GE-PAC * 30
CONTROL COMPUTER

GE-PAC 30-2 MICRO-PROGRAMMING MANUAL

GENERAL ELECTRIC

*Registered Trademark of General Electric Company

GE-PAC 30
CONTROL COMPUTER

GE-PAC 30-2 MICRO-PROGRAMMING MANUAL

GENERAL ELECTRIC

GE 29-047

CONTENTS

SECTION 1

INTRODUCTION TO MICRO-PROGRAMMING THE GE-PAC 30-2

The General Electric GE-PAC 30-2 computer is a very fast, simple and uncomplicated machine. This computer is controlled by a Read-Only-Memory (ROM). A series of programs wired into the ROM control the flow of information within the registers and core storage of the machine. These programs are a sequence of very simple and elementary steps. These steps, or micro-instructions, perform functions such as transferring the content of one register to another. A series of these micro-instructions can be combined to solve highly complex problems. The micro-program in the standard GE-PAC 30-2 is designed to emulate a third generation computer. This entire program, which handles all the instructions, interrupts and display function of the third generation computer represents less than one thousand micro-instructions in the ROM.

Any computer could be emulated by changing the micro-program directing the operation of the GE-PAC 30-2. Special instructions or functions can be added by developing a new micro-program. It is evident that the micro-program replaces the costly, complicated and failure-prone hardware of a much more sophisticated computer or controlling device. The basic speed and non-destructable characteristics of this technique combined with ease of implementation and low cost make it the most attractive alternative to special purpose hardware yet available.

The micro-instructions for the machine are quite similar to the instructions for a conventional machine. For example, they are located at various addresses in the Read-Only-Memory, and they consist of operation codes that operate on various operands. To write programs for the micro machine, it is convenient to use the same kind of symbolism that is used for writing programs on conventional machines, and this means it is convenient to use an assembler.

An assembler allows:

1. Operation codes to have symbolic names.
2. Operand to have symbolic names.
3. Numbers to be written in a natural way.
4. Memory locations to have symbolic names.
5. Error checking to be performed.

The GE-PAC 30-2 Micro-Code Assembler performs all of these functions and will run on any standard GE-PAC 30 computer with 8K bytes of memory and a teletype-writer.

Each micro-instruction is represented by a wire strung through the U-core ferrite transformers of the ROM. The data in a ROM is mechanically "loaded" during the manufacturing process by weaving the wires through an array of 32 U-cores. Each transformer corresponds to one binary bit of information. If the wire passes through the center of the U-core, a "one" will be read out of that bit position when that wire is pulsed. If the wire passes on the outside of the transformer, a "zero" will be read. Each wire is assigned a consecutive pair of hex addresses. The wire is woven according to the data to be stored in those addresses. Hence, each wire contains two 16-bit micro-instructions.

Since an error in a micro-program would require the restringing of the offending wires, it is highly desireable to wire a fully checked program. The GE-PAC 30-2 Micro-Code Simulator is used for testing and debugging GE-PAC 30-2

Micro-Code programs before they are wired into the ROM. It is an interactive program that enables the debugging process to proceed from a Teletypewriter keyboard under full control and continuous observation by the designer. The GE-PAC 30-2 Micro-Code Simulator will run on any standard GE-PAC 30 computer with 8K bytes of memory and a Teletypewriter.

To aid on the production of ROM's, a machine called ROMWATS (ROM Wiring Aid and Test Set) has been developed. This machine is directed by paper tapes produced from the object tapes output by the Micro-Code Assembler or the Simulator. The program that converts the object tape into the two tapes necessary to drive the machine is referred to as the ROMWATS program. The first tape produced by the ROMWATS program is used to wire the ROM. The second tape is used to check that the wires were strung correctly. The information presented in the GE-PAC 30-2 Micro-Programming Reference Manual will assist the designer in the production of good object tape, the production of the ROMWATS tapes is usually left to the manufacturer and should be of no concern to the user.

SECTION 2

THE MICRO-PROGRAMMED GE-PAC 30-2 PROCESSOR

The first problem a potential micro-programmer must encounter is that of visualizing the architecture of the micro-programmable machine. The information that is necessary is usually buried in a description of the instruction set or must be extracted from documents containing many references to boards, connectors, diodes, transistors and other such devices. We will try to solve this problem here and now.

The first hurdle that must be overcome is the confusion between the emulated computer and the micro-programmable computer. The computer that is described in most GE-PAC 30 documentation (Reference Manual 29-004) is an emulated computer. This machine is similar to the IBM 360 family of machines and has a very powerful instruction set. This machine does not exist in hardware. A smaller and much less sophisticated micro-programmable computer has been programmed to appear as though it has the capabilities of the larger machine. The operation of the smaller machine is directed by a program wired into a Read-Only-Memory. (See GE-PAC 30-2 Hardware Block Diagram.) When executing the instructions of the emulated computer, the micro-program directs the hardware to read from core memory the next instruction to be executed. The micro-program then decodes the emulated instruction by performing logical and arithmetic operations on the data that was obtained from memory. Having decoded the instruction, the micro-program will then enter a micro-subroutine that has been designed to perform the emulated instruction. The loop is then closed by incrementing the instruction counter of the emulated machine and returning to the point in the micro-program that will fetch the next instruction from memory. By adding the logic necessary to start, stop and select a starting address to the micro-program, the small machine is made to appear much larger without expending significant sums on hardware. All of this would be highly impractical if the micro-instructions were stored in a core memory. Core memories are either relatively slow devices or very expensive.

Read-Only-Memories (ROM) on the other hand are very fast, quite inexpensive and in addition, are non-volatile. It is evident, then, that this is an ideal device in which to store frequently used subroutines.

The small machine that uses the ROM must be designed to match the ROM's performance. The GE-PAC 30-2 has, therefore, been designed to execute most micro-operations in 400 nano seconds.

Now, the basic architecture will be described. The fine detail is left to the remaining sections of the Micro-Programming Manual.

The GE-PAC 30-2 has ten basic micro-instructions.

SYMBOL	DEFINITION
A	ADD
S	SUBTRACT
X	EXCLUSIVE OR
N	AND
O	INCLUSIVE OR
L	LOAD
C	COMMAND
T	TEST
B	BRANCH
D	DECODE

TABLE OF CONTENTS

APPENDICES

CHAPTER 1

INTRODUCTION AND BLOCK DIAGRAM ANALYSIS

GE-PAC 30-2 Processor operation centers around the Read-Only-Memory (ROM0). ROM locations are addressed by a 12 bit register consisting of an 8 bit incrementing register (RAL) and a 4 bit non-incrementing page register (RAS) which is loaded from the 4 bit 'outer-rank' (RAH) register when RAL is loaded.

Information read from ROM is placed in a 16 bit Data Register (RD). Bits 0:3 of RD specify a micro-operation to be performed which, in turn, defines the meaning of the remaining bits.

ROM with its pre-wired micro-program, directs the Processor through the control unit. Processor Control can, depending on the micro-operation code, set-up the Arithmetic Logic Unit (ALU) to the desired mode of operation, test for specified hardware conditions, issue functional commands to establish hardware conditions, initiate memory cycles, or set-up micro-instruction loops, or load and unload selected registers in the hardware register stacks.

There are five general purpose Micro-Registers labeled MR0 through MR4. Each has a capacity of 16 bits and is directly addressable from RD.

The Program Status Word (PSW) is a 16 bit register which indicates the system status relative to the user program being executed (see Reference Manual, Publication Number 29-004). Bits 0:11 of PSW define machine status. Bits 12:15 are set apart in the Condition Code Register (CCR) which may be loaded only from the Flag Register (FLR). When PSW is loaded, bits 12:15 of the S Bus are loaded into the FLR instead of the CCR. This

permutes user status to the micro-level. The Flag Register and ultimately the Condition Code Register reflect the results of the micro-instruction (or instructions in the case of a user micro-routine) just performed.

The Location Counter (LOC) is a 16 bit appendum to PSW which holds the address of the next user instruction to be performed. LOC is directly addressable by RD, however, it may be forcibly selected, regardless of RD in the Decode micro-instruction.

The Memory Address Register (MAR) is a 16 bit register used to address core memory locations. MAR appears twice -- on the Memory Interface and in the Processor register stacks. When MAR is loaded, both registers are loaded at the same time. It was duplicated in the register stacks because the Address Register on the Memory Interface cannot be unloaded to the B Bus.

The Memory Data Register (MDR) is a 16 bit register used to hold data read from or written into core memory. MDR is directly addressable by RD. It is separated into two bytes (MDH and MDL) which may be loaded separately on cross-shift operations.

The 16 user's General Registers each have a capacity of 16 bits. The user's registers (TR0:TR15) are not directly addressable from RD. To access a particular user's register, one must address the appropriate Instruction Register (IR) fields which contain the address of the desired user's register (TR0:TR15). To access the register specified by IR bits 8:11, User's Destination (YD) is addressed; to access the register specified by IR bits 12:15, User's Source (YS) is addressed; therefore, user's register selection is indirectly made. It is therefore necessary that IR contain the proper address before

a 'user's register' is used. This is an over simplified description; there are numerous conditions which affect indirect addressing and will be discussed further in Chapter 3.

The Instruction Register (IR) is a 16 bit register used to hold the user's instruction currently being processed. IR is directly addressable by RD. In addition, provision

is made for unloading only bits 8:11 to the B Bus bits 12:15 (IR4) for comparison between the Mask (M1) field and the Flag Register when executing user's branches. Bits 0:7 of IR (User's Operation Code) are used to address locations in the Decoder ROM (DROM). This is a separate Read-Only-Memory consisting of up to 128 pre-wired words, each 12 bits long.

DROM is interrogated only on a Decode micro-instruction and the resulting 12 bit read-out is immediately jammed into RAS and RAL. DROM holds the starting addresses of the micro-routines required to perform user's instructions. RAS and RAL may also be jammed with hardware generated addresses in the Decode micro-instruction.

The Counter Register (CNTR) is a 4 bit decrementing register. It may be pre-loaded with any number from 0 to 15 to count the number of repetitions of a single micro-instruction or a block of micro-instructions. The counter is used in the Multiply or Divide sequences to cause 16 interations of the micro-instruction sets.

The Arithmetic Register (AR) is a 16 bit register used to hold the first operand in arithmetic or logical micro-operations. It is one of two direct inputs to the Arithmetic Logic Unit (ALU). The other input to the ALU is the 16 bit B Bus which receives data from any

1 of 29 possible sources. The two bytes of the B Bus may also be swapped with the cross shift logic.

The ALU consists of a 16 bit parallel adder-subtracter logic network with a one bit look ahead carry. The 16 bit arithmetic or logical result is gated to the S Bus which in turn is gated to 1 of 33 possible destinations.

Input-Output transfer (I/O) is accomplished by a single micro-instruction. The I/O control lines are decoded from RD bits 14 and 15. Input data is taken from the Data Request Lines (DRL 0:7) and placed directly on B Bus bits 8:15. Output data is taken from S Bus bits 8:15 and loaded directly to the Data Available Lines (DAL 0:7).

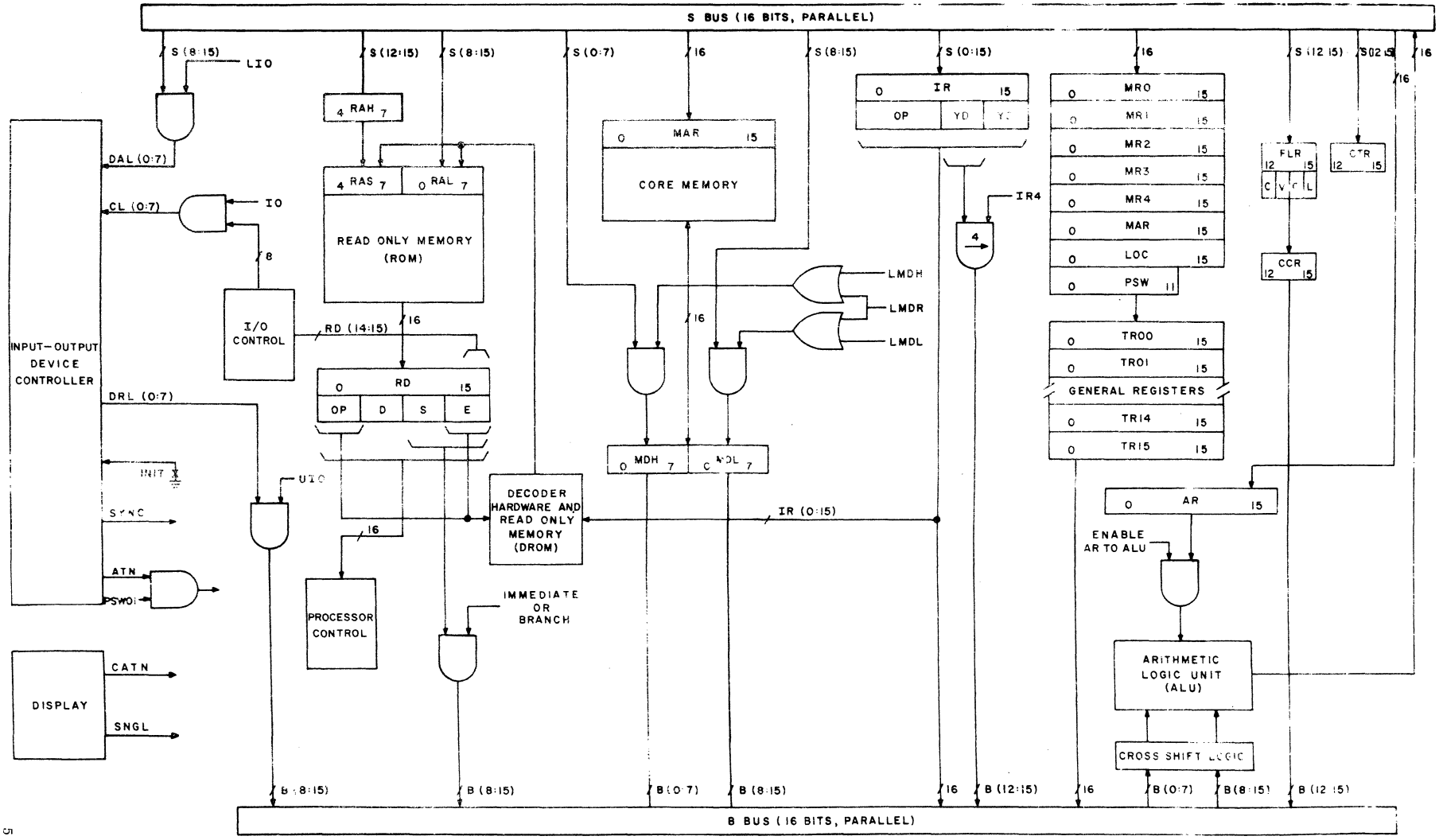


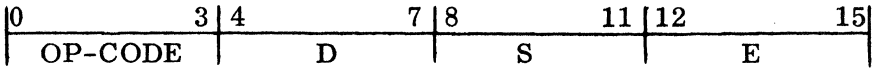
FIGURE 1 GE-PAC 30-2 HARDWARE BLOCK DIAGRAM

CHAPTER 2

WORD FORMAT

The GE-PAC 30 micro-instruction can have any one of four machine language formats, depending on the operation that the Op-Code specifies.

2.1 Arithmetic and Logic Format



S = Source field: the address of the register containing the second operand is in this field. The first operand comes from the A Register (AR).

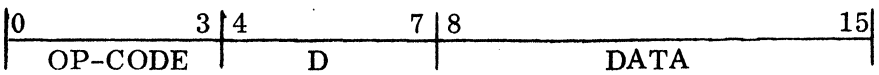
D = Destination field: the result of the operation will be placed into the register whose address is in this field.

E = Extended Operation field: specifies options within the same operation.

Bit 3 (of the Op-Code) in this format is always reset.

The following micro-instructions use the above format: Add, Subtract, Exclusive OR, AND, Inclusive OR, Load, and Decode.

2.2 Immediate Format



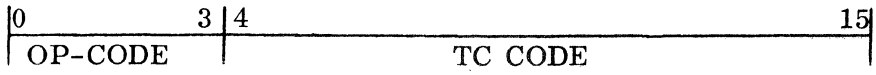
DATA = the second operand itself is in this field. The first operand comes from the A Register (AR).

D = Destination field: the result of the operation will be placed into the register whose address is in this field.

Bit 3 (of the Op-Code) in this format is always set.

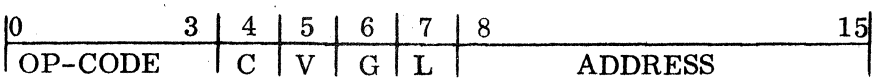
The following micro-instructions use the above format: Add Immediate, Subtract Immediate, Exclusive OR Immediate, AND Immediate, Inclusive OR Immediate, and Load Immediate.

2.3 Test and Command Format



TC CODE = Test or Command Code: specifies the signal to be tested or specifies the command to be performed.

2.4 Branch on Condition Format



ADDRESS = if conditions (C, V, G, or L) are met, the program is transferred to the 8 bit address specified by this field.

C = carry

V = overflow

G = greater, than zero

L = less, than zero

If bits 4:7 are zero, the state of the Counter Register is examined. If the counter does not equal 'one', the counter is decremented by one and the program is transferred to the 8 bit address specified by the ADDRESS field.

The "Branch on Condition" micro-instruction uses this format.

CHAPTER 3

SOURCE AND DESTINATION REGISTERS

3.1 Source registers are only available to non-immediate micro-instructions, (RD03 = 0). The following sources may be addressed.

RD	8	9	10	11	SYMBOLIC
	0	0	0	0	MR0
	0	0	0	1	MR1
	0	0	1	0	MR2
	0	0	1	1	MR3
	0	1	0	0	MR4
	0	1	0	1	MAR
	0	1	1	0	LOC
	0	1	1	1	PSW
	1	0	0	0	NULL
	1	0	0	1	IR
	1	0	1	0	MDR
	1	0	1	1	IR4
	1	1	0	0	NULL
	1	1	0	1	IO
	1	1	1	0	YD
	1	1	1	1	YDP1

3.2 There are two NULL sources. When address X'8' (the A Register) or address X'C' (the Counter) appear in the source field (RD 8:11), zeros are gated to all 16 bits of the B Bus.

3.3 When PSW is used as the source, bits 0:11 of the stack PSW are gated to B Bus bits 0:11 and the Condition Code Register (CCR bits 12:15) is gated to B Bus bits 12:15.

3.4 When IR4 is the source, IR bits 8:11 are gated to B Bus bits 12:15. The remainder of the B Bus is all zeros.

3.5 When IO (address X'D') appears as the source, an input operation is to be

performed. IO can only be a source in a Load instruction. The Processor is requesting a data response from an I/O device; IO is not a register. The nature of the request is encoded into the 'E' field of the Load instruction and will be discussed later in Chapter 4. The device response is gated from the Data Request Lines (DRL 0:7) to B Bus bits 8:15.

3.6 The 16 general purpose user's registers do not have individual source addresses. Instead, common symbolic addresses - YD (address X'E') and YDP1 (address X'F') - cause the general registers to be selected from the Instruction Register bits 8:11 or 12:15. The General Registers are Indirectly addressed.

3.7 There exists in the phase hardware, a flip-flop which specifies the IR field to be used for indirect source decoding. The flip-flop (PTYS), if set, causes the indirect source to be decoded from the User's Source (YS) field of IR, IR bits 12:15. If PTYS is reset, the indirect source is decoded from the User's Destination (YD) field of IR, IR bits 8:11. PTYS is set conditionally on a Decode micro-instruction and is reset when an indirect source is used.

3.8 If IR = X'nnA5 and PTYS is set when YD (address X'E') is the source, TR5 is unloaded to the B Bus. If PTYS is reset and YD is the source, TR10 is unloaded to the B Bus.

3.9 If YD plus 1 (YDP1) is the source, the indirect register decoding is done from the IR field YD or YS specified by the PTYS flip-flop. If the selected address is even, the next sequential address is forced and that register is unloaded to the B Bus. If the selected

address is odd, the next sequential address is not forced and the selected register is unloaded to the B Bus.

3.10 If IR = X'nn40' and PTYS is reset, YDP1 (address X'F') will cause TR5 to be unloaded to the B Bus. If IR = X'nn50' and YDP1 is the source, TR5 is unloaded to the B Bus.

3.11 The following Destinations may be addressed:

RD	4	5	6	7	SYMBOLIC	
	0	0	0	0	RAH*	MR0**
	0	0	0	1	RAL*	MR1**
	0	0	1	0	YS *	MR2**
	0	0	1	1	MR3	
	0	1	0	0	MR4	
	0	1	0	1	MAR	
	0	1	1	0	LOC	
	0	1	1	1	PSW	
	1	0	0	0	AR	
	1	0	0	1	IR	
	1	0	1	0	MDR	
	1	0	1	1	FLR	
	1	1	0	0	CNTR	
	1	1	0	1	IO	
	1	1	1	0	YD	
	1	1	1	1	YDP1	

* The Bank flip-flop must be reset to Load RAH, RAL, or YS.

** The Bank flip-flop must be set to Load MR0, MR1, or MR2.

3.12 Loading RAH loads the outer rank page register only of ROM address register. ROM decodes page from the inner rank (RAS). When RAL is loaded, the data in RAH are loaded into RAS at the same time. All 12 bits of ROM address are loaded.

3.13 When User's Source (YS) is the destination, the S Bus is loaded to the

General Register specified by the YS field of IR, IR bits 12:15.

When User's Destination (YD) is the destination, the S Bus is loaded to the General Register specified by the YD field of IR, IR bits 8:11. PTYS has no affect on destinations. When User's Destination plus 1 (YDP1) is the destination, the same even/odd anomoly must be observed: if the address is even, the next sequential general register is loaded; if the address is odd, that register is loaded.

3.14 When the Program Status Word (PSW) is loaded, S Bus bits 12:15 are loaded into the Flag Register (FLR).

3.15 When IO appears as the destination, an output operation is to be performed. IO can only be a destination in a Load micro-instruction. The Processor will transmit data to an external device. IO is <u>not</u> a register. The nature of the output data is encoded into the 'E' field of the Load instruction and will be discussed fully later in Chapter 4. The output byte is gated from S Bus bits 8:15 onto the Data Available Lines (DAL 0:7).

3.16 <u>For the cross shift option only</u>

If MDR is the source

- and MAR is even, the cross shift operates normally.
- and MAR is odd, the cross shift is inhibited; the instruction behaves as a normal load.

If MDR is the destination

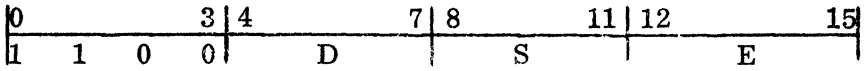
- and MAR is even, the lower byte of the source is stored as the higher byte in MDR - the lower byte of MDR remains unchanged.
- and MAR is odd, the lower byte of the source is stored as the lower byte in MDR - the upper byte of the MDR remains unchanged.

This option is used with MDR to get and store bytes in user memory.

CHAPTER 4

MICRO-INSTRUCTIONS

4.1 ADD



4.1.1 Description: the contents of the A Register (AR) are algebraically added to the contents of the register specified by the Source (S) field. The sum is then gated into the register specified by the Destination (D) field. For register addresses, see Appendix 1 and Chapter 3.

4.1.2 Options: Note that the character "x" in any bit position indicates a "don't care" state.

E field	Definition
1xxx	AR is not gated to the adder (Arithmetic Logic Unit, ALU): zero is added to the contents of the register specified by the Source (S) field.
x1xx	Set the flags.
xx1x	Carry flag (C), if set before this instruction, is added to the operands.
xxx1	If there is a carry from the most significant bit position, the Carry flag (C) is set.

4.1.3 Assembler Format: A D, S, E

A = symbol for Add

D = in this place there is a register symbol (e.g.: MAR, PSW, MR1), followed by a comma.

S = register symbol (e.g.: MR4, MDR, PSW), followed by a comma only if an assembler option is specified in the E field.

Symbol	Definition
CI	Carry in, but not carry out. Bits 14 and 15 of the instruction = 10.
CO	Carry out, but not carry in. Bits 14 and 15 of the instruction = 01.
NC	No carry. Bits 14 and 15 of the instruction = 00.
NF	No flags. Bit 13 of the instruction = 0.
NA	No AR to ALU. Bit 12 of the instruction = 1.

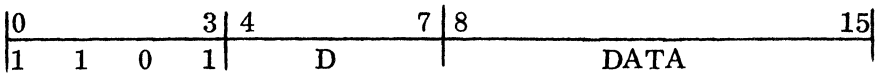
If there are no options stated in the E field of the symbolic assembler instruction, the following options are generated automatically: Carry in and out (bits 14 and 15 = 11), AR is an input to the ALU (bit 12 = 0), flags are changed (bit 13 = 1).

4.1.4 Examples:

	Binary					=	Decimal	
	Sign	Data					Sign	Data
	0	011	0101	0010	1000	AR	+	13,608
	0	010	1110	1001	0100	B Bus	+	11,924
	0	110	0001	1011	1100	S Bus	+	25,532
	0	011	0101	0010	1000	AR	+	13,608
	1	101	0001	0110	1100	B Bus	-	11,924
	0	000	0110	1001	0100	S Bus	+	1,684
	0	010	1110	1001	0100	AR	+	11,924
	1	100	1010	1101	1000	B Bus	-	13,608
	1	111	1001	0110	1100	S Bus	-	1,684
	0	111	1111	1111	1111		+	32,767
	0	000	0000	0000	0101		+	5
Overflow!	1	000	0000	0000	0100			33,472

Unlike carrys from the sign (bit 0) and the most significant magnitude bit (bit 1) result in overflow: the sum is greater than 16 bits (including sign).

4.2 ADD IMMEDIATE



4.2.1 Description: the contents of the A Register (AR) are algebraically added to the 8 bits in the DATA field of this instruction. The sum is then gated into the register specified by the Destination (D) field. For register addresses, see Appendix 1.

4.2.2 Options: None. Note that the flags (C, V, G, or L) are not changed by this micro-instruction.

4.2.3 Assembler Format: A D, DATA

A = symbol for Add Immediate

D = in this place there is a register symbol (e.g.: MR4, FLR, MAR), followed by a comma.

DATA = either the operand expressed in hexadecimal (e.g.: X'9A') or as the low order 8 bits, L (DATA), or the high order 8 bits, H(DATA), of the value of a symbol defined by DATA.

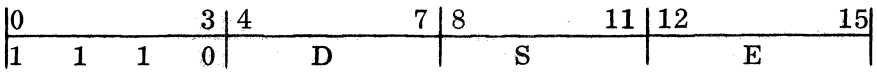
4.2.4 Examples:

Binary						=	Decimal	
Sign	Data						Sign	Data
0	011	0101	0010	1000	AR		+	13,608
0	000	0000	1011	1100	B Bus		+	188
0	011	0101	1110	0100	S Bus		+	13,796
1	100	1010	1101	1000	AR		-	13,608
0	000	0000	1011	1100	B Bus		+	188
1	100	1011	1001	0100	S Bus			13,420

Note: A negative B Bus is impossible.

	Binary					=	Decimal	
	Sign	Data					Sign	Data
	0	111	1111	1111	1111	AR	+	32,767
	0	000	0000	0000	0101	B Bus	+	5
Overflow!	1	000	0000	0000	0100	S Bus	+	33,472

4.3 SUBTRACT



4.3.1 Description: the contents of the register specified by the Source (S) field are algebraically subtracted from the contents of the A Register (AR). The difference is then gated into the register specified by the Destination (D) field. For register addresses, see Appendix 1.

4.3.2 Options: Note, that the character "x" in any bit position indicates a "don't care" state.

E field	Definition
1xxx	AR is not gated to the subtracter (ALU): the contents of the register specified by the Source (S) field are subtracted from zero.
x1xx	Set flags.
xx1x	Carry flag (C), if set before this instruction is also subtracted from the AR.
xxx1	If there is a borrow from the most significant bit position, the Carry flag (C) is set.

4.3.3 Assembler Format: S D,S,E

S = symbol for subtract

D = in this place there is a register symbol (e.g.: MR4, AR, MAR), followed by a comma.

S = register symbol (e.g.: MR3, MDR, MR2), followed by a comma only if an assembler option is specified in the E field.

The following symbolic options can appear in the E field:

Symbol	Definition
CI	Carry in, but not carry out. Bits 14 and 15 of the instructions = 10.
CO	Carry out, but not carry in. Bits 14 and 15 of the instructions = 01.
NC	No carry. Bits 14 and 15 of the instruction = 00.
NF	No flags. Bit 13 of the instruction = 0.
NA	No AR to ALU. Bit 12 of the symbolic instruction = 1.

If there are no options stated in the E field of the symbolic assembler instruction, the following options are generated automatically: carry in and out (bits 14 and 15 = 11), AR is an input to the ALU (bit 12 = 0), flags are changed (bit 13 = 1).

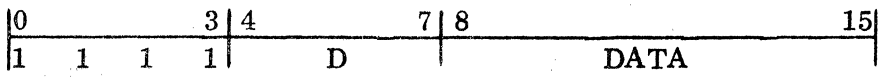
4.3.4 Examples:

Binary Sign	Binary Data					= Decimal Sign	Decimal Data
0	011	0101	0010	1000	AR	+	13,608
0	010	1110	1001	0100	B Bus	+	11,924
0	000	0110	1001	0100	S Bus	+	1,684
0	010	1110	1001	0100	AR	+	11,924
0	011	0101	0010	1000	B Bus	+	13,608
1	111	1001	0110	1100	S Bus	-	1,684

	Binary					=	Decimal	
	Sign	Data					Sign	Data
	0	011	0101	0010	1000	AR	+	13,608
	1	101	0001	0110	1100	B Bus	-	11,924
	0	110	0001	1011	1100	S Bus	+	25,532
	0	111	1111	1111	1111	AR	+	32,767
	1	111	1111	1111	1011		-	5
Overflow!	1	000	0000	0000	0100		+	33,472

Unlike borrows from the sign (bit 0) and the most significant magnitude bit (bit 1) result in overflow: the difference is greater than 16 bits (including sign).

4.4 SUBTRACT IMMEDIATE



4.4.1 Description: the contents of the DATA field are algebraically subtracted from the contents of the A Register (AR). The difference is then gated to the register specified by the Destination (D) field. For register addresses, see Appendix 1.

4.4.2 Options: None. Note, that the flags (C, V, G, or L) are not changed by this micro-instruction.

4.4.3 Assembler Format: S D, DATA

S = symbol for Subtract Immediate

D = there is a register symbol in this position (e.g.: MR4, LOC, MAR), followed by a comma

DATA = either the operand expressed in hexadecimal (e.g.: X'9A') or as the low order 8 bits, L (DATA), or the high order 8 bits, H (DATA, of the value of a symbol defined by DATA.

4.4.4 Examples:

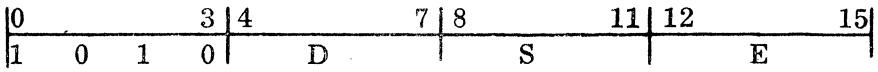
Binary						=	Decimal	
Sign	Data					Sign	Data	
0	011	0101	0010	1000	AR	+	13,608	
0	000	0000	1011	1100	B Bus	+	188	
0	011	0100	0110	1100	S Bus	+	13,420	
1	100	1010	1101	1000	AR	-	13,608	
0	000	0000	1011	1100	B Bus	+	188	
1	100	1010	0001	1100	S Bus	-	13,796	

Note: A negative B Bus is impossible.

	1	000	0000	0000	0001	AR	-	32,767
	0	000	0000	0000	0101	B Bus	+	5
Overflow!	0	111	1111	1111	1100		-	33,472

Unlike borrows from the sign (bit 0) and the most significant magnitude bit (bit 1) result in Overflow: the difference is greater than 16 bits (including sign).

4.5 EXCLUSIVE OR



4.5.1 Description: the contents of the register specified by the Source (S) field are logically subtracted (Exclusive ORed) from the contents of the A Register (AR). The logical difference (result) is then gated into the register specified by the Destination (D) field. For register addresses, see Appendix 1.

4.5.2 Options: Note, that the character "x" in any bit position indicates a "don't care" state.

E field	Definition
1xxx	AR is not gated to the ALU: the contents of the register specified by the Source (S) field

E field	Definition
	is gated into the register specified by the Destination (D) field without any change.
x1xx	Set flags.
xx1x	No effect on this instruction.
xxx1	No effect on this instruction.

4.5.3 Assembler Format: X D,S,E

X = symbol for Exclusive OR

D = in this place there is a register symbol (e.g.: MR4, LOC, MAR), followed by a comma

S = register symbol (e.g.: MR3, MDR, MR0), followed by a comma only if an assembler option is specified in the E field

The following symbolic options can appear in the E field.

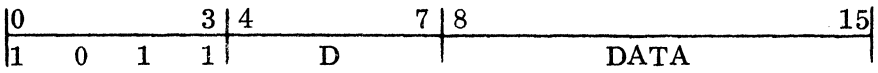
Symbol	Definition
CI	Carry in, but not carry out. Bits 14 and 15 of the instruction = 10.
CO	Carry out, but not carry in. Bits 14 and 15 of the instruction = 01.
NC	No carry. Bits 14 and 15 of the instruction = 00.
NF	No flags. Bit 13 of the instruction = 0.
NA	No AR to ALU. Bit 12 of the instruction = 1.

If there are no options stated in the E field of the symbolic assembler instruction, the following options are generated automatically: carry in and out (bits 14 and 15 = 11), AR is an input to the ALU (bit 12 = 0), flags are changed (bit 13 = 1).

4.5.4 Example:

0	101	1101	0010	0101	AR
0	010	1000	0101	1110	B Bus
0	111	0101	0111	1011	S Bus

4.6 EXCLUSIVE OR IMMEDIATE



4.6.1 Description: the contents of the DATA field are logically subtracted (Exclusive ORed) from the contents of the A Register (AR). The logical difference (result) is gated into the register specified by the Destination (D) field. For register addresses, see Appendix 1.

4.6.2 Options: None. Note that the flags (C, V, G, or L) are not changed by this micro-instruction.

4.6.3 Assembler Format: X D, DATA

X = symbol for Exclusive OR Immediate

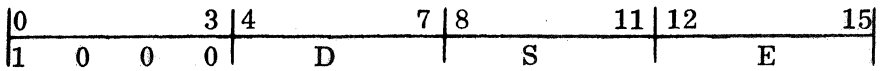
D = there is a register symbol in this position (e.g.: MR4, YD, MAR,) followed by a comma

DATA = either the operand expressed in hexadecimal (e.g.: X'9A'), or as the low order 8 bits, L (DATA), or high order 8 bits, H (DATA), of the value of a symbol defined by DATA.

4.6.4 Example:

0	101	1101	0010	0101	AR
0	000	0000	0101	1110	B Bus
0	101	1101	0111	1011	S Bus

4.7 AND



4.7.1 Description: the contents of the register specified by the Source (S) field are logically multiplied (ANDed) by the contents of the A Register (AR). The logical product (result) is then gated into the register specified by the Destination (D) field. For register addresses, see Appendix 1.

4.7.2 Options: Note, that the character "x" in any bit position indicates a "don't care" state.

E field	Definition
1xxx	AR is not gated to the ALU: the logical product is zero, and zero is gated into the register specified by the Destination (D) field.
x1xx	Set flags.
xx1x	No effect on this instruction.
xxx1	No effect on this instruction.

4.7.3 Assembler Format: N D,S,E

N = symbol for AND

D = in this place there is a register symbol (e.g.: MR4, YD, MAR,) followed by a comma

S = register symbol (e.g.: MR0, MDR, LOC,) followed by a comma only if an assembler option is specified in the E field

The following symbolic options can appear in the E field.

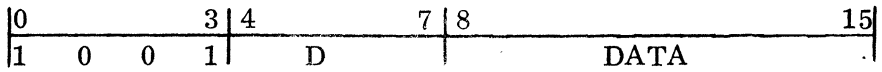
Symbol	Definition
CI	Carry in, but not carry out. Bits 14 and 15 of the instruction = 10.
CO	Carry out, but not carry in. Bits 14 and 15 of the instruction = 01.
NC	No carry. Bits 14 and 15 of the instruction = 00.
NF	No flags. Bit 13 of the instruction = 0.

If there are no options stated in the E field of the symbolic assembler instruction, the following options are generated automatically: carry in and out (bits 14 and 15 = 11), AR is an input to the ALU (bit 12 = 0), flags are changed (bit 13 = 1).

4.7.4 Example:

0	101	1101	0010	0101	AR
0	010	1000	0101	1110	B Bus
0	000	1000	0000	0100	S Bus

4.8 **AND** IMMEDIATE



4.8.1 Description: the contents of the DATA field are logically multiplied (ANDed) by the contents of the A Register (AR). The logical product (result) is gated into the register specified by the Destination (D) field. For register addresses, see Appendix 1.

4.8.2 Options: None. Note that the flags (C, V, G, or L) are not changed by this micro-instruction.

4.8.3 Assembler Format: N D, DATA

N = symbol for AND Immediate

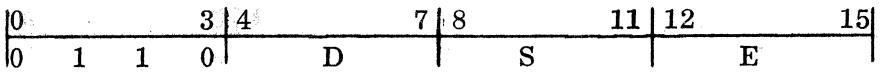
D = there is a register symbol in this position (e.g.: MR4, FLR, MAR,) followed by a comma

DATA = either the operand expressed in hexadecimal (e.g.: X'9A'), or as the low order 8 bits, L (DATA), or high order 8 bits, H (DATA), of the value of a symbol defined by DATA.

3.8.4 Examples:

0	101	1101	0010	0101	AR
0	000	0000	0101	1110	B Bus
0	000	0000	0000	0100	S Bus

4.9 INCLUSIVE OR



4.9.1 Description: the contents of the register specified by the Source (S) field are logically added (ORed) by the contents of the A Register (AR). The logical sum (result) is then gated into the register specified by the Destination (D) field. For register addresses, see Appendix 1.

4.9.2 Options: Note that the character "x" in any bit position indicates a "don't care" state.

E field	Definition
1xxx	AR is not gated to the ALU: the contents of the register specified by the Source (S) field is gated into the register specified by the Destination (D) field without any change.

E field	Definition
x1xx	Set flags.
xx1x	No effect on this instruction.
xxx1	No effect on this instruction.

4.9.3 Assembler Format: O D,S,E

O = symbol for Inclusive OR

D = in this place there is a register symbol (e.g.: MR0, LOC, MAR,) followed by a comma

S = register symbol (e.g.: MR4, MDR, YD,) followed by a comma only if an assembler option is specified in the E field

The following symbolic options can appear in the E field.

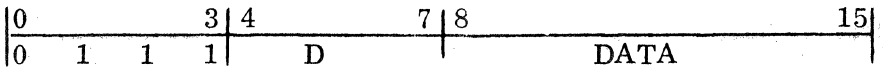
Symbol	Definition
CI	Carry in, but not carry out. Bits 14 and 15 of the instruction = 10.
CO	Carry out, but not carry in. Bits 14 and 15 of the instruction = 01.
NC	No carry. Bits 14 and 15 of the instruction = 00.
NF	No flags. Bit 13 of the instruction = 0.
NA	No AR to ALU. Bit 12 of the instruction = 1.

If there are no options stated in the E field of the symbolic assembler instruction, the following options are generated automatically: carry in and out (bits 14 and 15 = 11), AR is an input to the ALU (bit 12 = 0), flags are changed (bit 13 = 1).

4.9.4 Examples:

0	101	1101	0010	0101	AR
0	010	1000	0101	1110	B Bus
0	111	1101	0111	1111	S Bus

4.10 INCLUSIVE OR IMMEDIATE



4.10.1 Description: the contents of the DATA field are logically added (ORed) with the contents of the A Register (AR). The logical sum (result) is then gated into the register specified by the Destination (D) field. For register addresses, see Appendix 1.

4.10.2 Options: None. Note that the flags (C, V, G, or L) are not changed by this micro-instruction.

4.10.3 Assembler Format: O D, DATA

O = symbol for Inclusive OR Immediate

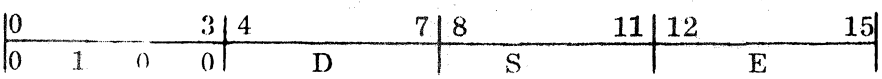
D = there is a register symbol in this position (e.g.: MR3, CNTR, MAR,) followed by a comma

DATA = either the operand expressed in hexadecimal (e.g.: X'9A'), or as the low order 8 bits, L (DATA), or high order 8 bits, H(DATA), of the value of a symbol defined by DATA.

4.10.4 Examples:

0	101	1101	0010	0101	AR
0	000	0000	0101	1110	B Bus
0	101	1101	0111	1111	S Bus

4.11 LOAD



4.11.1 The contents of the register specified by the Source (S) field are loaded unaltered into the register specified by the Destination (D) field. For register addresses, see Appendix 1.

4.11.2 Note that the character "x" in any bit position indicates a "don't care" state. If the Destination or Source is not IO, the E field has the following meaning:

E field	Definition
00xx	Load, no other options.
01xx	The contents of the Source register are shifted right one bit and the shifted number is loaded into the Destination register. Contents of source register are unchanged unless same register is both source and destination.
10xx	The contents of the Source register are shifted left one bit and the shifted number is loaded into the Destination register. Contents of source register are unchanged unless same register is both source and destination.
11xx	The source Data is byte swapped and loaded to the Destination register. If MDR is either Source or Destination, the cross shift will occur only if MAR is even. See Chapter 3.
xx1x	This is valid with the shift options only; if the Carry flag (C) is set prior to this instruction, a one will be shifted into the most significant bit on a shift right, or into the least significant bit on a shift left.
xxx1	With shift options only: if a one is shifted out, the Carry flag (C) is set. Non-shift options: the Carry flag (C) is reset.

4.11.3 If IO is Source or Destination, the E field takes on the following meaning. IO is Destination:

E field	Definition
xx01	The S Bus is loaded to the DAL's as Address (ADRS).
xx10	The S Bus is loaded to the DAL's as Data (DA).
xx11	The S Bus is loaded to the DAL's as a Command (CMD).

IO is Source:

E field	Definition
xx01	Device interrupt is Acknowledged (ACK). The interrupting device address is loaded from the DRL's to the Destination register.
xx10	Device Data is requested (DR). The data is loaded from the DRL's to the Destination Register.
xx11	Device Status is requested (STAT). The Status byte is loaded from the DRL's to the Destination Register.

4.11.4 Assembler Format: L D,S,E

L = symbol for Load

D = in this place there is a register symbol (e.g.: MR4, IO, MAR,) followed by a comma

S = register symbol (e.g.: MR4, MDR, IO,) followed by a comma only if an assembler option is specified in the E field.

The following symbolic options can appear in the E field.

Symbol	Definition
CI	Carry in, but not carry out. Bits 14 and 15 of the instruction = 10.
CO	Carry out, but not carry in. Bits 14 and 15 of the instruction = 01.

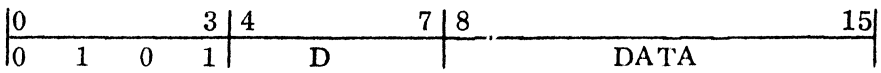
Symbol	Definition
NC	No carry. Bits 14 and 15 of the instruction = 00.
SR	Shift Right option.
SL	Shift Left option.
CS	Cross Shift option.
ADRS	I/O Address. I O in Destination.
DA	I/O Data Available. I O in Destination.
CMD	I/O Command. I O in Destination.
ACK	Acknowledge Interrupt. I O in Source.
DR	Data Request. I O in Source.
STAT	Status Request. I O in Source.

If there are no options stated in the E field of the symbolic assembler instruction, the following options are generated automatically: carry in and out (bits 14 and 15 = 11), no shifts (bits 12 and 13 = 00).

4.11.5 Examples:

Before	Symbolic Instruction	After
MR4 = X'1234'	L MR4, MR4, NC	MR4 = X'1234'
MR4 = X'1234'	L MR4, MR4, SL	MR4 = X'2468'
MR4 = X'1234'	L MR4, MR4, SR	MR4 = X'091A'
MR4 = X'1234'	L MR4, MR4, CS	MR4 = X'3412'

4.12 LOAD IMMEDIATE



4.12.1 Description: The 8 bits from the DATA field of this instruction are loaded into the register specified by the Destination (D) field. For register addresses, see Appendix 1. I/O is forbidden.

4.12.2 Options: None. Note that the flags (C, V, G, or L) are not changed by this micro-instruction.

4.13.3 Assembler Format: L D, DATA

L = symbol for Load Immediate

D = register symbol (e.g.: R5, SDR, MAH,) followed by a comma

DATA = either the operand expressed in hexadecimal (e.g.: X'9A') or as the low order 8 bits , L (DATA), or the high order 8 bits, H (DATA) of the value of a symbol defined by DATA.

4.13 COMMAND

4.13.1 Description: The command micro-instruction results in the performance of the following machine functions as specified by the state of the COMMAND CODE bits.

4.13.2 Options:

Bits set	Definition
5	Multiply Mode
4	Divide Mode
4, 5	Repeat Mode
7	Memory Read, Full Cycle
6, 7	Memory Write, Full Cycle
9	Reset Bank*
8	Set Bank*
8, 9	Trigger Bank
11	Reset Utility*
10	Set Utility*
10, 11	Trigger Utility*
12	Clear Memory Parity Fail*
13	Set Wait Alarm*
14	Reset Wait Alarm*
15	Power Down (this Command initializes the system)

* = flip-flops

4.13.3 Assembler Format: C F

C = symbol for Command

F = symbol for machine function

Symbol	Definition
MPY	Multiply
DIV	Divide
RPT	Repeat
MR	Memory Read, Full Cycle
MW	Memory Write, Full Cycle
CB	Reset Bank*
SB	Set Bank*
TB	Trigger Bank*
CUT	Reset (clear) Utility*
SUT	Set Utility*
TUT	Trigger Utility*
CMP	Clear Memory Parity Fail*
SWA	Set Wait Alarm*
CWA	Reset (clear) Wait Alarm*
POW	Power Down

* = flip-flops

A memory cycle consists of two half cycles: memory read, and memory write - in that order.

On a memory read, the memory location is read out, its contents are destroyed, then the memory location is restored and data is not lost.

On a memory write, the memory location is read out, but the data is not saved. Instead, the

contents of the data register MDR are written back into the addressed memory location.

The following rules should be followed when programming around memory Commands:

1. Memory address register must always be loaded with the desired address before the memory command.

2. If a write operation is intended, the data register must always be loaded with the desired data before the memory command.

3. Observe this general rule: If memory registers are addressed by any micro-instruction when a memory cycle is in progress, that operation is inhibited until the memory cycle is completed.

The Bank flip-flop controls the addressing of certain registers. See Appendix 1.

The Utility flip-flop has no hardware function assigned to it, it is only for program control. Its condition can be tested with the Test micro-instruction.

The Wait Alarm is a flip-flop whose output is tied to the "wait" lamp on the display. Its purpose is to give a visual indication to the programmer when the Processor is halted or idling.

The Power Down Command initializes the entire system, and is generally used when "house keeping" operations are completed after the detection of a power failure.

The Multiply command locks the Processor into a tight three instruction loop which is transversed 16 times until the Multiply function is completed. The command must be wired into an odd address. The Multiply loop would then follow:

odd address	C	MPY	not part of the loop
even address	L	YD, YD, SR	
odd address	A	YD, YD, CO	

The third instruction is not wired in. Instead, the Load is done twice: first with YD as Source and Destination, then with YDP1 as Source and Destination.

The add is done only if the hidden load resulted in a carry. If the above instruction set is not used, the results cannot be guaranteed.

The Divide command locks the Processor into a three instruction loop which is traversed 16 times until the Divide function is completed. The command must be wired into an odd address. The Divide loop would then follow:

```
C   DIV              not part of the loop
L   YD, YD, SL
A   YD, YD, CO
```

The third instruction is not wired in. Instead, the load is done twice: first with YDP1 as Source and Destination, then with YD as Source and Destination. The Add modifies YD only if it results in a carry. If the above instruction set is not used, the results cannot be guaranteed.

The Repeat command, if the counter is not empty, causes the Processor to repeat the next sequential micro-instruction the number of times previously set in the counter register. Each execution causes the counter to decrement until zero is reached when the Processor resumes normal instruction sequencing. If the counter is zero when the repeat command is given, the next sequential micro-instruction is skipped.

The command Repeat must be wired into an odd address. Any instruction may be repeated that does not result in a branch, e.g. Branch, Load RAL, Decode. Also, it is suggested that memory commands not be repeated.

4.14 TEST

4.14.1 Description: If any of the machine functions specified by the TEST CODE is true, the Greater flag (G) is set and the Less flag (L) is reset. If the function specified by the TEST CODE is false, the Less flag (L) is set and the Greater flag (G) is reset.

4.14.2 Options: The testable functions are:

Bit set	Definition
5	I/O Interrupt
6	Auto Restart
7	Console Interrupt
8	Console Single Instruction
9	Utility flip-flop
10	Memory Parity Fail flip-flop
11	Primary Power Fail

4.14.3 Assembler Format: T F

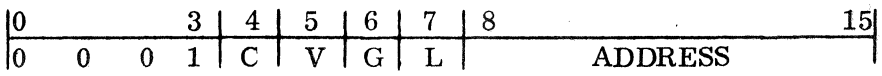
T = symbol for Test

F = symbol for machine function

Symbol	Definition
ATN	I/O Interrupt
ARST	Auto Restart
CATN	Console Interrupt

Symbol	Definition
SNGL	Console Single Instruction
UT	Utility flip-flop
MPF	Memory Parity Fail
PPF	Primary Power Fail

4.15 BRANCH ON CONDITION



4.15.1 Description: the Branch on Condition micro-instruction results in a transfer in the micro-program sequence, if any of the specified conditions (Carry, oVerflow, Greater than zero, Less than zero) are true. The flags (in the Flag Register) are tested against the flags specified by bits (4:7) of the instruction. If any flag and its corresponding bit in the instruction are both set, control of the program is given to the location specified by the ADDRESS field, otherwise the next sequential instruction will be performed.

4.15.2 Options: None.

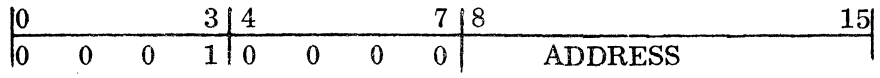
4.15.3 Assembler Format: B CC, ADDRESS

B = symbol for Branch on Condition

CC = Condition Code (C, V, G, or L: any one, any two e.g.: CV or VL etc., any three e.g.: CVG or VGL etc., or all four, CVGL, may be specified)

ADDRESS = symbolic or hexadecimal address where the micro-program transfers if conditions are met

4.16 BRANCH ON COUNTER



4.16.1 Description: the Branch on Counter micro-instruction results in a transfer in the micro-program sequence, if the Counter Register does not equal 'one'. If the condition is met, the Counter Register is decremented by 'one' and control of the program is given to the location specified by the ADDRESS field. If the counter does equal 'one', it is decremented to zero and the next sequential instruction will be performed.

4.16.2 Options: None.

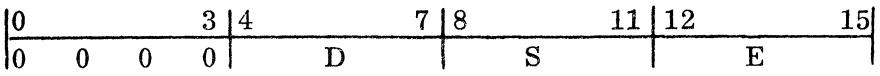
4.16.3 Assembler Format: B CTR, ADDRESS

B = symbol for Branch

CTR = Counter Register

ADDRESS = symbolic or hexadecimal address where the micro-program transfers if conditions are met.

4.17 DECODE



4.17.1 Qualifications: all zeros in RD are interpreted by the hardware as illegal and will result in an unconditional branch to location X'200' in ROM. Therefore, Decode is qualified by zeros in RD 0:3 and bits 4:15 are <u>not</u> all zeros. Only two sets of Destinations and Sources are allowed: the A Register (AR) as Destination and User's Destination (YD) as Source; or the Location Counter (LOC) as both Destination and Source.

4.17.2 Description: the Decode micro-instruction is primarily used to generate a phase change (see Chapter 5 for phase descriptions) although other options are available. Decodes are treated by the hardware as an Add or a Load. The Decode instruction terminating the Phase Zero micro-instruction set must have AR as its destination and YD as source. If the Instruction Register (IR) holds a user's Register to Register (RR) instruction, the contents of the General Register specified by IR 8:11 (YD) are loaded to the AR. If IR does not hold a user's RR instruction, the Location Counter (LOC) is forced to be both Source and Destination. The AR input to the ALU is forced to X'0002' and an Add is performed.

Only one Decode instruction is allowed in Phase Zero. Only the Phase Zero decode may result in a load or an add. Only the Phase Zero decode is written with AR the Destination and YD the Source. All other Decodes must have LOC as Source and Destination and are treated as Adds. The LOC will be incremented by two - AR input to ALU = X'0002' - unless we are exiting Phase Zero with user's RR instruction, exiting Phase One, or entering Phase Three. If the Decode is not to increment LOC, the AR input to ALU is X'0000' and LOC does not change. Other actions peculiar to the Decode micro-operation follow under options.

4.17.3 Options: Note that the character "x" in any bit position indicates a "don't care" state.

E field	Definition
1xxx	A Memory read is generated unless the conditions Phase Zero and RR user format exist.

E field	Definition
x1xx	The contents of the Flag Register (FLR) are jammed to the Condition Code Register (CCR)
xx1x	The Flag Register (FLR), Counter (CTR), Utility Switch (UT), and Register Bank (BANK) are cleared.
xxx1	The Phase Pointer and ROM Address will change. The indirect Source Pointer (PTYS) will set if we are exiting Phase Zero with an RR or indexed RS or RX user's operation in the Instruction Register. PTYS will reset the first time an indirect source is used.

CURRENT PHASE	IR	INDEX?	NEW PHASE	PTYS	NEW ROM ADDRESS
0	RR	NA	2	Sets	*
0	RX	YES	1	Sets	X'00C'
0	RX	NO	1	Stays Reset	X'008'
0	RS	YES	1	Sets	X'004'
0	RS	NO	2	Stays Reset	*
1	NA	NA	2	Resets	*

*ROM Address generated by DROM

On exiting Phase Two, the Decode instruction tests for ATN, CATN, SNGL, MPE, and PPF. If any condition is true, Phase Three is entered and ROM Address is forced to X'014'. If none are true, Phase Zero is entered and ROM Address is forced to X'010'.

4.17.4 Assembler Format: D D,S,E

D = symbol for Decode

D = A register symbol, either AR or LOC, followed by a comma

S = A register symbol, either YD or LOC, followed by a comma if an assembler option is specified in the E field

The following symbolic options may appear in the E field:

Symbol	Definition
MR	Memory Read. Bit 12 = 1
JAM	Load CCR from FLG. Bit 13 = 1
CL	Clear FLR, CNTR, UT, and BANK. Bit 14 = 1
PC	Phase Change. Bit 15 = 1
P0	Phase Zero. Bits 12:15 = 1011
P1	Phase One. Bits 12:15 = 1011
P2J	Phase Two, Jam. Bits 12:15 = 1111
P2N	Phase Two, No Jam. Bits 12:15 = 1011
P3	Phase Three. Bits 12:15 = 1011

CHAPTER 5

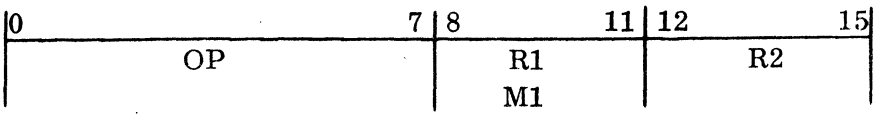
USER ORIENTED PHASE DESCRIPTION

The GE-PAC 30-2 'Micro-Processor' is, to a certain extent, oriented toward the standard GE-PAC 30 user's instruction set. The user's instruction is decoded to define many hardware and firmware functions before actually entering the micro-routine that will execute the instruction.

In the GE-PAC 30-2 there are four hardware conditions known as "phases". Each phase has corresponding sets of micro-instructions. In general, Phase Zero is dedicated to user's instruction fetch and class decoding: Phase One to indexing for the second operand; Phase Two to user's instruction execution; and Phase Three for interrupt service and display support. These phases effect and in-turn are effected only by the Decode micro-instruction. When the Decode micro-instruction is used to bring about a phase change, the subsequent state of the phase pointer is dependent on user's instruction format; whether or not the instruction is indexed and the current state of the phase pointer. (See Figure 1.)

User instruction format is specified by bits 0:3 of the instruction word. The three formats are:

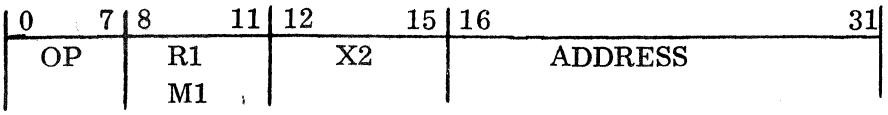
1. Register to Register (RR)



OP is an 8 bit field specifing, in all formats, the operation to be performed. R1 in all formats, is a 4 bit field containing the address of the register which holds the first operand or the mask (M1) for Branches. R2 is a 4 bit field containing the address of the

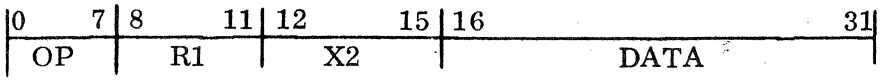
register which holds the second operand. Operation codes X'0n', '2n' and '9n' are recognized by the Processor as RR.

2. Register to Indexed Memory (RX)



X2 is a 4 bit field containing the address of the register which holds the index value. Bits 16:31 specify an address which can be modified by the index value to specify where in core memory the second operand resides. Operation codes X'4n', '6n', and 'Dn' are recognized by the Processor as RX.

3. Register to Storage (RS)



Bits 16:31 contain the data constant to be used as the second operand (immediate). Operation codes X'Cn' and 'En' are recognized by the Processor as RS.

If bits 12:15 of RX and RS instructions are zero, no indexing is to take place.

Within the user oriented firmware environment, user instruction execution begins when Phase Zero is entered. Memory has been read from the location specified by the Location Counter; the Location Counter has been incremented by two and the ROM address register was forced to X'010', the starting address of the Phase Zero micro-instruction sequence.

In Phase Zero, the Instruction Register (IR) is loaded from MDR and MAR is loaded from LOC. The instruction format is determined and on the Decode instruction terminating

Phase Zero, either Phase One or Phase Two will be entered. (See Figure 2.)

Phase One will be entered if the user's instruction format is RX or indexed RS. When Phase One is entered, memory is read putting the second half of the user's instruction in MDR, and LOC is incremented by two.

There are 3 Phase One instruction sets:

1. If indexed RX, ROM Address is forced to X'00C' where the contents of the General Register specified by bits 12:15 of IR (YS) are fetched and added to MDR. This 'effective address' is then loaded to MAR.

2. If non-indexed RX, ROM Address is forced to X'008' where MDR is loaded unaltered to MAR.

3. If indexed RS, ROM Address is forced to X'004' where the contents of the General Register specified by IR bits 12:15 (YS) are fetched and added to MDR, creating the 'effective data'.

Phase One is exited by a Decode micro-instruction which sets the Phase Pointer to Phase Two, and only for the two RX micro-routines, issues a memory read to fetch the second operand.

Phase One is skipped if the user's instruction format is RR or non-indexed RS. If RR, the AR is loaded from YD setting up the first operand; the LOC is not incremented; and Phase Two is entered. If non-indexed RS, the Location Counter is incremented by two and Phase Two is entered.

Anytime Phase Two is entered, either from Phase Zero or Phase One, the Decoder ROM (DROM) is interrogated. DROM is addressed by the operation code (bits 0:7) of IR. Each user's instruction has a unique 12 bit word wired into DROM. This word is the starting address of the micro-routine which will execute the specific user's instruction. The DROM readout is automatically jammed into ROM Address.

Illegal user's instructions do not have a DROM word. Interrogating a non-existant word results in zeros, which are jammed into ROM Address. ROM Location X'000' is wired with all zeros. When location zero is read, a hardware defined illegal condition arises, forcing ROM Address to X'200' and setting the Phase Pointer to Phase Three.

Phase Two is dedicated to the execution of user's instructions. When Phase Two is exited, the Decode instruction automatically tests for IO interrupt, Console Attention, Console Single Mode, Memory Parity Fail, and Primary Power Fail. If any of these conditions exists, the Phase Pointer is set to Phase Three, the Location Counter is not incremented, and ROM Address is jammed to X'014', where attempt is made to service the interrupting condition.

If none of the tested items are true, no interrupts are pending so the Phase Pointer is set to Phase Zero, the Location Counter is incremented by two, the next user's instruction is extracted from memory, and ROM Address is jammed to X'010' to re-enter the Phase Zero micro-instruction set.

In Phase Three, after successfully servicing the interrupting condition, the Phase Pointer is set to Phase Zero, the Location Counter is incremented by two, the next user's instruction is extracted from memory and ROM Address is jammed to X'010' to re-enter the Phase Zero micro-instruction set. If the interrupt cannot be resolved, the Processor remains in Phase Three.

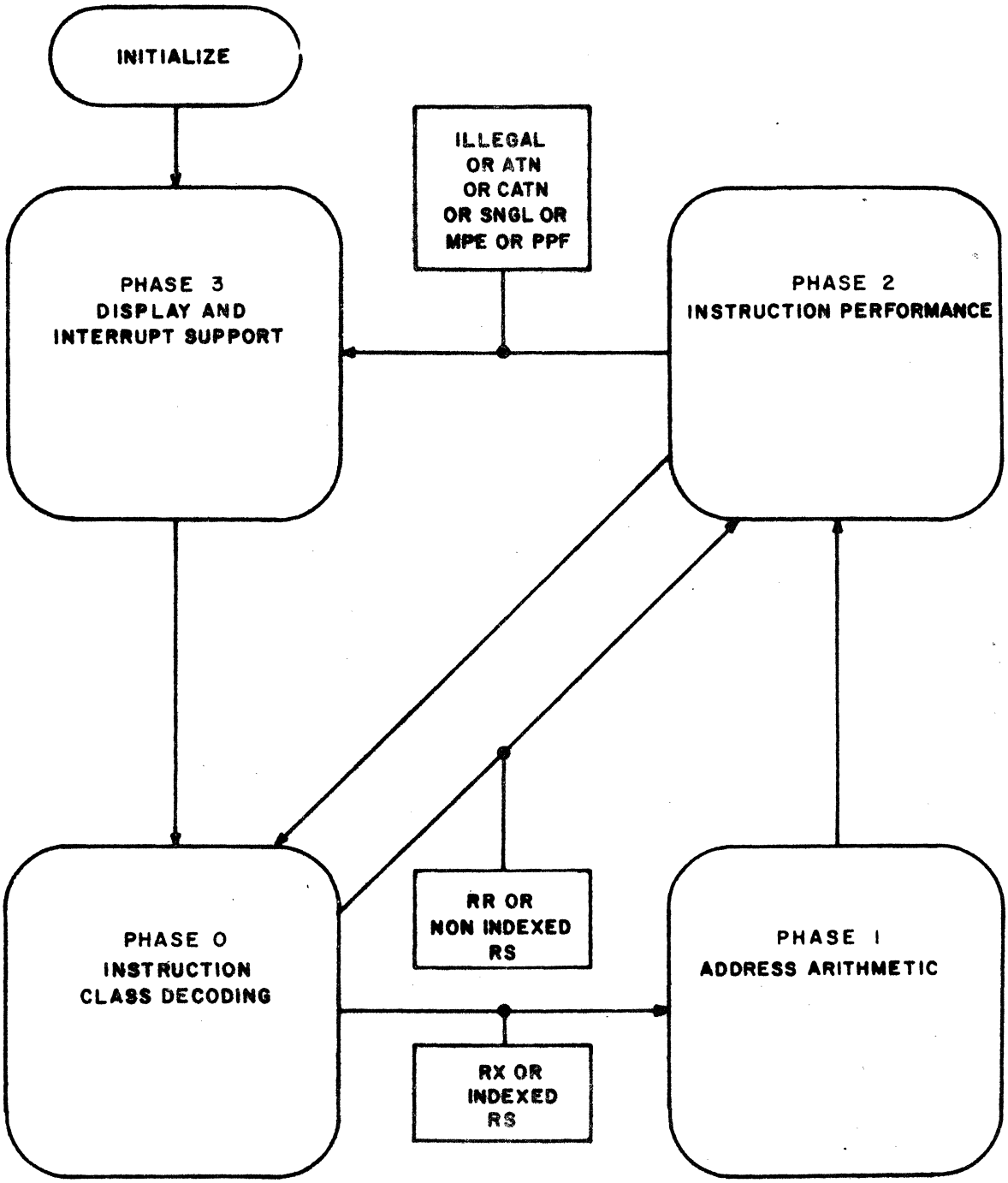


FIGURE I. GENERAL PHASE ANALYSIS

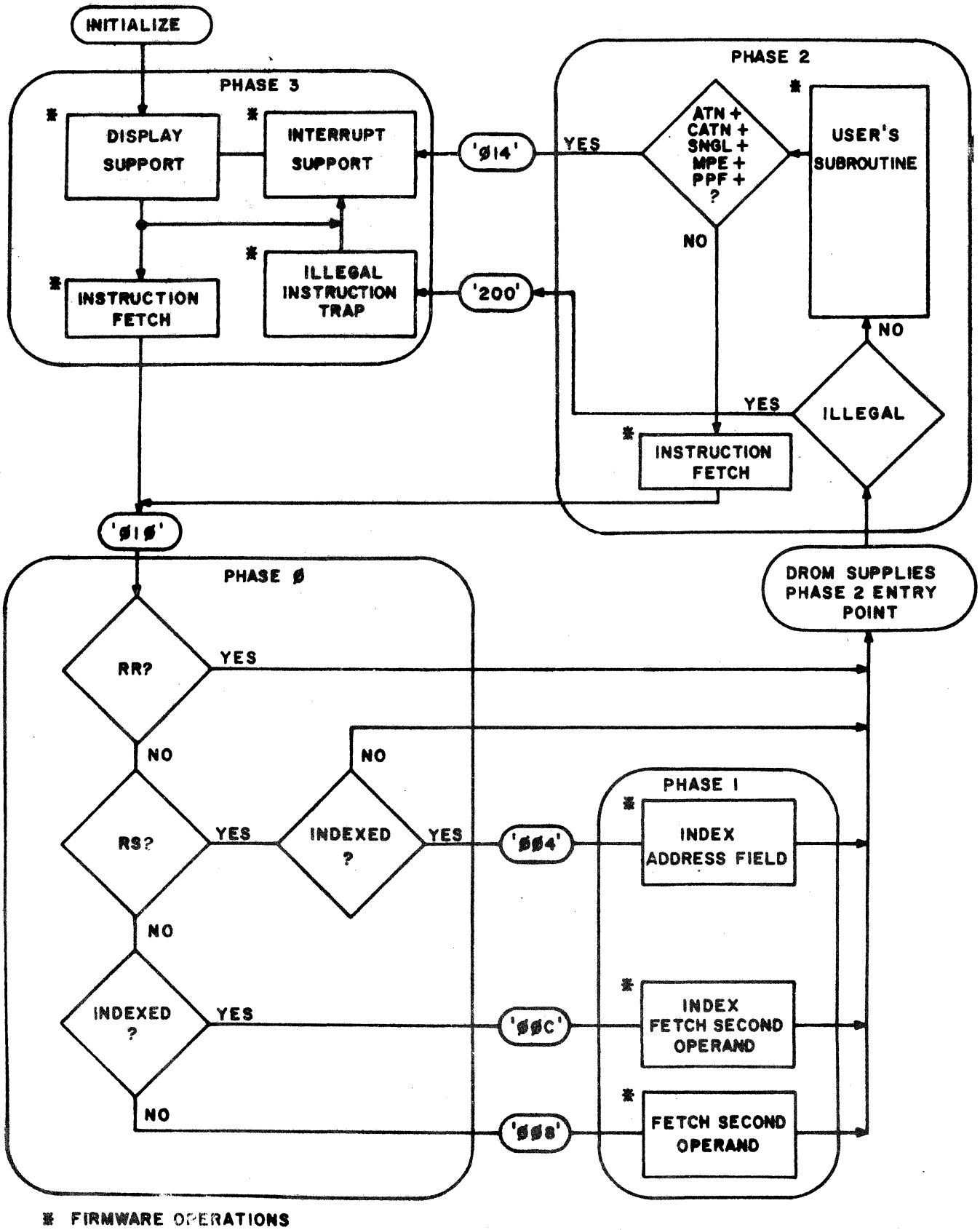


FIGURE 2. GENERAL FLOW OF USER INSTRUCTION

CHAPTER 6

MICRO-PROGRAMMING INPUT/OUTPUT

All input/output support in the GE-PAC 30-2 is done via single load micro-instructions. The position of IO in the Load instruction - Destination or Source - plus the encoding of the E field, define the operation of the interface. Specifically, one of eight control lines is activated.

control line	0	1	2	3	4	5	6	7
	ADRS	DA	DR	STAT	CMD	ACK	-	PPF *

*PPF is generated not by the Load I/O instruction, but by the system Primary Power Fail detection networks.

If IO is the destination and the E field equals xx01, then the source register contains the address of a device.

xx10, then the source register contains data for the device previously addressed.

xx11, then the source register contains a command for the device previously addressed.

If the IO is the source and the E field equals xx01, then the existing external interrupt is recognized and the address of the interrupting device is gated to the destination register.

xx10, then the Processor is requesting data from the device previously addressed. The Data byte is gated to the destination register.

xx11, then the Processor is requesting the status of the device previously addressed. The status byte is gated into the destination register.

To address a device, the address must be established in some register prior to entering the I/O instruction, e.g.

```
L   MR2,X'nn'          DEVICE ADDRESS
L   IO,MR2,ADRS
B   V,ERROR            TEST FOR FSYN
```

If the device is operational, it responds with a sync (SYN). If sync is not received in approximately 50 microseconds, a false sync (FSYN) is generated which sets the V flag in PLR. Note that I/O operations will not terminate until a SYN is received or a FSYN is generated. Devices must be addressed before a command or Data byte can be sent.

```
L   MR2,X'nn'          DEVICE ADRS
L   IO,MR2,ADRS        ADDRESS THE DEVICE
B   V,ERROR            TEST FOR FSYN
L   MR3,IO,STAT        STATUS REQUEST
```

The micro-program can test for external interrupts. If an interrupt is indicated, the micro-program can find out which device interrupted by loading from I/O with Acknowledge (ACK) specified. The sync resulting from ACK loads the interrupting devices address to the Destination Register. The ACK line is not fanned out to all device controllers as are ADRS, DA, DR, STAT, CMD, and PPF. Instead, it is connected to the Device Controller of highest priority. If an interrupt condition is not present in the first Device Controller, the signal (ACK) is passed on to the next Device Controller. The interrupting Device Controller captures the ACK signal and responds with a sync and its address.

```
    T   ATN
    B   G,SRVC
    -   --------
    -   --------

SRVC    L   MR3,IO,ACK      ACKNOWLEDGE
```

NOTE: It takes somewhat less than 100 nanoseconds to pass ACK through a non-interrupting device controller.

The delay of sync response is 200 nanoseconds to cover any skew on the data lines during transfer.

CHAPTER 7

ADDITIONAL SPECIFICATIONS

7.1 Multiple E field definitions are connected with the + sign in the assembler format, e.g.: L MR1, MR2, SL+CO+CI.

7.2 The following registers, if they are specified by the Source field, will result in zeros as the Source Operand: AR, CTR.

7.3 In order for MR0, MR1, or MR2 to be addressed as a Destination Register, the Bank flip-flop must be set.

7.4 In order for RAL, RAH, or YS to be addressed as a Destination Register, the Bank flip-flop must be reset.

7.5 IO must not be both a Source and a Destination in the same micro-instruction.

7.6 On flags: In general, the flags remain unchanged except as follows:

A. Load (with no shift) clears the C flag if Carry Out is specified in the E field unless the load involves IO.

B. Load (with shift) adjusts the C flag if Carry Out is specified in the E field unless the Load involves IO.

C. Add and Subtract adjust all flags if Set Flags is specified in the E field. See item E.

D. AND, OR, and Exclusive OR adjust the G and L flags and clear the V flag if Set Flags is specified in the E field. See item E.

E. The adjustement of the G and L flags by A, S, N, O, X instructions is governed by the following rules:

$$G = \overline{S_0} \cdot (S_1 + S_2 + \text{---} + S_{14} + S_{15} + G_p + L_p)$$

$$L = S_0$$

where S_n = bit n of resulting data
G_p = previous setting of G flag
L_p = previous setting of L flag

F. Test adjusts the G and L flags as follows:

If any tested-for condition is true, G = 1 and L = 0.

If all tested-for conditions are false, G = 0 and L = 1.

7.7 When FLR is the destination register of an instruction that modifies flags, the result in FLR will be the logical OR of the resulting data and the resulting condition code bits (CVGL).

7.8 Flags are not changed by any "immediate" operation unless FLR is the destination register.

7.9 On a Status Request (STAT), FSYN in addition to setting the V flag, sets only bit 13 in the Destination.

7.10 The Initialize Button and the Initialize Relay preset the starting conditions. On power-up, the Initialize Relay remains closed to ground until voltages have reached normal levels. A short period after opening, the system clock starts.

The initialized state is as follows:

BANK flip-flop	Cleared
RAL	Cleared
Utility flip-flop	Cleared
RAS	Cleared
RAH	Cleared
FLR	Cleared
CNTR	Cleared
Phase Pointer	Phase 3
PTYS	Reset
Memory Control	Cleared (inactive)
ROM Data Register	Cleared
Core Current Drivers	Off

Note that the ROM initialized will not cause any problems. All zeros in RD do not cause an illegal, if it is the result of Initialize.

All other registers are in a non-defined state.

The initialize line is also distributed to the I/O devices.

7.11 PPF is a condition testable by the T instruction. It is generated by an optional Primary Power Fail Detector. ROM sequences can test this and take appropriate action. A set of contacts on the POWER switch parallels the PPF condition, thus initializing an orderly shutdown. PPF is distributed to I/O devices on Control Line 7.

7.12 The ROM sequence is allowed about 1 millisecond for orderly shutdown, after which it sets the POW flip-flop (see Command instruction). This stops the system clock and closes the initialize contacts.

7.13 On memory parity fail (option) the memory sets the parity fail (MPF) flip-flop. This can be tested by the ROM Program with a TEST instruction. It can be cleared with a COMMAND instruction.

7.14 The Wait flip-flop is associated with a front panel lamp that is on when the Wait flip-flop is set. If the ROM is waiting for further inputs from the console, it goes into a minor loop and sets this flip-flop with a SWA command. It is cleared with a CWA command.

7.15 The Utility flip-flop can be used by the ROM for program control. It can be set, cleared, or triggered (complemented) via the COMMAND instruction.

7.16 If, during a memory cycle, the attempt is made to use MDR as a Source Register, the execution of the micro-instruction is inhibited, until the "read" portion of the memory cycle is finished.

7.17 If, during a memory cycle, the attempt is made to use MDR or MAR as Destination Registers, the execution of the micro-instruction is inhibited, until the entire memory cycle is finished.

7.18 If, during a memory cycle, the attempt is made to initiate another memory cycle with a Command or Decode micro-instruction, the execution of the micro-instruction is inhibited, until the current memory cycle is finished.

7.19 To implement a transfer within the ROM micro-program, special care should be taken in the loading of the ROM address registers (RAL, RAH).

The decoding network, to fetch an instruction from the ROM, has inputs from the RAL and RAS registers. The RAS register receives information from the RAH register at the same time that the RAL register is loaded.

Consequently, two items are of importance:

A. Loading RAH has no immediate affect on the address for the ROM.

B. Before loading RAL, the RAH register must contain the correct page number.

7.20 One and only one Decode micro-instruction may be used in Phase Zero. That Decode must have AR as the Destination and YD as the Source. All other Decode micro-instructions must have LOC as Destination and Source. There are no quantity restrictions in any other phases.

APPENDIX 1

REGISTER ADDRESSES

CODE	DESTINATION	SOURCE
0000	RAH*MR0**	MR0
0001	RAL*MR1**	MR1
0010	YS* MR2**	MR2
0011	MR3	MR3
0100	MR4	MR4
0101	MAR	MAR
0110	LOC	LOC
0111	PSW	PSW
1000	AR	NULL
1001	IR	IR
1010	MDR	MDR
1011	FLR	IR4
1100	CNTR	NULL
1101	IO***	IO***
1110	YD	YD
1111	YDP1	YDP1

* Bank must be reset
** Bank must be set
*** Not a register

APPENDIX 2
GE-PAC 30-2 MICRO-CODE SUMMARY

OP-CODE	INSTRUCTION
0 0 0 0	DECODE
0 0 0 1	BRANCH
0 0 1 0	TEST
0 0 1 1	COMMAND
0 1 0 0	LOAD
0 1 0 1	LOAD IMMEDIATE
0 1 1 0	OR
0 1 1 1	OR IMMEDIATE
1 0 0 0	AND
1 0 0 1	AND IMMEDIATE
1 0 1 0	EXCLUSIVE OR
1 0 1 1	EXCLUSIVE OR IMMEDIATE
1 1 0 0	ADD
1 1 0 1	ADD IMMEDIATE
1 1 1 0	SUBTRACT
1 1 1 1	SUBTRACT IMMEDIATE

E FIELD	DEFINITION
For A, S, X, N, O:	
1 x x x	No AR to ALU
x 1 x x	Set Flags
x x 1 x	Carry Into ALU
x x x 1	Carry Out of ALU
For L ONLY:	
0 0 x x	Load
0 1 x x	Shift Right
1 0 x x	Shift Left
1 1 x x	Cross Shift
On Shifts Only:	
x x 1 x	Carry Into ALU
x x x 1	Carry Out of ALU
On Non-Shifts:	
x x x 1	Clear Carry

x = "don't care" condition

Commands

Bits Set	Definition
5	Multiply
4	Divide
4, 5	Repeat
7	Mem. Read
6, 7	Mem. Write
9	Reset Bank*
8	Set Bank*
8, 9	Trigger Bank*
11	Reset Utility*
10	Set Utility*
10, 11	Trigger Utility*
12	Clear Mem. Parity*
13	Set Wait Alarm*
14	Reset Wait Alarm*
15	Power Down

* = flip-flops

Tests

Bits Set	Definition
5	I/O Int. (ATN)
6	Auto-restart (ARST)
7	Cons. Ex. (CATN)
8	Cons. Sngl. (SNGL)
9	Utility flip-flop (UT)
10	Mem. Par. Fail (MPF)
11	Prim. Pwr. Fail (PPF)

Load, I/O = Destination

E field	Definition
x x 0 1	Address
x x 1 0	Data Available
x x 1 1	Command

Load, I/O = Source

x x 0 1	Acknowledge
x x 1 0	Data Request
x x 1 1	Status Request

APPENDIX 3

GE-PAC 30-2 INSTRUCTION EXECUTION TIMES

Add	800 nsec
Add Immediate	800 nsec
Subtract	800 nsec
Subtract Immediate	800 nsec
Exclusive OR	400 nsec
Exclusive OR Immediate	400 nsec
AND	400 nsec
AND Immediate	400 nsec
Inclusive OR	400 nsec
Inclusive OR Immediate	400 nsec
Load	400 nsec
Load Immediate	400 nsec
Load I/O	1200 nsec
Command	400 nsec
Test	400 nsec
Branch on Condition	800 nsec
Decode	800 nsec

Exceptions: The instruction takes 800 nsec: if RAL is specified as a Destination Register.

GE-PAC 30-2 MICRO-INSTRUCTION ASSEMBLER MANUAL

TABLE OF CONTENTS

TABLE OF CONTENTS
(Continued)

CHAPTER 1

INTRODUCTION TO THE MICRO-CODE ASSEMBLER

1.1 INTRODUCTION

The GE-PAC 30-2 computer is controlled by Read-Only-Memory programs. These programs control the flow of information within the registers and core storage. The instructions that make up these micro-programs are quite similar to the instructions for a conventional machine. Micro-instructions are located at various addresses in the ROM and bear the same grammatical structure as more conventional instructions. It is convenient to use the same kind of symbolism that is used in conventional assembler programming. Therefore, micro-programs may be written in a more natural way using symbolic operation codes, operands, and memory locations.

1.2 FEATURES OF THE MICRO-CODE ASSEMBLER

The GE-PAC 30 Micro-Code Assembler assembles Source instructions that have been prepared as a tape or a deck of cards. A listing is generated on the teletype-writer or line-printer. This listing lists the input statements and the generated code. An object tape is produced in a machine readable form. This tape is used as an input to a simulator so that the logic may be checked before the ROM is built. The same tape, or a modified version produced by the simulator, is then used in the manufacturing of the specified Read-Only-Memory.

An assembly is performed in two "passes". This means that the source tape or cards must be read twice. On the first pass, a symbol table is built. This table contains definitions of all names that occur in the program. On the second pass, code is generated. The generated code is placed in a buffer as the listing is created. The buffer holds 256 micro-instructions. The contents of this buffer are punched either when it becomes full, or when an ORG pseudo-instruction changes the origin of code generation.

During the first pass the user has the option of stopping on recognized errors. This permits corrections to be made as the pass proceeds.

1.3 INPUT FORMAT

The micro-program may be prepared as a deck of cards or as a source tape. If cards are to be the source input, they will be prepared with one instruction per card. The following format is recommended, although the assembler will accept free format.

1. Columns 1 through 6 - Label
2. Column 9 - 0.2.8 "space" character
3. Columns 10 through 14 - Symbolic Operation Code
4. Column 15 - 0-2-8 "space" character
5. Column 16 on - Symbolic Operands
6. Column 35 - 0-2-8 "space" character
7. Column 36 on - Comments

Items 1, 6, and 7 are optional. Items 2, 4, and 6 are optional depending on the type of card reader used. If the card reader used generates a column strobe, the 0-2-8 "space" character may be replaced by blank columns - keypunch space bar. The cards must be front slashed (IBM form X28-6509-2), and prepared on an IBM 029 keypunch. If the operands occupy more space than that suggested above, the comment field may be moved right.

The comment field is restricted in length because of the narrow pages that a teletypewriter types. Longer statements will be truncated to 67 listing spaces total.

The Assembler needs to see only one space between source statement fields. Superfluous blank columns (Spaces) are ignored. The Assembler reformats the input instructions which may cause the appearance of the listing to differ from the appearance of the source cards.

Source tapes are prepared by the Editor in the same format as the user code assembler. Each field of the source tape is separated by one or more blank characters. The code is ASCII. Each instruction is terminated by a carriage return. If the teletypewriter is the source input device, each instruction on the source tape will be separated from the next by a minimum of six delete characters. This is necessitated because of the start and stop characteristics of the teletypewriter paper tape reader. This format is normally produced by the Editor and is of no real concern to the user.

CHAPTER 2

GE-PAC 30-2 MICRO-CODE ASSEMBLY LANGUAGE

2.1 LABELS

The Read-Only-Memory contains sixteen-bit words. Each word forms one instruction. The words are referred to by addresses. The purpose of the label field is to give symbolic names to the addresses. Labels are formed by using one to six alphanumeric characters. Longer labels are truncated to six characters. The following symbols are valid labels:

T2
LOOP25
N
STOP

The following symbols are invalid as labels for the reasons given:

2TOP	first character is not alphabetic
COMMAND	more than six characters
A to D	contains a blank
X4.2	contains a special character, a period

2.2 OPERATIONS

The micro-assembler will interpret the following micro-operations:

SYMBOL	DESCRIPTION
L	Load
A	Add
S	Subtract
N	AND
O	OR
X	Exclusive OR
B	Branch
T	Test
C	Command
D	Decode

The single letter symbols are the mnemonics that must be used in the operation field to describe a micro-operation.

CHAPTER 3

INSTRUCTION FORMATS

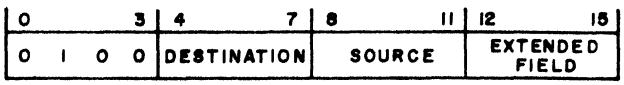
3.1 LOAD INSTRUCTION

The basic action of the Load instruction is to load one register with the contents of another. At the same time, one or more optional operations may be performed. Thus, there may be a one place shift, or a Carry may be shifted in. The Load instruction is written:

```
      L   REG1, REG2
or    L   REG1, REG2, OPT
or    L   REG1, REG2, OPT1+OPT2+...OPTn
```

where REG1 and REG2 are Register names and OPT, OPT1, OPT2, ... OPTn are the names of extended field options.

The format of the assembled instruction is:



The names and hexadecimal addresses of the registers that are used in the Load instruction are shown on Table 1.

In the absence of specific extended field options, the L instruction assumes Carry In and Out only. The extended option field of the instruction is split into subfields. Each subfield controls a specific operation. The occurrence of an extended option in the Source instruction causes the appropriate subfield to be set with the appropriate value. The list of extended field options is scanned from left to right, and accordingly the right-most operation is used in the event of conflict.

In the absence of further specification, the extended operation field contains a 3. The set of extended field options is listed in Table 2.

Typical instructions are:

```
L   MAR, LOC
L   AR, YD, SR
L   IO, MDR, ADRS+CS
L   FLR, IO, STAT+SL
```

3.2 INSTRUCTIONS A, S, N, O, X

The basic action of the instructions A, S, N, O, and X is to combine the contents of the Source and the AR Registers, and to return the result to the Destination Register. At the same time, one or more extended field options may be performed. Accordingly, these instructions are written:

```
      I   REG1, REG2
or    I   REG1, REG2, OPT
or    I   REG1, REG2, OPT1+OPT2+...OPTn
```

where REG1 and REG2 are register names; OPT, OPT1, OPT2, and OPTn are names of extended field options; and I stands for the Op-Code.

The format of an assembled instruction is:

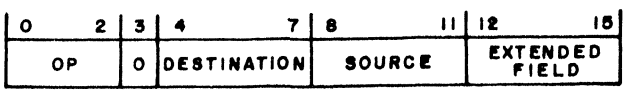


TABLE 1. LOAD INSTRUCTION REGISTER ASSIGNMENTS

FUNCTION	NAME	ADDRESS
Micro-Register 0	MR0	0
ROM address high	RAH	0
Micro-Register 1	MR1	1
ROM address lower	RAL	1
Micro-Register 2	MR2	2
The General Register addressed by the YS field of IR	YS	2
Micro-Register 3	MR3	3
Micro-Register 4	MR4	4
Memory Address Register	MAR	5
Location Counter	LOC	6
Program Status Word	PSW	7
A Register	AR	8
Instruction Register	IR	9
Memory Data Register	MDR	A
Flag Register	FLR	B
The YD field of IR	IR4	B
Counter	CNTR	C
I/O Bus	IO	D
The General Register addressed by the YS or YD field of IR	YD	E
The odd member of the General Register pair addressed by the YS or YD field of IR	YDP1	F

TABLE 2. EXTENDED FIELD OPERATIONS FOR LOAD

FUNCTION	NAME	MASK	VALUE
Carry In but not Out	CI	3	2
Carry Out but not In	CO	3	1
No Carries	NC	3	0
Carry In and Out	C	3	3
Shift Left	SL	C	8
Shift Right	SR	C	4
Cross Shift	CS	C	C
Address	ADRS	3	1
Data Available	DA	3	2
Command	CMD	3	3
Acknowledge	ACK	3	1
Data Request	DR	3	2
Status Request	STAT	3	3

where the operation codes are such that the leading four bits are as follows:

SYMBOL	HEX CODE
A	C
S	E
N	8
O	6
X	A

All the registers permitted on the Load instruction and listed on Table 1 except IO, may be used by these instructions.

In the absence of specific extended field options, this set of instructions assumes Carry In and Out and set flags. The extended field of the instruction is split into subfields. Each subfield controls a specific operation. The occurence of an extended option in the Source instruction causes the appropriate subfield to be set to the corresponding value. Table 3 lists the extended options and shows both the values and masks defining the subfields. If conflicting options are specified, those to the right take precedence. In the absence of further specification, the extended field equals 7.

TABLE 3. EXTENDED OPTIONS FOR A, S, N, O, X

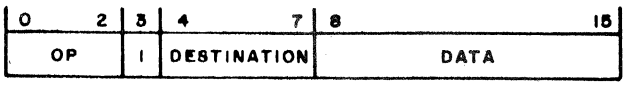
FUNCTION	NAME	MASK	VALUE
Carry In but not Out	CI	3	2
Carry Out but not In	CO	3	1
No Carries	NC	3	0
Carry In and Out	C	3	3
No flags	NF	4	0
No A Register	NA	8	8

Typical instructions are:

```
A    MR2, MR3, NA+NF+CI
S    YD, MR4, CO
S    MAR, MAR, CI+NF
N    AR, MR0
O    MDR, MDR, NA+NC
X    MAR, LOC
```

3.3 IMMEDIATE INSTRUCTIONS

The set of operations L, A, S, N, O, and X also occur in an immediate form. In this form, the Source Data does not come from a register, but from eight bits of the instruction itself. When this happens, extended field options are forbidden. The format of an assembled Immediate instruction is:



where the operation code generates the first four bits as follows:

SYMBOL	HEX CODE
L	5
A	D
S	F
N	9
O	7
X	B

The assembler allows the eight-bit Immediate constant to be specified in a variety of ways. The simplest is a hexadecimal constant, in which case the instruction takes the form:

```
     I    REG, X'n'
or   I    REG, =X'n'  (the equal
               sign is optional)
```

The 'n' is a number written as one or more hexadecimal digits. A single digit produces a right-justified constant. If more digits are written, the right-most two of the numeric field, truncated to four digits are used to generate the eight-bit Immediate operand

	SYMBOLIC	GENERATED CODE
L	MR3, X'1'	5301
L	MR3, X'3F'	533F
L	MR3, X'26A'	536A
L	MR3, X'A5672'	5367

It is convenient to use the Immediate constant to represent an address. Addresses in the micro-machine are sixteen bits long, and the Immediate constant is only eight. Therefore, provision is made to select either the high or low order eight-bits of an address as an Immediate constant. Instructions with such Immediates are written:

```
      I     REG, L(S)
or    I     REG, H(S)
or    I     REG, =L(S)      the equal sign is
or    I     REG, =H(S)      optional
```

where L stands for the low order byte, and H for the high order byte. S is a symbol that is defined in the label field of another instruction. Typical instructions are:

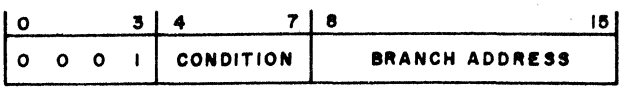
SYMBOLIC			GENERATED CODE	
Name	Op	Operand	Adrs	Data
ABX	L	RAH, H(CDY)	0408	5004
CDY	S	MR4, =X'2A	0409	F42A
	L	MR4, L(ABX)	040A	5408

3.4 BRANCH INSTRUCTIONS

The Branch instruction is written:

```
      B     COND, S
```

where COND is the name of a condition, S is the label of the instruction to which control may be given. The format of the assembled instruction is:



The Branch instruction contains only an eight-bit field for the Branch address, the eight low order bits of the required address. Typical Branch instructions are:

SYMBOLIC			GENERATED CODE	
Name	Op	Operand	Adrs	Data
POS	B	G, LOC	0027	1235
LOC	B	CG, POS	0035	1A27

The Branch instruction may cause the Flag Register to be tested. The output of the test causes either the next sequential instruction to be performed, or control to pass to the specified place in the Read-Only-Memory.

The tests made have the symbolic names listed on Table 4. Compound conditions are also listed.

The Branch instruction may also test the state of the Counter register. This is done by specifying CTR as the condition. The output of the test causes either the next sequential instruction to be performed, or control to pass to the specified place in the Read-Only-Memory.

MEANING	SYMBOL	HEX CODE
Counter not ONE	CTR	0

3.5 COMMAND INSTRUCTIONS

The Command instruction is written:

```
      C     COM
or    C     COM1+COM2+.....COMn
or    C     X'n'
```

where COM, COM1, COM2 ..... COMn are the names of specific commands, and 'n' is a hexadecimal constant useful for setting up non-standard bit configurations. The first form is used for a single command, and the second form is used for multiple commands. If conflicting multiple commands are given, the ones to the right in the string of commands take precedence. The format of the assembled instruction is:

Typical Commands are:

```
      C     MR
      C     RPT + CUT
      C     X'5AA'
```

The Commands listed on Table 5 may be given. The mask indicates the bit positions of the twelve-bit command literal that are cleared before insertion of the value.

TABLE 4. BRANCH TESTS

MEANING	SYMBOL	HEX CODE
Less than zero	L	1
Greater than zero	G	2
Greater than or less than zero	GL	3
Overflow	V	4
Overflow or less than zero	VL	5
Overflow or greater than zero	VG	6
Overflow or greater than or less than zero	VGL	7
Carry	C	8
Carry or less than zero	CL	9
Carry or greater than zero	CG	A
Carry or greater than or less than zero	CGL	B
Carry or Overflow	CV	C
Carry or Overflow or less than zero	CVL	D
Carry or Overflow or greater than zero	CVG	E
Carry or Overflow or greater than or less than zero	CVGL	F

TABLE 5. COMMANDS

FUNCTION	NAME	MASK	VALUE
Divide	DIV	C00	800
Multiply	MPY	C00	400
Repeat	RPT	C00	C00
Memory Read	MR	300	100
Memory Write	MW	300	200
Priviledged Write	PW	300	300
Clear the Bank	CB	0C0	040
Set the Bank	SB	0C0	080
Clear Utility flip-flop	CUT	030	010
Set Utility flip-flop	SUT	030	020
Trigger Utility flip-flop	TUT	030	030
Power Down	POW	001	001
Clear Wait Alarm	CWA	00C	004
Set Wait Alarm	SWA	00C	008
Clear Memory Parity	CMP	002	002

3.6 TEST INSTRUCTIONS

The Test instruction is written:

```
     T      TEST
or   T      TEST1+TEST2....+TESTn
or   T      X'n'
```

where TEST, TEST1, TEST2, ...TESTn are the names of specific tests and 'n' is a hexadecimal constant useful for creating non-standard configurations. The first form is used for a single test and the second form is used for multiple tests. All the tests are independent, and there is no possibility of conflict. The format of the assembled instruction is:

Typical Test instructions are:

```
T     UT
T     ATN+CATN+PPF
T     X'408'
```

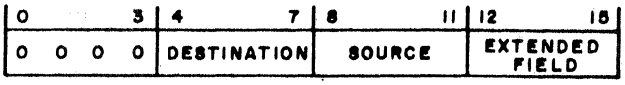
The Tests that may be made are listed on Table 6.

3.7 DECODE INSTRUCTIONS

The action of the Decode instruction is determined by the hardware at execution time. One or more extended field options may be performed. The Decode instruction is written.

```
      D   REG1,REG2
or    D   REG1,REG2,OPT
or    D   REG1,REG2,OPT1+OPT2+...OPTn
```

where REG1 and REG2 are register names and OPT, OPT1, OPT2....OPTn are extended field options. The format of the assembled instruction is:



All registers allowed on A, S, N, O, and X may be used on Decode.

The extended field is split into single-bit subfields, each bit controlling a specific operation. The occurrence of an extended field option in the Source instruction causes the appropriate subfield to be set to the corresponding value. The list of extended options on Table 7 shows both the values and masks defining the subfields. If conflicting options are specified, those to the right take precedence. In the absence of further specification, the extended field is set to zero.

Typical instructions are:

```
D     AR,YD,P0
D     LOC,LOC
D     LOC,LOC,MR
D     LOC,LOC,P2J
```

TABLE 6. TESTS

FUNCTION	NAME	MASK	VALUE
Unassigned	800	800	800
I/O Interrupt	ATN	400	400
Auto Restart	ARST	200	200
Console Interrupt	CATN	100	100
Console Single Mode	SNGL	080	080
Utility flip-flop	UT	040	040
Memory Parity Fail	MPF	020	020
Primary Power Fail	PPF	010	010
Fast I/O Interrupt	FAST	008	008
Unassigned	004	004	004
Unassigned	002	002	002
Unassigned	001	001	001

TABLE 7. DECODE EXTENDED OPTIONS

FUNCTION	NAME	MASK	VALUE
Memory Read	MR	008	008
Jam FLR to CCR	JAM	004	004
Clear FLR, CNTR, BANK, UT	CL	002	002
Phase Change	PC	001	001
Phase Zero	P0	00F	00B
Phase One	P1	00F	00B
Phase Two Jam	P2J	00F	00F
Phase Two, No Jam	P2N	00F	00B
Phase Three	P3	00F	00B

CHAPTER 4

PSEUDO INSTRUCTIONS

4.1 ORG

ORG specifies the position in the Read-Only-Memory where the assembled program will begin. ORG pseudo-instructions can be used freely to cause different parts of the program to be located in different places. The format of the ORG instruction is:

```
ORG    X'n'
```

where'n' is a number of four or fewer digits written in hexadecimal form. (Longer numbers are truncated to the left most four digits.) Typical ORG instructions are:

```
ORG    X'12B'
ORG    X'62'
```

4.2 EQU AND SYN

EQU and SYN are used to define new symbols. Once a symbol is defined, it may be used in exactly the way labels are used. EQU is used to define a symbol to have a literal value. SYN is used to equate a new symbol to a previously defined one. The formats are:

```
S      EQU    X'n'
S1     SYN    S2
```

where S, S1, and S2 are names of symbols composed of one to six alphanumeric characters, and 'n' is a hexadecimal number composed of four or fewer digits. Typical examples of EQU and SYN are:

SYMBOLIC			GENERATED CODE
CONS1	EQU	X'1A98'	(no code generated)
SUBK	SYN	CONS1	(no code generated)
	L	AR, H(CONS1)	581A
	O	MAR, L(SUBK)	7598

4.3 P

The P pseudo-instruction is used to output the current content of the punch buffer and suppress punching origin addresses when subsequent punching occurs. The format of the P instruction is:

```
P    (no argument is necessary)
```

4.4 Z

The Z pseudo-instruction is used to define a hexadecimal constant. This instruction has an argument which is either a hexadecimal constant or a symbolic address. The format of the Z instruction is:

```
     Z     X'n'
or   Z     S
```

where 'n' is a hexadecimal constant of four or fewer digits and S is a previously defined symbol of one to six alphanumeric characters.

Typical Z instructions are:

SYMBOLIC			GENERATED CODE	
			Adrs	Data
TALLY	Z	TOTAL	0282	0283
TOTAL	Z	X'A12F'	0283	A12F
SUM	Z	TALLY	0284	0282

In normal use, the P instruction is followed by a block of Z pseudo-instructions defining Decoder ROM data. For example:

```
P
Z   X'0000'
Z   SYMB1
Z   SYMB2
Z   SYMBn
END
```

The P instruction causes that executable part of the micro-program remaining in the punch buffer to be output.

The subsequent Z pseudo-instructions begin filling the punch buffer with their respective hexadecimal arguments.

The END statement causes the assembly process to terminate. Depressing the EXECUTE switch will cause the data in the punch buffer to be output without an origin address.

4.5 END

The END pseudo-operation is used to indicate the end of a Source program. No special instruction is required at the beginning of a micro-code Source program.

CHAPTER 5

OUTPUT FORMAT AND ERROR MESSAGES

5.1 OUTPUT FORMAT

The listing is printed on the teletypewriter or line printer. The listing contains two columns of four hexadecimal digits each. The left hand column contains the address of the instruction in storage. The right hand column contains the sixteen bits of the instruction in hexadecimal form. Following the generated data, and on the same line, the corresponding source data is printed.

Paper tape object code is punched on the teletypewriter or high speed punch. The tape is punched in blocks; each block corresponding to a 256 word section of code or a partial page of code that is headed by an ORG pseudo-instruction. Blocks are separated by blank tape. Each character on the tape represents a hexadecimal digit. Four tape characters are punched in the tape for each sixteen bit word; most significant hexadecimal digit first. The first four characters in a block give the starting address of the block. Subsequent groups of characters give successive words of the generated code. The paper tape codes used for the hexadecimal digits are listed on Table 8.

A block of object information generated by P and Z pseudo-instructions uses the same paper tape codes as instruction data, but no orgin addresses appear.

5.2 ERROR MESSAGES

The following errors are detected and flagged by the characters shown.

ERROR	CHARACTER
Illegal operation code	O
Instruction format error	F
Multiple defined symbol	M
Undefined symbol	U
Bad character in source card	B

When the error option is used to detect errors on the first pass, multiple defined symbols are not flagged until their second occurrence. Undefined symbols are not detected until the symbol table is printed.

Errors generally cause an instruction word of all zeros to be generated. A source card that contains an illegal (bad) Hollerith character is listed up to the illegal character for easy identification.

If the symbol table overflows, the message "OVERFLOW" is printed and the assembly process terminates.

TABLE 8. PAPER TAPE CODES

HEX DIGIT	CODE	HEX DIGIT	CODE
0	1001 0000	8	1001 1000
1	1000 0001	9	1001 1001
2	1000 0010	A	1001 1010
3	1000 0011	B	1001 1011
4	1000 0100	C	1001 1100
5	1001 0101	D	1001 1101
6	1001 0110	E	1001 1110
7	1001 0111	F	1001 1111

CHAPTER 6

MICRO-ASSEMBLER SYNOPSIS

6.1 INPUT FORMAT

If tape is the source, the normal Editor output is accepted as the input format. If cards are the source, the following format is suggested.

COL	DESCRIPTION
1-6	*Symbolic Label
9	*0-2-8 "space"
10-14	Symbolic operation code
15	*0-2-8 "space"
16-71	Symbolic operand field terminated by (0-2-8)* if comments follow

*optional

If column 1 is an asterisk (*), the card generates no code, but is listed.

6.2 OUTPUT LISTING FORMAT

The output listing contains the following information for each input source statement:

1. Absolute location in hexadecimal

2. Absolute content of the location, in the form:

   XXXX

   where X is a hexadecimal digit.

3. Error messages:

   F - Format Error
   O - Op-Code Error
   U - Undefined Symbol
   M - Multiple Defined Symbol
   B - Bad Hollerith Character

4. Symbolic (source) image.

6.3 OUTPUT PAPER TAPE FORMAT

The paper tape is organized into blocks, each representing:

1. all words between one ORG pseudo-instruction and the next.

2. all words up to page end.

3. P pseudo-generated data.

Each sixteen-bit number is punched in four frames of paper tape, each frame representing one hexadecimal digit. Each block on paper tape, in item 1 and 2 above, begins with a sixteen-bit word (four frames) giving the initial location of the block. The punched address must be a halfowrd address. It is therefore punched as twice the actual ROM address. Succeeding words on paper tape represent succeeding locations in memory.

Eight inches of blank tape are left between blocks. The paper tape code used for the hexadecimal digits is as shown earlier on Table 8.

6.4 INSTRUCTION FORMATS

Instruction Class 1

A OP D , S , E (Register to Register)

B OP D , C , or OP D,=C (Immediate)

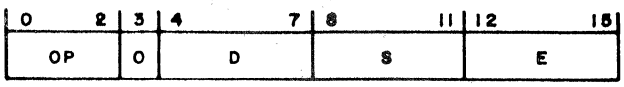
where OP is A Add
N AND
S Subtract
O OR
X Exclusive OR
L Load
D Decode

D is a register name (see 6.6)
S is a register name (see 6.6)
E is the extended option field which has the form M, + ... +Mn where M are modifiers
C is an eight-bit constant which can take the form:

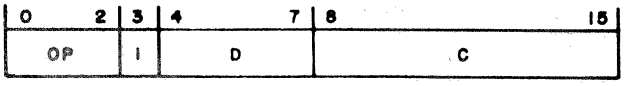
1. X'nn' (n is a hexadecimal digit)
2. H(s) (High order address of a word in ROM; 's' is a symbolic name of a ROM location
3. L(s) (Low order address of a word in ROM)

Assembled format:

A.



B.



Instruction Class 2:

Symbolic Format:

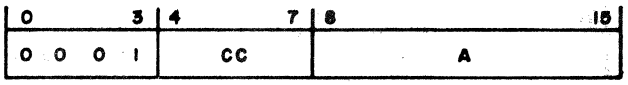
B CC, A

where B is the Branch Op-Code (0001)

CC is the symbolic condition

A is the symbolic address (always uses low order address bits)

Assembled Format:



Instruction Class 3:

Symbolic Format:

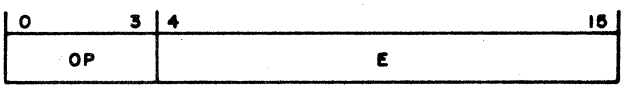
OP E or OP X'n'

where OP is C - command (0011)
T - test (0010)

E is the extended form M,+ ..... +Mn are modifiers for test or command.

'n' is a hexadecimal constant

Assembled Format:



6.5 PSEUDO-OPERATIONS

ORG ORG X'N'

set assembly location counter to N (hexadecimal)

EQU S EQU X'N'

Set the symbol S to the hexadecimal value N.

SYN S_1 SYN S_2

Set the symbol S_1 to the same value as S_2, which must have been previously defined.

Z Z X'N'

Define Constant. Set the current assembly location to Hexadecimal N or the hexadecimal address associated with symbol S_2. The assembled form becomes XXXX where X are hexadecimal digits.

P P

Punch the content of the buffer and suppress punching origin address when subsequent punching occurs.

END END

End current pass of assembly.

6.6 REGISTER NAMES AND EQUIVALENTS

Table 9 lists all register names and equivalents.

TABLE 9. REGISTERS

SYMBOLIC	ABSOLUTE	MEANING
MR0	0	Micro-Register 0
RAH	0	ROM address High
MR1	1	Micro-Register 1
RAL	1	ROM address low
MR2	2	Micro-Register 2
YS	2	General Register addressed by YS field of IR
MR3	3	Micro-Register 3
MR4	4	Micro-Register 4
MAR	5	Memory Address Register
LOC	6	Location Counter
PSW	7	Program Status Word
AR	8	A Register
IR	9	Instruction Register
MDR	A	Memory Data Register
FLR	B	Flag Register
IR4	B	YD field of IR
CNTR	C	Counter
IO	D	I/O Bus
YD	E	General Register addressed by the YS or YD field of IR
YDP1	F	Odd member of the General register pair addressed by the YS or YD field of IR.

6.7 ADD CLASS SUBFIELD MODIFIERS (A, S, N, O, X)

E field is normally 0111 unless further specified. Each modifier has a mask M and a value V as shown on Table 10. Each modifier converts current E code from E_b to E_a as follows:

$$E_a = E_b \cdot \overline{M} + V.M$$

6.8 MODIFIERS FOR L OP-Code

E code is normally 0011. Table 11 lists the modifiers.

6.9 MODIFIERS FOR D OP-CODE

E code is normally 0000. Table 12 lists the modifiers.

TABLE 10. ADD CLASS MODIFIERS

SYMBOLIC MODIFIER	MASK M	VALUE V	MEANING
CI	0011	0010	Carry In but not Out
CO	0011	0001	Carry Out but not In
NC	0011	0000	No Carries
C	0011	0011	Carry In and Out
NF	0100	0000	No Flags
NA	1000	1000	No A Register

TABLE 11. LOAD MODIFIERS

SYMBOLIC MODIFIER	MASK M	VALUE V	MEANING
CI	0011	0010	Carry In but not Out
CO	0011	0001	Carry Out but not In
NC	0011	0000	No Carries
C	0011	0011	Carry In and Out
SL	1100	1000	Shift Left
SR	1100	0100	Shift Right
CS	1100	1100	Cross Shift
ADRS	0011	0001	Address
DA	0011	0010	Data Available
CMD	0011	0011	Command
ACK	0011	0001	Acknowledge
DR	0011	0010	Data Request
STAT	0011	0011	Status Request

TABLE 12. DECODE MODIFIERS

SYMBOLIC MODIFIER	MASK	VALUE	MEANING
MR	1000	1000	Memory Read
JAM	0100	0100	Jam FLR to CCR
CL	0010	0010	Clear
PC	0001	0001	Phase Change
P0	1111	1011	Phase Zero
P1	1111	1011	Phase One
P2J	1111	1111	Phase Two, JAM
P2N	1111	1011	Phase Two, No JAM
P3	1111	1011	Phase Three

6.10 SYMBOLIC CONDITION CODE FOR B OP-CODE

The following coding applies to the Branch micro-op.

SYMBOLIC CC	ABS VALUE	MEANING
CTR	0000	Counter not ONE
L	0001	Less than zero
G	0010	Greater than zero
V	0100	Overflow
C	1000	Carry

The conditions C, V, G, and L may be specified in any combination.

6.11 MODIFIERS FOR C OP-CODE

Table 13 lists the modifiers for the Command op-code.

6.12 MODIFIERS FOR T OP-CODE

Table 14 lists the modifiers for the Test micro-op.

TABLE 13. COMMAND MODIFIERS

SYMBOLIC	MASK	VALUE	MEANING
DIV	C00	800	Divide
MPY	C00	400	Multiply
RPT	C00	C00	Repeat
MR	300	100	Memory Read
MW	300	200	Memory Write
PW	300	300	Priviledge Write
CB	0C0	040	Clear Register Bank
SB	0C0	080	Set Register Bank
TB	0C0	0C0	Trigger Register Bank

TABLE 13. COMMAND MODIFIERS (Continued)

SYMBOLIC	MASK	VALUE	MEANING
CUT	030	010	Clear Utility flip-flop
SUT	030	020	Set Utility flip-flop
TUT	030	030	Trigger Utility flip-flop
POW	001	001	Power Down
CWA	00C	004	Clear Wait Alram
SWA	00C	008	Set Wait Alram
CMP	002	002	Clear Memory Parity

TABLE 14. TEST MODIFIERS

SYMBOLIC	MASK	VALUE	MEANING
800	800	800	unassigned
ATN	400	400	I/O Interrupt
ARST	200	200	Auto-Restart
CATN	100	100	Console Interrupt
SNGL	080	080	Console single mode
UT	040	040	Utility flip-flop
MPE	020	020	Memory Parity Error
PPF	010	010	Primary Power Fail
FAST	008	008	Fast I/O Interrupt
004	004	004	unassigned
002	002	002	unassigned
001	001	001	unassigned

APPENDIX 1
GE-PAC 30-2 REGISTER ADDRESSES

CODE	DESTINATION	SOURCE
0000	RAH*MR0**	MR0
0001	RAL*MR1**	MR1
0010	YS* MR2**	MR2
0011	MR3	MR3
0100	MR4	MR4
0101	MAR	MAR
0110	LOC	LOC
0111	PSW	PSW
1000	AR	NULL
1001	IR	IR
1010	MDR	MDR
1011	FLR	IR4
1100	CNTR	NULL
1101	IO***	IO***
1110	YD	YD
1111	YDP1	YDP1

* Bank must be reset
** Bank must be set
*** Not a register

APPENDIX 2

GE-PAC 30-2 SUMMARY OF INSTRUCTIONS

OP-CODE	INSTRUCTION
0 0 0 0	DECODE
0 0 0 1	BRANCH
0 0 1 0	TEST
0 0 1 1	COMMAND
0 1 0 0	LOAD
0 1 0 1	LOAD IMMEDIATE
0 1 1 0	OR
0 1 1 1	OR IMMEDIATE
1 0 0 0	AND
1 0 0 1	AND IMMEDIATE
1 0 1 0	EXCLUSIVE OR
1 0 1 1	EXCLUSIVE OR IMMEDIATE
1 1 0 0	ADD
1 1 0 1	ADD IMMEDIATE
1 1 1 0	SUBTRACT
1 1 1 1	SUBTRACT IMMEDIATE

Commands

Bits Set	Definition
5	Multiply
4	Divide
4, 5	Repeat
7	Mem. Read
6	Mem. Write
6, 7	Privileged Write
9	Reset Bank*
8	Set Bank*
8, 9	Trigger Bank*
11	Reset Utility*
10	Set Utility*
10, 11	Trigger Utility*
14	Clear Mem. Parity*
12	Set Wait Alarm*
13	Reset Wait Alarm*
15	Power Down

* = flip-flops

E FIELD	DEFINITION
For A, S, X, N, O:	
1 x x x	No AR to ALU
x 1 x x	Set Flags
x x 1 x	Carry Into ALU
x x x 1	Carry Out of ALU
For L ONLY:	
0 0 x x	Load
0 1 x x	Shift Right
1 0 x x	Shift Left
1 1 x x	Cross Shift
On Shifts Only:	
x x 1 x	Carry Into ALU
x x x 1	Carry Out of ALU
On Non-Shifts:	
x x x 1	Clear Carry

x = "don't care" condition

Tests

Bits Set	Definition
5	I/O Int. (ATN)
6	Auto-restart (ARST)
7	Cons. Ex. (CATN)
8	Cons. Sngl. (SNGL)
9	Utility flip-flop (UT)
10	Mem. Par. Fail (MPF)
11	Prim. Pwr. Fail (PPF)
12	Fast I/O Int-(FAST)

Load, I/O = Destination

E field	Definition
x x 0 1	Address
x x 1 0	Data Available
x x 1 1	Command

Load, I/O = Source

x x 0 1	Acknowledge
x x 1 0	Data Request
x x 1 1	Status Request

APPENDIX 3
TELETYPEWRITER/ASCII/HEX CONVERSION TABLE

HEX (MSD) →					8	9	A	B	C	D	E	F
(LSD) ↓	Teletypewriter Tape Channels →			8	DEPENDS UPON PARITY							
				7	0	0	0	0	1	1	1	1
				6	0	0	1	1	0	0	1	1
				5	0	1	0	1	0	1	0	1
	4	3	2	1								
ø	0	0	0	0	NULL	DC_0	SPACE	0	@	P		
1	0	0	0	1	SUM	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_0	S_0	(	8	H	X		
9	1	0	0	1	HT/SK	S_1	)	9	I	Y		
A	1	0	1	0	LF	S_2	*	:	J	Z		
B	1	0	1	1	VT	S_3	+	;	K	[		
C	1	1	0	0	FF	S_4	,	<	L	\		ACK
D	1	1	0	1	CR	S_5	-	=	M	]		ALT. MODE
E	1	1	1	0	SO	S_6	.	>	N	↑		ESC
F	1	1	1	1	SI	S_7	/	?	O	←		DEL

APPENDIX 4
ASCII/CARD CODE CONVERSION TABLE

GRAPHIC	8-BIT ASCII CODE	7-BIT ASCII CODE	CARD CODE	GRAPHIC	8-BIT ASCII CODE	7-BIT ASCII CODE	CARD CODE
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

MICRO-ASSEMBLER OPERATING PROCEDURES

1. GE-PAC 30 Micro-Assemblers are absolute and may be loaded by any of the GE-PAC 30 loaders. Instructions for loading tapes are contained in the Loader Descriptions, Publication Number 06-025A12.

2. Place the tape or deck to be assembled in the symbolic input device.

3. Set data Switch 15 on the computer's console if error detection on the first pass of the assembly is not required. Otherwise, make sure that data Switch 15 is reset.

4. The Micro-Assembler begins at location X'80'. The paper is advanced one page and a title is printed. It is important to position the paper before starting.

5. The first pass proceeds with the source statements being read until the END statement is reached. If data Switch 15 is not set, errors cause the system to halt. When this happens, the offending instruction is listed together with an error flag which describes the error.

6. If Switch 15 is not set and error detection causes a halt, the offending statement must be reprocessed either by replacing the erroneous statement with a corrected one, or re-reading the bad statement with Switch 15 set. Processing is resumed by depressing the EXECUTE button.

7. When the END card is reached, the symbolic table is printed. Errors are flagged. If error correction is being attempted, and the symbol table contains "undefined" symbols, abandon the assembly.

8. Replace the tape or deck in the symbolic input device. When the computer halts after printing the symbol table, press EXECUTE to perform the second pass.

9. During the second pass, listing is performed. If the computer halts, it is because it is ready to produce an object tape. With the GE-PAC 30-1 Micro-Assembler only, verify that the previous instruction listed was ROM or END. Prepare the binary output device for operation. Press EXECUTE. The tape is produced and the computer will again halt.

10. Check that the symbolic output device is ready for printing. Press EXECUTE, listing again proceeds.

11. Repeat steps 9 and 10 as many times as necessary.

12. After step 10 has been performed the final time, the system types ASSEMBLY COMPLETE.

13. To perform another assembly, place the new tape or deck in the symbolic input device and press EXECUTE. Repeat the previous steps.

14. If, for any reason, the second pass must be aborted and restarted without repeating pass one, address X'84' will permit this restart without changing the symbol table.

TABLE OF CONTENTS

APPENDICES

ILLUSTRATIONS

CHAPTER 1

GENERAL DESCRIPTION

1.1 PURPOSE

The GE-PAC 30-2 Micro-Code Simulator is used for testing and debugging GE-PAC 30-2 micro-code programs. Micro-code programs, when implemented, become permanently wired instructions in a Read-Only-Memory (ROM). The purpose of the Simulator is to enable a micro-code program to be tested before it is wired into an ROM. The use of the Simulator minimizes the testing and debugging of a program in its wired form.

1.2 MICRO-CODE SOFTWARE

The Simulator is but on program in a family of GE-PAC 30 micro-code support programs, all of which run on any GE-PAC 30 Processor which has at least 8K bytes of core. A standard GE-PAC 30-2 Processor, therefore, can be used to develop and test special micro-code programs for use with any GE-PAC 30 Processor. Other support programs are:

1. The GE-PAC 30-2 Micro-Code Assembler, which accepts micro-code instructions in symbolic form and generates an ROM binary object tape. This Assembler is described in Publication Number 05-012A12.
2. The ROMWATS program, which accepts an ROM binary object tape and generates a wiring tape for the machine used to wire and test the physical ROM. This program is described in Publication Number 29-047.

1.3 OPERATION

The Simulator is interactive in nature, with many similarities to CLUB, the Hexadecimal Debug Program. It is operated and controlled from the teletypewriter keyboard. All numerical quantities are expressed in hexadecimal notation. Most directives are expressed with a single letter codes, many of which conform to the repertoire of CLUB commands.

The Simulator is organized around a set of data areas which are maintained in core memory. The five principal data areas are

1. Simulator registers and parameters
2. simulated Processor registers
3. simulated ROM
4. simulated DROM
5. simulated core memory

Refer to Figure 1-1.

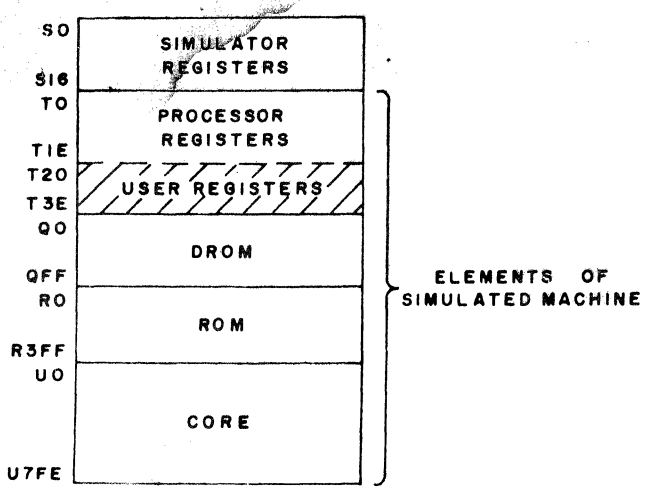


Figure 1-1. Primary Data Areas

The data areas are images of the various elements of the machine being simulated. Commands are provided for the display and manipulation of the various data areas. Other commands cause the Simulator to fetch micro-code instructions from the simulated ROM and perform the action specified. Features are provided for continuous or single-step execution.

1.4 RELATED DOCUMENTS

This document assumes a knowledge of the GE-PAC 30-2 Processor. For more information on the GE-PAC 30-2, refer to the Ge-PAC 30 Reference Manual, Publication Number 29-004, or the Micro-Instruction Reference Manual, Publication Number 29-032. For a description of CLUB, refer to the Hexadecimal Debug Program Description, Publication Number 03-002R03A12.

CHAPTER 2

ORGANIZATION OF THE SYSTEM

2.1 MAJOR COMPONENTS

The system is composed of three major components:

1. The interaction control
2. The micro-code execution control
3. The data areas

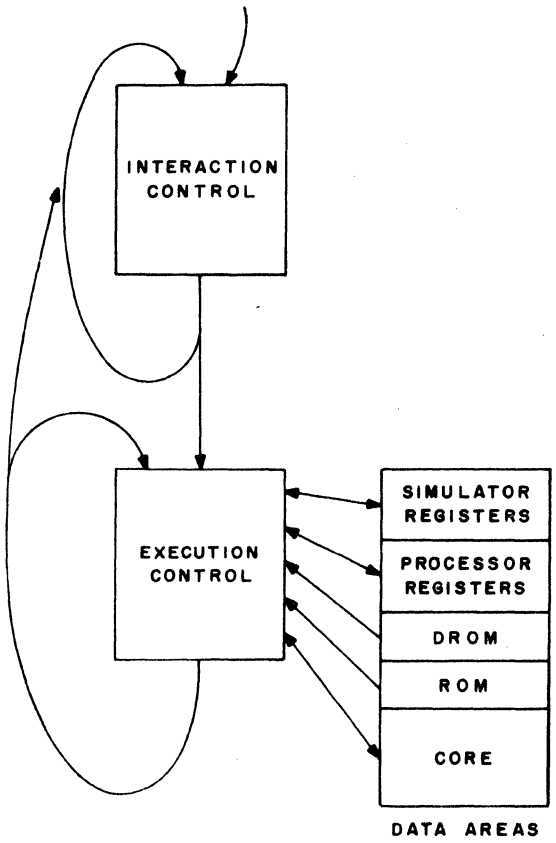


Figure 2-1. Major Components of GE-PAC 30-2 Simulator

These major components are represented in Figure 2-1. The interaction control portion accepts commands from the teletype, and performs the action specified. All of the interactive features of the Simulator reside in this section. Control remains in this section unless micro-code execution is explicitly started by a keyboard command.

The execution control portion contains the routines which simulate the GE-PAC 30-2 Processor. When given control, the execution routines operate on the data areas as though they were the elements of a machine. A number of methods are provided for terminating execution and returning control to the user at the keyboard. Specific methods are discussed in the next chapter. The details of the data areas, shown in Figure 2-2, are discussed in the following sections.

2.2 SIMULATOR REGISTERS

Twelve halfwords are used as special registers by the Simulator. The first four halfwords concern Display Panel parameters:

S0 = Display Panel switches. When a micro-code program inputs data from Device Number 1, the data is obtained from this halfword.

S2 = Display Byte Count/Status. The byte count is used for input/output transfers with Device Number 1. The Simulator adjusts this count as needed. The status byte is used whenever a micro-code program reads status from Device Number 1.

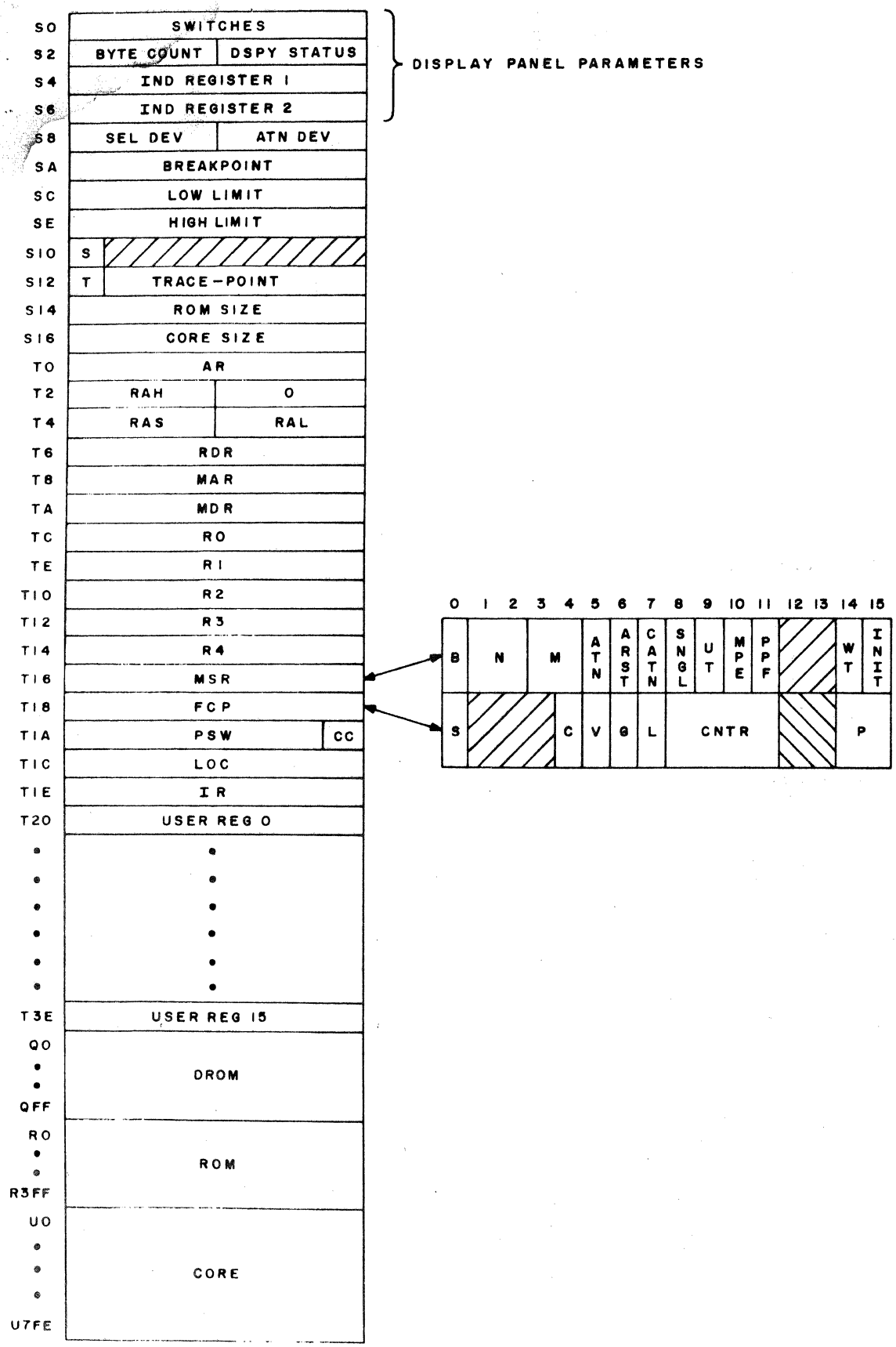


Figure 2-2. GE-PAC 30-2 Simulator Data Areas

S4 = Display Panel indicator registers. Data outputs to Device Number 1 are stored here.
S6

The remaining halfwords are as follows:

S8 = Selected Device/Attention Device. The Selected Device field is used by the Simulator to remember which device was last selected. The Attention Device is used by the Simulator to define which device was the source of an ATN signal. The Attention Device should be specified by the user.

SA = Breakpoint location. This halfword can contain an address of a micro-instruction in the ROM data area. During micro-code execution, the contents of the simulated RAS/RAL is compared to this breakpoint value. When a match is found, execution terminates, and a message is typed.

SC = Low Limit.

SE = High Limit. These values can be set by the operator. These limit values are used during print, output, and trace operations.

S10 Single Step Flag. Bit 0 of this halfword is set for single step execution. In this mode, execution of micro-instructions terminates at the completion of each instruction.

S12 = Trace Mode Flag. Bit 0 of this halfword is set for Trace mode during execution. In this mode, following the execution of each micro-instruction, the Simulator prints the data between the Low and High limits. If Bit 0 is not set, this halfword can contain an address of a micro-instruction in the ROM data area. During micro-code execution, the content of the simulated RAS/RAL is compared to this tracepoint value. When a match is found, the Simulator prints the data between the Low and High limits.

S14 = ROM Size. This halfword contains the number of micro-instructions to be stored in the simulated ROM. When the Simulator is loaded, this number is set at 400, which defines a 1K ROM.

S16 = Core Size. This halfword contains the number of bytes required for the simulated core memory. When the Simulator is loaded, this number is set at 800, which defines a 2K byte core memory.

2.3 PROCESSOR REGISTERS

Each halfword in this area contains the processor registers as shown in Figure 2-3. With two exceptions, each halfword contains one processor register. The exceptions are:

T2 = RAH/0. The left half of this halfword contains the high-order ROM Address Register. Note that this is the outer rank of the Address Register. The right half of this halfword is always zero.

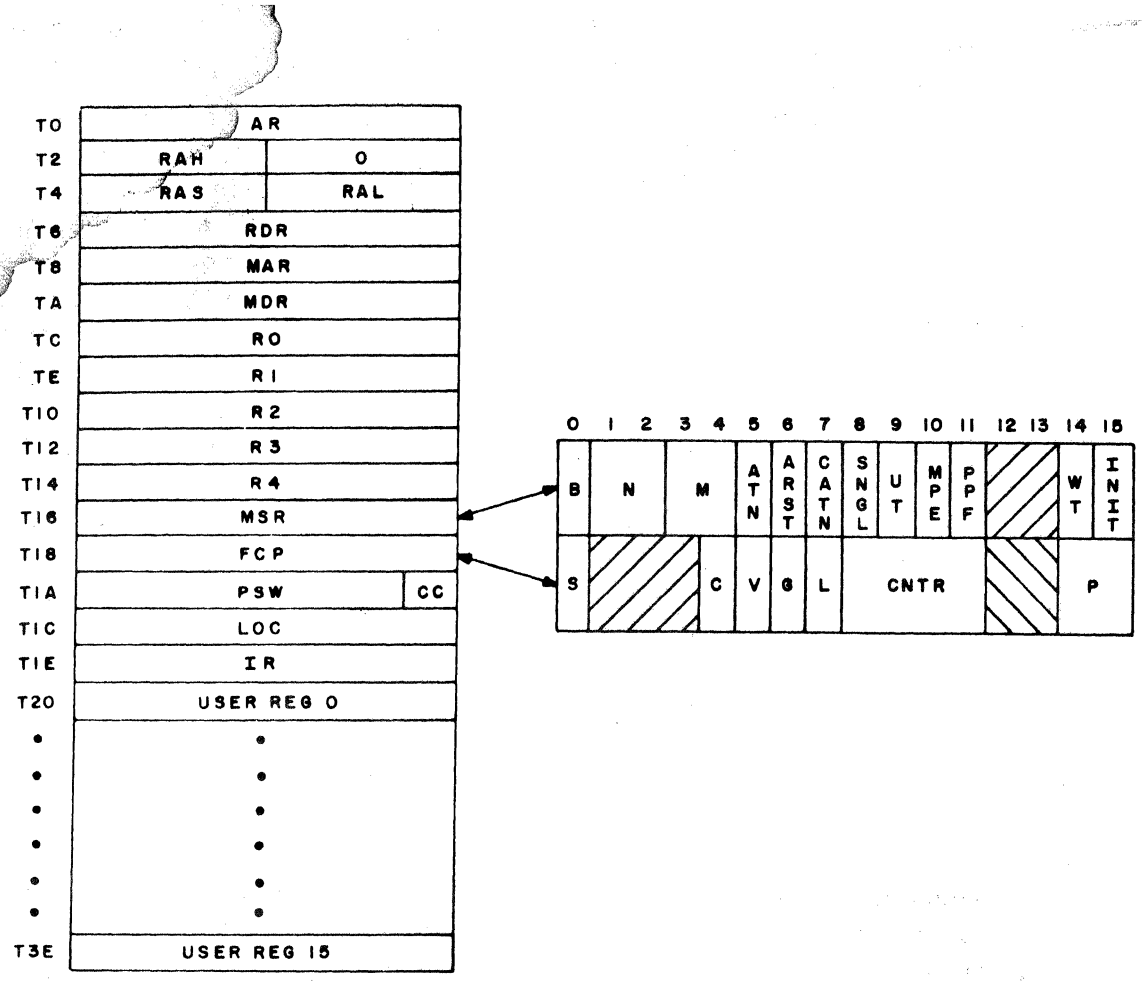


Figure 2-3. Simulated Processor Registers

T4 = RAS/RAL. The left half of this halfword contains the high-order ROM Address Register. Note that this is the inner rank of the ROM address page register. The right half of this halfword contains the ROM Address Lower Register (RAL).

T16 = Micro-Status Register (MSR) fields are as follows:

B = Bank Switch which affects Register Destination Addresses 0, 1, and 2. When B is set, addresses 0, 1, and 2 mean MR0, MR1, and MR2. When B is reset, addresses 0, 1, and 2 mean RAH, RAL, and YS.

M = Mode definition. This 2-bit field contains 1 for MPY mode, 2 for DIV mode, 3 for RPT mode and 0 when no counter modes are in affect.

N = A number associated with the counter modes. This field is adjusted by the Simulator and should not be changed by the user.

ATN = Device Attention.

ARST = Auto Restart.

CATN = Console Attention.

SNGL = Console Single Mode

UT = Utility flip-flop

MPE = Memory Parity Error

PPF = Primary Power Fail

WT = Wait Indicator

INIT = Initialize Switch. This bit is set by the I Ø keyboard command or a POW command. This switch is reset by the first 0000 ROM micro-instruction.

T18 = Flags, Counter, Phase (FCP). The fields contained in this register are:

S = Source Flag. When this switch is set, the source reference addresses E or F imply the YS field of IR. When reset, the E or F source addresses are taken from the YD field of IR.

CVGL = Flag Register (FLR)

CNTR = Four-Bit Counter Register.

P = Phase Pointer associated with Decode instructions.

2.4 USER'S REGISTERS

Data areas T 20 through T 3E contain the sixteen 16-bit general purpose User's Registers.

2.5 DROM

Each halfword in the DROM data area contains the Phase Two ROM entry point associated with a particular user's operation code. DROM data can be loaded from a binary object tape. This data can be displayed or changed using keyboard operations.

2.6 ROM

Each halfword in the ROM data area contains one micro-instruction. The micro-instructions can be loaded from an ROM binary object tape. These instructions can be displayed and changed using keyboard operations. These instructions are accessed during micro-code execution.

2.7 CORE MEMORY

Each halfword of the CORE data area contains one halfword of the core memory for the simulated machine. Whenever the microprogram accesses core memory, this data area is used.

CHAPTER 3

USER COMMANDS

3.1 GENERAL SYNTAX

The command format is very similar to CLUB. All numbers and addresses are expressed in hexadecimal. Commands from the keyboard are buffered and not processed until a delimiter character is typed. The principal delimiters are:

␣	blank (space bar)
CR	carriage return
LF	line feed
.	decimal point

In this discussion, the term cell refers to a halfword in memory. Whenever a cell is displayed, it becomes the open cell. The open cell is then available for modification.

The RUB OUT key can be used to correct typing mistakes. Whenever RUB OUT is depressed, the current command is ignored, and the open cell is closed.

Keyboard commands can be up to 8 characters long. If more than 8 characters are entered before a delimiter, or if the command is not proper, the Simulator will type a question mark (?) and not process the command. After an error message, the open cell is closed.

3.2 CELL EXAMINATION

The space bar (␣) is the delimiter used for cell examination. The general command is of the form

address ␣

where the address specifies which cell to open. The address is expressed in hexadecimal with an optional prefix letter which identifies the data area of interest. The prefix letters are:

Q	DROM
R	ROM
S	Simulator Registers
T	Processor Registers
U	(User) Core Memory

Sample commands are:

Q2B␣	open cell 2B in the DROM
R27␣	open cell 27 in the ROM
S16␣	open Simulator register 16 (Core Size)
T4␣	open cell 4 of the Processor registers (RAS/RAL)
U678␣	open halfword 678 of simulated core
345␣	open halfword 345 of actual core

When a cell is opened, the system responds by typing the address of the opened cell, and the contents of the open cell. For example:

U678␣

U0678 C820

The line feed (LF) key can be used to open the next sequential cell in memory. For example:

```
T4Ø
TØØØ4      Ø243      LF
TØØØ6      53FF
```

Similarly, the carriage return (CR) key can be used to open the previous cell in memory. For example

```
R36Ø
RØØØ36     2345      CR
RØØ35      5432
```

Note that the ROM and DROM data areas are addressed by word; that is a 1K ROM has addresses R0, R1, R2, R3, ..., R3FF. The 256 word DROM area has addresses Q0, Q1, Q2, ..., QFF. Other data areas in memory are addressed by byte; that is, a 2K Core has addresses U0, U2, U4, ..., U7FE.

When LF or CR commands are used, any characters which precede the LF or CR are ignored. The LF or CR operations should not be used to cross boundaries from one data area to another. If this is attempted, the data displayed will be correct, but the address indicated will not be correct.

3.3 CELL CHANGE

The decimal point (.) is the delimiter used to change the content of a cell. The general command is of the form

data.

where the specified data is to be deposited in the open cell. A cell must have been previously opened to use this command. The data must be expressed in hexadecimal form, composed of the characters 0 - 9 and A - F. Leading zeros are not required. When this command is used, the Simulator responds with the address of the open cell and the new content of that cell. For example:

```
T8Ø
TØØØ8      Ø123      7777.
TØØØ8      7777      LF
TØØØA      4567      89.
TØØØA      ØØ89
```

3.4 LETTER COMMANDS

Other commands to the Simulator are expressed with one or two characters, followed by a space. These commands are listed on Table 3-1 in alphabetical order.

TABLE 3-1. LETTER COMMANDS

Command	Meaning	Explanation
GØ	GO	Begin micro-code execution with the micro-instruction specified by RAS/RAL.
HØ	Set High Limit	The address of the open cell is defined as the high limit. This limit is used in print, output, and trace operations. A cell must be open when the H command is used. The high limit is recorded in Simulator cell SE.

TABLE 3-1. LETTER COMMANDS
(CONTINUED)

Command	Meaning	Explanation
I⍉	Initialize	The simulated processor is initialized as follows: 1. The RAH, RAS, RAL, CNTR, FLR, and RD registers are cleared. The Phase Pointer is set to X'3'. 2. The INIT indicator is set, but all other bits of the Micro-Status Register are cleared. (See Figure 2-2.)
IQ⍉	Input to DROM	The Simulator contains a micro-code loader which reads a DROM tape from the binary input device and loads the data into the simulated DROM. When this command is used, the Simulator halts to allow a tape to be placed in the tape reader. When the EXECUTE button is pushed, the loader reads the DROM binary tape. Refer to Chapter 5 for details.
IR⍉	Input to ROM	The Simulator contains a micro-code loader which reads an ROM object tape from the binary input device and loads the data into the simulated ROM. When this command is used, the Simulator halts to allow a tape to be placed in the reader. When the EXECUTE button is pushed, the loader reads the ROM binary tape. Refer to Chapter 5 for details.
IU⍉	Input to Core	The Simulator contains an 8-bit loader which reads an 8-bit binary tape into memory as specified by the Low and High Limits. When this command is used, the Simulator halts to allow a tape to be placed in the reader. When EXECUTE is pushed, the tape is read into memory. Refer to Chapter 5 for details.
J⍉	Set Tracepoint	The address of the open cell is used to define a tracepoint. The open cell must lie within the ROM data area. The address of the ROM tracepoint is stored in Simulator cell S12. During execution, the Simulator prints between Low and High Limits whenever RAS/RAL matches the tracepoint, or whenever the Trace mode is set.

TABLE 3-1. LETTER COMMANDS
(CONTINUED)

Command	Explanation	Meaning
K/	Kill Trace	The Trace mode is reset, or the tracepoint is cleared. The Simulator cell S12 will contain 7FFF after the Kill operation.
L/	Set Low Limit	The address of the open cell is defined as the low limit. This limit is used in print, output, and Trace operations. A cell must be open when the L command is used. The low limit is recorded in Simulator cell SC.
M1/ M2/ M3/ M4/ M5/ M6/ M7/ M8/	Set VARI FIX Mode Set HALT FIX Mode Set RUN Mode Set ADRS Mode Set MEMR Mode Set MEMW Mode Set HALT FLT Mode Set VARI FLT Mode	These commands set the Display Panel Status Byte in Simulator cell S2 to reflect the specified mode. The right-most four bits of the status byte are unchanged.
OQ/ OR/ OU/	Output DROM Output ROM Output Core	The contents of the DROM, ROM, or Core as defined by the low and high limits are punched on to paper tape. When this command is used, the Simulator halts to allow the punch to be prepared. When EXECUTE is pushed, the specified area of memory is punched with both leader and trailer. Details of device selection and tape format are discussed in Chapter 5.
P/	Print	The contents of the cells defined by the low and high limits (inclusive) are printed on the teletype. The print format consists of one address and 8 values per line.
Q/	Query Core Location	This command causes the Simulator to type out the address in actual core memory at which the simulated core memory begins. This address may be needed by loader programs to load binary object tapes into simulated core. The address is typed in hexadecimal. If the ROM Size specified in Simulator cell S14 is changed, the location of simulated core in actual core also changes.
V/	Set Single Mode	In single mode, execution of micro-instructions is performed one-at-a-time. That is, after each micro-operation is executed, control returns to the user at the teletype. Bit 0 of Simulator cell S10 is set during single mode.

TABLE 3-1. [illegible]
(CONTINUED)

Command	Meaning	Explanation
W/	Wipe Out Single	This command resets single mode. Simulator cell S10 contains zero after this operation.
X/	Set Breakpoint	The address of the open cell is used to define a breakpoint. The open cell must lie within the ROM data area. The address of the ROM breakpoint is stored in Simulator cell SA. During execution, RAS/RAL is compared to cell SA following each micro-instruction. When a match is detected, execution terminates and a message is typed.
Y/	Set Trace Mode	In the Trace mode, the Simulator prints all cells between the low and high limits after each micro-instruction is executed. Bit 0 of Simulator cell S12 is set during Trace mode.
Z/	Zap the Breakpoint	This command clears any existing ROM breakpoint. Simulator cell SA contains FFFF after this operation.

3.5 SPECIAL CONTROLS

The Micro-Status Register (MSR) represents various signals and indicators associated with the Processor. Some of the MSR bits must be set by the user to simulate the occurrence of the corresponding signals. For operator convenience, the commands listed on Table 3-2 are provided.

No delimiter is required for these commands, and any characters which precede the special command characters (!, *, ", #, %) are ignored. The Simulator responds with a carriage return and line advance to acknowledge these special controls.

TABLE 3-2. SPECIAL CONTROLS

Command	Meaning	Explanation
%	Set ARST	This command sets bit 6 of the Micro-Status Register (MSR, T16) to simulate the Automatic Restart function.
!	Set ATN	This command sets bit 5 of the MSR to simulate the occurrence of a device interrupt.

Table 3-2. SPECIAL CONTROLS
(Continued)

Command	Meaning	Explanation
*	Set CATN	This command sets bit 7 of the MSR to simulate the display panel EXECUTE button. If the display status byte in Simulator cell S2 contains 4X, which means VARI mode, then SNGL in bit 8 of the MSR is also set. If the display status byte does not equal 4X, then bit 8 of the MSR is reset.
"	Set MPF	This command sets bit 10 of the MSR to simulate the occurrence of memory parity failure.
#	Set PPF	This command sets bit 11 of the MSR to simulate the occurrence of primary power failure.

CHAPTER 4

MICRO-INSTRUCTIONS

4.1 EXECUTION CYCLE

Execution of micro-instructions is started with the GO (G∅) command at the keyboard. The execution cycle for each micro-instruction is illustrated in Figure 4-1. The micro-op to be executed next is indicated by RAS/RAL. Before fetching the micro-op, the contents of RAS/RAL is compared to the ROM Size specified by Simulator cell S14. If the RAS/RAL exceeds the ROM Size, the Simulator types

ROM ADRS ERR

and terminates execution. If the RAS/RAL is within limits, the micro-op is fetched from the simulated ROM, and stored in RD. At this time, RAL is incremented by 1. Note that there is no carry to RAS. This results in a page wrap-around effect. The specific instruction fetched is then executed. When the instruction is complete, the tests indicated in Figure 4-1 are performed. A print between limits results if either the Trace mode is in effect, or a tracepoint has been encountered. Control returns to the keyboard is either single mode prevails, Switch 15 is depressed, or a breakpoint has been encountered. In the case of Switch 15 being set, or the breakpoint being found, the message

BREAKPOINT

is typed. For single mode, the Simulator simply outputs a carriage return and line advance to indicate that control has returned to the keyboard.

The execution of specific micro-instructions is discussed in the following sections. Only special features or exceptions are mentioned; the Simulated execution in most cases is identical to that of a GE-PAC 30-2 Processor.

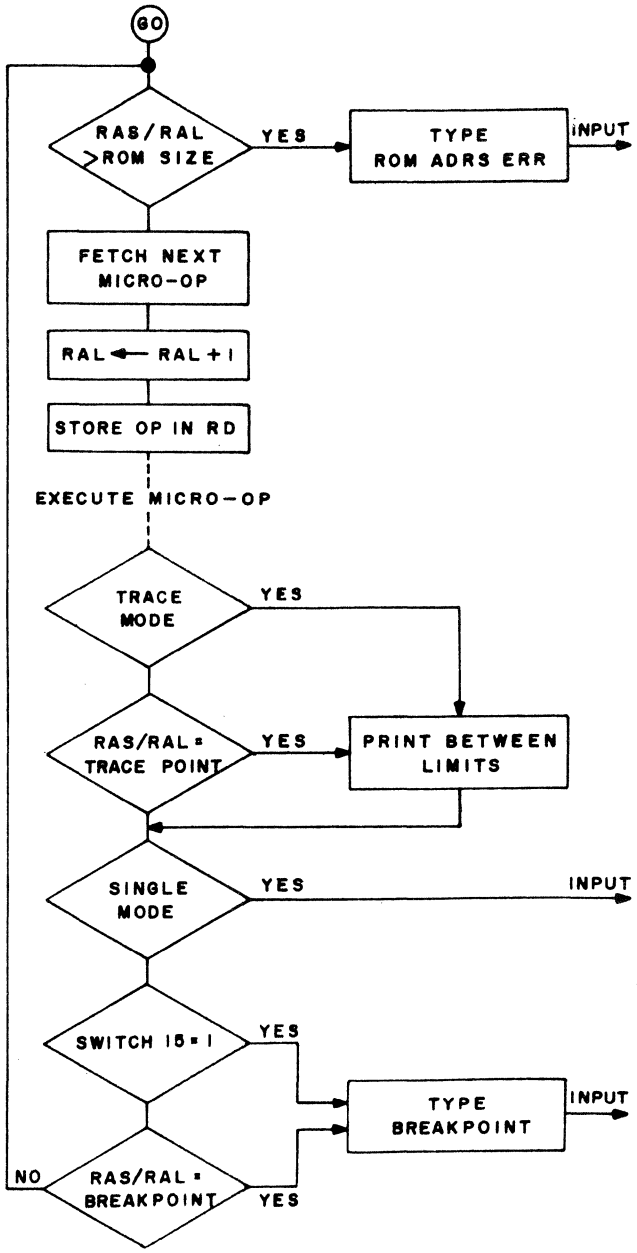


Figure 4-1. Execution Cycle

4.2 MEMORY OPERATIONS

All memory operations with simulated core memory act exactly the same as with an actual memory. These operations are listed on Table 4-1.

Memory Write (MW) and Priviledged Write (PW) are not separate to the Simulator as memory protect hardware is not simulated.

If the core address exceeds the Core Size parameter in Simulator cell S16 during Read operations, zero data is fetched. If the address exceeds the Core Size during Write operations, the information is deposited into the bit bucket.

4.3 REGISTER ADDRESSING

The Simulator includes the same source and destination restrictions as an actual machine. These restrictions are:

1. The Bank Switch (Bit 0 in MSR) affects the destination address only, not the source address. The Bank Switch must be set to store into MR0, MR1, or MR2 and reset to store into RAH, RAL, or YS. If the source address 0, 1, or 2 is used, the MR0, MR1, or MR2 is implied independent of the Bank Switch.

2. If the AR or CTR registers are addressed as Source registers, zero data will be fetched.

3. IO cannot be used as a source and destination of one instruction. If attempted, the Simulator types

   IO ERROR

   and the instruction is not executed.

4. When the RAL is loaded with data, the RAH is copied into the RAS.

4.4 FLAG SETTINGS

The C, V, G, L flags reside in bits 4 through 7 of the FCP register. These flags get set as a result of various micro-instructions, or by explicitly loading the FLR register. In general, if the FLR is not the explicit destination, the flags remain unchanged except as follows:

1. Load adjusts C flag if Carry Out is specified in the E field.

2. Add, Subtract adjust the C flag if Carry Out is specified in the E field.

3. Add, Subtract adjust the V, G, L flags if Set Flags is specified in the E field.

4. AND, OR, Exclusive OR adjust the G, L flags if Set Flags is specified in the E field.

5. Test Adjusts the G, L flags.

TABLE 4-1. MEMORY OPERATIONS

Operation	before		after	
	MD	Memory	MD	Memory
Full Read	A	B	B	B
Full Write	A	B	A	A
Priviledged Write	A	B	A	A

If the FLR is the explicit destination, the flags get set by either the destination data or the setting conditions as defined previously in items 1-4.

When executing A, S, N, O, X instructions with Set Flags specified in the E field, the G and L flags are adjusted as follows:

$$G = \overline{So}.(S_1+S_2+\ldots+S_{15}+G_p+L_p)$$

$$L = So$$

where S_n bit n of resulting data

G_p = previous G flag

L_p = previous L flag

4.5 INPUT-OUTPUT

All input-output operations are initiated when a Load instruction is done with IO as source or destination. The first five cells in the Simulator data area are:

S0	SWH	SWL	DISPLAY PANEL SWITCHES
S2	N	STATUS	N = BYTE COUNT
S4	B3	B2	INDICATOR REGISTER 1
S6	B1	B0	INDICATOR REGISTER 2
S8	SELDEV	ATNDEV	

The result of an IO load is summarized in Table 4-2.

The Simulator differs from a real processor in that it does not rely on an external device to return sync. Note that there is no such thing as a time-out with the Simulator.

This technique of combining a sequence of micro-instructions into one user-instruction makes I/O possible with certain restrictions. Namely, the timing will not be realistic. The simulated I/O operations will run much slower than normal. No test is made in the Simulator for SELDEV = 0. A zero or improper device number or an unavailable device should not be referenced.

TABLE 4-2. IO SUMMARY

IO is the Source		
E field	Operation	Explanation
xx00	none	IO ERROR message.
xx01	Acknowledge	Copy ATNDEV to Destination, and clear ATN (bit 5 of MSR).
xx10	Data Request	If SELDEV ≠1, execute a Read Data instruction with device number from SELDEV and put data into the Destination Register specified. If SELDEV = 1, copy SWH into Destination if N is odd; copy SWL into SDR if N is even. Increment N, and reset if N = 4.
xx11	Status Request	If SELDEV ≠ 1, execute a Sense Status instruction with device number from SELDEV and put the status into the Destination Register. If SELDEV = 1, copy STATUS into the Destination.

TABLE 4-2. IO SUMMARY
(CONTINUED)

IO is the Destination		
E field	Operation	Explanation
xx00	none	IO ERROR message.
xx01	Address	Copy Source Register into SELDEV. If Source byte = 1, reset CATN and clear N, the byte count.
XX10	Data Available	If SELDEV ≠ 1, execute a Write Data instruction with the device number from SELDEV and data from the Source Register specified. If SELDEV = 1, copy the Source byte into B_n, incremented N and reset if N = 4.
XX11	Command	Execute an Output Command instruction with Device Number from SELDEV and data from the Source Register specified. If SELDEV = 1, the command is disregarded.

4.6 OTHER MICRO-INSTRUCTIONS

If a Command instruction is executed with bit 15 1, which specifies POW, the Simulator types

POWER DOWN

and terminates execution.

4.6.1 Counter Modes

The Counter modes (MPY, DIV, RPT) affect execution in the following ways.

1. The Command operation sets the M field of the MSR according to the mode specified, and clears N, the counter phase count. The M field is set to 1 for MPY, 2 for DIV, and 3 for RPT.

2. At the end of each instruction, the M field is tested. If a counter mode is specified by a non-zero M, the action shown in Figure 4-2 occurs.

3. During the micro-op fetch from the simulated ROM, the incrementing of the RAL is suppressed if MPY Mode with N = 1 or 3, DIV Mode with N = 1 or 3, or RPT Mode with N = 1.

4. During data fetch from the source registers, any reference to YD implies YDP1 if MPY Mode with N = 2, or DIV Mode with N = 1.

5. During data store into the destination register, any reference to YD implies YDP1 if MPY Mode with N = 2, or DIV Mode with N = 1. If DIV Mode with N = 3, the store is suppressed if Carry in FLG register is zero.

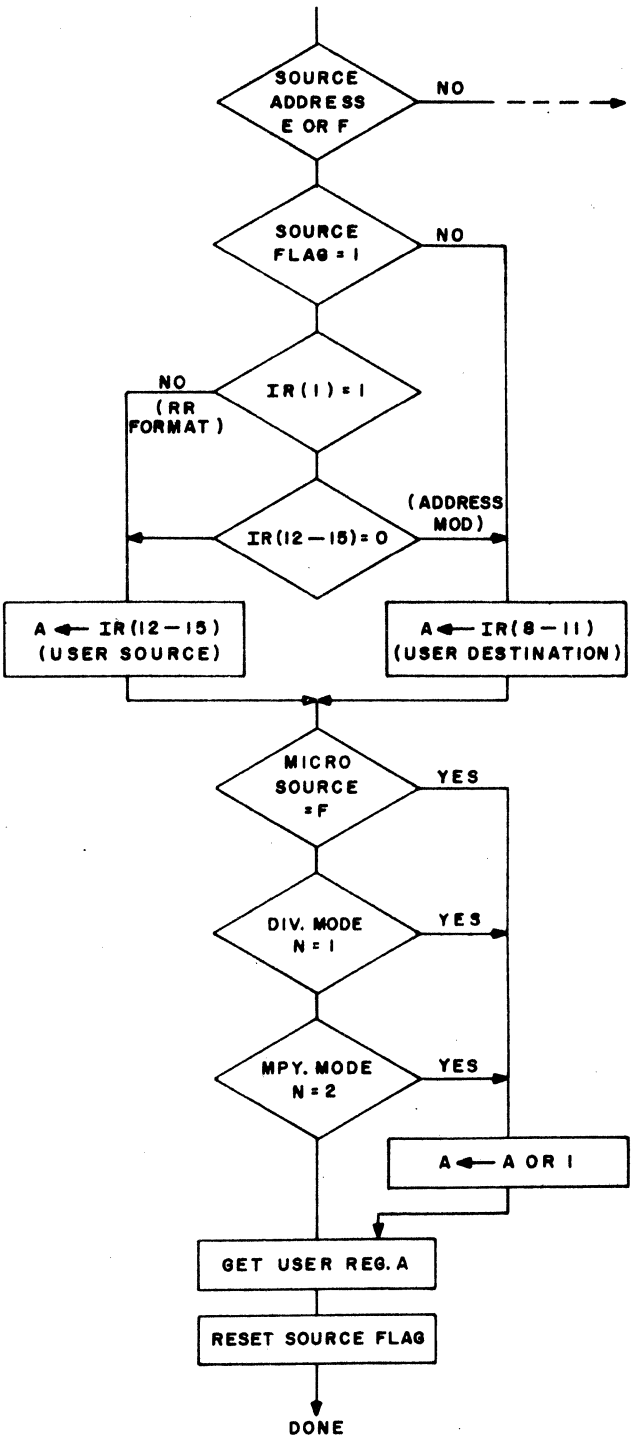
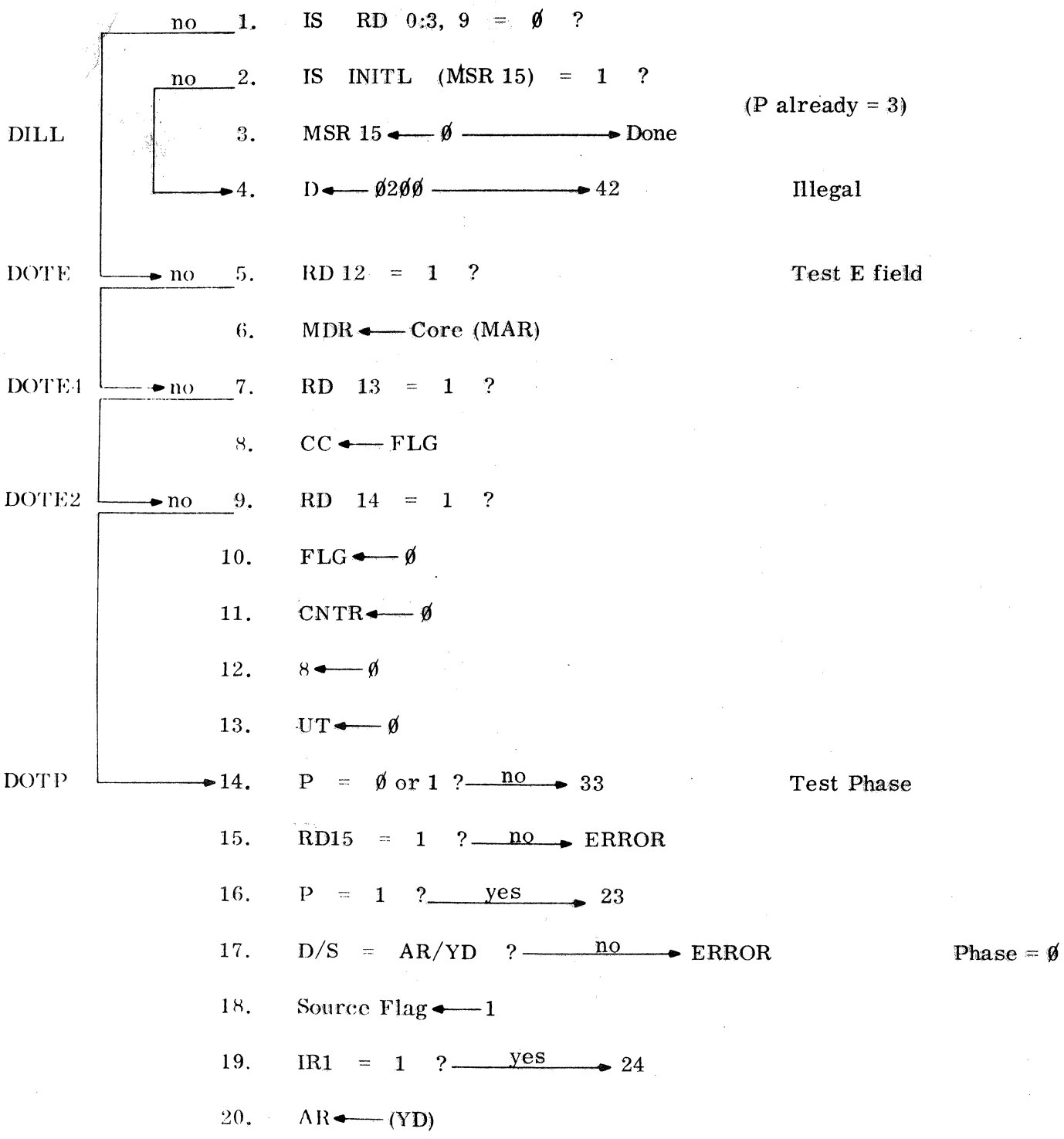
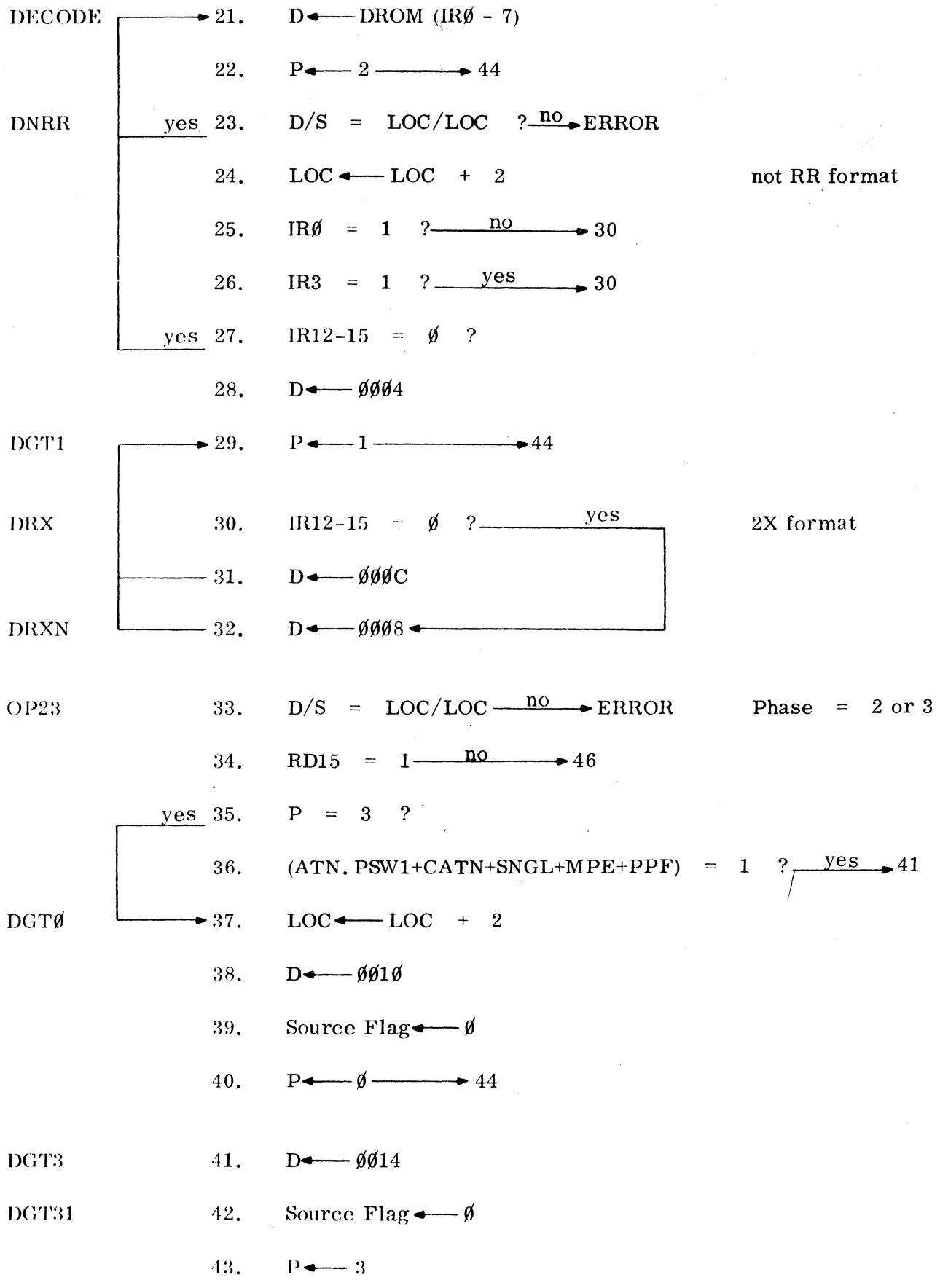


Figure 4-2. Post Execution Counter Mode Handler

4.6.2 Decode Instruction



4.6.2 Decode Instruction (Continued)



4.6.2 Decode Instruction (Continued)

DOJ 44. RAH ← Ø

45. RAS/RAL ← D → DONE

DZ3NE 46. LOC ← LOC + 2 → DONE

4.6.3 Test Instruction

The Test micro-op tests bits in the Micro-Status Register. The ATN bit is tested only if specified by the Test micro-op and bit 4 of the PSW is set.

4.6.4 User Destination/Source

The use of addresses E or F in the Source field of a micro-op can refer to either the user's Destination or Source register. The logic associated with this register selection is shown in Figure 4-3

Recall that with the Decode instruction, the Source Flag is set - when leaving Phase 0, and reset on entry to Phase 0 or Phase 3.

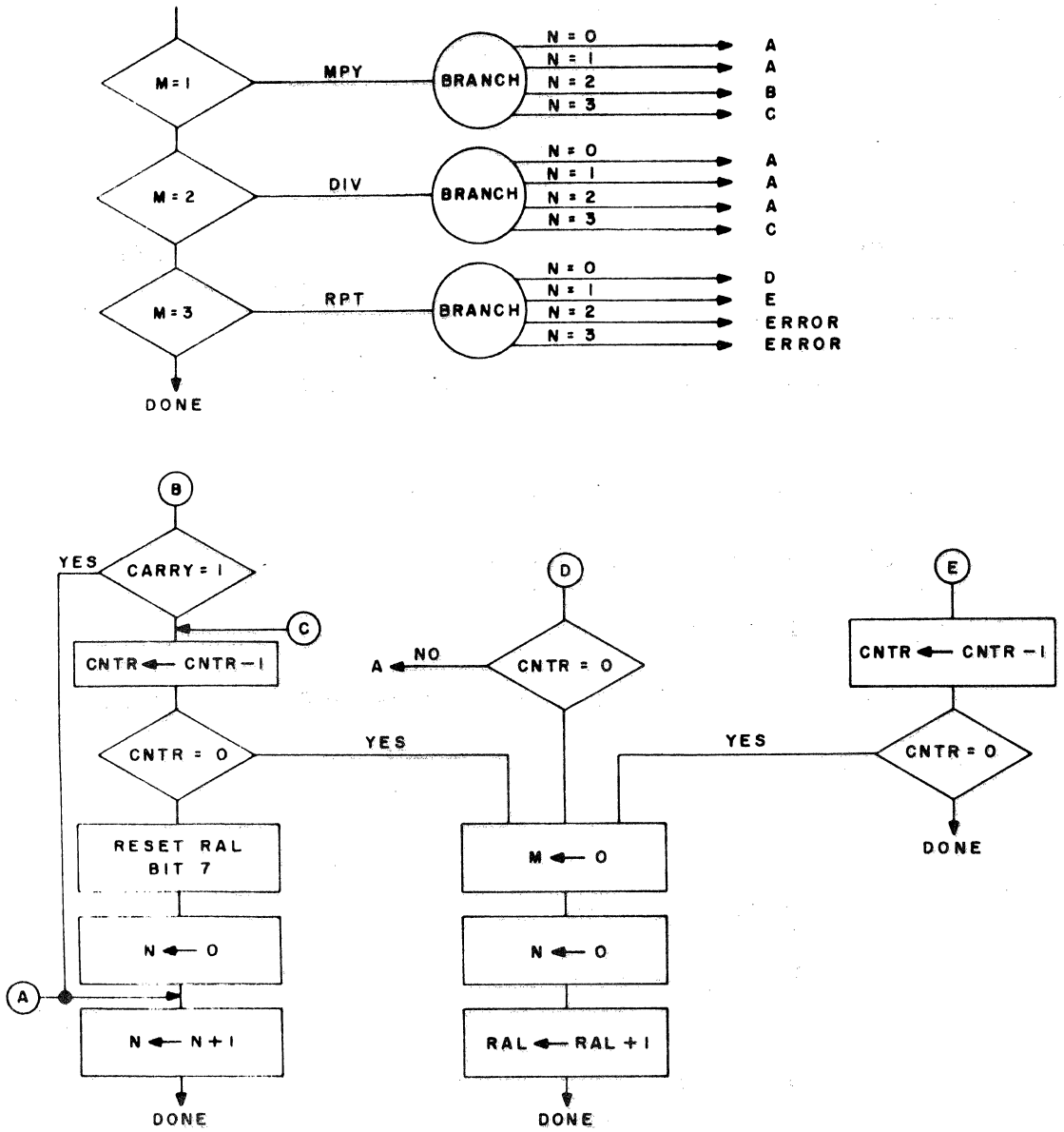


Figure 4-3. Source/Destination Logic

CHAPTER 5

USE OF THE SIMULATOR

5.1 CONFIGURATION

The GE-PAC 30-2 Simulator program, 05-014, runs on any GE-PAC 30 Processor which has 8K bytes or more of core memory. The Simulator assumes that a teletypewriter is interfaced to the Processor as Device Number 2; the Display Panel is referenced as Device Number 1.

5.2 LOADING THE SIMULATOR

The Simulator Tape, 05-014R02M09, is an absolute tape using the normal binary object tape format. The Simulator can be loaded by either the 8K Absolute Loader or the General Loader. Refer to the Loader Descriptions, Publication Number 06-025A12, for a detailed explanation of the loading procedures.

5.3 STARTING PROCEDURES

The starting location is 100. The specific procedures to start are:

1. Set the Display Panel switches to X'0100'.
2. Select ADRS mode and depress EXECUTE.
3. Select RUN mode and depress EXECUTE.

When started, the Simulator types a carriage return and line advance on the teletypwriter which makes an audible click. This sound means the Simulator is ready for use. When started at 100, the current state of the data areas is unchanged. This means that immediately after loading, the content of the simulated ROM, DROM, and Core areas is unpredictable. Once the ROM, DROM, and Core data areas have been set up, however, their state is not affected by restarting at 100.

5.4 MEMORY ALLOCATION

The Simulator itself requires a little less that 5K bytes of memory, starting at 100. The remaining core memory can be allocated such that

$$2*(\text{ROM Size}) + (\text{Core Size}) \leq M$$

where ROM Size is defined in Simulator cell S14, Core Size is defined in Simulator cell S16, and M is the remaining memory available. In an 8K memory, M is about 3K, and the equation becomes:

$$2*(\text{ROM Size}) + (\text{Core Size}) \leq 3K$$

After loading, ROM Size is 1K and Core Size is 2K. If these numbers are satisfactory for the job at hand, the parameters do not have to be adjusted. If a different allocation is perferred, the ROM Size parameter should be adjusted prior to loading the Simulated Core area with data. Once the Core area has been loaded, the ROM Size parameter is Simulator cell S14 should not be changed. The Core Size parameter affects only simulated memory operations, and can be changed anytime. Memory operations are discussed in Section 4.2

5.5 LOADING THE ROM

The keyboard command IRØ is used to input data to the ROM data area. When this command as used, the Simulator will read a ROM binary object tape from the Binary Input Device as defined in the Device Definition Table from X'78' to X'7F'. Specifically, the halfword at X'78' is interpreted as follows:

78	Dev No.	Command

This halfword for various devices is shown below.

TTY	0294
HSPTR	0399

The ROM binary tape format must conform to that generated by the GE-PAC 30-2 Micro-Code Assembler, as defined in Publication Number 05-012A12.

Basically, the tape must be organized in blocks, where each block begins with an address. The address is to be the ROM address times 2. Each block must be followed by blank tape.

When the IRØ command is given, the Simulator will halt to allow the tape to be placed in the tape reader. When the EXECUTE button is depressed, the tape is read. Leading blank tape is skipped. Data is read and stored in the ROM data area until blank tape is encountered. If the data on the tape was not an even multiple of 4 characters, the Simulator types a question mark (?) to indicate an improper tape format. If the tape format was proper, the Simulator types the ROM address and content of the last ROM instruction loaded from the tape.

5.6 LOADING THE DROM

The Keyboard command, IQØ, is used to input data to the DROM data area. The Simulator will read a DROM binary object tape from the Binary Input Device. The tape read must be one continuous block. The tape format should be identical to that of an ROM tape.

When the IQØ command is given, the Simulator will halt to allow the tape to be placed in the tape reader. When the EXECUTE button is depressed, the tape is read. Leading blank tape is skipped. At completion of the Load, the Simulator types the DROM address and content of the last word loaded from the tape.

5.7 LOADING THE CORE

The Simulator contains an 8-bit loader which can be used to load the simulated core memory. The loader reads 8-bit tapes from the Binary Input Device as defined in the Device Definition Table from X'78' to X'7F'. No address information is required on the tape. The tape is loaded into the area of memory as indicated by the Low and High Limits. When the EXECUTE button is depressed, the tape is read. Leading blank tape is skipped. Loading proceeds until the High Limit is reached.

5.8 EXECUTION

Once an ROM program has been loaded, the simulated machine should be prepared for execution. The Initialize (IØ) command is available for this purpose. Other items which may require set-up are:

S2	Display Panel Status
S0	Display Panel Switches
S8	ATN Device Definition

Figure 2-2 should be kept in view during the simulation process to assist in the identification of the data areas.

The command G⌀ causes ROM execution to begin at the micro-instruction indicated by RAS/RAL. Note that if RAS is changed to some value from the keyboard, it is in general, wise to give RAH the same value.

The techniques available to control and monitor the execution process are summarized below.

Technique	Set Command	Clear Command
breakpoint	Rnnnn⌀----X⌀	Z⌀
tracepoint	Rnnnn⌀----J⌀	K⌀
trace mode	Y⌀	K⌀
single mode	V⌀	W⌀

In addition, Switch 15 on the (actual) Display Panel can be used to interrupt execution and return control to the keyboard. When Switch 15 is depressed, the Simulator types

BREAKPOINT

and terminates execution.

5.9 ROM OUTPUT

After a program has been tested, it may be desirable to punch a new ROM binary tape. The keyboard command OR⌀ is for this purpose. When this command is used, the Simulator punches an ROM binary tape to the Binary Output Device as defined in the Device Definition Table from X'78' to X'7F'. Specifically, the halfword at X'7A' is interpreted as follows:

7A	Dev No.	Command

This halfword for various devices is shown below.

TTY	0298
HSPTP	0392

The information to be punched is defined by the Low and High limits inclusive. The tape is punched in a proper ROM binary format. Note that if this tape is to be used as ROMWATS input, unused locations in the ROM should contain zero. This procedure to dump an ROM tape, therefore is as follows:

1. Set Low Limit with L⌀.
2. Set High Limit with H⌀.
3. Clear unused locations in the ROM block to be punched.
4. Make sure cell X'7A' defines the proper device.
5. Give OR⌀ command. The Simulator will halt to allow device preparation. Turn tape punch on.
6. Depress EXECUTE to start punching. Leader and trailer are punched before and after the data. The Simulator will halt after punching is complete.
7. Depress EXECUTE to regain control at the keyboard.

5.10 OTHER OUTPUT OPERATIONS

The contents of the DROM can be punched on tape with the OQ½ command. The procedure for this operation is exactly the same as for ROM outputs.

The contents of the Core can be punched on tape with the OU½ command. The procedure for this operation is similar to the above, except that an 8-bit tape will be punched. It is not necessary in this case to clear any unused core locations. Due to certain punch characteristics, it will not be possible to use this operation with some ASR 33 Teletypes.

Following all output operations, it is possible to verify that the tape was punched properly. To verify a tape, the proper input operation should be performed with Switch 15 on the Display Panel depressed. With this switch set, the IQ, IR, or IU commands will compare rather than load. If any errors are detected, a COMPARE FAIL message is typed.

APPENDIX 1
COMMAND SUMMARY

nnnn ␣	Display halfword of actual core
Qnnnn ␣	Display simulated DROM location
Rnnnn ␣	Display simulated ROM location
Snnnn ␣	Display Simulator Register
Tnnnn ␣	Display two simulated Mod 4 registers
Unnnn ␣	Display halfword of simulated core
nnnn.	deposit nnnn into open cell
nnnn LF	ignore nnnn, display next cell
nnnn CR	ignore nnnn, display previous cell
nnnn RO	ignore nnnn, close the open cell

Note that n = 0, 1, 2, ... , 9, A, B, C, D, E, F
␣ = blank (space bar)
LF = line feed
CR = carriage return
RO = rub out

G ␣	Go, start micro-code execution
H ␣	High; set high limit
I ␣	Simulate Processor power-up
IQ ␣	Input DROM binary tape
IR ␣	Input ROM binary tape
IU ␣	Input 8-bit Core tape
J ␣	Set tracepoint
K ␣	Kill tracepoint or trace mode
L ␣	Low; set low limit
M1 ␣	Set VARI FIX mode
M2 ␣	Set HALT FIX mode
M3 ␣	Set RUN mode
M4 ␣	Set ADRS mode
M5 ␣	Set MEMR mode
M6 ␣	Set MEMW mode
M7 ␣	Set HALT FLT mode
M8 ␣	Set VARI FLT mode
OQ ␣	Output DROM binary tape
OR ␣	Output ROM binary tape
OU ␣	Output 8-bit Core tape
P ␣	Print between limits
Q ␣	Query Core location
V ␣	Set Single mode
W ␣	Wipe out Single mode
X ␣	Set breakpoint
Y ␣	Set trace mode
Z ␣	Zap breakpoint

!	Set ATN
*	Set CATN
"	Set MPF
#	Set PPF
%	Set ARST

APPENDIX 2
ERROR MESSAGES

BREAKPOINT	An ROM breakpoint was encountered during simulated ROM execution. The instruction at the breakpoint was not executed. This message also occurs when Switch 15 on the Display Panel is depressed to interrupt ROM execution.
ROM ADRS ERR	During execution, this message occurs if the contents of RAS/RAL are not less than the ROM Size defined in Simulator cell S14.
POWER DOWN	A Command micro-instruction set the POW bit in the Micro-Status Register. In acutal operation, this action would shut down the Processor power.
IR RACE CONDITION	The command following an IR load refers to a user register as a data source.
IO ERROR	The IO code was used as Source and Destination, or the operation was not a Load, or the E field specified XX00.
MODE ADRS ERR	A RPT or MPY or DIV Command did not fall on an odd address.
NO SUCH MODE	The N field in the Micro-Status Register contains an improper value. The occurrence of this message implies a problem in the Simulator itself.
MODE CONFLICT	A command specifying a Counter mode occurred during a Counter mode.
DO IT ERR	A Do micro-op occurred during phase 0 or 1 with RD15 not set, or during phase 0 and the Destination/Source field did not contain AR/YD, or during phase 1, 2, or 3 and the Destination/Source field did not contain LOC/LOC.
COMPARE FAIL	An ROM Input (IR⅛), DROM Input (IQ⅛), or Core Input (IU⅛) operation with Switch 15 depressed detected a mismatch between the tape and memory.

APPENDIX 3
DISPLAY PANEL STATUS

STATUS BYTE

0	1	2	3	4	5	6	7
MODE				REG			

Mode Control Switch

VARI FLT	0	1	1	0
HALT FLT	1	1	1	0
VARI FIX	0	1	0	0
HALT FIX	1	1	0	0
RUN	1	0	0	0
ADRS	0	0	1	1
MEMR	0	0	1	0
MEMW	0	0	0	1

Register Select Switch

0	0	0	0	OFF
0	0	0	1	Register Display
0	0	1	0	INST
0	1	0	0	PSW
1	0	0	0	R0/1
1	0	0	1	R2/3
1	0	1	0	R4/5
1	0	1	1	R6/7
1	1	0	0	R8/9
1	1	0	1	R10/11
1	1	1	0	R12/13
1	1	1	1	R14/15

APPENDIX 4
MICRO-INSTRUCTION SUMMARY

OP-CODE	INSTRUCTION
0 0 0 0	DECODE
0 0 0 1	BRANCH
0 0 1 0	TEST
0 0 1 1	COMMAND
0 1 0 0	LOAD
0 1 0 1	LOAD IMMEDIATE
0 1 1 0	OR
0 1 1 1	OR IMMEDIATE
1 0 0 0	AND
1 0 0 1	AND IMMEDIATE
1 0 1 0	EXCLUSIVE OR
1 0 1 1	EXCLUSIVE OR IMMEDIATE
1 1 0 0	ADD
1 1 0 1	ADD IMMEDIATE
1 1 1 0	SUBTRACT
1 1 1 1	SUBTRACT IMMEDIATE

Commands

Bits Set	Definition
5	Multiply
4	Divide
4, 5	Repeat
7	Mem. Read
6	Mem. Write
6, 7	Priv. Write
9	Reset Bank*
8	Set Bank*
8, 9	Trigger Bank*
11	Reset Utility*
10	Set Utility*
10, 11	Trigger Utility
12	Clear Mem. Parity*
13	Set Wait Alarm*
14	Reset Wait Alarm*
15	Power Down

* — flip-flops

E FIELD	DEFINITION
For A, S, X, N, O:	
1 x x x	No AR to ALU
x 1 x x	Set Flags
x x 1 x	Carry Into ALU
x x x 1	Carry Out of ALU
For L ONLY:	
0 0 x x	Load
0 1 x x	Shift Right
1 0 x x	Shift Left
1 1 x x	Cross Shift
On Shifts ONLY:	
x x 1 x	Carry Into ALU
x x x 1	Carry Out of ALU
On Non-Shifts:	
x x x 1	Clear Carry

x — "don't care" condition

Tests

Bits Set	Definition
5	I/O Int. (ATN)
6	Auto-restart (ARST)
7	Cons. E. (CATN)
8	Cons. Sngl. (SNGL)
9	Utility flip-flop (UT)
10	Mem. Par. Fail (MPF)
11	Prim. Pwr. Fail (PPF)
12	Fast I/O Int. (FAST)

Load, I/O = Destination

E field	Definition
x x 0 1	Address
x x 1 0	Data Available
x x 1 1	Command

Load, I/O	Source
x x 0 1	Acknowledge
x x 1 0	Data Request
x x 1 1	Status Request

ROMWATS PROGRAM (05-005)

1. PURPOSE

This program converts ROM object tapes as generated from the Micro-Code Assembler or Micro-Code Simulator into a form which is acceptable to the automated ROM wiring machine.

2. PROGRAM DESCRIPTION

The ROMWATS program requires 2724 bytes of core in addition to 2048 bytes for storage of the ROM program.

Two ROMWATS tapes are generated for each ROM program. The first tape (wire tape) is used to wire the ROM. The second tape (check tape) is used to check the ROM after it has been wired.

3. PROGRAM INPUT

The ROMWATS program consists, in part, of a loader which is able to accept ROM object tapes. Figure 1 shows the form of this tape.

The object tape input must possess the following characteristics; no other limitations are imposed.

1. Four-level code. Tape channels 8-5 are ignored, hexadecimal data resides in channels 1-4.
2. Each non-contiguous block of data must be preceded by a minimum of six frames of blank tape and a four frame ROM address.
3. Since core memory is byte addressable and ROM is halfword addressable, ROM addresses which are punched on the object tape must be multiplied by two (left shifted one place).
4. The ROM object image buffer allows storage of 1024 ROM words segmented into four pages of 256 words each. In order to allow for ROM programs of 2048 words, each core buffer page is assigned two ROM addresses as shown in Table 1. Once a core buffer page has been filled, a ROMWATS tape will be dumped before the buffer can be utilized for the other ROM page. For this reason, it is recommended that the addresses on the ROM object tape be in ascending order.
5. Since each ROMWATS tape generates a full 1024 words of ROM wiring data, it is imperative that the ROM object tape contain only that data which is to be wired. Extraneous data appearing on the object tape will be wired; unused ROM locations should be signified by not appearing on the object tape or by containing zeros.

TABLE 1. ROM ADDRESSES

CORE BUFFER PAGE	CORE LOC. HEX	CORRESPONDING ROM PAGES	
		UNSHIFTED	SHIFTED
0	800-9FE	000-0FF, or 400-4FF,	000-1FE 800-9FE
1	A00-BFE	100-1FF, or 500-5FF,	200-3FE A00-BFE
2	C00-DFE	200-2FF, or 600-6FF,	400-5FE C00-DFE
3	E00-EFE	300-3FF, or 700-7FF,	600-7FE

The ROMWATS program reserves storage space for Absolute CLUB W/O Output (03-003). The location in core of the object image can be found by taking the ROM address as it appears on the Micro-Code Assembler listing, shifting it left once and adding a displacement of X'800'. CLUB may be used to modify its contents. Changes should be made after ROMWATS prints the message "TURN PUNCH ON". After making the necessary changes, turn the punch on and execute at location (X'184A').

4. PROGRAM OUTPUT

The output of the ROMWATS program will consist of two tapes for each 1024 words of ROM code. The first tape is a wire tape which is used to wire the ROM. The second tape is a check tape which is used to check the ROM once it is wired. Two sets of ROMWATS tapes must be generated for ROM programs that are in excess of 1024 words.

Figure 2 shows a section of ROMWATS output tape corresponding to the object input tape section shown in Figure 1. Figure 3 shows the core memory allocation for ROMWATS. Each record on the ROMWATS output tape consists of the data shown on Table 2.

1. Odd parity is punched in column 8 and generated over columns 4 through 1 only.

2. Records are normally separated by 10 frames of leader. Every thirty second record is followed by 20 frames of leader. This signifies an address bank change.

3. The addresses generated by the ROMWATS tape are not linear. This is done to facilitate wiring of the ROM whose address decoding is non-linear due to hardware considerations.

5. OPERATING PROCEDURE

Use the following procedure to generate the ROMWATS tape.

1. The ROMWATS program is in absolute form and should be loaded with the ABSOLUTE LOADER. The program occupies location X'1010' to X'1AB4' inclusive. Locations X'800' to X'1000' are reserved for the ROM object image. Locations X'80' to X'800' are reserved for Absolute CLUB W/O Output (03-003) if it is necessary.

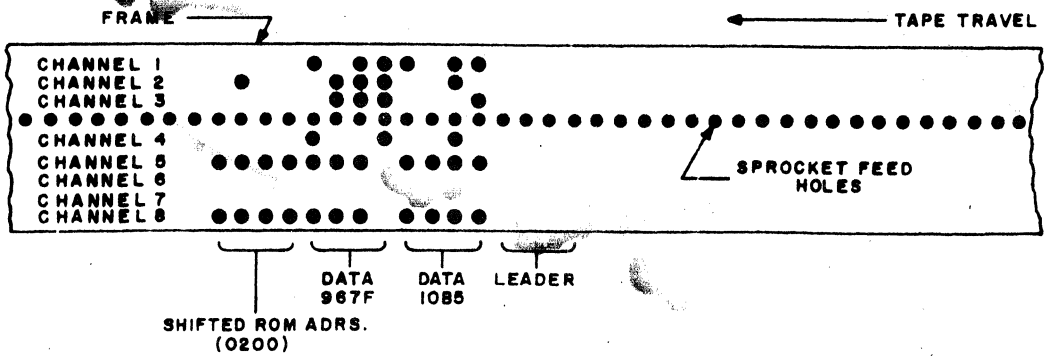


Figure 1. ROMWATS Object Tape Input

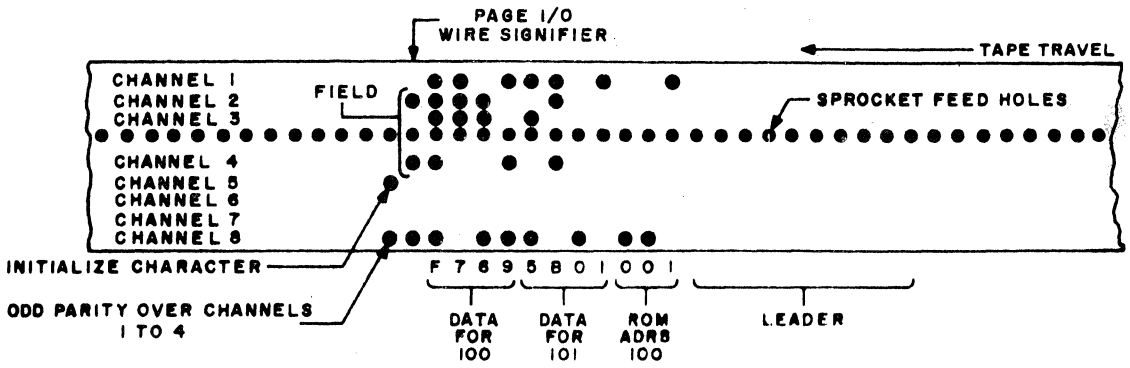


Figure 2. ROMWATS Output Tape

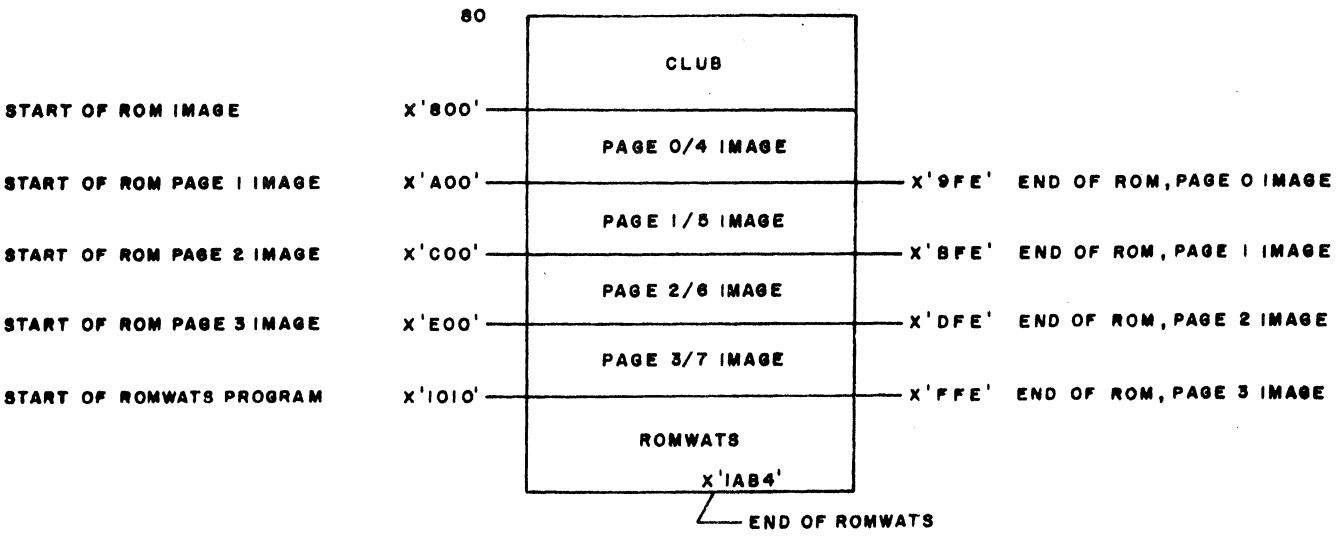


Figure 3. Core Memory Allocation

TABLE 2. ROMWATS TAPE FORMAT

FRAME	TYPE CHARACTER	PURPOSE	TAPE COLUMNS 8 7 6 5 4 . 3 2 1
1	INITIALIZE	SIGNIFIES RECORD TO FOLLOW	1 0 1 0 0 . 0 0 0
2	Directive	Signifies wire or check record, page 0, 1 or 2, 3	
		page 0, 1 wire	1 0 0 0 1 . 0 1 0
		page 2, 3 wire	1 0 0 0 0 . 1 1 0
		page 0.1 check	0 0 0 0 1 . 0 1 1
		page 2, 3 check	0 0 0 0 0 . 1 1 1
3-10	8 Frames of data: 3-6 even ADRS data, MSB6. 7-10 odd ADRS data, MSB 10	Words to be wired	
11-13	3 frames of ADRS specifying even ADRS, MSB 13	ADRS where data is to be wired	
14-23	10 frames of leader	inter-record gap	

2. Location X'78' specifies the input device number. Location X'79' specifies the input device command. The input device is used to read the ROM object tape into the ROM image buffer. Location X'7A' specifies the output device number. Location X'7B' specifies the output device command. The input device is normally a paper tape reader. The output device is always some sort of paper tape punch. Location X'78' to X'7B' must be set by the operator before the program can be executed. Table 3 lists the most used device numbers and commands.

3. The printing of all messages is done on a TTY (Device 2). If the ASR 35 is used as the output device, the mode selector should be switched to TTR when the message "TURN PUNCH ON" is received. If the ASR 35 is used as an input device, the mode switch should be in the TTS position after the message "INITIALIZE, EXECUTE AND READER ON" is received. Otherwise, the ASR 35 should be left in the K mode to receive printed messages.

TABLE 3. DEVICE NUMBERS

INPUT DEVICE	(X'78') NUMBER	INPUT (X'79') DEVICE CMND	OUTPUT DEVICE	(X'7A') NUMBER	OUTPUT (X'7B') DEVICE CMND
TTY	X'02'	X'94'	TTY	X'02'	X'18'
HIGH SPEED READER	X'03'	X'99'	H S PUNCH	X'05'	X'9A'

4. After the program has been loaded, address X'1110' should be selected and the program executed in RUN mode.

5. The message "PLACE OBJECT TAPE IN READER INITIALIZE, EXECUTE AND READER ON" will be printed out and the machine will go into a Wait state. Load the tape with the leader over the read heads, release Data Switch 15 (far right switch) and depress EXECUTE.

6. If a message stating "THE BUFFER FOR THIS PAGE IS FULL" is printed and the full object tape has not been read, it means that the object program is in excess of 1024 words and two ROMWATS passes will be necessary.

7. If no message is printed out during the loading procedure, the system will remain in the Read mode until Data Switch 15 is depressed. This is done to allow multiple tapes to be loaded.

8. In any case, after the statement "TURN PUNCH ON" is printed, either by virtue of Data Switch 15 being depressed, or buffer full condition, the punch should be turned on and the EXECUTE switch depressed (the machine will be in the Wait state after the punch on statement). The ROMWATS tape will then be punched.

9. If the ROM object image must be modified before dumping the ROMWATS tape, this should be done after the statement "TURN PUNCH ON", but prior to depressing the EXECUTE switch. This procedure is described in Section 3. After modification of the object program, address X'184A' should be selected, the punch turned on and the program executed in the Run mode.

10. The output tape will consist of two sections spaced by about one foot of leader. The first section is a wire tape, and the second section is a check tape. The tapes should be separated and identified by job title, data, and pass number, if applicable.

11. If the buffer full message was received, another pass will be necessary to generate the full complement of ROMWATS tapes. The tapes just generated should be removed and marked Pass 1 wire tape and Pass 1 check tape. The input object tape should be repositioned to the leader just preceding the four address frames which were read, and the EXECUTE button depressed. Steps 9 and 10 should then be repeated and will result in the generation of two more tapes. These tapes should be marked Pass 2 wire tape and Pass 2 check tape.

12. After completion of the program, the machine will be left in a Wait state at Location X'1110'. Depressing EXECUTE will cause re-execution of the program.

13. If a tape jam occurs during the punching of the ROMWATS tape, or for some reason another set of ROMWATS tapes is desired, address X'184A' should be selected and the program executed.

READER COMMENTS

The General Electric Company solicits your help in providing complete and accurate technical publications covering our Process Computer equipment. Please answer the questions listed here by checking the appropriate block. If your answer to any of these questions is "NO", please explain in "Comments" section below. Your comments and suggestions become the property of General Electric Company.

	YES	NO
• Is this publication adequate for your needs?	☐	☐
• Is the material		
Presented in clear text?	☐	☐
Conveniently organized?	☐	☐
Adequate detail?	☐	☐
Adequately illustrated?	☐	☐
Suitable for the technical level desired?	☐	☐

- What is your computer application? _____

_____

- What is your position? (Supervisor, Programmer, Technician, etc.) _____
- How is this publication used:

Familiarization of the subject?	☐	As reference material?	☐
For training purposes?	☐	For maintenance of equipment?	☐

Other (explain) _____

- Please give complete references (page number, line, etc.) with your comments. Please indicate if a reply is desired and include your proper mailing address.
- Your cooperation will be appreciated.

COMMENTS:

No postage necessary if mailed in the U. S. A.

Staple

YOUR ASSISTANCE, PLEASE

This document has been generated to help us serve you better. Your answers to the questions on the reverse side of this form, together with comments and recommendations, will be of great value to us in providing the best possible publications for your use. Your answers and comments will be carefully reviewed by the person who generated this publication, and may result in a revised publication. Your comments and recommendations become the property of General Electric Company.

Communications concerning Technical Publications should be directed to:

Manager, Technical Publications
GE Process Computer Department
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 DEPARTMENT
2255 West Desert Cove Road
Phoenix, Arizona 85029

Attention: Technical Publications

Cut Along Line

Fold Fold

Additional Comments: