 INTERRUPT SYSTEM

## 2.1 INTERRUPT PROCEDURE

The Interrupt structure of the GE-PAC 30-2E provides rapid response to external and internal 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 several types of interrupts in the GE-PAC 30-2E are listed in Table 2.1 along with their associated enable/disable PSW bits. 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 Interrupts 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 I/O Termination Interrupt can also occur during a Load Program Status Word or Exchange Program Status instruction.

The Interrupt Procedure of the GE-PAC 30-2E 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 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	PSW CONTROL BIT
External	1
Machine Malfunction	2
Fixed-Point Divide Fault	3
Automatic I/O Service	4
Floating-Point Arithmetic Fault	5
I/O Termination	6
Protect Mode	7
Illegal Instruction	
Termination Queue Overflow	
Supervisor Call	

Table 2.1 Interrupts

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 indicated core locations for Old and New Program Status Words are shown in Section 1. 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.

## 2.2 INTERNAL INTERRUPTS

The GE-PAC 30-2E can generate six Internal Interrupts. Of these the Illegal Instruction and the Supervisor Call cannot be inhibited. Inhibited Internal Interrupts are not queued.

### 2.2.1 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  $2^{15}$ . 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.

### 2.2.2 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  $\pm X'7FFF\ FFFF'$ . 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 more explanation of floating-point instructions.

### 2.2.3 Machine Malfunction Interrupt

Bit two of the Current Program Status Word controls the Machine Malfunction Interrupt. This error can occur on either a memory parity error or during the restart process following a power down. If the memory is equipped with the Parity Option, the parity bit of each memory byte is set to maintain odd parity. This bit is recomputed during each memory read; if the computed bit is not equal to the transferred bit, and if bit two 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. It is not possible to guarantee programmed recovery from a parity error.

In a power fail situation, the GE-PAC 30-2E 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 returns, the registers are reloaded and the Current PSW is restored from locations X'0024' through X'0027'. If bit two of this (Machine Malfunction) 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 memory parity error and power failure by comparing the location counter of the Machine Malfunction Old PSW with the contents of locations X'0026' and X'0027' (Power Fail PSW save area). If they are equal, a power failure has occurred. Pressing the Initialize switch on the console causes the Processor to clear device interrupts and go through the power fail and restart sequence.

## 2.2.4 Illegal Instruction

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 cannot be decoded into a legal operation for processing. 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.

## 2.2.5 Protect Mode Violation

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, and the Old PSW then points to the Privileged instruction that caused the interrupt.

## 2.2.6 Supervisor Call (SVC)

This interrupt occurs as the result of an SVC Instruction, which is used to communicate between running programs and operating systems. When an SVC Instruction is executed, the following action takes place:

1. Current PSW is stored at the Supervisor Call Old PSW location.
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 Status location, location X'009A'.

4. The Current Location Counter is loaded from one of the Supervisor Call New instruction address locations. The Supervisor Call Interrupt is not Inhibitable.

## 2.3 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. 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 30-2E has two classes of interrupts directly related to peripheral device handling. These are External Interrupt and the Immediate Interrupt. Two other classes, the I/O Termination Interrupt and the Termination Queue Overflow Interrupt can occur upon termination of an Interrupt Service Block 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 I/O Termination Interrupt. The Termination Queue Overflow Interrupt is always enabled.

### 2.3.1 External Interrupt

If Bit 1 of the Current PSW is set, and if 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.

### 2.3.2 Immediate Interrupt

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 Service Pointer Table in locations X'00D0' to X'02CF', see Section 3. The Service Pointer obtained is the address of either an Old PSW save area or a Function Word of an Interrupt Service Block. If Bit 15 of the service pointer is reset, the address contained is that of an area of core defined as follows:

<u>ADDRESS</u>	<u>CONTENTS</u>
(Service Pointer)	"Old" PSW Status and Condition Code.
(Service Pointer) +2	"Old" PSW Location Counter.

(Service Pointer) +4	"New" PSW Status and Condition Code.
(Service Pointer) +6	First Instruction in Interrupt Service Subroutine.

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)  $\rightarrow$  [(SERVICE POINTER)]  
 PSW (0:15)  $\rightarrow$  [(SERVICE POINTER)+4]  
 PSW (16:31)  $\leftarrow$  (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 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 implying that Automatic I/O service is required. This Processor activity is described in Section 3.

### 2.3.3 I/O Termination Interrupt

The termination of an Interrupt Service Block (ISB) 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 I/O Termination Old PSW location and the Current PSW is loaded from the I/O Termination New PSW location. In this way, the control software is notified of the completion of an ISB 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 loaded PSW is set, and there is an entry in the queue, this interrupt is taken. This is described in more detail in Section 3.

### 2.3.4 Termination Queue Overflow Interrupt

If the Processor attempts to enter an 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 the Queue Overflow Old PSW location; and loads the Current PSW from the Queue Overflow New PSW location. This interrupt cannot be disabled.

## 2.4 SPECIAL INTERRUPTS

The GE-PAC 30-2E Processor provides one special interrupt. This is the Console Interrupt.

### 2.4.1 Console Interrupt

The GE-PAC 30-2E provides for operator intervention in the following manner. If Bit 4 of the Current PSW is set, the Register Display switch in the OFF position (no registers selected), and the Mode Control switch in RUN, depressing the EXECUTE switch causes an Interrupt signal from Device 01. Servicing this signal can be accomplished through the Immediate Interrupt or the Interrupt Service Block.

### 2.4.2 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 user at this time attempt to store into a protected core area (as defined by the mask in the Memory Protect Controller), 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.



## SECTION 3

# INPUT/OUTPUT

### 3.1 INTRODUCTION

The input/output structure of the GE-PAC 30-2E provides a high degree of flexibility in controlling and communicating with peripheral devices. It can perform data transfers in any of several ways. The choice of which input/output method to use depends on the particular application and on the characteristics of the external devices. The primary methods of data transfer between the Processor and peripheral devices on the Multiplexor Bus are:

1. One byte or one halfword to or from any of the General Registers.
2. One byte or one halfword to or from core memory.
3. A block of data to or from core memory under direct Processor Control.
4. A block of data to or from core memory under control of an Interrupt Service Block (ISB).
5. A block of data to or from core memory under control of the Selector Channel.

Up to 256 devices or device controllers can be connected to the Multiplexor Bus. In general, GE-PAC 30 standard device controllers expect a predetermined sequence of commands to effect data transfers. These commands address the device, put it in the correct mode, cause data to be transferred, and in some cases, deactivate the device. The proper sequence of commands for any device can be initiated through Program Controlled I/O or ISB programming techniques. For details on any particular device, refer to the Programming Specifications or manual for the device.

### 3.2 PROGRAM-CONTROLLED I/O

Programmed-Controlled I/O makes use of programmed instructions to transmit a proper sequence of commands to device controllers. The exact order of instructions depends on the nature of the device in use. There are two types of operations for program-controlled I/O: status monitoring and interrupt drive. Status monitoring I/O operates with the I/O Interrupt enable bit (bit 1) of the Current Program Status Word in the disabled state. The program repeatedly tests device status, waiting for the device controller to be ready to accept commands. The following steps describe a general approach to this type of programming:

1. An Output Command instruction addresses the device and sets the proper mode.
2. A Sense Status instruction tests the state of the device; i. e. ; busy, device unavailable, etc.
3. A Conditional Branch instruction loops back to the Sense Status instruction until the device is ready to transfer data.
4. When the device is ready, a Read or Write instruction causes the data transfer to take place.

All Input/Output instructions are described in Section 4.10. The I/O instructions to devices on the Multiplexor Bus address the device and perform the specified operation in a request/response fashion. If a non-existent device is addressed, or if the device fails to respond in time, the instruction "times out", and indicates this action to the program in the Condition Code. This time-out feature assures that a program can never "hang up" due to an I/O programming error or a device failure.

Note that all I/O Instructions are privileged and program-controlled I/O can be performed only with the Processor in the Supervisor Mode.

### 3.3 INTERRUPT-DRIVEN I/O

Interrupt Driven I/O takes advantage of the interrupt structure of the GE-PAC 30-2E to overlap data transfers with program execution. Two types of I/O Interrupts can be used:

1. The External Interrupt, which is enabled if bit 1 of the Current PSW is set and bit 4 of the PSW is reset, this is compatible with the GE-PAC 30-1 and 30-2.
2. The Immediate Interrupt, which is enabled if bit 1 and bit 4 of the Current PSW are set, and is unique to the GE-PAC 30-2E.

These interrupts are described in Section 2. The programming techniques for both are similar. Both require that before I/O transfer is initialized, the device be initialized with the Output Command instruction. Both require that the programs overlapped with data transfers run with the I/O Interrupt bit (bit 1) of the Current Program Status Word in the enabled state.

When an External Interrupt occurs, the New PSW at X'0044' must have bit 1 reset, and the interrupt service routine must acknowledge the interrupt with an

Acknowledge Interrupt Instruction. This instruction returns the address of the interrupting device which the program uses to vector the appropriate routine for data transfer. The program exits by executing a Load Program Status Word instruction specifying the old External Interrupt location at 0040.

When an Immediate Interrupt occurs, the Processor has already acknowledged the interrupt condition, obtained the device number, stored the Current PSW at the origin of the interrupt service routine, and loaded the New Program Status Word from next sequential location. The program can service the device without further identification and exit by executing a Load Program Status Word Instruction specifying its own origin as the load location.

In general, Interrupt Driven I/O takes more program steps to set up the appropriate pointers, and New PSWs, but much less processor execution time is required. This is because programs can avoid using Sense Status instruction loops to determine the conditions of a peripheral device, which is normally very slow compared to the execution rate of the Processor.

### 3.4 BLOCK I/O

The Read Block (RB or RBR) and Write Block (WB or WBR) instructions are appropriate for record-oriented devices which operate at moderate to high speed, which is approximately 20K-150K bytes per second. For data transfers at this rate, Interrupt-Driven I/O is not appropriate since all the Processor time would be consumed servicing interrupts.

The Read Block and Write Block instructions greatly simplify programming in this case. A single instruction causes information to be transferred between a device and sequential location in core memory. Transfer is terminated when a predetermined location is reached, or when an unusual device status is encountered.

Prior to block transfer, Output Command and Sense Status instructions are used to specify the function and test the status of the device. The block transfer instruction can then perform all remaining steps of input/output. Note that the complete attention of the Processor is given to the data block transfer and that normal processing will not resume until completion of this instruction. Also, this instruction is non-interruptable. An alternate method of handling block transfers is to use the Interrupt Service Block as described in the next section. For very high speed devices, up to 500K bytes per second, the Selector Channel is required, see Section 3.8.

## 3.5 AUTOMATIC I/O PROGRAMMING

The GE-PAC 30-2E Automatic I/O controls the activities of peripheral devices. The execution of the

Automatic I/O takes place in between the execution of user instructions and results in a program delay rather than a program interrupt with an exchange of Program Status Words. The Automatic I/O may generate an interrupt because of abnormal conditions or because of the occurrence of an event for which the software had requested an interrupt. Bits 1 and 4 of the Current Program Status Word control the operation of the Automatic I/O. Both of these bits must be set to permit Automatic I/O operations. Operation also depends on the Service Pointer Table, and the Interrupt Service Block with its associated Function Word, and the I/O Termination Queue, see Fig. 3.1.

### 3.5.1 Service Pointer Table

The Service Pointer Table starts at location X'00D0'. 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 as described in Section 2. If bit 15 of the entry is set, then the entry minus one is the address of an ISB Function Word.

### 3.5.2 Interrupt Service Block

The Interrupt Service Block contains the ISB Function Word plus the storage locations and data required for the operation. The ISB 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 Service Pointer Table. A complete Interrupt Service Block is shown in Fig. 3.2.

### 3.5.3 I/O Termination Queue

The I/O Termination Queue is a circular list identical to those described in Section 4 under "list 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 ISB function. The GE-PAC 30-2E uses the queue to indicate termination of an ISB operation.

### 3.5.4 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





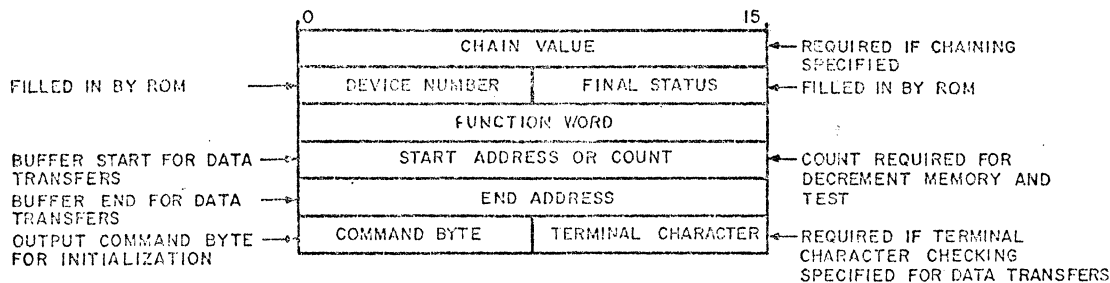


Fig. 3.2 Interrupt Service Block

The action taken by the ISB depends on the bit configuration of the Function Word. Fig. 3.1 shows the ISB Operation in block diagram form. Appendix 7 contains complete flow charts of the operation.

In the queuing operation, a Queue Overflow Interrupt is generated if the queue is full when an ISB attempts to make an entry. This interrupt is described in Section 2.

### 3.6 ISB FUNCTION WORD

There are three phases involved in ISB 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 Word is shown in Fig. 3.3.

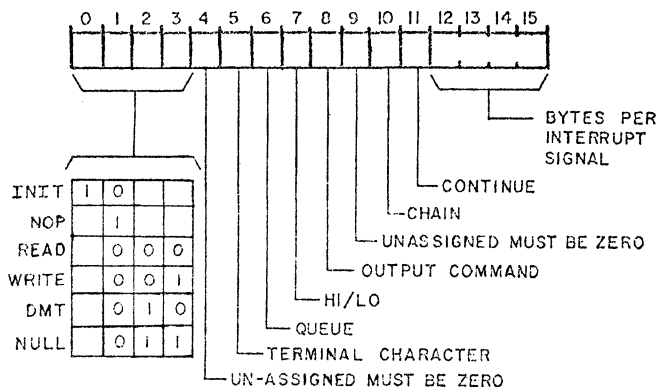


Fig. 3.3 Bit Configuration for ISB Function Word

#### 3.6.1 Initialization

Bits 0 (INIT) and 8 (Output Command) of the Function Word control the initialize phase of channel operations. If bit 0 (INIT) is set when the ROM decodes the command word, it resets bit 0 (INIT) and checks bit 8 (Output Command). If bit 8 is set, the ROM issues the output command located at the start of the Interrupt Service Block plus ten and returns control to the Processor. Operations with the device resume when an interrupt signal occurs from the device. Since the ROM 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 I/O operation. The bit configurations of the Function Word for the Initialization phase are illustrated in Fig. 3.4.

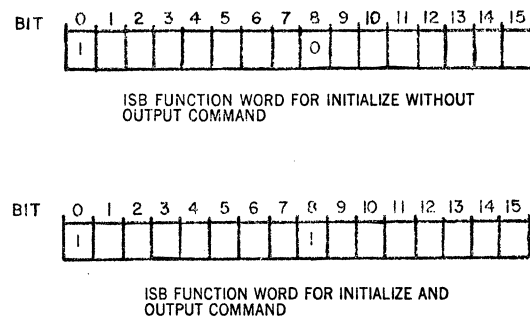


Fig. 3.4 Function Word for Initialize and Output Command

#### 3.6.2 I/O Operations

There are five distinct types of I/O operations the GE-PAC 30-2E can perform. These are:

1. Read
2. Write

3. Decrement Memory Test
4. No Operation
5. Null

The ISB configurations for these operations are illustrated in Fig. 3.5.

The ROM decodes bits 1, 2, and 3 of the Function Word to determine which of these operations to perform. It uses bit 5 and bits 12 through 15 for additional information on Read/Write 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 positions indicate 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 have been transferred, the starting address is incremented by the appropriate amount and compared to the ending address. If it is greater, the termination phase is entered. If it is not greater, control returns 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 termination phase is entered. In this way, termination can be terminated because the buffer exhausted or a terminal character has been found in the data stream.

Before starting a data transfer, the GE-PAC 30-2E checks the device status. Any non-zero status condition will stop the transfer and cause the termination phase to be entered. Before entering the termination phase, the Initial (INIT) Bit and No Operation bit is set in the Function Word, the Queue bit is set to force an entry in the I/O 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 halfword 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, control returns to the Processor for program execution. If equal to zero, the termination phase is entered 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 ROM is to ignore any interrupt signal from the associated device. The ROM 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 Service Pointer Table should contain pointers to "no operation" control words for all non-existent devices.

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

### 3.6.3 Termination

The GE-PAC 30-2E 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, control returns to the Processor. The two termination functions are Queue and Chain. The Function Word bit configuration for Queuing and chaining is shown in Fig. 3.6. Bit 6 of the Function Word controls queuing. If this bit is set, the ROM on entering the termination phase stores the address of the Function Word in the I/O Termination Queue. The condition of bit 7 of the Function Word controls positioning within 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.

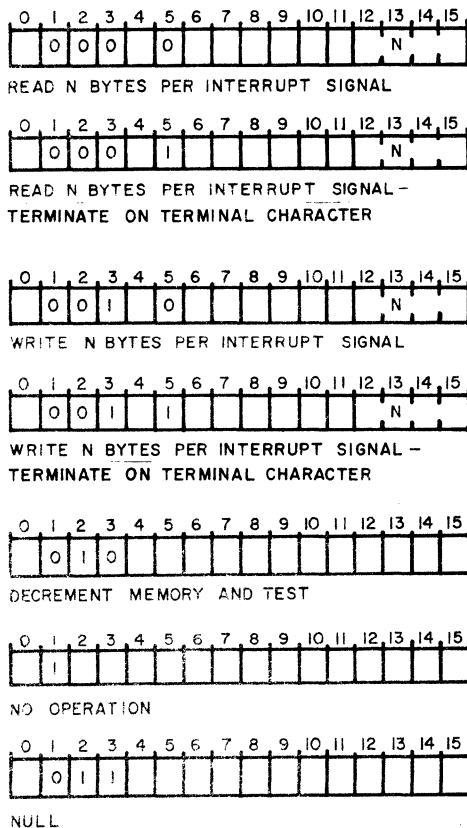


Fig. 3.5 Function Words for I/O Operation

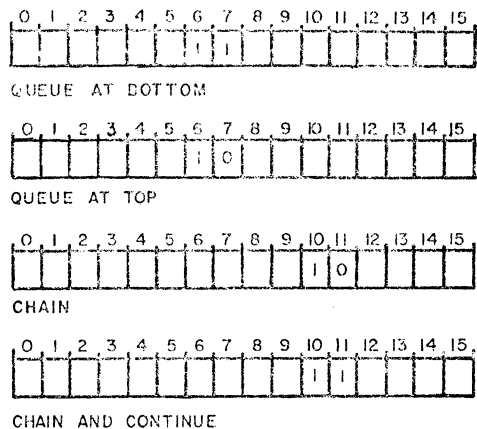


Fig. 3.6 Function Words for Termination

Bit 10 of the Function Word controls chaining. In this operation, the ROM stores the first halfword of the Interrupt Service Block in the appropriate location in the Service Pointer 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 ROM 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 ROM can either generate an Immediate Interrupt or start another I/O operation.

### 3.7 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 I/O 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 Fig. 3.7A. The Chain value is set to point to the second Interrupt Service Block. The Status Byte and Device

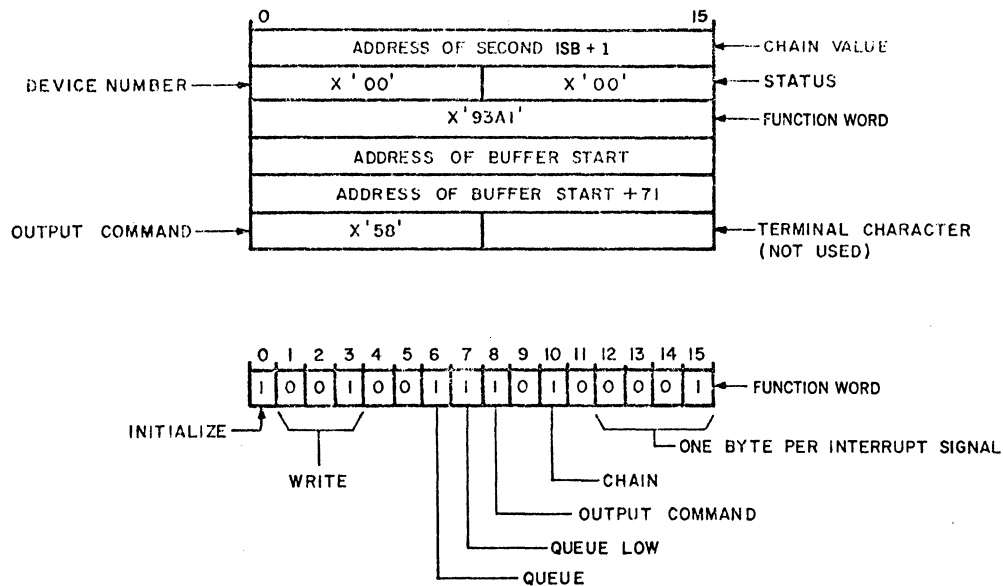
Number are set to zero. The Function Word is set for Initialize, Write, Queue, Queue Low, Output Command, Chain, and transfer one byte per interrupt signal. The next two halfwords point to the beginning and end of the 72 byte buffer. The Output Command byte is set to enable and write. The user program 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 ROM issues the Output Command and resets the Initialize bit. It gives control to the Processor for the execution of normal instruction. As each subsequent interrupt signal is received from the device, the ROM 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 I/O 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 ROM is directed to the second Interrupt Service Block. This block is illustrated in Fig. 3.7B. 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 ROM 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 ROM 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, which causes the ROM to generate the Immediate Interrupt on the same interrupt signal that caused output of the last data byte. If this bit were reset, the next interrupt signal from the device would generate the Immediate Interrupt.

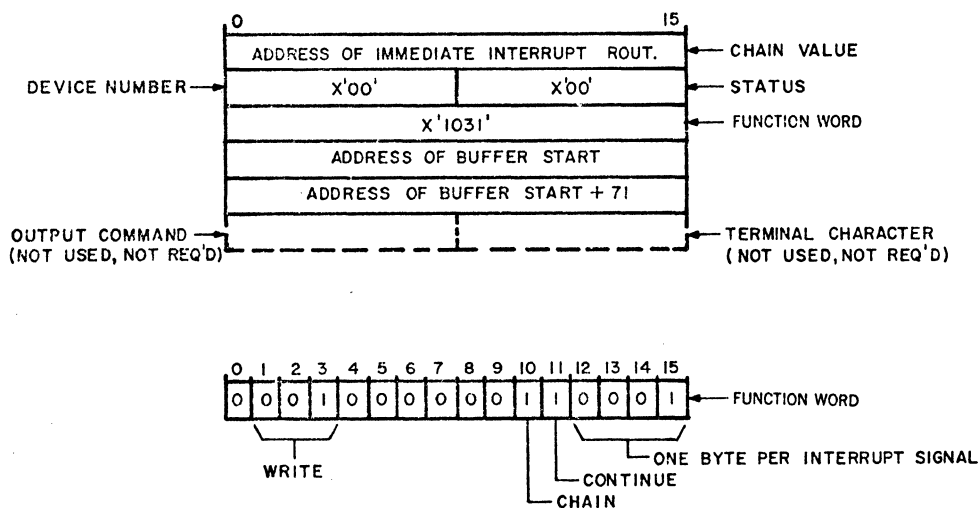
### 3.8 SELECTOR CHANNEL I/O

#### 3.8.1 Introduction

The Selector Channel controls the transfer of data between I/O devices and core memory at rates of up to 500K bytes per second. Up to 25 I/O devices can be connected to the Selector Channel, but only one device can transfer data at a time. More than one Selector



(A) First Interrupt Service Block



(B) Second Interrupt Service Block

Fig. 3.7 First/Second Interrupt Service Blocks

Channel can be incorporated in the system. The advantage gained in using the Selector Channel is that other program processing can occur simultaneously with the transfer of data between the I/O device and core. This is accomplished by allowing the Selector Channel and the Processor to access memory on a cycle-stealing basis. In some instances, the execution time of the program in process is affected, while in others, the effect is negligible. This depends upon the rate at which the Selector Channel and Processor both compete for access to memory.

Fig. 3.8 is a block diagram which shows the incorporation of the Selector Channel into the GE-PAC 30-2E system.

### 3.8.2 Operation

Programming a device on the Selector Channel consists of setting up the device, setting up the Selector Channel, and sending a GO command to the Selector

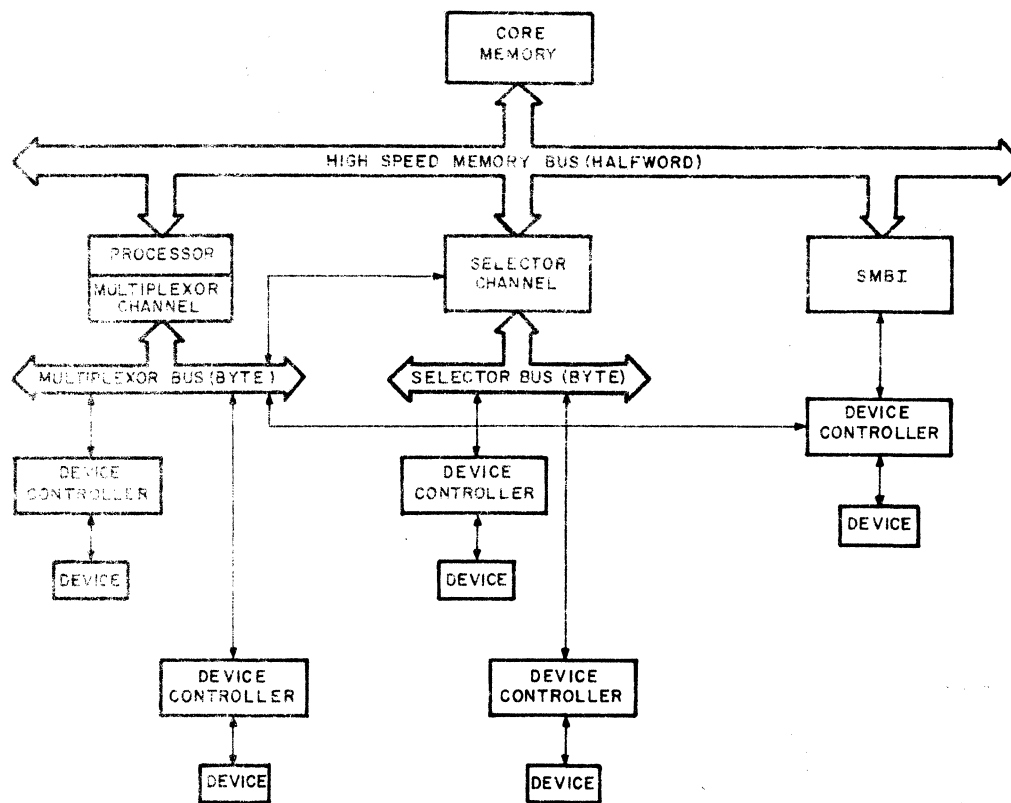


Fig. 3.8 Systems Interface Block Diagram

Channel. When all devices on the Selector Channel are idle, the Selector Bus becomes a part of the Multiplexor Bus. This provides the path to set up the device and the Selector Channel. The last device addressed prior to sending the GO command is the device the Selector Channel controls, assuming that the device is connected to the Selector Channel. The program must, therefore, send the GO command before addressing any other devices.

During the data transfer, the Selector Channel provides a direct data path between the device and core memory. Until the transfer is completed, no I/O instructions can be issued to any device on the Selector Channel, including the device transferring data.

If devices on the Selector Channel are referenced while the Channel is busy, the False Sync (V condition code) bit is set. The setting up or the initialization of the device is accomplished by executing an Output Command (OC or OCR) instruction. Refer to the Programming Manual for the device to be controlled for the bit configuration of the Output Command. Note that the Selector Channel has a unique device number just as all other I/O devices. Output Commands, as with all Input/Output instructions, affect only the device addressed.

The Selector Channel has a 16-bit incrementing address register and a 16-bit final address register. The user program loads the starting core address into the incrementing register and the final core address into the final address register. Transfer is completed when the incrementing address register matches the final address register. The address limits are expressed inclusively; transfers begin and end on the addresses placed in the starting and final address registers.

Core memory in the GE-PAC 30-2E Processors is addressed on halfword boundaries; that is, each time memory is accessed two bytes or a halfword are obtained. A 16-bit address register is used, with the least significant bit, bit 15, being used to determine the byte desired. See Fig. 3.9.

Each time the Selector Channel accesses core memory, two bytes (a halfword) are transmitted. It is mandatory that data transfers begin on a halfword boundary. The following results if data transfers are ended on byte boundaries:

1. Write Mode (Core to Device) ending on byte boundary (bit 15 = 1) has no effect.

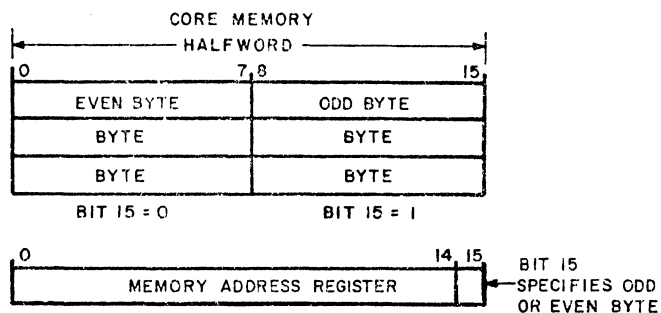


Fig. 3.9 Memory Address

2. Read Mode (Device to Core) ending on byte boundary (bit 15 = 1) causes the previous contents of the last odd byte in core to be written into the current odd byte in core, see Fig. 3.10.

The user program specifies the mode, either Read or Write, and gives the GO command. The following sections provide details for programming the Selector Channel.

**NOTE**

When executing programs that involve the use of the Selector Channel, the Processor may not be run in the Variable Mode.

### 3.8.3 Address Set-Up

An Output Command stop should be issued prior to starting any operation on the Selector Channel. Four successive bytes are required to specify the starting and final addresses. Either the Write Data (WD or WDR), Write Block (WB or WBR), or Write Halfword

(WH or WHR) Instructions may be used to send the starting and final addresses to the Selector Channel Controller. Fig. 3.11 illustrates the meaning of four bytes used for addressing. In addition it illustrates the meaning of Data Bytes when setting Start and Final Address.

1. Starting Address High (bits 0-7)
2. Starting Address Low (bits 8-15)
3. Final Address High (bits 0-7)
4. Final Address Low (bits 8-15)

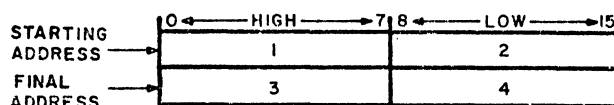


Fig. 3.11 Selector Channel Controller

### 3.8.4 Termination

Data transmission between the Selector Channel and the device presently connected to it is halted if any of the following conditions occur:

1. The starting address matches the final address. This would be considered a normal termination.
2. The starting (incrementing) address goes from all ones to zero (maximum count). In this case, no match occurred and this would be considered an abnormal termination.
3. Any of the DU, EOM, or EX status bits of the device presently connected to the Selector Channel changes to a ONE. This is also an abnormal termination.

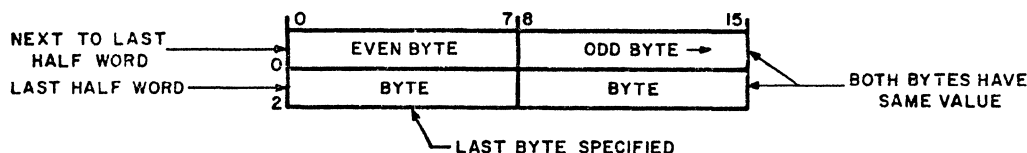


Fig. 3.10 Core Memory Configuration

4. A STOP Command is sent to the Selector Channel Controller via a user program.

The termination condition is determined one of two ways: by a status loop, or by the interrupt method. An Output Command STOP, should be issued to the Selector Channel following its termination.

### 3.8.5 Reading the Final Address

After Selector Channel termination, the last processor core location either written into or read from may be determined by executing a pair of Read Data (RD or RDR) instructions or a Read Block (RB or RBR) or a Read Halfword (RH or RHR) instruction. This information permits a user program to verify a successful

data transmission or determine at what address termination occurred.

Fig. 3.12 illustrates the meaning of the order in which the data is read into the Processor, and the order in which Read Data Instructions are executed.

1. Final Address High (Bits 0-7).
2. Final Address Low (Bits 8-15).

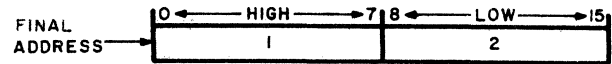


Fig. 3.12 Read Data Instructions Configurations

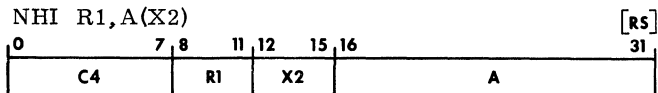
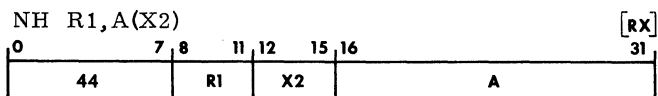
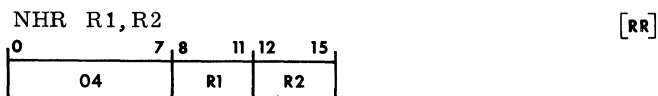


# SECTION 4 INSTRUCTION REPERTOIRE

## 4.1 INTRODUCTION

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

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



2. A description of instruction operation.

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.

3. A diagrammatic representation of instruction operation.

NHR: (R1) ← (R1) AND (R2)

NH: (R1) ← (R1) AND [ A + (X2) ]

NHI: (R1) ← (R1) AND 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 1 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.

Resulting Condition Code:

12	13	14	15
C	V	G	L
		0	0
		0	1
		1	0

LOGICAL PRODUCT IS ZERO

LOGICAL PRODUCT IS NOT ZERO

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.

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

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 4 bits specifying Branch or 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 4-bit second operand used with Short Format immediate instructions.
D	The 4-bit displacement value used with Short Format branch instructions.
(0:7)	A bit grouping within a byte, a halfword, or a fullword. Read as "0 thru 7 inclusive".
(8:15)	"bits 8 thru 15 inclusive", etc.
(16:31)	
PSW	Program Status Word of 32 bits containing the Status, Condition Code, and current instruction address.
CC	Condition Code of 4 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.

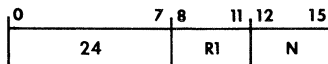
## 4.2 FIXED-POINT LOAD/STORE INSTRUCTIONS

The Fixed-Point Load/Store Instructions are used to transfer data between the General Registers and core memory. The instructions described in this section are:

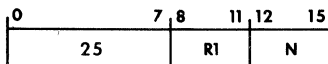
LIS	Load Immediate Short
LCS	Load Complement Short
LHR	Load Halfword RR
LH	Load Halfword
LHI	Load Halfword Immediate
LM	Load Multiple
STH	Store Halfword
STM	Store Multiple

### 4.2.1 Load Halfword

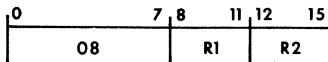
LIS R1, N [SF]



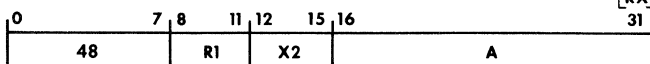
LCS R1, N [SF]



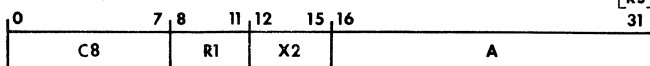
LHR R1, R2 [RR]



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
		0	1
		1	0
		1	1

OPERAND IS ZERO  
 OPERAND IS LESS THAN ZERO  
 OPERAND IS GREATER THAN ZERO

Programming Note:

Load Immediate Short (LIS) causes the 4-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.

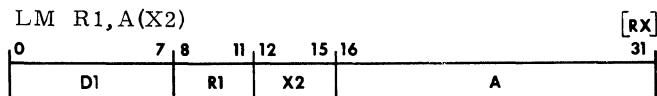
Load Complement Short (LCS) causes the 4-bit second operand to be expanded to a 16-bit halfword with high order bits set to zero. The two's complement of this halfword is loaded into the General Register specified by R1.

These instructions may be used to preset a register with an index value, load a register with the first operand for a supplemental Arithmetic operation (e.g. Add, Multiply), or set the Condition

Code for supplemental testing by a Branch or Condition Instructions.

The Load Immediate Short (LIS) and Load Complement Short (LCS) instructions are unique to the GE-PAC 30-2E.

### 4.2.2 Load Multiple



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.

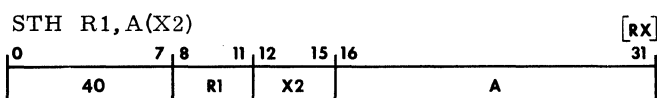
Note that any number of sequential General Registers can be loaded in this manner.

1. (R1)  $\longleftarrow$  [ A + (X2) ]
2. R1: X'F'  
if R1 = X'F', the 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.

### 4.2.3 Store Halfword



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

(R1)  $\longrightarrow$  [ A + (X2) ]

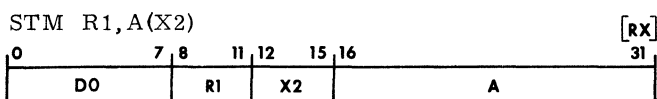
Resulting Condition Code:

Unchanged.

Programming Note:

This instruction is subject to Memory Protect.

### 4.2.4 Store Multiple



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 Instruction in conjunction with the Load Multiple 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 contained in the General Registers and/or core memory. Also included are logical and Arithmetic compare operations. The instructions described in this section are:

AIS	Add Immediate Short
AHR	Add Halfword RR
AH	Add Halfword
AHI	Add Halfword Immediate
AHM	Add Halfword to Memory
ACHR	Add with Carry Halfword RR
ACH	Add with Carry Halfword
SIS	Subtract Immediate Short
SHR	Subtract Halfword RR
SH	Subtract Halfword
SHI	Subtract Halfword Immediate

SCHR	Subtract with Carry Halfword RR
SCH	Subtract with Carry Halfword
CLHR	Compare Logical Halfword RR
CLH	Compare Logical Halfword
CLHI	Compare Logical Halfword Immediate
CHR	Compare Halfword RR
CH	Compare Halfword
CHI	Compare Halfword Immediate
MHR	Multiply Halfword RR
MH	Multiply Halfword
MHUR	Multiply Halfword Unsigned RR
MHU	Multiply Halfword Unsigned
DHR	Divide Halfword RR
DH	Divide Halfword

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			ARITHMETIC OVERFLOW
1				CARRY

Programming Note:

Add Immediate Short (AIS) causes the 4-bit second operand (N) to be added to the contents of the General Register specified by R1. The result replaces the contents of R1.

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

Add Halfword to Memory (AHM) 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.

The Add Immediate Short (AIS) and Add Halfword to Memory (AHM) instructions are unique to the GE-PAC 30-2E.

### 4.3.1 Add Halfword

AIS R1,N	[SF]
0 7 8 11 12 15	
26 R1 N	

AHR R1,R2	[RR]
0 7 8 11 12 15	
0A R1 R2	

AH R1,A(X2)	[RX]
0 7 8 11 12 15 16 31	
4A R1 X2 A	

AHI R1,A(X2)	[RS]
0 7 8 11 12 15 16 31	
CA R1 X2 A	

AHM R1,A(X2)	[RX]
0 7 8 11 12 15 16 31	
61 R1 X2 A	

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

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

### 4.3.2 Add with Carry Halfword

ACHR R1,R2	[RR]
0 7 8 11 12 15	
OE R1 R2	

ACH R1,A(X2)	[RX]
0 7 8 11 12 15 16 31	
4E R1 X2 A	

The 16-bit second operand and the carry bit of the condition code are algebraically added 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	13	14	15	
C	V	G	L	
0	0	0	0	SUM IS ZERO
0	0	1	0	SUM IS LESS THAN ZERO
0	1	0	0	SUM IS GREATER THAN ZERO
1	1	1	1	ARITHMETIC OVERFLOW
1	1	1	0	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 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 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 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.

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

Resulting Condition Code:

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

Programming Note:

The Subtract Immediate Short (SIS) instruction causes the 4-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 Subtract Immediate Short (SIS) instruction is unique to the GE-PAC 30-2E.

### 4.3.3 Subtract Halfword

SIS R1, N					[SF]
0	7	8	11	12	15
27			R1	N	

SHR R1, R2					[RR]
0	7	8	11	12	15
OB			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

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

### 4.3.4 Subtract with Carry Halfword

SCHR R1, R2					[RR]
0	7	8	11	12	15
OF			R1	R2	

SCH R1,A(X2)							[RX]
0	7	8	11	12	15	16	31
4F			R1	X2	A		

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:  $(R1) \leftarrow (R1) - (R2) - C$   
 SCH:  $(R1) \leftarrow (R1) - [A + (X2)] - C$

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:

See Add with Carry Halfword.

### 4.3.5 Compare Logical Halfword

CLHR R1,R2

[RR]

0	7, 8	11, 12	15
05	R1	R2	

CLH R1,A(X2)

[RX]

0	7, 8	11, 12	15, 16	31
45	R1	X2	A	

CLHI R1,A(X2)

[RS]

0	7, 8	11, 12	15, 16	31
C5	R1	X2	A	

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
		1	0	OPERAND
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]

0	7, 8	11, 12	15
09	R1	R2	

CH R1,A(X2)

[RX]

0	7, 8	11, 12	15, 16	31
49	R1	X2	A	

CHI R1,A(X2)

[RS]

0	7, 8	11, 12	15, 16	31
C9	R1	X2	A	

The first operand specified by R1 is compared to the sixteen-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:

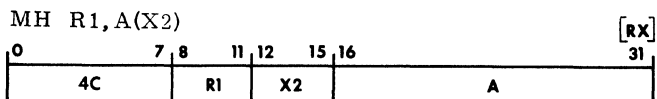
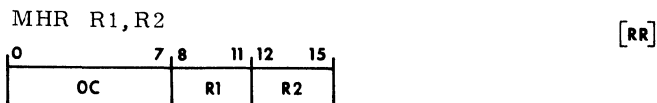
12	13	14	15	
C	V	G	L	
		0	0	FIRST OPERAND EQUAL TO SECOND OPERAND
		0	1	FIRST OPERAND LESS THAN SECOND OPERAND
		1	0	FIRST OPERAND GREATER THAN SECOND OPERAND
1				FIRST OPERAND LESS THAN SECOND OPERAND
0				FIRST OPERAND EQUAL TO OR GREATER THAN SECOND OPERAND

Programming Note:

The Compare Halfword Instructions permit arithmetic comparison of signed two's complement sixteen-bit integers. They facilitate fast comparisons for DO loop and IF statement processing in FORTRAN.

The Compare Halfword Instructions are unique to the GE-PAC 30-2E.

### 4.3.7 Multiply Halfword



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.

$$\begin{array}{lcl} \text{MHR:} & (R1, R1 + 1) & \longleftarrow (R1 + 1) * (R2) \\ \text{MH:} & (R1, R1 + 1) & \longleftarrow (R1 + 1) * [A + (X2)] \end{array}$$

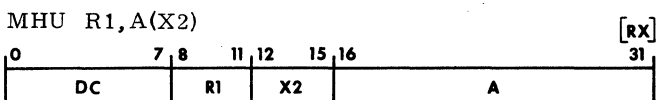
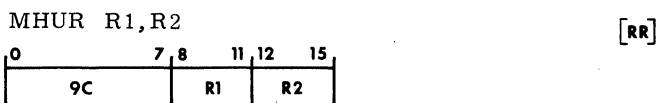
Resulting Condition Code:

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



The 16-bit second operand is multiplied by the contents of the General Register specified by  $R1 + 1$ . All sixteen 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:

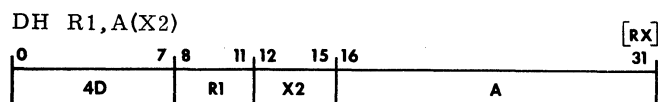
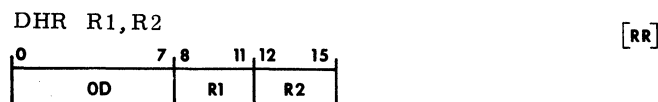
Unchanged.

### Programming Note:

This instruction is most useful in applications requiring multiple precision multiply capability. Typically, a multiply halfword 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 instruction. The partial products could then be summed.

The Multiply Halfword Unsigned instructions are unique to the GE-PAC 30-2E.

### 4.3.9 Divide Halfword



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:	$(R1 + 1) \longleftarrow (R1, R1 + 1) / (R2)$
	$(R1) \longleftarrow \text{Remainder}$
DH:	$(R1 + 1) \longleftarrow (R1, R1 + 1) / [A + (X2)]$
	$(R1) \longleftarrow \text{Remainder}$

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 is enabled by bit 3 of the Program Status Word. The operands remain unchanged.

## 4.4 LOGICAL INSTRUCTIONS

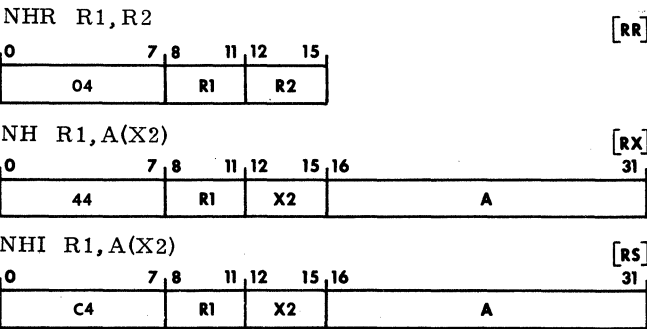
The Logical instructions combine each bit of the first operand with the corresponding bit in the second operand. The instructions described in this section are:

NHR AND Halfword RR

NH AND Halfword

NHI	AND Halfword Immediate
OHR	OR Halfword RR
OH	OR Halfword
OHI	OR Halfword Immediate
XHR	Exclusive OR Halfword RR
XH	Exclusive OR Halfword
XHI	Exclusive OR Halfword Immediate
THI	Test Halfword Immediate

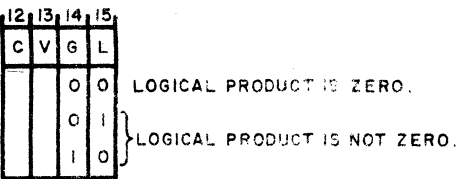
4.4.1 AND Halfword



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)
MH:	(R1)←(R1) AND [ A + (X2) ]
NHI:	(R1)←(R1) AND A + (X2)

Resulting Condition Code:



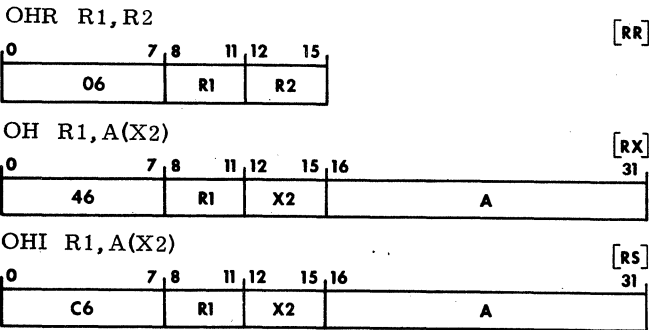
Programming Note:

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

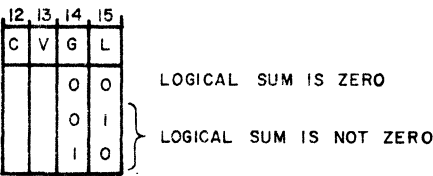
4.4.2 OR Halfword



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:



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]

0	7	8	11	12	15
07			R1		R2

XH R1,A(X2)

[RX]

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

XHI R1,A(X2)

[RS]

0	7	8	11	12	15	16	31
C7			R1		X2		A

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
		0	1
		1	0

LOGICAL DIFFERENCE IS ZERO

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]

0	7	8	11	12	15	16	31
C3			R1		X2		A

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
		0	1
		1	0

NONE OF THE BITS OF THE RESULT SET

BIT 0 OF THE RESULT SET

ONE OR MORE OF BITS 1-15 OF THE RESULT SET AND BIT 0 RESET

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:

LBR Load Byte RR  
 LB Load Byte  
 STBR Store Byte RR  
 STB Store Byte  
 EXBR Exchange Byte RR  
 CLB Compare Logical Byte

### 4.5.1 Load Byte

LBR R1,R2

[RR]

0	7	8	11	12	15
93			R1		R2

LB R1,A(X2)

[RX]

0	7	8	11	12	15	16	31
D3			R1		X2		A

The 8-bit second operand is loaded into the rightmost (least significant) 8 bits of the General Register specified by R1. The left-most (most significant) 8 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]

0	7,8	11,12	15
92	R1	R2	

STB R1,A(X2) [RX]

0	7,8	11,12	15,16	31
D2	R1	X2	A	

The rightmost (least significant) 8-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 leftmost byte, R2 (0:7), is unchanged.

The RX Store Byte (STB) instruction is subject to memory protect.

## 4.5.3 Exchange Byte

EXBR R1,R2

0	7,8	11,12	15
94	R1	R2	

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.

The Exchange Byte Instruction is unique to the GE-PAC 30-2E.

## 4.5.4 Compare Logical Byte

CLB R1,A(X2) [RX]

0	7,8	11,12	15,16	31
D4	R1	X2	A	

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) : [ A + (X2) ]

Resulting Condition Code:

12	13	14	15
C	V	G	L
		0	0
		0	1
		1	0
0			
1			

FIRST OPERAND EQUALS SECOND OPERAND  
 FIRST OPERAND DOES NOT EQUAL SECOND OPERAND  
 FIRST OPERAND IS EQUAL TO OR GREATER THAN SECOND OPERAND  
 FIRST OPERAND IS LESS 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 unnormalized 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 Section 1.5.2. 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 separate from the General Registers. Quantities in floating-point registers can be manipulated only with floating-point instructions. The instructions described in this section are:



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]

0                      7,8                      11,12                      15
2B                      R1                      R2

SE R1,A(X2) [RX]

0                      7,8                      11,12                      15,16                      31
6B                      R1                      X2                      A

The exponents of the two operands are compared. If the exponents differ, the fraction with the smaller exponent is right shifted hexadecimally (4 bits at a time) and its exponent is incremented by one for each hexadecimal shift until the two exponents agree. The fractions are then algebraically subtracted. If a carry results, the exponent of the difference is incremented by one and the fraction (result) is shifted right one hexadecimal position (4 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 (4 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]

0                      7,8                      11,12                      15
29                      R1                      R2

CE R1,A(X2) [RX]

0                      7,8                      11,12                      15,16                      31
69                      R1                      X2                      A

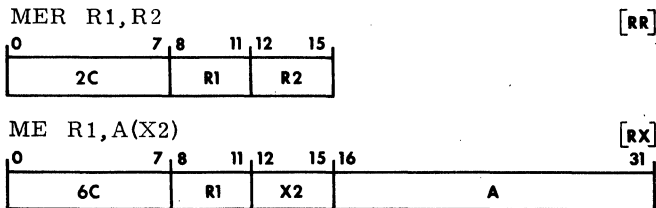
The first operand is compared to the second operand. Comparison is algebraic, taking into account the sign, fraction, and exponent of each number. The result is indicated by the setting of the condition code (PSW 12:15). Both operands remain unchanged.

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

Resulting Condition Code:

12	13	14	15	
C	V	G	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	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



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 (4 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.

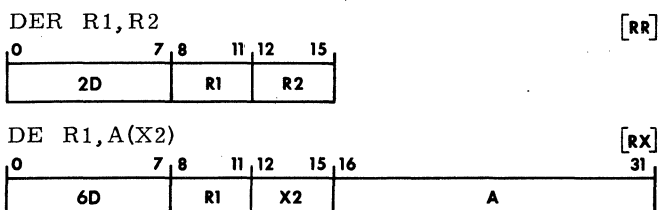
If normalization causes the exponent to underflow, the entire floating-point result is set to zero and the Overflow flag is set.

MER: (R1) ← (R1)\*(R2)  
ME: (R1) ← (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

#### 4.6.7 Floating-Point Divide



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) ← (R1)/(R2)  
DE: (R1) ← (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
0	1	X	X	EXPONENT OVERFLOW
0	1	0	0	EXPONENT UNDERFLOW
1	1	0	0	DIVISOR EQUAL TO ZERO

Programming Note:

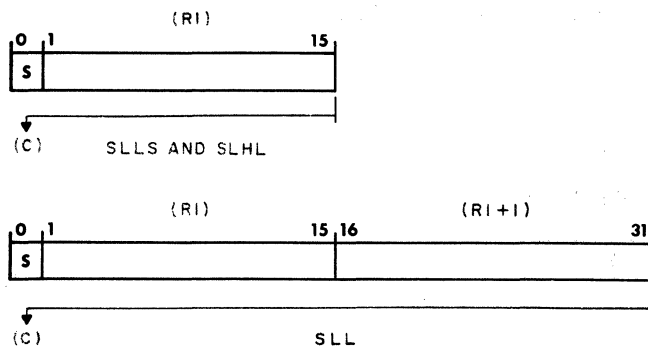
Division by zero, overflow, or underflow cause a Floating-Point Arithmetic Fault Interrupt if enabled by bit 5 of the PSW. Inspection of the Condition Code of the Old PSW indicates the actual cause of the interrupt. If the carry flag is set, then the divisor was zero. If the carry flag is not set, then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

## 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 end 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:

SLLS	Shift Left Logical Short
SLHL	Shift Left Halfword Logical
SLL	Shift Left Logical
SRLS	Shift Right Logical Short
SRHL	Shift Right Halfword Logical
SRL	Shift Right Logical
RLL	Rotate Left Logical
RRL	Rotate Right Logical
SLHA	Shift Left Halfword Arithmetic
SLA	Shift Left Arithmetic
SRHA	Shift Right Halfword Arithmetic
SRA	Shift Right Arithmetic



Resulting Condition Code:

12	13	14	15	
C	V	G	L	
0	0	0	0	RESULT IS ZERO
0	0	0	1	RESULT IS LESS THAN ZERO
0	0	1	0	RESULT IS GREATER THAN ZERO
0	1	0	0	LAST BIT THAT WAS SHIFTED OUT WAS A ZERO
1	0	0	0	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 specified the number of positions the content of R1 is to be shifted.

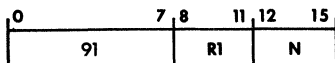
For the Shift Left Halfword Logical (SLHL) instruction only the low order 4-bits (12 through 15) of  $A + (X2)$  are used for the shift count.

The Shift Left Logical and Shift Left Logical Short instructions are unique to the GE-PAC 30-2E.

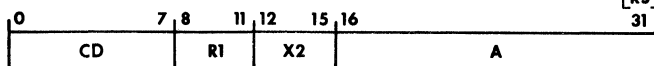
The Shift Left Logical (SLL) instruction shifts registers R1 and R1 + 1, an even odd pair. The R1 field of the instruction must specify an even register. The shift count is specified by the low order 5-bits (11 through 15) of the value  $A + (X2)$ . The carry is formed by the output of R1.

### 4.7.1 Shift Left Logical

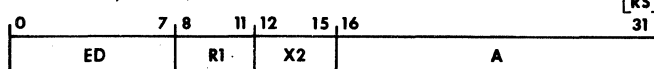
SLLS R1,N [SF]



SLHL R1,A(X2) [RS]



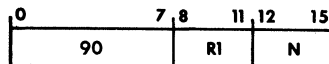
SLL R1,A(X2) [RS]



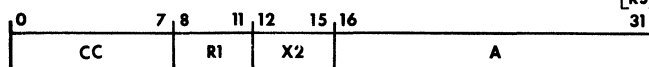
The content of the first operand is shifted left the number of positions specified by the second operand. High order bits shifted out of position 0 are shifted thru the carry bit of the PSW and then lost. Zeros are shifted into the low order bit position.

### 4.7.2 Shift Right Logical

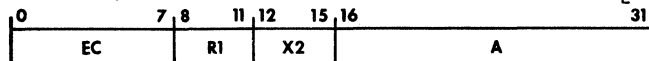
SRLS R1,N [SF]



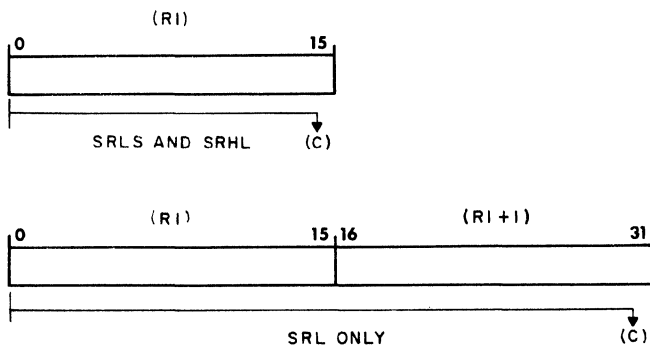
SRHL R1,A(X2) [RS]



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 zero.



Resulting Condition Code:

12	13	14	15
C	V	G	L
		0	0
		0	1
		1	0
0			
1			

RESULT IS ZERO

RESULT IS LESS THAN ZERO

RESULT IS GREATER THAN ZERO

LAST BIT THAT WAS SHIFTED OUT WAS A ZERO

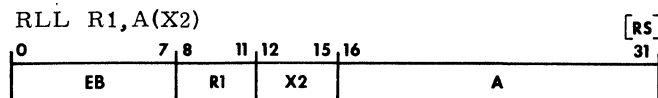
LAST BIT THAT WAS SHIFTED OUT WAS A ONE

Programming Note:

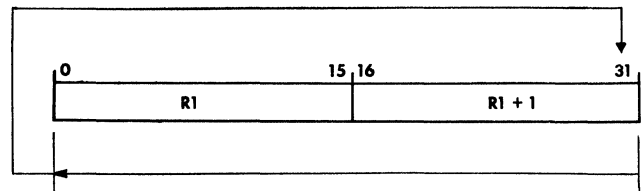
The programming notes on 4-14 for Shift Left Logical Instructions apply to these instructions as well.

The Shift Right Logical and Shift Right Logical Short instructions are unique to the GE-PAC 30-2E.

### 4.7.3 Rotate Left Logical



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 sixteen 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
		1	0
		0	1

RESULT IS ZERO.

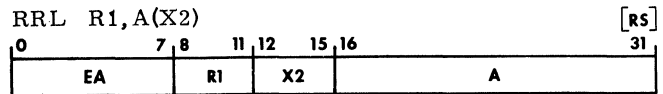
RESULT IS GREATER THAN ZERO.

RESULT IS LESS THAN ZERO.

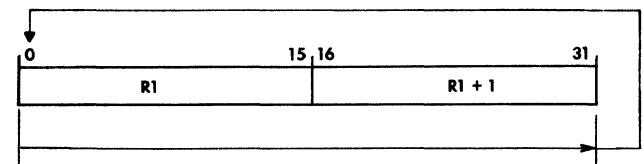
Programming Note:

The Rotate Left Logical instruction is unique to the GE-PAC 30-2E.

### 4.7.4 Rotate Right Logical



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 sixteen places interchanges the two halves (R1, R1 + 1) of the first operand.



Resulting Condition Code:

12	13	14	15
C	V	G	L
		0	0
		1	0
		0	1

RESULT IS ZERO

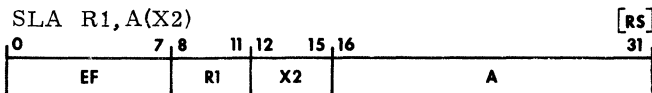
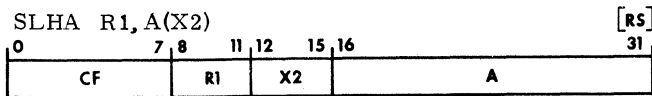
RESULT IS GREATER THAN ZERO

RESULT IS LESS THAN ZERO

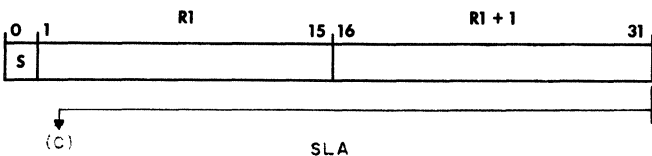
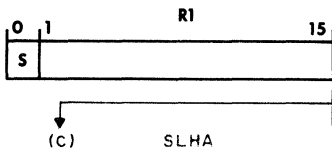
Programming Note:

The Rotate Right Logical instruction is unique to the GE-PAC 30-2E.

## 4.7.5 Shift Left Arithmetic



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 thru 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	0	0
0	0	0	1
0	0	1	0
0	1	0	0
1	0	0	0

RESULT IS ZERO

RESULT IS LESS THAN ZERO

RESULT IS GREATER THAN ZERO

LAST BIT THAT WAS SHIFTED OUT WAS A ZERO

LAST BIT THAT WAS SHIFTED OUT WAS A ONE

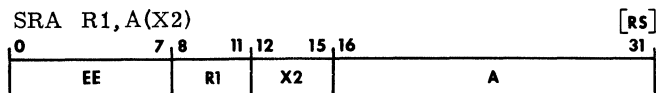
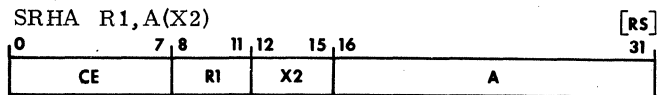
Programming Note:

For the Shift Left Halfword Arithmetic (SLHA) instruction the shift count is specified by the low order 4 bits (12 through 15) of the value of A + (X2).

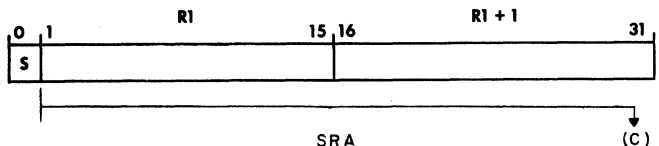
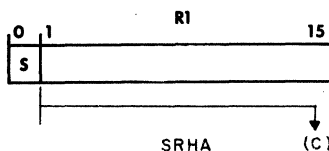
The Shift Left Arithmetic (SLA) instruction shifts Registers R1 and R1 + 1, an even odd pair. R1 must specify an even register. The shift count is specified by the low order 5-bits (11 through 15) of the value of A + (X2).

The Shift Left Arithmetic instruction is unique to the GE-PAC 30-2E.

## 4.7.6 Shift Right Arithmetic



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	0	0
0	0	0	1
0	0	1	0
0	1	0	0
1	0	0	0

RESULT IS ZERO

RESULT IS LESS THAN ZERO

RESULT IS GREATER THAN ZERO

LAST BIT THAT WAS SHIFTED OUT WAS A ZERO

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 4 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 5-bits (11 through 15) of the value of  $A + (X2)$ . The Carry is formed by the output of R1 + 1 instead of R1.

The Shift Right Arithmetic instruction is unique to the GE-PAC 30-2E.

## 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).

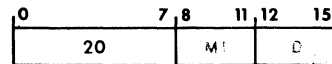
The 4-bit M1 field is not a register address, but rather an image of the condition code to be tested. The instructions described in this section are:

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

### 4.8.1 Branch on True Condition

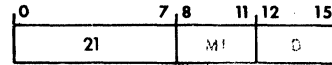
BTBS M1,D

[SF]



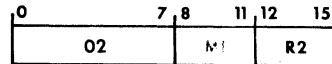
BTFS M1,D

[SF]



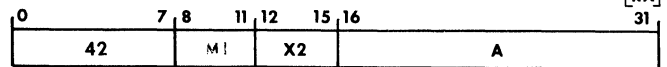
BTCCR M1,R2

[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  
 BTCCR: [ PSW (16:31) ] ← (R2)  
 BTC: [ PSW (16:31) ] ← A + (X2)

Tested Condition False:

BTBS: } [ PSW (16:31) ] ← [ PSW (16:31) ] + 2  
 BTFS: }  
 BTCCR: }  
 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 held 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 2) 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 2) is added to the present location counter to generate the address of the next instruction.

The Short Branch instructions (e.g. BTBS) are core economical for branches which specify small displacements from the present location counter. For example in sense status loops used for program controlled I/O.

Branch on true condition with a mask of 0 is a no-operation.

The Branch on True Backward Short and the Branch on True Forward Short instructions are unique for the GE-PAC 30-2E.

#### 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 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 2) 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 2) 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.

The Branch on False Condition Backward and the Branch on False Condition Forward instructions are unique to the GE-PAC 30-2E.

### 4.8.2 Branch on False Condition

BFBS M1,D [SF]

0	7,8	11,12	15
22	M1	D	

BFFS M1,D [SF]

0	7,8	11,12	15
23	M1	D	

BFCR M1,R2 [RR]

0	7,8	11,12	15
03	M1	R2	

BFC M1,A(X2) [RX]

0	7,8	11,12	15,16	31
43	M1	X2	A	

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)  $\leftarrow$  [PSW (16:31)] - 2D  
 BFFS: PSW (16:31)  $\leftarrow$  [PSW (16:31)] + 2D  
 BFCR: PSW (16:31)  $\leftarrow$  (RR)  
 BFC: PSW (16:31)  $\leftarrow$  A + (X2)

#### Tested Condition True

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

### 4.8.3 Branch on Index

BXH R1,A(X2) [RS]

0	7,8	11,12	15,16	31
CO	R1	X2	A	

BXLE R1,A(X2) [RS]

0	7,8	11,12	15,16	31
CI	R1	X2	A	

Prior to execution of this instruction, the General Register specified by the first operand (R1) must contain a 16-bit starting count 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 count (R1) to be incremented by (R1 + 1) and logically compared to the index limit, (R1 + 2).

BNH: (R1)  $\leftarrow$  (R1) + (R1 + 1)  
           (R1) : (R1 + 2)  
           if (R1) > (R1 + 2)  
           [ PSW (16:31) ]  $\leftarrow$  A + (X2)  
           if (R1)  $\leq$  (R1 + 2);  
           [ PSW (16:31) ]  $\leftarrow$  PSW (16:31) + 4

BNLE: (R1) (R1) + (R1 + 1)  
           (R1) : (R1 + 2)  
           if (R1)  $\leq$  (R1 + 2)  
           [ PSW (16:31) ]  $\leftarrow$  A + (X2)  
           if (R1) > (R1 + 2);  
           [ PSW (16:31) ]  $\leftarrow$  PSW (16:31) + 4

Resulting Condition Code:

Unchanged.

Programming Note:

For the Branch on Index High (BNH) instruction, the contents of R1 + 1 should be negative. As long as the count (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 (BNLE) instruction the contents of R1 + 1 should be positive. As long as the count (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]  

0	7	8	11	12	15
O1			R1		R2

BAL R1,A(X2) [RX]  

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

The Branch and Link instruction is executed in two phases. The instruction address field of the Program Status Word [PSW (16:31)] is incremented and transferred to the General Register specified by the first operand (R1). Then the second operand is loaded into the instruction address field [PSW (16:31)]. The next instruction executed will be accessed from the location specified by the new instruction address.

BALR: (R1)  $\leftarrow$  [ PSW (16:31) ] + 2  
           PSW (16:31)  $\leftarrow$  (R2)  
 BAL: (R1)  $\leftarrow$  [ PSW (16:31) ] + 4  
           PSW (16:31)  $\leftarrow$  A + (X2)

Condition Code:

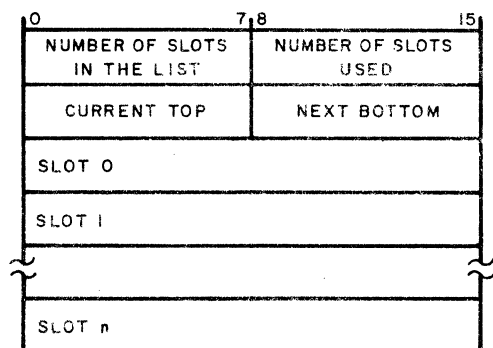
Unchanged.

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 current instruction address field 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.

## 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 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 Fig. 4.1.

The instructions described in this section are:

- ATL            Add to Top of List
- ABL            Add to Bottom of List
- RTL            Remove from Top of List
- RBL            Remove from Bottom of List

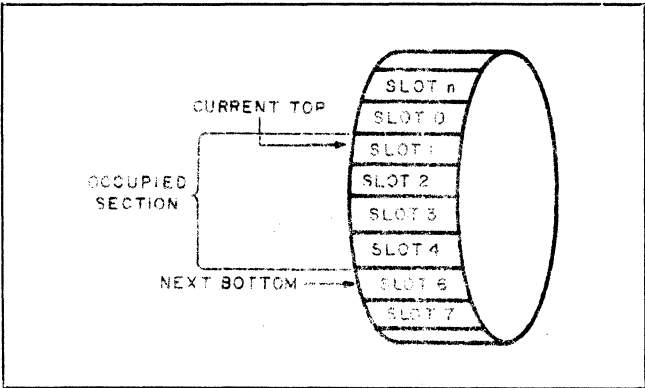


Fig. 4.1 Circular List

4.9.1 Add to Top/Bottom of List

ATL R1,A(X2)																[RX]	
0	7	8	11	12	15	16										31	
64				R1				X2				A					

ABL R1,A(X2)																[RX]	
0	7	8	11	12	15	16										31	
65				R1				X2				A					

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 greater than 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 or equal to 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 (1) 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 Top/Bottom of List instruction is unique to the GE-PAC 30-2E.

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 (1). 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.

These instructions are 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 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 Top/Bottom of List instructions are unique to the GE-PAC 30-2E.

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.

These instructions are 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 the 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 GE-PAC 30 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:

AIR	Acknowledge Interrupt RR
AI	Acknowledge Interrupt
SSR	Sense Status RR
SS	Sense Status
OCR	Output Command RR
OC	Output Command
RDR	Read Data RR
RD	Read Data
WDR	Write Data RR
WD	Write Data
RBR	Read Block RR
RB	Read Block
WBR	Write Block RR
WB	Write Block
RHR	Read Halfword RR
RH	Read Halfword
WHR	Write Halfword RR
WH	Write Halfword
AL	Autoload

### 4.10.1 Acknowledge Interrupt

AIR R1,R2						[RR]
0	7	8	11	12	15	
9F		R1		R2		

AI R1,A(X2)										[RX]
0	7	8	11	12	15	16				31
DF			R1		X2		A			

The address of the interrupting device replaces the content of the 16-bit General Register specified by the first operand (R1). The 8-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			
	1		
		1	
			1

DEVICE BUSY (BSY)  
 EXAMINE STATUS (EX) OR TIME OUT  
 END OF MEDIUM (EOM)  
 DEVICE UNAVAILABLE (DU)

Programming Note:

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

Resulting Condition Code:

12	13	14	15
C	V	G	L
1			
	1		
		1	
			1

DEVICE BUSY (BSY)  
 EXAMINE STATUS (EX) OR TIME OUT  
 END OF MEDIUM (EOM)  
 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]
0	7	8	11	12	15
9E		R1		R2	

OC R1,A(X2)																[RX]		
0	7	8	11	12	15	16										31		
DE				R1			X2			A								

The 16-bit General Register specified by the first operand (R1) contains the device address. The device is addressed and the 8-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.

### 4.10.2 Sense Status

SSR R1,R2						[RR]
0	7	8	11	12	15	
9D		R1		R2		

SS R1,A(X2)																[RX]
0	7	8	11	12	15	16										31
DD				R1		X2		A								

The 16-bit General Register specified by the first operand (R1) contains the device address. The device is addressed and the 8-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)

### 4.10.4 Read Data

RDR R1,R2						[RR]
0	7	8	11	12	15	
9B			R1	R2		

RD R1,A(X2) [RX]															
0	7	8	11	12	15	16	31								
DB				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 8-bit data byte is transmitted from the device replacing the content of the location specified by the second operand.

RDR:  $\left\{ \begin{array}{l} R2 (8:15) \\ R2 (0:7) \end{array} \right\} \leftarrow \begin{array}{l} \text{Data byte} \\ \text{Zero} \end{array}$   
 RD:  $\left\{ \begin{array}{l} R2 (8:15) \\ A + (X2) \end{array} \right\} \leftarrow \text{Data byte}$

Resulting Condition Code:

12	13	14	15
C	V	G	L
	I		

INSTRUCTION TIME OUT

Programming Note:

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

#### 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 8-bit data byte is transmitted to the device. Both operands remain unchanged.

WDR:  $\left\{ \begin{array}{l} R2 (8:15) \end{array} \right\} \longrightarrow (\text{Device})$

WD:  $\left\{ \begin{array}{l} A + (X2) \end{array} \right\} \longrightarrow (\text{Device})$

Resulting Condition Code:

12	13	14	15
C	V	G	L
	I		

INSTRUCTION TIME OUT

Programming Note:

These instructions are privileged.

#### 4.10.6 Read Block

RBR R1,R2 [RR]

0	7 8	11 12	15
97	R1	R2	

RB R1,A+(X2) [RX]

0	7 8	11 12	15 16	31
D7	R1	X2	A	

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 8-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	13	14	15
C	V	G	L
C	O	O	O
I			
I			
I			

BLOCK DATA TRANSFER COMPLETED CORRECTLY  
 DEVICE BUSY (BSY)  
 EXAMINE STATUS (EX) OR TIME OUT  
 END OF MEDIUM (EOM)  
 DEVICE UNAVAILABLE (DU)

Programming Note:

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

#### 4.10.7 Write Block

WBR R1,R2 [RR]

0	7 8	11 12	15
96	R1	R2	

WB R1,A(X2) [RX]

0	7 8	11 12	15 16	31
D6	R1	X2	A	

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 8-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
1			
	1		
		1	
			1

BLOCK DATA TRANSFER COMPLETED CORRECTLY  
 DEVICE BUSY (BSY)  
 EXAMINE STATUS (EX) OR TIME OUT  
 END OF MEDIUM (EOM)  
 DEVICE UNAVAILABLE (DU)

Programming Note:

These instructions are privileged.

#### 4.10.8 Read Halfword

RHR R1,R2 [RR]

0	7,8	11,12	15
99	R1	R2	

RH R1,A(X2) [RX]

0	7,8	11,12	15,16	31
D9	R1	X2	A	

The 16-bit General Register specified by R1 contains the device address. The device is addressed and two eight-bit bytes are received from the device replacing the contents of the second operand.

RHR: R2 (0:7) ← First Data Byte  
 R2 (8:15) ← Second Data Byte  
 RH: [A + (X2)] ← First Data Byte  
 [A + (X2) + 1] ← Second Data Byte

Resulting Condition Code:

12	13	14	15
C	V	G	L
	1		

INSTRUCTION TIME OUT

Programming Note:

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

#### 4.10.9 Write Halfword

WHR R1,R2 [RR]

0	7,8	11,12	15
98	R1	R2	

WH R1,A(X2) [RX]

0	7,8	11,12	15,16	31
D8	R1	X2	A	

The 16-bit General Register specified by R1 contains the device address. The device is addressed and two eight-bit bytes are transmitted to the device from the location specified by the second operand.

WHR: [R2 (0:7)] → Device  
 [R2 (8:15)] → Device  
 WH: [A + (X2)] → Device  
 [A + (X2) + 1] → Device

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, sixteen-line Interrupt Module, conversion equipment (i. e. D/A and A/D Converters), card reader, and display panel.

These instructions are privileged.

#### 4.10.10 Autoload

AL A(X2) [RX]

0	7,8	11,12	15,16	31
D5	0	X2	A	

The Autoload instruction loads memory with a block of data from a byte oriented input device (e. g. Teletype, 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. (X'80' n = 0) ← byte #n
2. n ← n + 1
3. (X'80' + n) ← byte #n
4. If A + (X2) = X'80' + n, instruction is finished, otherwise return to equation 2.



Resulting Condition Code:

(2)	(3)	(4)	(5)	
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 must contain 0.

## 4.11 SYSTEM CONTROL INSTRUCTIONS

The set of System Control instructions provide a means for the program to set the Program Status Word, swap PSWs, 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). The instructions described in this section are:

LPSW	Load Program Status Word
EPSR	Exchange Program Status
SINT	Simulate Interrupt
SVC	Supervisor Call

### 4.11.1 Load Program Status Word

LPSW A(X2)				[RX]
0	7, 8	11, 12	15, 16	31
C2		X2		A

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

$[PSW(0:31)] \leftarrow [A + (X2)]$

Resulting Condition Code:

Determined by PSW loaded by the instruction.

Programming Note:

This instruction is privileged.

The R1 field of a load PSW instruction should contain 0.

### 4.11.2 Exchange Program Status

EPSR R1, R2

0	7, 8	11, 12	15
95	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)  $\longrightarrow$  R1  
PSW (0:15)  $\longleftarrow$  R2

Resulting Condition Code:

Determined by New PSW.

Programming Note:

This instruction is privileged.

The Exchange Program Status instruction is unique to the GE-PAC 30-2E.

### 4.11.3 Simulate Interrupt

SINT A(X2)				[RS]
0	7, 8	11, 12	15, 16	31
E2		X2		A

The least significant 8-bit of the second operand, A + (X2), is presented to the Interrupt Handler as a device number. The device number indexes the Service Pointer Table at X'00D0' and results in either an Immediate Interrupt or an I/O operation.

Programming Note:

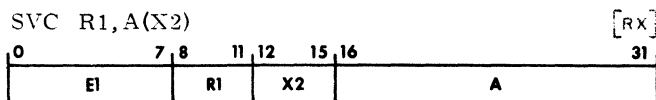
This instruction is privileged.

The Simulate Interrupt instruction is unique to the GE-PAC 30-2E.

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

#### 4.11.4 Supervisor Call

SVC R1,A(X2)



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'0090' through X'008B' 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'.

$(X'0094') \leftarrow A + (X2)$   
 $(X'0096') \leftarrow \text{PSW } (0:31)$   
 $(X'009A) \leftarrow \text{PSW } (0:15)$   
 $(X'009C + 2 * R1) \leftarrow \text{PSW } (16:31)$

Resulting Condition Code:

Defined by New PSW.

Programming Note:

The Supervisor Call instruction is unique to the GE-PAC 30-2E.

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.

# SECTION 5

## CONSOLE OPERATING PROCEDURES

### 5.1 INTRODUCTION

The GE-PAC 30 display panel and the various controls associated with it, are shown in Fig. 5.1. The control console includes the following:

1. Control Switches: OFF-ON, INT and EXE
2. Mode Controls: RUN, ADR, etc.
3. Sixteen latching Data/Address Switches
4. Display Control Switch
5. Two 16-bit Display Registers

### 5.2 CONTROL SWITCHES

The OFF-ON switch controls power to the Processor and peripheral device controllers. Associated with the switch is a POWER indicator lamp in the lower left corner of the Display Panel. Whenever power is applied, the POWER indicator is illuminated.

The momentary INT switch is used to bring the Processor and device controllers to an initial state. Depressing this switch turns off power to the system for approximately a 2-second interval. When the power is restored, the initial state for the system is established. This switch is normally used when the Processor is in a Halt or Wait condition. The effect of the INT switch on a running program is discussed in Section 5.6. The momentary EXE switch causes the Processor to perform the function specified by the Mode Control switch as discussed in the next section.

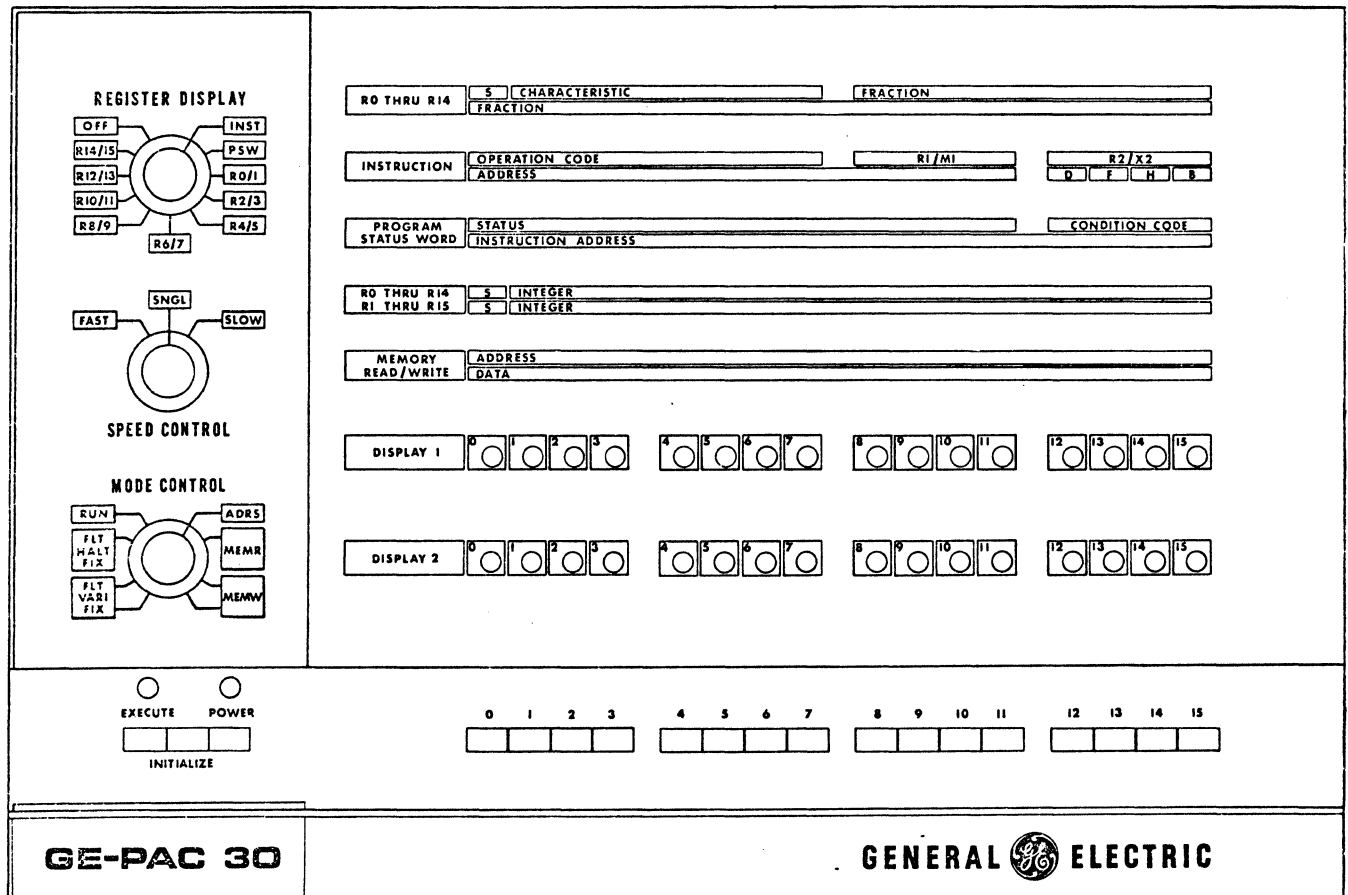


Fig. 5.1 Display Panel

## 5.3 MODE CONTROL

The Mode Control switch is labeled VARI, RUN, ADRS, MEMR, MEMW, and HALT. The Processor is controlled by selecting the appropriate switch position, and then depressing the EXE switch to activate the function. The meaning of each of the latching Mode Control switches is as follows:

### VARI:

The Variable switch specifies variable speed execution of programs. When VARI is selected, and the SPEED CONTROL is in the single position, the Processor executes a single instruction each time the EXE switch is depressed. When VARI is selected, and the SPEED CONTROL is between FAST and SLOW, the Processor executes instructions at a variable rate from 1 to 1000 instructions per second. The speed is governed by the Speed Control knob. The Display Registers are active during either single or variable rate execution. The Display Registers are updated as specified by the Display Control Switch following the execution of each instruction.

### RUN:

The RUN position specifies program execution at normal speed. Note that in this case, the Display Registers are not activated or controlled by the Processor. Rather, during normal program execution, the display panel is available for use as an I/O device. If the running program does not output to the Display Registers, then these registers retain the last value displayed prior to starting the execution of the program.

### ADRS:

The Address position is used to enter a 16-bit address from the Data/Address switches into the address portion of the Current Program Status Word PSW (16:31). The address specified can be used to read data from memory, write data into memory, or start the execution of a program. When the address is transferred from the switches to the PSW, the least significant bit (PSW bit 31) is cleared so that the resulting address is always even. Display Registers 1 and 2 reflect the New PSW contents if the PSW display position is selected.

### MEMR:

The Memory Read position is used to read data from the specified address in memory. After EXE is depressed, the memory address is shown in Display Register 1 (top) and the contents of that location are shown in Display Register 2 (bottom). After the values are displayed, the address portion of the PSW is incremented by 2. Depressing EXE repeatedly displays consecutive locations from core memory.

### MEMW:

The Memory Write position is used to enter data from the Data/Address switches into a specified location in memory. After EXE is depressed, the memory address is shown in Display Register 1 (top) and the data written into that location is shown in Display Register 2 (bottom). After the values are displayed, the address portion of the PSW is incremented by 2. Depressing EXE repeatedly writes data into consecutive memory locations.

### HALT:

To stop program execution, select the HALT position and depress EXE. This action puts the GE-PAC 30-2E into a Halt Mode, which is non-interruptible. Performing ADRS, MEMR, or MEMW operations also puts the Processor into the Halt Mode. In the Halt Mode, the Wait indicator over the EXE switch is illuminated. The Wait indicator is also on between instructions during either single or variable speed execution, or when an executing program enters the Wait state by setting bit 0 of the PSW. Note that the program-initiated Wait state is interruptible, while the console-initiated Halt Mode is not interruptible.

## 5.4 DISPLAY REGISTERS

The two 16-bit Display Registers are used as described above for Address, Memory Read, and Memory Write operations. Whenever EXE is depressed with Mode Control switch on the Halt Mode, or following each instruction during either single or variable speed execution, the Display Registers are used as specified by the Display Control Switch. The Display Control Switch Positions are INS, PSW, 0-1, 2-3, 4-5, 6-7, 8-9, A-B, C-D, and E-F. These positions select the registers within the Processor to be dis-

played in the Display Registers. The information displayed is as follows:

INS:

When the Instruction position is selected, the Processor displays the contents of two consecutive halfwords from memory as specified by the address portion of the PSW. This information is the next instruction to be executed if program execution is in progress. The first halfword from memory is shown in Display Register 1, and the second halfword is shown in Display Register 2.

PSW:

When the PSW position is selected, the Processor displays the contents of the Current Program Status Word. Display Register 1 contains the program status and condition code, and Display Register 2 contains the location counter.

0-1, 2-3, . . . E-F:

When the FLT on the Mode Switch is not selected, these switches cause the Processor to display the General Registers as indicated on the switch. For example: switch 2-3 causes General Register 2 to be shown in Display Register 1, and General Register 3 to be shown in Display Register 2. When the FLT on the Mode Switch is selected, these switches refer to the 32-bit Floating-Point Register indicated by the even number on the switch. For example: switch E-F causes the most signif-

icant half of floating register E to be shown in Display Register 1, and the least significant half to be shown in Display Register 2.

When the Display Control Switch is in the OFF position, the Display Registers are considered OFF by the Processor, and no information is displayed. In this case, the Display Registers may still reflect data written to the display panel from a running program. The OFF position is also used in relation to console interrupts. See Section 5.6.

## 5.5 OPERATING PROCEDURES

### 5.5.1 Initialization

To bring up power and initialize the system:

1. Place the Mode Control switch in the HALT position.
2. Push the Power Switch in.
3. Depress the momentary Initialize (INT) switch.

This action establishes electrical power, and leaves the Processor in the Halt Mode. Before the system can be used, it is necessary to initialize a few important pointers and New PSWs in core memory. The locations in memory to be adjusted are shown in Table 5.1.

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.

Table 5.1 Memory Core Locations

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

1. Enter 0022 (0000 0000 0010 0010) into the Data/Address switches, select Mode Control switch position ADRS, and depress EXE.
2. Enter 0058 (0000 0000 0101 1000) into the Data/Address switches, select Mode Control switch position MEMW, and depress EXE. This enters value 0058 into location 0022.
3. Enter 0034 into the Data/Address switches, select Mode Control switch position ADRS, and depress EXE.
4. Enter 8000 into the Data/Address switches, select Mode Control switch position MEMW 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 the above locations are set, their contents can be verified using Memory Read operations as follows:

1. Enter 0022 into the Data/Address switches, select Mode Control switch position ADRS, and depress EXE.
2. Select Mode Control switch position MEMR, and depress EXE. At this point, the address (0022) should be displayed in Display Register 1, and the contents (0058) should be displayed in Display Register 2.
3. Enter 0034 into the Data/Address switches, select Mode Control switch position ADRS, and depress EXE.
4. Select Mode Control switch position MEMR, and depress EXE. At this point, Display Register 1 should show the address (0034) and Display Register 2 should show its contents (8000).
5. To examine the next location, depress EXE. At that time, Display Register 1 should show the address (0036), and Display Register 2 should show the contents (0050).
6. Follow similar steps until all appropriate locations have been verified.

Once core 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 ADRS Mode Control, there is no way to directly adjust the program status PSW (0:15) from the display 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, select Mode Control switch position ADRS, and depress EXE.
2. Select Mode Control switch position RUN, 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.

## 5.5.2 Program Loading

There are many ways to load the GE-PAC 30-2E memory with programs and/or data. Most programs are loaded using one of the program loaders associated with the GE-PAC 30-2E system software. Refer to programming publications for details on loaders or 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 in Table 5.2.

This sequence is based on the Auto-load instruction, which is described in Section 4.10. This instruction reads 8-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

Location	Contents	Instruction		
50	D500 00CF	AL	O, X'CF'	AUTO LOAD
54	4300 0080	B	X'80'	BRANCH TO 80
78	XXYY	DC	X'XXYY'	DEV NO AND CMND

Table 5.2 Core Memory Description

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 teletype, paper type, or magnetic tape devices that transfer 8-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 the specified program length was loaded. A non-zero condition code implies the device terminated the load sequence before the program length was satisfied.

### 5.5.3 Program Execution

To start a program executing at normal speed, two steps are required:

1. Enter the starting address of the program into the Data/Address switches, select the Mode Control switch position ADRS, and depress EXE.
2. Select the Mode Control switch position RUN and depress EXE.

To halt the execution of a running program, select Mode Control switch position HALT and depress EXE. When the Processor enters the Halt Mode, the Display Registers are updated as specified by the Display Control Switch. For example: if EXE is depressed with the Mode Control switch position HALT, and the

Display Control Switch position PSW, execution is halted and the Current PSW is displayed in the Display Registers. Each time EXE is depressed in the Halt Mode, the Display Registers are updated as specified by the Display Control Switch. Therefore, to alter the Display Registers, once the Processor is in the Halt Mode, simply change the Display Control Switch and depress EXE to change the data displayed.

To execute a program a single instruction at a time, the following steps are required:

1. Enter the starting address of the program into the Data/Address switches, select the Mode Control switch position ADRS, and depress EXE.
2. Select the Mode Control switch position VARI, and depress EXE. This executes one instruction at the address specified, and returns the Processor to the Halt Mode with the Display Registers updated as specified by the Display Control Switches.
3. Depress EXE for each subsequent instruction of the program to be executed.

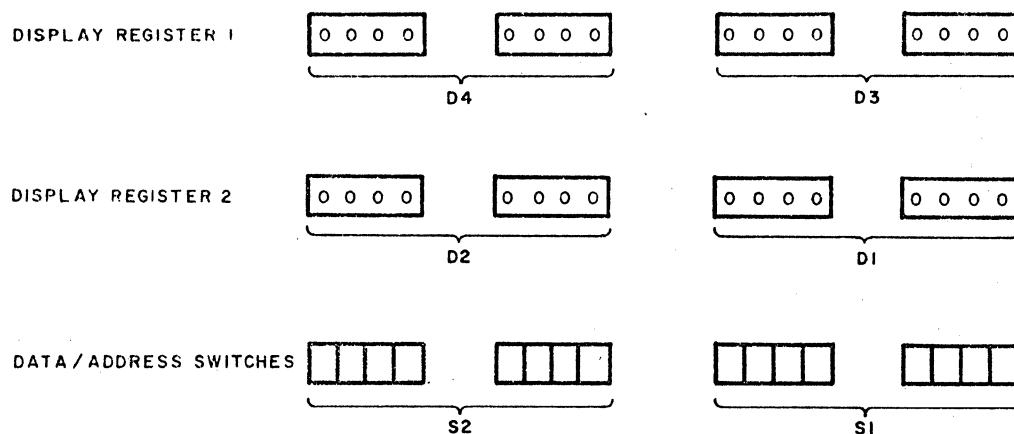
At any time during the single-step sequence, the Display Control Switch can be adjusted to change the selection of registers to be displayed. After any instruction execution, to examine more than one register without executing more instructions, place the Mode Control switch in the Halt Mode. Then select the Display Control Switch position desired and depress EXE to look at other registers. Place the Mode Control Switch in VARI to resume single-step execution.

## 5.6 PROGRAMMING CONSIDERATIONS

### 5.6.1 Display Panel I/O

The Display Panel is available to any running program as an I/O device with device address 01. The status and command bytes for the display panel are summarized in Appendix 8. The status byte for the display indicates the setting of the Mode Control and the Dis-

play Control 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 Display Registers is transferred -- is reset every time the Display Panel is addressed on the Multiplexor Bus. The Display Panel is addressed by every I/O instruction using device address 01. Subsequent read or write instructions transfer subsequent bytes as shown in Fig. 5.2. Normal I/O instructions, therefore, can be used to input data from the Data/Address switches, and output data to the display registers.



Instructions Executed (RR or RX)	Data Transferred	
	Normal Mode	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

Fig. 5.2 Display Panel/Data Transferred



### 5.6.2 Console Interrupt

In the GE-PAC 30-2E, an interrupt can be generated from the display panel 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 Mode Control Switch in the RUN position, and the Display Control Switch OFF, 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.

### 5.6.3 Wait State

The running program can put the GE-PAC 30-2E Processor into the Wait State by setting bit 0 of the Current PSW. The operator is informed of this action by the Wait indicator 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 execution of a routine to service the interrupt. When the routine restores the original PSW, the Wait State will be re-established.
2. The operator selects RUN and depress EXE on the display panel, which causes 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 or variable-speed executions. That is, with single-step execution it is possible to "step through" the Wait State inadvertently through rapid or continuous use of the EXE switch. Similarly, variable-speed execution results from automatic generation of EXE signals from a hardware timer, and the same phenomenon can occur.

### 5.6.4 Power Fail

Depressing the INT switch of the display panel causes removal of electrical power from the system for a brief interval (about 2 seconds). When power is removed, 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 core memory. When power is restored, the PSW and General Registers are re-loaded from core memory, and the device controllers are all initialized. At this point, the Processor interrogates the Display Panel mode control switches. If Mode Control switch RUN is not selected, the Processor enters the Halt Mode and the Wait indicator is illuminated. If Mode Control switch RUN is selected, the Processor examines bit 2 of the restored Current PSW. If 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 fail/restore has occurred. The program must compare the Old PSW for machine malfunctions with the PSW save area at 0024 to make this determination. 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 on the display panel should be considered carefully when bit 2 of the Current PSW is enabled.



# APPENDIX 1

## INSTRUCTION SUMMARY - ALPHABETICAL

INSTRUCTION	OP CODE	MNEMONIC
Acknowledge Interrupt	DF	AI
Acknowledge Interrupt RR	9F	AIR
Add Halfword	4A	AH
Add Halfword Immediate	CA	AHI
Add Halfword RR	0A	AHR
◆ Add Halfword Memory	61	AHM
◆ Add Immediate Short	26	AIS
◆ Add to Bottom of List	65	ABL
◆ Add to Top of List	64	ATL
Add with Carry Halfword	4E	ACH
Add with Carry Halfword RR	0E	ACHR
AND Halfword	44	NH
AND Halfword Immediate	C4	NHI
AND Halfword RR	04	NHR
Autoload	D5	AL
Branch and Link	41	BAL
Branch and Link RR	01	BALR
*Branch on False Condition	43	BFC
*Branch on False Condition RR	03	BFCR
*Branch on True Condition	42	BTC
*Branch on True Condition RR	02	BTCR
◆ Branch on True Backword Short	20	BTBS
◆ Branch on True Forward Short	21	BTFS
◆ Branch on False Backword Short	22	BFBS
◆ Branch on False Forward Short	23	BFFS
Branch on Index High	C0	BXH
Branch on Index Low or Equal	C1	BXLE

\*See Extended Branch Mnemonics Section for thirty (30) additional symbolic instructions for the GE-PAC 30-2E.

◆ Extended GE-PAC 30-2E Feature.

INSTRUCTION	OP CODE	MNEMONIC
◆ Compare Halfword	49	CH
◆ Compare Halfword Immediate	C9	CHI
◆ Compare Halfword RR	09	CHR
◆ Compare Logical Byte	D4	CLB
Compare Logical Halfword	45	CLH
Compare Logical Halfword Immediate	C5	CLHI
Compare Logical Halfword RR	05	CLHR
Divide Halfword	4D	DH
Divide Halfword RR	0D	DHR
◆ Exchange Byte RR	94	EXBR
◆ Exchange Program Status RR	95	EPSR
Exclusive OR Halfword	47	XH
Exclusive OR Halfword Immediate	C7	XHI
Exclusive OR Halfword RR	07	XHR
Floating - Point Add	6A	AE
Floating - Point Add RR	2A	AER
Floating - Point Compare	69	CE
Floating - Point Compare RR	29	CER
Floating - Point Divide	6D	DE
Floating - Point Divide RR	2D	DER
Floating - Point Load	68	LE
Floating - Point Load RR	28	LER
Floating - Point Multiply	6C	ME
Floating - Point Multiply RR	2C	MER
Floating - Point Store	60	STE
Floating - Point Subtract	6B	SE
Floating - Point Subtract RR	2B	SER
Load Byte	D3	LB
Load Byte RR	93	LBR
◆ Load Complement Short	25	LCS

◆ Extended GE-PAC 30-2E Feature.

INSTRUCTION	OP CODE	MNEMONIC
Load Halfword	48	LH
Load Halfword Immediate	C8	LHI
Load Halfword RR	08	LHR
◆ Load Immediate Short	24	LIS
Load Multiple	D1	LM
Load Program Status Word	C2	LPSW
Multiply Halfword	4C	MH
Multiply Halfword RR	0C	MHR
◆ Multiply Halfword Unsigned	DC	MHU
◆ Multiply Halfword Unsigned RR	9C	MHUR
OR Halfword	46	OH
OR Halfword Immediate	C6	OHI
OR Halfword RR	06	OHR
Output Command	DE	OC
Output Command RR	9E	OCR
Read Block	D7	RB
Read Block RR	97	RBR
Read Data	DE	RD
Read Data RR	9B 9B	RDR
◆ Read Halfword	D9	RH
◆ Read Halfword RR	99	RHR
◆ Rotate Left Logical	EB	RLL
◆ Rotate Right Logical	EA	RRL
◆ Remove from Bottom of List	67	RBL
◆ Remove from Top of List	66	RTL
Sense Status	DD	SS
Sense Status RR	9D	SSR
◆ Shift Left (Fullword) Arithmetic	EF	SLA
◆ Shift Left (Fullword) Logical	ED	SLL
Shift Left (Halfword) Arithmetic	CF	SLHA
Shift Left (Halfword) Logical	CD	SLHL
◆ Shift Left Logical Short	91	SLLS
◆ Shift Right (Fullword) Arithmetic	EE	SRA
◆ Shift Right (Fullword) Logical	EC	SRL
Shift Right (Halfword) Arithmetic	CE	SRHA
Shift Right (Halfword) Logical	CC	SRHL
◆ Shift Right Logical Short	90	SRLS

◆ Extended GE-PAC 30-2E Feature.

INSTRUCTION	OP CODE	MNEMONIC
◆ Simulate Interrupt	E2	SINT
Store Byte	D2	STB
Store Byte RR	92	STBR
Store Halfword	40	STH
Store Multiple	D0	STM
Subtract Halfword	4B	SH
Subtract Halfword Immediate	CB	SHI
Subtract Halfword RR	0B	SHR
◆ Subtract Immediate Short	27	SIS
Subtract with Carry Halfword	4F	SCH
Subtract with Carry Halfword RR	0F	SCHR
◆ Supervisor Call	E1	SVC
◆ Test Halfword Immediate	C3	THI
Write Block	D6	WB
Write Block RR	96	WBR
Write Data	DA	WD
Write Data RR	9A ✓	WDR
◆ Write Halfword	D8	WH
◆ Write Halfword RR	98	WHR

◆ Extended GE-PAC 30-2E Feature.

## APPENDIX 2

### INSTRUCTION SUMMARY - NUMERICAL

OP CODE	MNEMONIC	INSTRUCTION
01	BALR	Branch and Link RR
02	BTCR	Branch on True Condition RR
03	BFCR	Branch on False Condition RR
04	NHR	AND Halfword RR
05	CLHR	Compare Logical Halfword RR
06	OHR	OR Halfword RR
07	XHR	Exclusive OR Halfword RR
08	LHR	Load Halfword RR
09	CHR	Compare Halfword RR
0A	AHR	Add Halfword RR
0B	SHR	Subtract Halfword RR
0C	MHR	Multiply Halfword RR
0D	DHR	Divide Halfword RR
0E	ACHR	Add with Carry Halfword RR
0F	SCHR	Subtract with Carry Halfword RR
20	BTBS	Branch on True Backward Short
21	BTFS	Branch on True Forward Short
22	BFBS	Branch on False Backward Short
23	BFBS	Branch on False Forward Short
24	LIS	Load Immediate Short
25	LCS	Load Complement Short
26	AIS	Add Immediate Short
27	SIS	Subtract Immediate Short
28	LER	Floating-Point Load RR
29	CER	Floating-Point Compare RR
2A	AER	Floating-Point Add RR
2B	SER	Floating-Point Subtract RR
2C	MER	Floating-Point Multiply RR
2D	DER	Floating-Point Divide RR
40	STH	Store Halfword
41	BAL	Branch and Link
42	BTC	Branch on True Condition
43	BFC	Branch on False Condition
44	NH	AND Halfword
45	CLH	Compare Logical Halfword
46	OH	OR Halfword
47	XH	Exclusive OR Halfword
48	LH	Load Halfword
49	CH	Compare Halfword
4A	AH	Add Halfword
4B	SH	Subtract Halfword
4C	MH	Multiply Halfword

OP CODE	MNEMONIC	INSTRUCTION
4D	DH	Divide Halfword
4E	ACH	Add with Carry Halfword
4F	SCH	Subtract with Carry Halfword
60	STE	Floating-Point Store
61	AHM	Add Halfword Memory
64	ATL	Add to Top of List
65	ABL	Add to Bottom of List
66	RTL	Remove from Top of List
67	RBL	Remove from Bottom of List
68	LE	Floating-Point Load
69	CE	Floating-Point Compare
6A	AE	Floating-Point Add
6B	SE	Floating-Point Subtract
6C	ME	Floating-Point Multiply
6D	DE	Floating-Point Divide
90	SRLS	Shift Right Logical Short
91	SLLS	Shift Left Logical Short
92	STBR	Store Byte RR
93	LBR	Load Byte RR
94	EXBR	Exchange Byte RR
95	EPSR	Exchange Program Status RR
96	WBR	Write Block RR
97	RBR	Read Block RR
98	WHR	Write Halfword RR
99	RHR	Read Halfword RR
9A	WDR	Write Data RR
9B	RDR	Read Data RR
9C	MHUR	Multiply Halfword Unsigned RR
9D	SSR	Sense Status RR
9E	OCR	Output Command RR
9F	AIR	Acknowledge Interrupt RR
CO	BXH	Branch on Index High
C1	BXLE	Branch on Index Low or Equal
C2	LPSW	Load Program Status Word
C3	THI	Test Halfword Immediate
C4	NHI	AND Halfword Immediate
C5	CLHI	Compare Logical Halfword Immediate
C6	OHI	OR Halfword Immediate
C7	XHI	Exclusive OR Halfword Immediate
C8	LHI	Load Halfword Immediate
C9	CHI	Compare Halfword Immediate
CA	AHI	Add Halfword Immediate
CB	SHI	Subtract Halfword Immediate
CC	SRHL	Shift Right (Halfword) Logical



OP CODE	MNEMONIC	INSTRUCTION
CD	SLHL	Shift Left (Halfword) Logical
CE	SRHA	Shift Right (Halfword) Arithmetic
CF	SLHA	Shift Left (Halfword) Arithmetic
DO	STM	Store Multiple
D1	LM	Load Multiple
D2	STB	Store Byte
D3	LB	Load Byte
D4	CLB	Compare Logical Byte
D5	AL	Auto Load
D6	WB	Write Block
D7	RB	Read Block
D8	WH	Write Halfword
D9	RH	Read Halfword
DA	WD	Write Data
DB	RD	Read Data
DC	MHU	Multiply Halfword Unsigned
DD	SS	Sense Status
DE	OC	Output Command
DF	AI	Acknowledge Interrupt
E1	SVC	Supervisor Call
E2	SINT	Simulate Interrupt
EA	RRL	Rotate Right Logical
EB	RLL	Rotate Left Logical
EC	SRL	Shift Right (Fullword) Logical
ED	SLL	Shift Left (Fullword) Logical
EE	SRA	Shift Right (Fullword) Arithmetic
EF	SLA	Shift Left (Fullword) Arithmetic



# 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	028	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 RR	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	022	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	033	BNZR	R2
No Operation	4200	NOP	
No Operation RR	0200	NOPR	

# EXTENDED SHORT BRANCH MNEMONICS



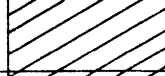
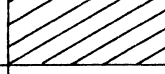
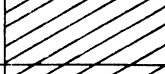
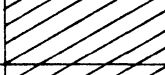
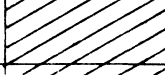
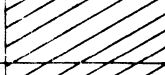
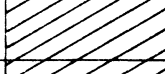
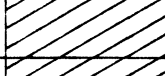
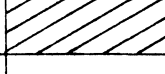


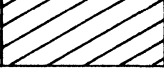
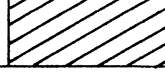
INSTRUCTION	OP CODE	MNEMONIC <sup>(1)</sup>	OPERANDS <sup>(2)</sup>
Branch Absolute Back	220	BABS	*-A
Branch Absolute Forward	230	BAFS	A-*
Branch on Zero Back	223	BZBS	*-A
Branch on Zero Forward	233	BZFS	A-*
Branch on Not Zero Back	203	BNZBS	*-A
Branch on Not Zero Forward	213	BNZFS	A-*
Branch on Plus Back	202	BPBS	*-A
Branch on Plus Forward	212	BPFS	A-*
Branch on Not Plus Back	222	BNPBS	*-A
Branch on Not Plus Forward	232	BNPFS	A-*
Branch on Minus Back	201	BMBS	*-A
Branch on Minus Forward	211	BMFS	A-*
Branch on Not Minus Back	221	BNMBS	*-A
Branch on Not Minus Forward	231	BNMFS	A-*
Branch on Carry Back	208	BCBS	*-A
Branch on Carry Forward	218	BCFS	A-*
Branch on Overflow Back	204	BOBS	*-A
Branch on Overflow Forward	214	BOFS	A-*
Branch on Low Back	208	BLBS	*-A
Branch on Low Forward	218	BLFS	A-*
Branch on Not Low Back	228	BNLBS	*-A
Branch on Not Low Forward	238	BNLFS	A-*
Branch on Equal Back	223	BEBS	*-A
Branch on Equal Forward	233	BEFS	A-*
Branch on Not Equal Back	203	BNEBS	*-A
Branch on Not Equal Forward	213	BNEFS	A-*

## Notes:

1. These Mnemonics are not recognized by the Stand Alone Assembler.
2. The Operand may be a displacement number or the present location minus a label or a label minus the present location.

# APPENDIX 4

## OP CODE MAP

	0	2	4	6	9	C	D	E
0		BTBS	STH	STE	SRLS	BXH	STM	
1	BALR	BTFS	BAL	AHM	SLLS	BXLF	LM	SVC
2	BTCR	BFBS	BTC		STBR	LPSW <sup>P</sup>	STB	SINT <sup>P</sup>
3	BFCR	BFBS	BFC		LBR	THI	LB	
4	NHR	LIS	NH	ATL	EXBR	NHI	CLB	
5	CLHR	LCS	CLH	ABL	EPSR <sup>P</sup>	CLHI	AL <sup>P</sup>	
6	OHR	AIS	OH	RTL	WBR <sup>P</sup>	OHI	WB <sup>P</sup>	
7	XHR	SIS	XH	RBL	RBR <sup>P</sup>	XHI	RB <sup>P</sup>	
8	LHR	LER	LH	LE	WHR <sup>P</sup>	LHI	WH <sup>P</sup>	
9	CHR	CER	CH	CE	RHR <sup>P</sup>	CHI	RH <sup>P</sup>	
A	AHR	AER	AH	AE	WDR <sup>P</sup>	AHI	WD <sup>P</sup>	RRL
B	SHR	SER	SH	SE	RDR <sup>P</sup>	SHI	RD <sup>P</sup>	RLL
C	MHR	MER	MH	ME	MHUR	SRHL	MHU	SRL
D	DHR	DER	DH	DE	SSR <sup>P</sup>	SLHL	SS <sup>P</sup>	SLL
E	ACHR		ACH		OCR <sup>P</sup>	SRHA	OC <sup>P</sup>	SRA
F	SCHR		SCH		AIR <sup>P</sup>	SLHA	AI <sup>P</sup>	SLA
	RR	RR	RX	RX	RR	RS	RX	RS

P = Privileged Instructions



## APPENDIX 5

### INSTRUCTION EXECUTION TIMES

	<u>RR or SF</u>	<u>RS</u>	<u>RS Indexed</u>	<u>RX</u>	<u>RX Indexed</u>	
ABL				8.4/23.6/23.6	8.8/24.0/24.0	OVF/NORM/WRAP
ACH	3.6			6.0	6.4	
AE	47.6/54.4/72.4			46.8/53.6/71.6	47.2/54.0/72.0	MIN/AVE/MAX
AH	3.2	4.0	6.0	5.6	6.0	
AHM				6.8	7.2	
AI	6.8			9.2	9.6	
AIS	4.4					
AL				11.2+9.2n	11.6+9.2n	n=no. of bytes
ATL				8.4/21.6/22.0	8.8/22.0/22.4	OVF/NORM/WRAP
BAL	4.8			6.0	6.4	
BFBS	4.4/8.8					No BR/BR
BFC	4.8/4.8			6.0/6.0	6.4/6.4	No BR/BR
BFFS	4.4/8.4					No BR/BR
BTBS	4.8/9.2					No BR/BR
BTC	4.4/5.2			5.6/6.4	6.0/6.8	No BR/BR
BTFS	4.8/8.8					No BR/BR
BXH		10.4/11.2	12.4/13.2			No BR/BR
BXLE		11.2/11.2	13.2/13.2			No BR/BR
CE	20.0/23.2/28.0			19.2/22.4/27.2	19.6/22.8/27.6	+, -/+, +/-, -
CH	7.2/6.8	6.8/6.4	8.8/8.4	8.4/8.0	8.8/8.4	Signs differ/ Signs alike
CLB				6.8	7.2	
CLH	3.2	4.0	6.0	5.6	6.0	
DE	182.0/204.0/239.6			181.2/203.2/238.8	181.6/203.6/239.2	MIN/AVE/MAX
DH	38.0/39.2/41.6			39.2/40.4/42.8	39.6/40.8/43.2	MIN/AVE/MAX
EPSR	7.6					
EXBR	2.8					
LB	3.2			5.2	5.6	
LCS	4.0					
LE	24.8/28.8/35.6			25.2/29.2/36.0	25.6/29.6/36.4	MIN/AVE/MAX
LH	2.8	3.2	5.2	4.8	5.2	
LIS	3.6					
LM				8.0+3.2n	8.4+3.2n	n=no. of regs.
LPSW		8.8	10.8			
ME	131.2/144.0/173.6			130.4/143.2/172.8	130.8/143.6/173.2	MIN/AVE/MAX
MH	22.8/28.4/36.8			24.0/29.6/38.0	24.4/30.0/38.4	MIN/AVE/MAX
MHU	18.4/24.8/30.8			20.0/25.6/32.0	20.4/26.0/32.4	MIN/AVE/MAX
NH	2.8	3.6	5.6	5.2	5.6	
OC	5.6			6.8	7.2	
OH	2.8	3.6	5.6	5.2	5.6	
RB	14.0+6.4n			12.8+6.4n	13.2+6.4	n=no. of bytes
RBL				8.8/22.8/23.2	9.2/23.2/23.6	OVF/NORM/WRAP
RD	5.6			8.0	8.4	
RH	7.2			9.6	10.0	
RLL		10.4+2n	12.4+2n			n=no. of shifts Note 1
RRL		10.4+2n	12.4+2n			n=no. of shifts Note 1
RTL				8.8/24.4/24.4	9.2/24.8/24.8	OVF/NORM/WRAP

	<u>RR or SF</u>	<u>RS</u>	<u>RS Indexed</u>	<u>RX</u>	<u>RX Indexed</u>	
SCH	3.6			6.0	6.4	
SE	47.6/62.8/72.8			46.8/62.0/72.0	47.2/62.4/72.4	MIN/AVE/MAX
SH	3.2	4.0	6.0	5.6	6.0	
SINT				See Note 5		
SIS	4.8					
SLA		10.4+1.6n	12.4+1.6n			n=no. of shifts Note 2
SLHA		5.2+.4n	7.2+.4n			n=no. of shifts
SLHL		4.0+.4n	6.0+.4n			n=no. of shifts
SLL		10.4+1.6n	12.4+1.6n			n=no. of shifts Note 3
SLSL	4.4+.4n					n=no. of shifts
SRA		10.4+2n	12.4+2n			n=no. of shifts Note 4
SRHA		4.8+.4n	6.8+.4n			n=no. of shifts
SRHL		4.0+.4n	6.0+.4n			n=no. of shifts
SRL		10.4+1.6n	12.4+1.6n			n=no. of shifts Note 2
SRSL	4.4+.4n					n=no. of shifts
SS	6.4			8.0	8.4	
STB	4.4			6.4	6.8	
STE				12.4	12.8	
STH				6.4	6.8	
STM				8.0+3.6n	8.4+3.6n	n=no. of regs.
SVC		14.0	15.6			
THI		3.6	5.6			
WB	13.2+6.0n			12.0+6.0n	12.4+6.0n	n=no. of bytes
WD	5.6			6.8	7.2	
WH	6.8			8.4	8.8	
XH	2.8	3.6	5.6	5.2	5.6	

Note 1: Add .4 if  $n > 0$ , add .8 if  $n > 16$

Note 2: Add 3.2 if  $n > 0$ , add 6.4 if  $n > 16$

Note 3: Add .8 if  $n > 0$ , add 1.6 if  $n > 16$

Note 4: Add 1.2 if  $n > 0$ , add 2.4 if  $n > 16$

Note 5: SINT Execution times same as Automatic I/O processing times. Add 2.0 if indexed.

#### NOTE

These times assume the standard 1 microsecond core memory with no interference from a Selector Channel or any other device on the memory bus.



# 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 <sup>n</sup>							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 ADDITION TABLE

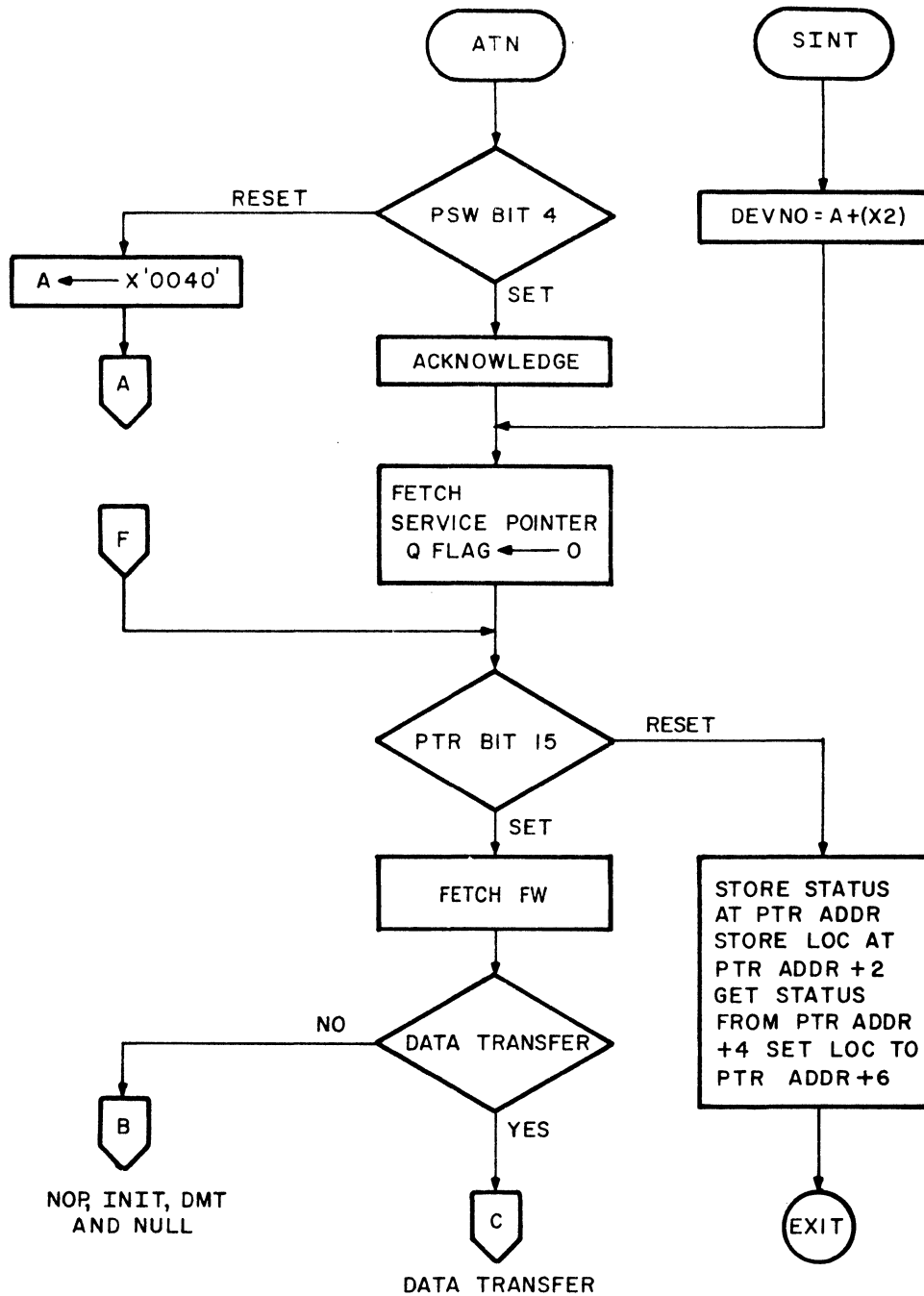
	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	
1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	10	1
2	3	4	5	6	7	8	9	A	B	C	D	E	F	10	11	2
3	4	5	6	7	8	9	A	B	C	D	E	F	10	11	12	3
4	5	6	7	8	9	A	B	C	D	E	F	10	11	12	13	4
5	6	7	8	9	A	B	C	D	E	F	10	11	12	13	14	5
6	7	8	9	A	B	C	D	E	F	10	11	12	13	14	15	6
7	8	9	A	B	C	D	E	F	10	11	12	13	14	15	16	7
8	9	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	

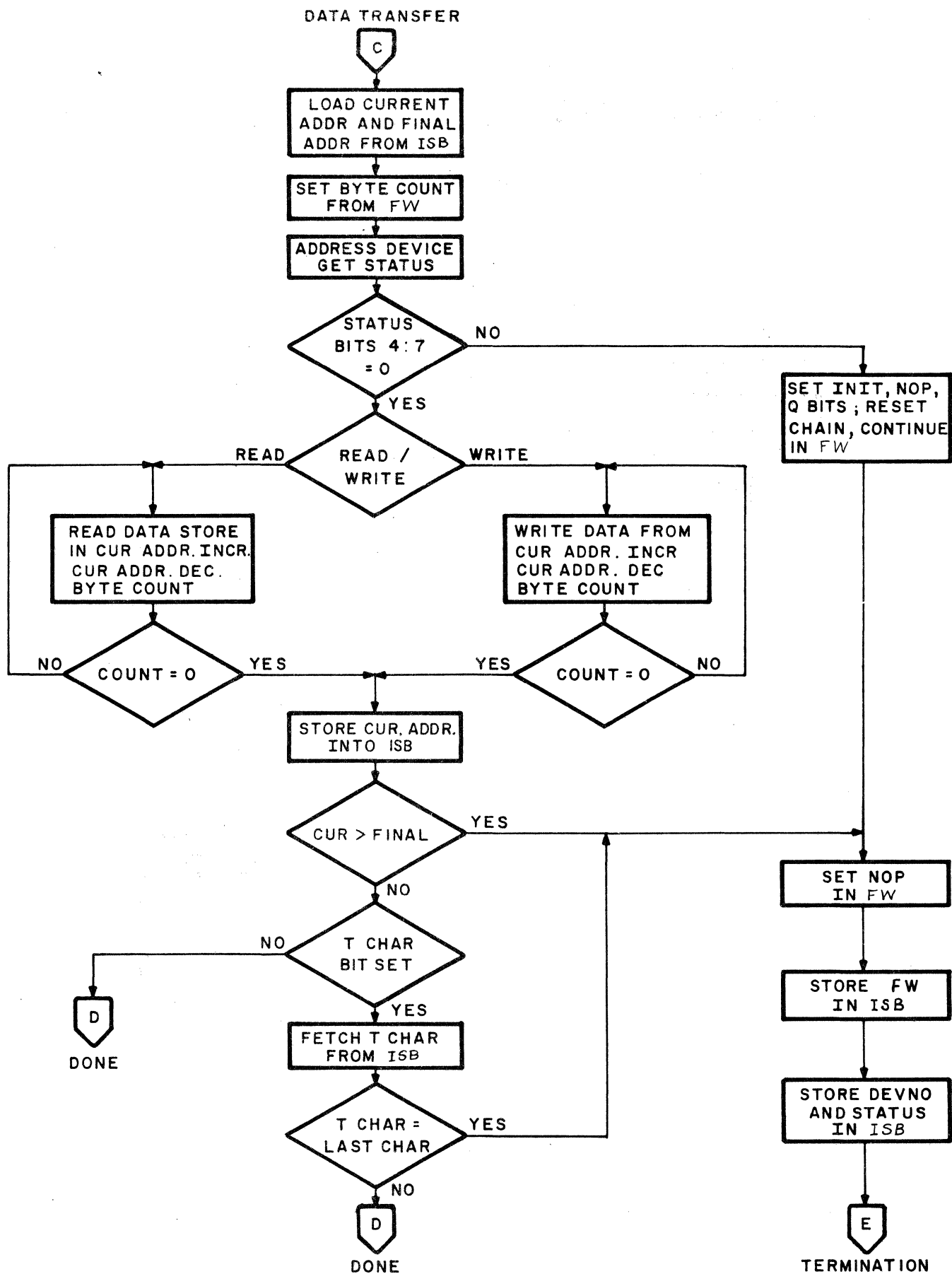
HEXADECIMAL MULTIPLICATION TABLE

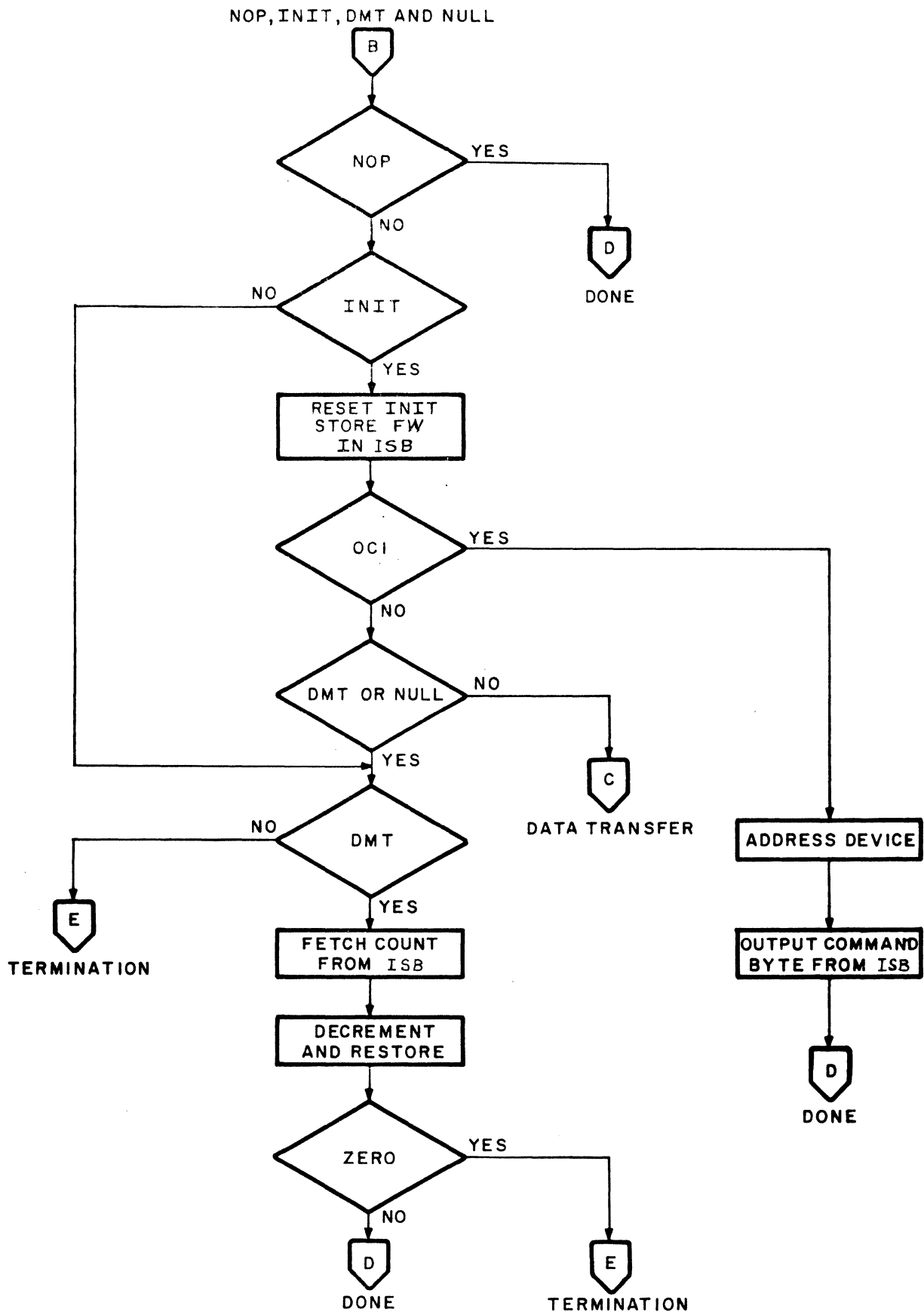
	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	
1	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	1
2	2	4	6	8	A	C	E	10	12	14	16	18	1A	1C	1E	2
3	3	6	9	C	F	12	15	18	1B	1E	21	24	27	2A	2D	3
4	4	8	C	10	14	18	1C	20	24	28	2C	30	34	38	3C	4
5	5	A	F	14	19	1E	23	28	2D	32	37	3C	41	46	4B	5
6	6	C	12	18	1E	24	2A	30	36	3C	42	48	4E	54	5A	6
7	7	E	15	1C	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	

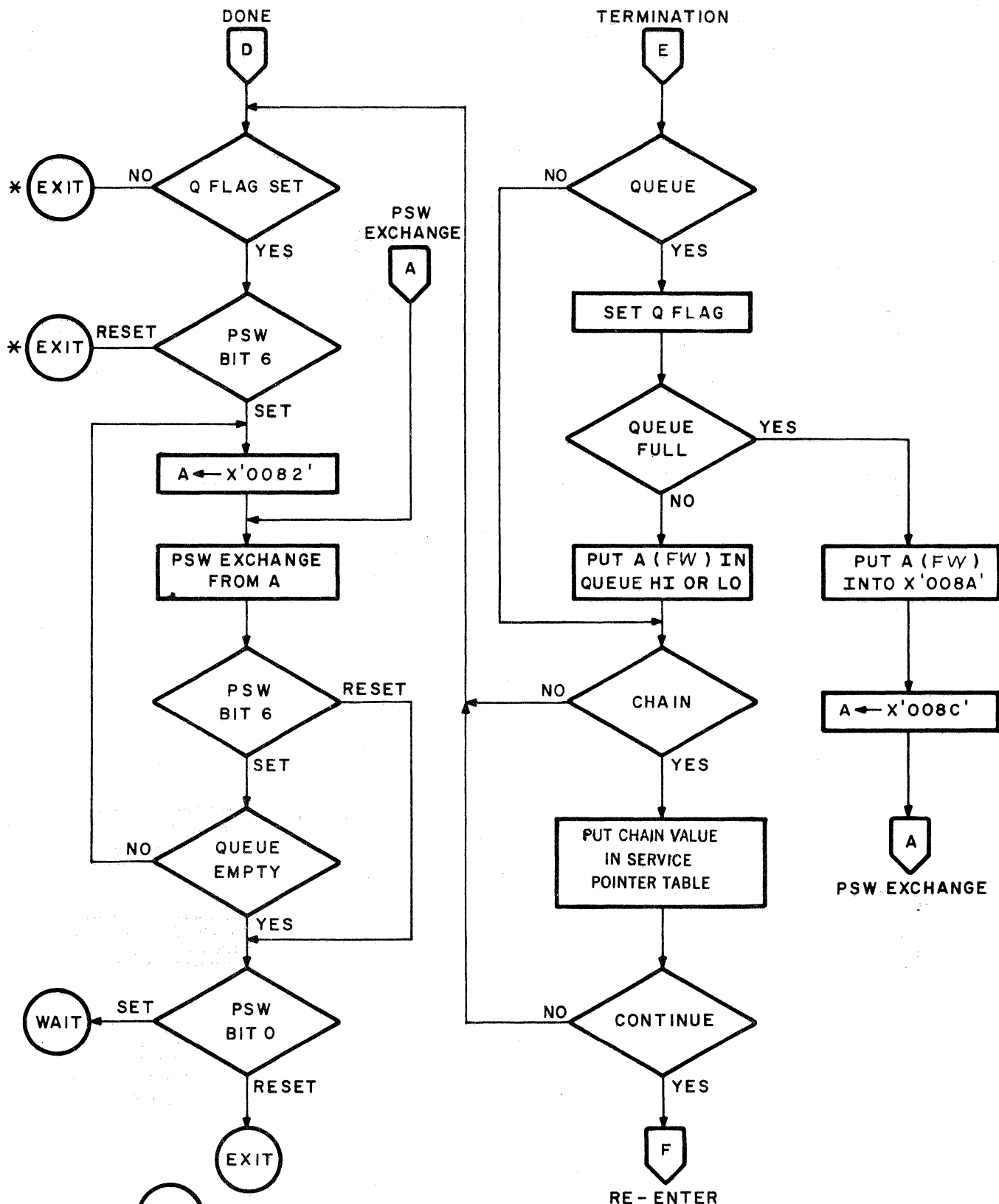
# APPENDIX 7

## AUTOMATIC I/O OPERATION AND TIMING DATA









- MEANS EXECUTE NEXT INSTRUCTION AS SPECIFIED BY PSW.

\*- IF INTERRUPT SIGNAL IS PRESENT, FIRMWARE WILL SERVICE IT BEFORE EXECUTION OF NEXT INSTRUCTION.



## AUTOMATIC I/O INTERRUPT SERVICE TIMES

[illegible]

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

32.0 (base)	
	plus <u>4.8 (TCHAR no match)</u>
	36.8 microseconds

SINT Execution time is same. Add 1.6 microseconds if indexed.

Read and Write times are for one byte per interrupt.

Add 6.0 microseconds for each additional byte on Read.

Add 4.0 microseconds for each additional byte on Write.

Note that Continue contains a negative number.



# APPENDIX 8

## I/O REFERENCES

### DISPLAY 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 CONTROL SWITCH	VARI (FIX)	0	1	0	0				
	VARI FLT	0	1	1	0				
	RUN	1	0	0	0				
	HALT (FIX)	1	1	0	0				
	HALT FLT	1	1	1	0				
	MEM WRITE	0	0	0	1				
	MEM READ	0	0	1	0				
	ADRS	0	0	1	1				
REGISTER DISPLAY SWITCH	OFF					0	0	0	0
	REG DISPLAY					0	0	0	1
	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
	R10, R11					1	1	0	1
	R12, R13					1	1	1	0
	R14, R15					1	1	1	1

#### COMMAND:

- NORM** In the Normal Mode, Byte 0 of the registers or switches is accessed each time an I/O operation is directed to the Display Panel.
- INC** In the Incremental Mode, subsequent I/O operations access subsequent bytes of the registers or switches.

TELETYPEWRITER STATUS AND COMMAND BYTE DATA  
(HEX ADDRESS 02)

BIT NUMBER	0	1	2	3	4	5	6	7
STATUS BYTE			BRK		BSY	EX		DU
COMMAND BYTE	DISABLE	ENABLE	UNBLOCK	BLOCK	WRT	READ		

- BRK**            The Break bit is set when the Break key on the Teletypewriter is depressed, or the Teletypewriter is logically disconnected from the Controller.
- BSY**            The significance of the Busy bit depends upon whether a Read or a Write operation is in progress. During Write mode, BSY is normally low, and goes high only while data is being received by the device. During Read mode, BSY is normally high, and goes low only when data has been received from the device, but not yet been transferred to the Processor. During Read mode, BSY goes high again as soon as the Processor accepts the data.
- EX**             The Examine bit is set whenever BRK is set.
- DU**             The Device Unavailable bit is set whenever the Teletypewriter is in the OFF or LOCAL mode, or power is not connected to the Teletypewriter.
- DISABLE**       This command disables the Device Interrupt to the Processor from the Device Controller.
- ENABLE**        This command enables the Device Interrupt to the Processor from the Device Controller.
- UNBLOCK**      This command enables the printer to print data entered via either the keyboard or the tape reader.
- BLOCK**         This command disables the feature described above.
- WRT**    {        The Write and Read commands are used to define the significance of the  
**READ** }        BSY bit.

# TELETYPEWRITER/ASCII/HEX CONVERSION TABLE

HEX (MSD) →					8	9	A	B	C	D	E	F
(LSD) ↓	Teletype- writer Tape Channels → ↓				8	DEPENDS UPON PARITY *						
					7	0	0	0	0	1	1	1
					6	0	0	1	1	0	0	1
					5	0	1	0	1	0	1	0
	4	3	2	1								
0	0	0	0	0	NULL	DC <sub>0</sub>	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 <sub>0</sub>	S <sub>0</sub>	(	8	H	X		
9	1	0	0	1	HT/SK	S <sub>1</sub>	)	9	I	Y		
A	1	0	1	0	LF	S <sub>2</sub>	*	:	J	Z		
B	1	0	1	1	VT	S <sub>3</sub>	+	;	K	[		
C	1	1	0	0	FF	S <sub>4</sub>	,	<	L	\		ACK
D	1	1	0	1	CR	S <sub>5</sub>	-	=	M	]		ALT. MODE
E	1	1	1	0	SO	S <sub>6</sub>	.	>	N	↑		ESC
F	1	1	1	1	SI	S <sub>7</sub>	/	?	O	←		DEL

\* Parity bit adjusted for even parity (even number of 1's) on input from Teletype keyboard. Parity bit is ignored on output to Teletype printer.

HIGH SPEED PAPER TAPE READER/PUNCH  
STATUS AND COMMAND BYTE DATA (HEX ADDRESS 13)

BIT NUMBER	0	1	2	3	4	5	6	7
STATUS BYTE	OV			NMTN	BSY	EX		DU
COMMAND BYTE	DISABLE	ENABLE	STOP	RUN	INCR	SLEW	WRITE	READ

<u>BIT</u>	<u>READER</u>	<u>PUNCH</u>
OV	The Overflow bit is set when the Buffer Register is loaded from the Reader before the previous character has been transferred. This condition can only happen in the SLEW Mode.	The Overflow bit is always reset in the Write Mode.
NMTN	The No Motion bit is set when the Reader has been issued a STOP Command and the tape has stopped on the next character.	The No Motion bit is always reset in the Write Mode.
BSY	The Busy bit is set when the Buffer Register is empty, waiting for an output from the Reader.	The Busy bit is set when the Buffer Register is full, waiting for an Unload signal from the Punch.
EX	The Examine bit is set whenever OV = 1 or NMTN = 1.	The Examine bit is always reset in the Write Mode.
DU	The Device Unavailable bit is set when the power to the Reader motor is off, or the Reader lever is in the LOAD position (straight up).	The Device Unavailable bit is set when the Punch is in the LOCAL state (switch is released), or a low tape condition exists on the tape reel inside the cabinet. There is no low tape sensor on the fan fold bins.
DISABLE	This command inhibits interrupts from the Device Controller from interrupting the Processor. Interrupts are queued.	Same as for the HSPTR.
ENABLE	This command permits interrupts from the Device Controller to interrupt the Processor.	Same as for the HSPTR.
STOP	This Command bit halts the motion of the tape after the next character has been read. The next character to be read is positioned over the sense lights when the tape stops.	This Command bit turns the Punch motor off.
RUN	This Command starts the tape moving and leaves the Controller in the RUN Mode.	This bit starts the Punch motor.
INCR	In this mode of operation, the tape is advanced one character when the controller is in the RUN Mode and BSY = 1. The tape stops after encountering one character. The tape remains stopped until a Read Data Instruction, which resets BSY and starts the tape moving again.	Not used.
SLEW	In this mode of operation, the tape is advanced continuously until stopped.	Not used.
WRITE		Designates the High Speed Paper Tape Punch.
READ	Designates the High Speed Paper Tape Reader.	

CARD READER STATUS AND COMMAND BYTE DATA  
(HEX ADDRESS 04)

BIT NUMBER	0	1	2	3	4	5	6	7
STATUS BYTE	EOV	TBL	HE	NMTN	BSY	EX	EOM	DU
COMMAND BYTE	DISABLE	ENABLE	FEED					

**EOV**      The EOV bit is set when the data is not taken from the Device Controller buffer before the next column of data arrives from the read station. This bit is reset by a FEED Command.

**TBL/DU**      These bits are set when the Card Reader fails to pick a card upon command, or when an error condition occurs in the Card Reader. The error conditions are:

1. Card Motion Error
2. Light Current Error
3. Dark Current Error

These error conditions prevent the reading of any more cards until manually reset by the operator.

**HE**      The HE bit is set when the last card in the input hopper has been read. When HE sets, NMTN is set. The HE bit must be manually reset by the operator.

**NMTN**      The NMTN is set except for the time between a FEED command and the time it takes for a card to pass through the read station.

**BSY**      The BSY bit is set while the Device Controller is awaiting data from the Card Reader. It resets when the data is available to be transferred.

**EX**      The EX bit sets when any one of the upper four (4) bits of the Status byte is set.

**EOM**      The EOM bit is set whenever NMTN is set, and when the input hopper becomes empty. Reset when FEED command is issued.

**DISABLE**      This command disables the Card Reader Device Interrupt.

**ENABLE**      This command enables the Card Reader Device Interrupt.

**FEED**      This command initiates a new card feed cycle; however, no action occurs if TBL, DU, or HE is set.

# ASCII CARD CODE CONVERSION TABLE

<u>GRAPHIC</u>	<u>8-BIT ASCII CODE</u>	<u>7-BIT ASCII CODE</u>	<u>CARD CODE</u>	<u>GRAPHIC</u>	<u>8-BIT ASCII CODE</u>	<u>7-BIT ASCII CODE</u>	<u>CARD CODE</u>
SPACE	A0	20	0-8-2	@	C0	40	8-4
!	A1	21	12-8-7	A	C1	41	12-1
"	A2	22	8-7	B	C2	42	12-2
#	A3	23	8-3	C	C3	43	12-3
\$	A4	24	11-8-3	D	C4	44	12-4
%	A5	25	0-8-4	E	C5	45	12-5
&	A6	26	12	F	C6	46	12-6
'	A7	27	8-5	G	C7	47	12-7
(	A8	28	12-8-5	H	C8	48	12-8
)	A9	29	11-8-5	I	C9	49	12-9
*	AA	2A	11-8-4	J	CA	4A	11-1
+	AB	2B	12-8-6	K	CB	4B	11-2
,	AC	2C	0-8-3	L	CC	4C	11-3
-	AD	2D	11	M	CD	4D	11-4
.	AE	2E	12-8-3	N	CE	4E	11-5
/	AF	2F	0-1	O	CF	4F	11-6
0	B0	30	0	P	D0	50	11-7
1	B1	31	1	Q	D1	51	11-8
2	B2	32	2	R	D2	52	11-9
3	B3	33	3	S	D3	53	0-2
4	B4	34	4	T	D4	54	0-3
5	B5	35	5	U	D5	55	0-4
6	B6	36	6	V	D6	56	0-5
7	B7	37	7	W	D7	57	0-6
8	B8	38	8	X	D8	58	0-7
9	B9	39	9	Y	D9	59	0-8
:	BA	3A	8-2	Z	DA	5A	0-9
;	BB	3B	11-8-6	[	DB	5B	12-8-2
<	BC	3C	12-8-4	\	DC	5C	11-8-1
=	BD	3D	8-6	]	DD	5D	11-8-2
>	BE	3E	0-8-6	↑	DE	5E	11-8-7
?	BF	3F	0-8-7	←	DF	5F	0-8-5



# SELECTOR CHANNEL STATUS AND COMMAND BYTE DATA

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 data transmission has been completed, 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'FO', 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 pushbutton 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.

# PROGRAMMABLE MEMORY PROTECT STATUS AND COMMAND BYTE DATA

BIT NUMBER	0	1	2	3	4	5	6	7
STATUS BYTE		BS	P ON	PWF		EX		
COMMAND BYTE	DISARM	ARM	P ON	P OFF	BS OFF	BS ON		

- BS            The Block Switch bit is set when the upper 64KB of a 128KB memory system is selected. This bit has no meaning with 64KB or less memory.
- 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.
- 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.
- BS ON       This Command bit sets the Block Switch to select the upper block of memory in a large memory system (over 64K bytes).
- BS OFF     This Command bit resets the Block Switch to select the lower block of memory in a large memory system (over 64K bytes).

## 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 <input type="checkbox"/>	Reference <input type="checkbox"/>
Training <input type="checkbox"/>	Maintenance <input type="checkbox"/>
Other (Explain) _____	
- Does this publication meet your requirements 

YES <input type="checkbox"/>	NO <input type="checkbox"/>
------------------------------	-----------------------------
- Is the material:

1) Presented in clear text	<input type="checkbox"/>	<input type="checkbox"/>
2) Conveniently organized	<input type="checkbox"/>	<input type="checkbox"/>
3) Adequately detailed	<input type="checkbox"/>	<input type="checkbox"/>
4) Adequately illustrated	<input type="checkbox"/>	<input type="checkbox"/>
5) Presented at appropriate technical level	<input type="checkbox"/>	<input type="checkbox"/>
- Please provide specific text references (page number, line, etc.) with your comments.

NAME \_\_\_\_\_ DATE \_\_\_\_\_  
TITLE \_\_\_\_\_  
COMPANY NAME \_\_\_\_\_  
AND ADDRESS \_\_\_\_\_

**COMMENTS:**

Staple

Communications concerning Technical Publications should be directed to:

Manager, Technical Publications  
GE Process Computer Products Dept.  
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**  
**PROCESS COMPUTER PRODUCTS DEPT**  
2255 West Desert Cove Road  
Phoenix, Arizona 85029

Attention: Technical Publications

Fold

Fold

Cut Along Line

Additional Comments:



*Progress Is Our Most Important Product*

**GENERAL  ELECTRIC**

