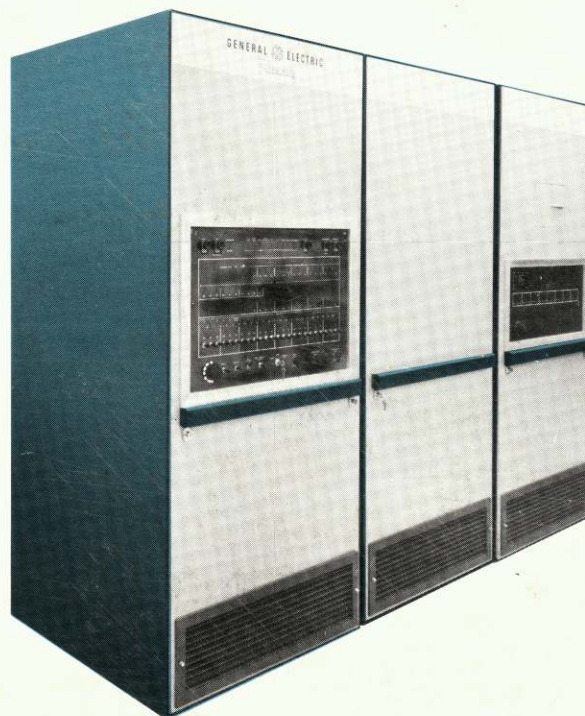


GE PAC 4000

PROGRAMMING MANUAL



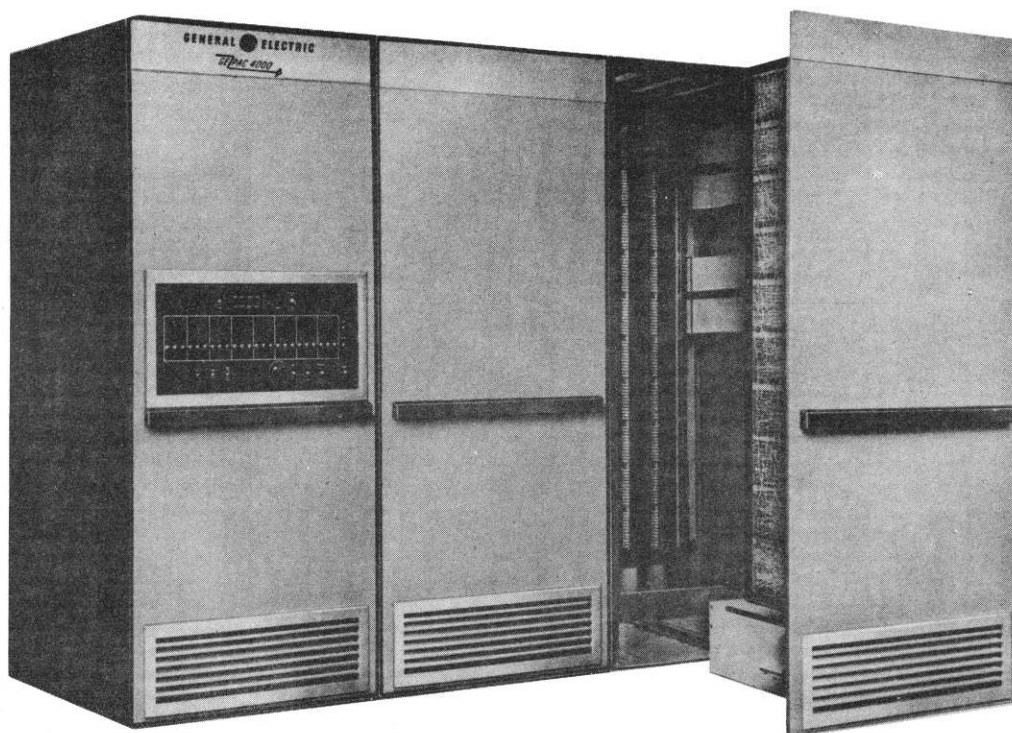
GENERAL  **ELECTRIC**

PROCESS COMPUTER SECTION
INDUSTRY CONTROL DEPARTMENT • PHOENIX, ARIZONA



GE PAC 4000^{*}

PROGRAMMING MANUAL



CONTENTS

I	FOREWORD	I:1
II	GE/PAC SYSTEM DESCRIPTION	II:1
III	GE/PAC 4040 SYSTEM	III:1
	HARDWARE	III:1
	PROGRAMMING INSTRUCTIONS	III:4
	INTERNAL EFFECT INSTRUCTIONS	III:5
IV	GE/PAC 4050/4060 SYSTEM	IV:1
	APPLICATION	IV:1
	FEATURES	IV:1
	DESCRIPTION	IV:1
	INSTRUCTIONS	IV:3
V	DATA MANIPULATION	V:1
	GENERAL	V:1
	DATA REPRESENTATION	V:1
	ARITHMETIC OPERATIONS	V:4
	LOGIC AND BIT LOGIC OPERATIONS	V:5
	LISTS AND THEIR USAGE	V:6
	GEN 1 INSTRUCTIONS ¹	V:8
	A NOTE ON THE COUNT INSTRUCTIONS	V:10
VI	PROGRAM CONTROL	VI:1
	INTRODUCTION	VI:1
	PROGRAM CONTROL	VI:1
	MEMORY ADDRESSABILITY	VI:2
	ADDRESS MODIFICATION	VI:3
	SUBROUTINE LINKAGE	VI:4
	PROGRAM SWITCHES	VI:5
	TESTS AND THE CONDITIONAL BRANCH BITS	VI:5
	INDEX AND LOOP CONTROL	VI:6
	QUASI INSTRUCTIONS (or Extended Function Commands)	VI:6
	XEC AND ITS USAGE	VI:6
	PROGRAM INTERRUPT	VI:8
	RELATIVE, ABSOLUTE AND COMMON SYMBOLS	VI:9
VII	INPUT AND OUTPUT CONTROL	VII:1
	GEN 2 INSTRUCTIONS	VII:1
	I/O COMMUNICATION	VII:1
VIII	DATA COMMUNICATION	VIII:1
	BULK STORAGE DRUM MEMORY (4500)	VIII:1
	TIMING	VIII:2
IX	PROCESS COMMUNICATION	IX:1
	DIGITAL INPUT SCANNER (4400)	IX:1
	ANALOG INPUT SCANNER (MODEL 4100)	IX:1
	ACCURACY IN MEASUREMENT OF ANALOG QUANTITIES	IX:8
	MULTIPLE OUTPUT DISTRIBUTOR (4300)	IX:8
	TIMED OUTPUT CONTROLLER (4302)	IX:10
X	MAN COMMUNICATION	X:1
	PERIPHERAL BUFFER (MODEL 4201)	X:1
	OPERATORS CONSOLE	X:7

CONTENTS CONTD.

XI	GE/PAC PROGRAMMER'S CONSOLE	XI:1
	DESCRIPTION OF GE/PAC 4040 CONSOLE	XI:1
	OPERATION OF GE/PAC 4040 CONSOLE	XI:3
	USE OF PROGRAM LOAD OPTION	XI:3
XII	OCTAL DECIMAL CONVERSION TABLE	XII:1
XIII	FLOW CHART SYMBOLS	XIII:1
XIV	GE/PAC INSTRUCTION SUMMARY	XIV:1

I FOREWORD

The GE/PAC 4000 Programming Techniques Manual is devoted to the more advanced techniques of real-time programming. Many of the distinctive features of the GE/PAC 4000 for computer control applications are presented. The material in this manual is directed to the reader with a basic understanding of programming fundamentals. Additional GE/PAC 4000 information is available in the following manuals:

- Reference Manual - system hardware.
- Process Assembler Language - assembly program.
- Monitor System Manual - monitor program.
- Fortran II Compiler - compiler language.

Information presented in this manual and the programming examples provided are written in the assembly language (PAL). The assembly language uses a symbolic form that is convenient for the programmer. The PAL Statement Coding Form is shown in Figure 1. Each line of coding represents one instruction to the assembler. Four fields are used by the programmer as follows:

1. Location field - columns 1 through 6.

Symbolic locations of six or less alphanumeric characters are used.

2. Instruction (TYPE) field - columns 8 through 10.

Mnemonic instruction codes of two or three alphabetic characters are used.

3. Statement field - columns 12 through 68.

Symbolic addresses of six or less alphanumeric characters are used. Decimal integer values, or the combination of parameters connected by arithmetic operations, may be used. Indexing is specified by a comma and the appropriate index. Comments are also included in this field.

4. Identification field - columns 70 through 80.

Column 70 defines the language used and the remainder of the field identifies the project number, program number and card sequence number.

The symbolic program must be converted into actual computer (machine) language by the PAL assembler before execution. During the assembly validity tests are made on each instruction to indicate simple programming errors and omissions.

GENERAL ELECTRIC
PROCESS COMPUTER SECTION
PHOENIX, ARIZONA

**PROCESS LANGUAGE STATEMENT
CODING FORM**

PROCESSOR KEYS

GE 312	0 - Delete
GE 312	3 - NGAP
GE 412	2 - PAF
GE 412	4 - COBL
GE 7PAC	6 - Assembler
GE 7PAC	7 - Fortran

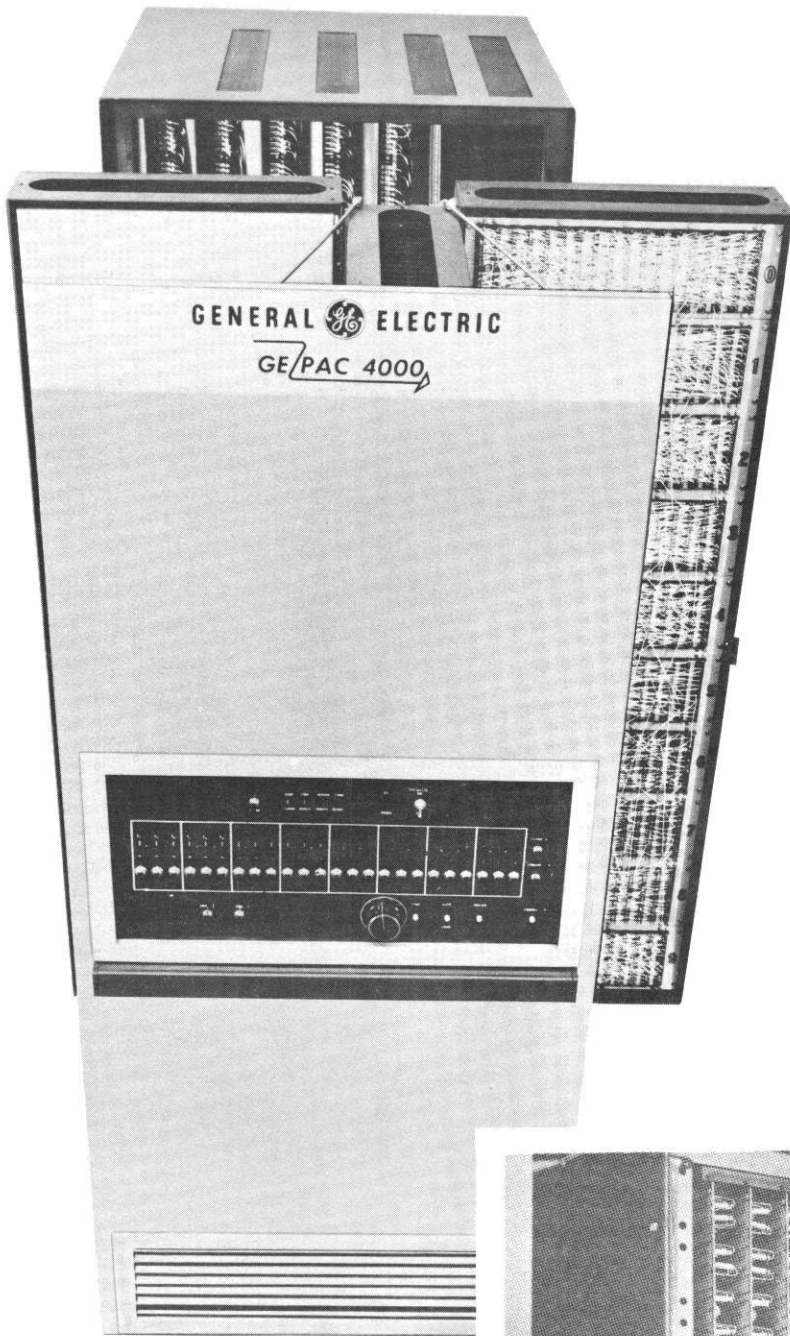
Project Name: _____

Page: _____ of _____ Date: _____

Programmer: _____

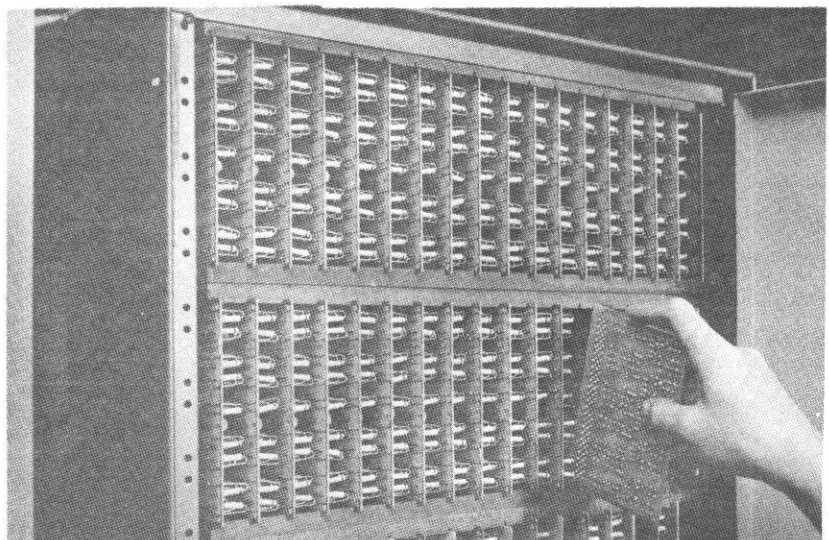
LOCATION*	TYPE OP CODE	STATEMENT OFFERINGS	BRANCH CONTROL FIELD		Any Case	P R	Prog #	Sequence #
			C	70				
1								
2								
3								
4								
5								
6								
7								
8								
9								
0								
1								
2								
3								
4								
5								
6								
7								
8								
9								
0								
1								
2								
3								
4								
5								
6								
7								

* Restricted to five characters for COBL. Shared space in GE/PAC manual restrictions.



Central processor with
roll-out assembly withdrawn
and pages swung out

Typical page (front view)
showing card being inserted



II GE/PAC SYSTEM DESCRIPTION

GENERAL

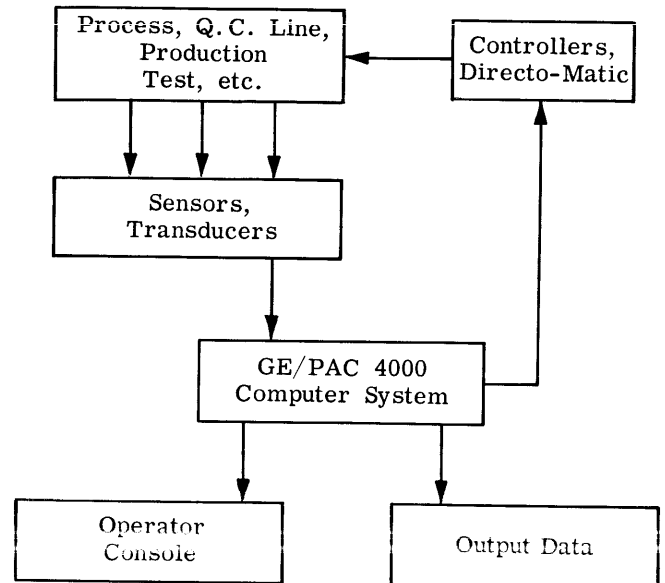
The GE/PAC 4000 line is an entirely new process computer system especially designed to fulfill computer system needs across the complete range of process control applications. This is accomplished in GE/PAC by the concepts of modularity and expandability. Modularity allows offering customers precisely the equipment that is needed for an initial installation; expandability permits building on GE/PAC to meet future requirements.

GE/PAC systems are less expensive than competitive equipment for other reasons:

1. Interface costs are reduced because GE/PAC is compatible with Directo-Matic[®] II control, GE/TAC telemetering and control, GE/MAC process instruments, and G-E sensors.
2. Costs for a controlled environment are reduced because GE/PAC, utilizing silicon semiconductors, will provide reliable operations in ambient temperature ranging from 32°F to 131°F.
3. Computer room costs can be reduced because GE/PAC requires less space than most computers and full front access saves wall space.

GE/PAC 4000 system is comprised of a complete family of standard computer modules that can be organized into many configurations to meet a wide variety of applications. GE/PAC systems can be used as an off-line computer, for on-line data acquisitions and logging, for data processing, and for process monitoring and control. GE/PAC may be integrated in systems with Directo-Matic, GE/MAC, X-ray sensors, and many other forms of business and process devices.

A simplified layout of a process computer application is shown in the following sketch. The upper block represents the process, or function, which is being monitored or controlled by GE/PAC. Pertinent variables which provide the necessary information about the performance of the process are sent into the GE/PAC system by sensors and transducers. The sensors and transducers mainly serve to produce the kinds of signals that the computer can work with. The signals are converted into digital form and used by GE/PAC under direction of the internally stored program. Outputs from GE/PAC can be



communication with operators by lights and displays or by printout devices. Outputs to the process from GE/PAC can be direct or through controllers.

Benefits from GE/PAC controlled systems are many; some are listed below:

- Reduced operating costs
- Improved quality
- Increased throughput rate
- More process information
- Greater management control
- Safer operation

DESCRIPTION

The GE/PAC 4000 is a fast, highly reliable, flexible family of standard modules especially designed to fit industrial applications. A list of the family of standard modules is tabulated in Figures II -1 and II -2.

There are four integral parts of the GE/PAC 4000:

1. Memory - where data and instructions are stored.
2. Arithmetic and Control Unit - which performs calculations and controls the execution of programs.

3. Input - which translates sensor readings, contact closures, manual switches, and other data into usable digital information.
4. Output - which translates digital information into logs, analog set points, lighted displays, and other forms of information usable outside the system.

A diagram showing the organization of the modules of the GE/PAC 4000 system is illustrated in Figure II -3. The memory and the arithmetic and control unit communicate with each other via the memory multiplexer, providing that, in addition to the arithmetic unit, a drum controller or controller selector is used. If the arithmetic unit is the only memory "user device," the memory multiplexer is not required. The arithmetic unit in turn communicates with the automatic program interrupt module, with the input/output buffer, and with the peripheral buffer. The I/O buffer communicates with the analog and digital inputs and with analog and digital outputs.

The arithmetic and control unit performs calculations and other logical operations, and sequences and distributes data throughout the system. It supplies and receives information from the memory, the automatic priority interrupt, the I/O buffer, and the peripheral buffer.

The memory module contains the clock that provides the timing pulses for the system. Memory is a random access storage characterized by its high-speed capability. In addition to its storage feature, it serves to validate information for the system.

The automatic priority interrupt detects and locates ready and complete signals from devices that require testing at long-time intervals. It is also used to give instantaneous signals to the computer for changes in process conditions. When the interrupt is detected, a subroutine is initiated that takes priority over the mainline program, and with the use of the AU services, the requesting interrupt. The API may be expanded from eight interrupt levels to any required number of levels up to 64.

The I/O buffer is the communication link between the AU and the system input and output devices. It acts as a multiplexer for digital and analog inputs and as a multiplexer and amplifier for output signals.

The peripheral control buffer communicates with the AU and is used as a data buffer, translator, and sequencer device for the various peripherals.

Signal inputs may be from contact closures, pulses, or measuring devices. All input information is converted to usable digital information before it is analyzed by the AU. The AU uses the logic and

equations stored in memory to decide whether any control actions are required. If corrective action is needed, the AU provides the necessary information to the digital and analog output circuits to change the process control variables.

	<u>Model Number</u>
<u>Central Processor</u>	
4040 (1-16K)	C4040
4050 (1-64K)	C4050
4060 (1-64K)	C4060
Automatic Program Interrupt (API) (8-32) interrupts	4031
Automatic Restart	4797
Stall Alarm	4091
I/O Expander Channels	4070
Program Load	4020
<u>Process Communication</u>	
<u>Analog Inputs</u>	
Scanner Control	4100
A/D Converter (succ. approx.)	4130
A/D Converter (integrating)	4135
Channel Selection	4140
High-Level Amplifier	4125
Low-Level Amplifier	4120
Switch Matrix Termination	4180
Signal Conditioning	4150
Thermocouple Reference	4160
Power Supplies	4190
<u>Digital Inputs</u>	
Digital Control	4400
Digital Signal Conditioning	4450
Terminations	4480
Status Change Detection	4455
<u>Digital Outputs</u>	
Multiple-Output Control	4300
Binary-Contact Outputs	4362
Time-Contact Output Control	4302
Contact Outputs (Timed)	4366
Termination	4380
Analog Outputs	4364
Decimal Outputs	4360
<u>Man Communication</u>	
Typewriter (fixed carriage)	4270
Paper-Tape Reader	4211
Paper-Tape Punch	4251
Card Reader	4240
Typewriter (long carriage)	4223
<u>Data Communication</u>	
Drum File and Control	4520
Magnetic Tapes and Control	4530
Disc Memories and Control	4540

Figure II-1. GE/PAC Standard Package Model Numbers

<u>Module</u>	<u>Function</u>	<u>Remarks</u>
Scanner Control (4100) and A/D Converter (4130A10 or 4135A10)	Analog inputs	Two-way communication over one channel. Output a scanner control word to S Register (4100), input digitized conversion from C Register (4130A10 or 4135A10). Up to 256 analog inputs each group. Options of 1, 2, and 4 groups are available.
Peripheral Control (4200)	Typewriter, paper-tape punch, paper-tape reader, card reader	Two-way communication over one channel. Up to 8 peripherals.
Contact Status Scanner (4400)	Contact or digital inputs	Input only to A Register. Up to 64 groups of 21 contact inputs each group.
Pulse Counter Hardware (4153)	Pulse inputs	Input only to A Register. One counter, 10 bits long.
Output Controller (4300)	Latched-output contacts, latched-output digital, latched-output analog	Output only from A Register. Up to 64 groups, 16 bits each group.
Timed-Output Controller	Timed-output contacts	Output only from A Register. Up to 64 timed contact outputs.
Latched-Output Contact (4177)	Latched-output contacts, latched-output digital, latched-output analog	Output only from A Register. Use in place of 4300 small OC system. One group of 18 bits.

Figure II-2. I/O Modules

SPECIFICATIONS

1. ENVIRONMENT

The cabinet, whether opened or closed, operates within the following limits:

1. temperatures from 0 to 55C (32F to 131F)
2. relative humidity from 5 to 95%
3. atmosphere where dust particles are no larger than 10 microns
4. vibration between 5 and 33 CPS with a 0.015-inch displacement
5. shock less than 15 G's for 250 micro-seconds.

2. POWER REQUIREMENTS

Power supply required for the system is 230 volts, 3-wire grounded neutral 60 cycles, with permitted voltage variation of $\pm 10\%$ and frequency variation ± 1 cycle per second. Optional 50-cycle systems are available. Internal system voltages are +12 volts, -12 volts, +5 and 0 volts for digital circuits, +28 volts for peripheral devices, +50, -50, 0, +12 and -12 volts for analog circuits. Power requirements vary from 1 to 1.5 KW for a small system with blower up to 4 to 7 KW for a system with 16K core.

3. PACKAGING

Printed circuit boards have a copper laminate for ground plane on one side. Use of proved flow soldering and component mounting techniques on all circuit boards provide high reliability and low maintenance expense. Optimized combinations of flip flops, gates, and other circuits on the boards results in a low percentage of unused components. Some components on the board are mounted on end (to make optimum space usage) with the lead on the high end of the component brought back through the board for soldering and to give support to the cantilever-mounted component. All boards are marked with an easily understood alphanumeric designation for ease of identification. The boards have provision for 51 pin terminations which fit securely in the receptacle which also has provision for labeling the designation of the board it holds. Circuit boards are 5.4" x 4" with maximum component height of 3/4".

The circuit board receptacle containing 17 cards is of molded Lexan® to provide high strength and rigid support for printed circuit boards. It has the card guides integrally molded in its structure with cards spaced on 1" centers. Lexan's high strength and non-fracturability protects against malformation of the board receptacle, thus protecting the boards as well.

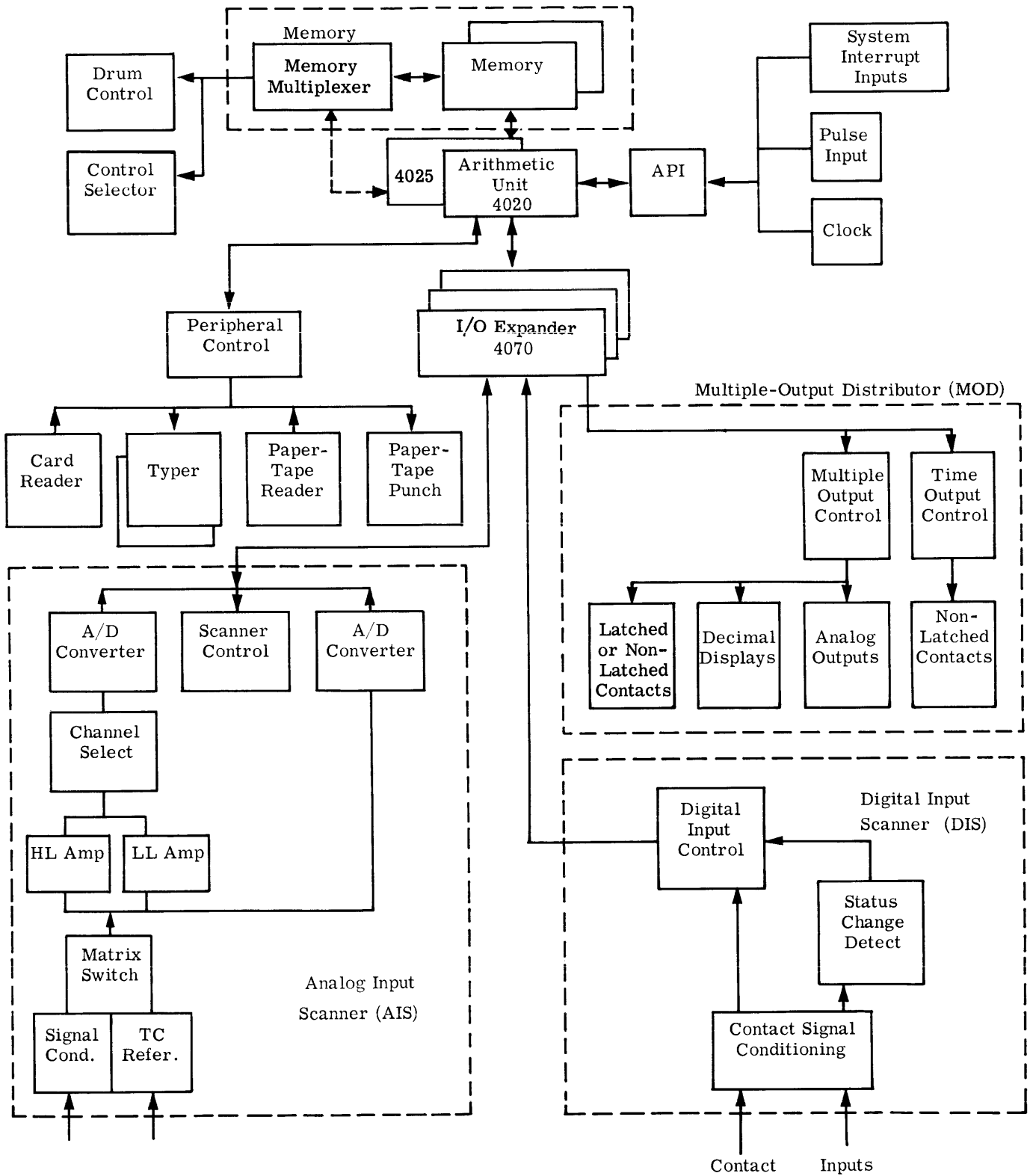


Figure II-3. GE/PAC 4000 Modules

The back panel is fitted to the number of card rows required for the particular module. Modules can thus be assembled, wired, and tested in the factory before mounting in the cabinet. The availability of prewired assemblies for simple field expansion makes feasible supplying initially only the portion of the system required for the original automation effort.

A GE/PAC cabinet is an enclosure of steel which mounts against the wall with all customer terminations and certain components, such as power supplies, mounted on the interior back wall or on the left side.

The electronic portion of the system is mounted in frames which contain 10 card holders, a maximum of 170 cards per frame. They can also accommodate 19" relay racks or certain specially mounted devices. These frames are installed similar to book pages on a roll-out assembly which is wheeled (on 4" wheels) into the cabinet enclosure. The outer two pages are hinged at the back to permit access to both sides and to the fixed center page and to facilitate short lead lengths to terminal boards on the fixed page. Certain devices, such as the computer console and display lamps, are mounted on the front panel which is rigidly supported on the roll-out base. Air circulation for cooling of the circuit components is provided from a blower mounted in a plenum in the assembly base. The blower supplies about 200 CFM to each page frame.

Air enters through grillwork in the front of the cabinet, passes through the page frames and out through a drip-proof exit in the top of the cabinet. A certain portion (about 100 CFM) of the air in the base is diverted to the left to cool some electronic devices mounted on the inside cabinet walls.

This cabinet and assembly design has several advantages. It provides single-side access to all system components, saving floor space normally needed for a rear aisle. It allows easy access to all components by simply withdrawing the assembly and either opening the book-page frames or by entering the cabinet to gain access to terminations, power supplies, or other devices mounted on the walls. The cabinet dimensions are 32" wide by 36" deep by 76" high providing plenty of working space. Leads which must go from the cabinet wall over to the assembly use a multiconductor belt or ribbon which is flexible horizontally. These are close to and parallel to the right side of the cabinet to permit unobstructed entry to personnel into the cabinet when the assembly is withdrawn.

The matrix relays for the analog scanner and customer terminations are mounted on the back or side wall of the cabinet. They are accommodated in groups of 16 by a basic mounting block which accepts 16 customer terminations (2 or 3 wire) up to #12 conductor and plug-in capability for 16 double-pole relays and 16 signal-conditioning elements. The mounting module contains an additional relay for matrix switching to provide isolation of the individual matrix. The computer cabinet can contain up to 128 analog signal inputs or combinations of inputs and outputs. This permits use of the single-cabinet design for a small system. The single cabinet houses the terminations of sensor leads, analog scanner modules and computer -- the complete process computer system with the exception of typewriters and paper-tape equipment in one compact enclosure. The functional modularity of the system means only those functions needed must be purchased initially; future expansion in the field, as automation requirements grow, is easy. Where a greater number of input terminations are needed than can be accommodated in one cabinet, a cabinet 20" in depth is available which provides additional mounting space. The 20" deep front-access cabinet can also contain one frame, or page, for mounting electronic modules.

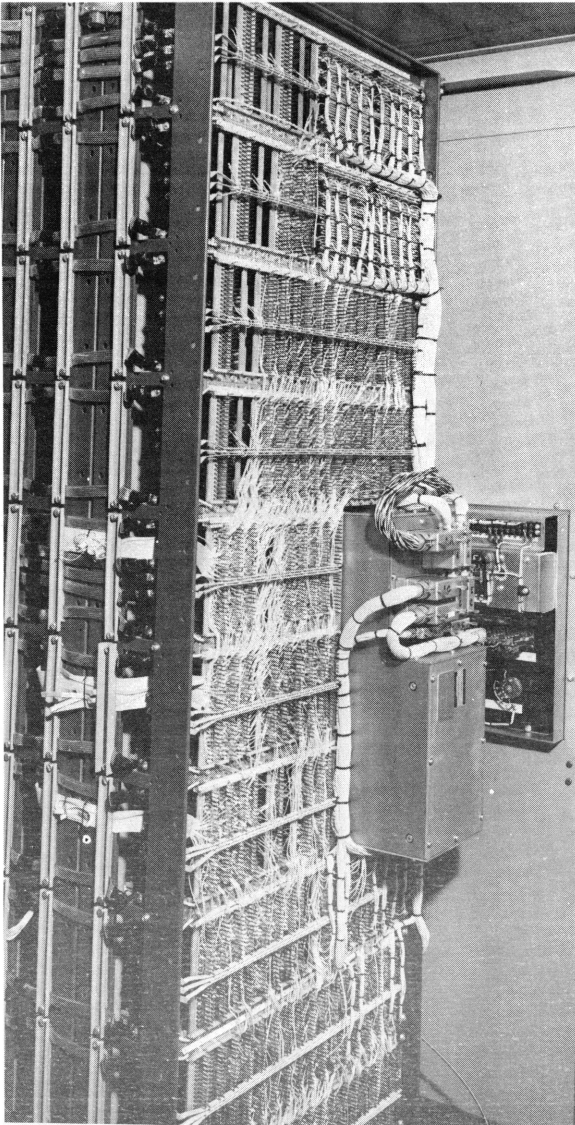
4. CIRCUITS

Electronic circuits are contained on boards within functional modules located on the assembly pages. The solid-state silicon components are unaffected by ambient temperature variants (0 to 55C) and thermal shock (20C in 5 minutes), and in addition, provide fast switching time. The NAND-type diode logic employed exhibits the following special features:

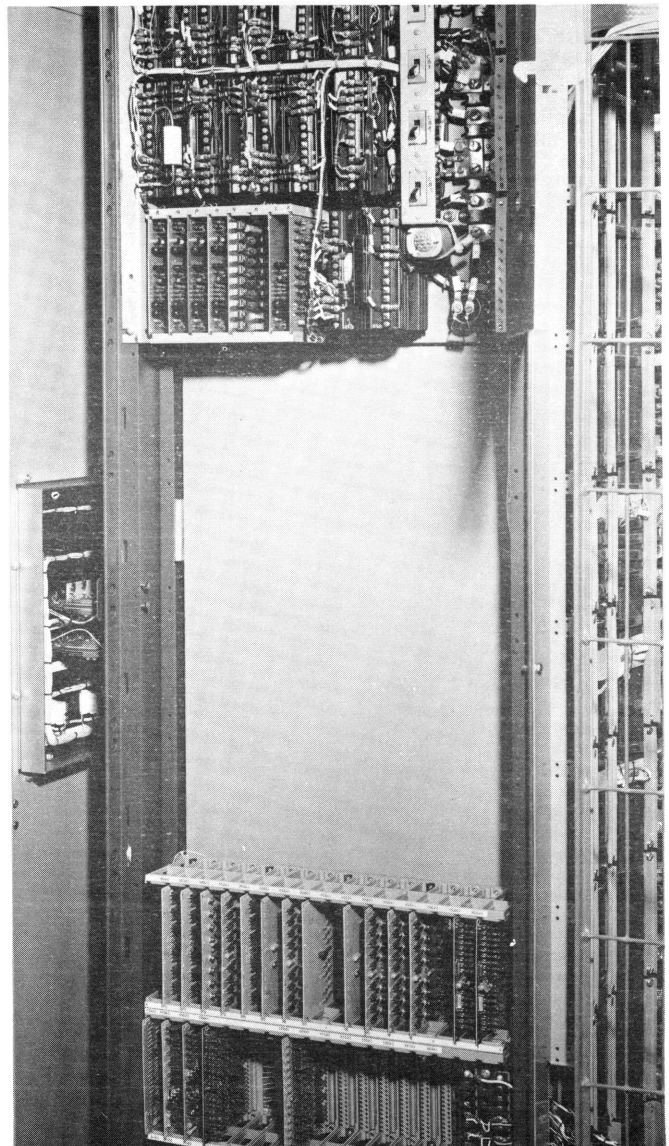
1. "AND" and "OR" logic functions with inversion and power-gain output
2. output rise time nominally 25 millimicro (nano) seconds
3. output-to-input delay less than 50 nanoseconds

These components are packaged in a manner to utilize a minimum of space.

Compatible 3MC, 300 KC, and 20 KC logic circuit families permit using the circuit speed best suited to a given module's requirements. These differ for various modules depending on speed of operation and noise susceptibility. The wide choice of available circuit boards enhances the flexibility of the system in meeting a wide variety of applications. Logic is designed to be capable of operation when all varying parameters are at the most unfavorable limit of their tolerance at the same time. This is known as "worst case" design and contributes to high reliability.



Back side of page with core memory

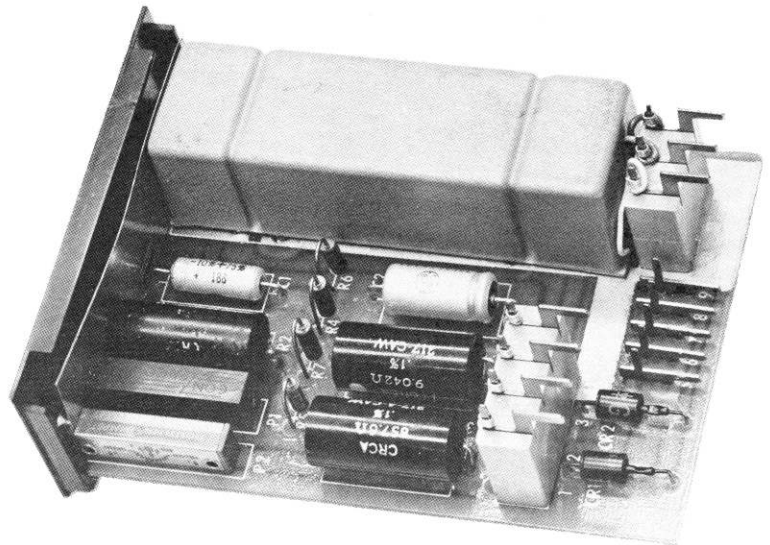


Front of page showing power supply (top) and peripheral buffer (bottom). Blank space available for future expansion.



Arrangement of input terminations and signal conditioning units for GE/PAC scanner

Typical signal conditioning card





III GE/PAC 4040 SYSTEM

HARDWARE

APPLICATION

The 4040 System (model 4020 arithmetic and control unit) is used in GE/PAC systems as the computation center of the computer. It is used to perform a wide variety of arithmetical and logical operations at high speed.

FEATURES

1. The normal operation time for most unmodified instructions is less than 16 microseconds.
2. The normal operation time for most modified instructions is approximately 22 microseconds.
3. 2's complement arithmetic is employed.
4. 7 indexing address modification memory cells are provided.
5. 118 defined instructions, including extended function commands, are available.
6. 16,384 memory locations are directly addressable.
7. Contains relative-addressing ability.

DESCRIPTION

Description of the model 4020 will be subdivided into two parts: (1) Elements Comprising the Arithmetic Unit, and (2) Instruction Format and Sequencing.

1. ELEMENTS COMPRISING THE ARITHMETIC UNIT

The model 4020 arithmetic and control unit is comprised of data registers, full address, and control flip flops, as shown in the block diagram of Figure III-1. The registers used are as follows: A, B, I, P, and J. Control flip flops, along with the full adders, work in conjunction with the registers to perform arithmetic and bit manipulation operations, and for checking and remembering conditions occurring in the AU. Names of the flip flops and full adders employed are: Flip Flops, Test (TSTF), Overflow (OVRF), Permit Automatic Interrupt (PAIF), Priority

Interrupt First (PIIF), Execution (XECF), Demand (DEMF), and Carry - Adders; A, B, and P. The function and description of each of these elements are discussed in the succeeding paragraphs.

A Register — the A register is the primary working register for the arithmetic unit. It is comprised of 24 high-speed flip flops in a bit configuration numbered 0-23; bit 23 is the most significant. Functionally, it acts as temporary storage for data coming from or going to the input/output equipment of the computer and is the accumulator register during arithmetic and bit-manipulation operations. Transfer of data from A to internal registers of the AU is accomplished in serial. Data transferred from A to registers and devices external to the AU is accomplished in parallel.

B Register — the B register is a 24-bit parallel-entry buffer register used between core memory and the AU. It is made up of 24 high-speed flip flops in the same configuration as the A register. B is the communication link for information transfer between memory and the AU registers A, I, and P.

I Register — the I register is a 24-bit register comprised of 24 high-speed flip flops arranged in the same bit configuration as the A and B registers. It is the holding register for the bits that control the operation of the AU.

Adder A — the FAA is used in any AU operation involving the A register.

Adder B — the FAB is used in any AU operation involving the I and B registers.

Adder P — the FAP is used in any AU operation involving the modification of the contents of the P register.

2. INSTRUCTION FORMAT AND SEQUENCING

There are four types of instructions employed by the arithmetic and control unit model 4020 (4040 System): Full Operand, GEN 1, GEN 2 and GEN 3. Each distinct instruction involves either memory addressing, input/output device selection, bit manipulation of the A register, or extended-function commands. The bit configuration of each type of instruction determines the operation to be performed and the control necessary to perform it. Figure III-2 summarizes these types.

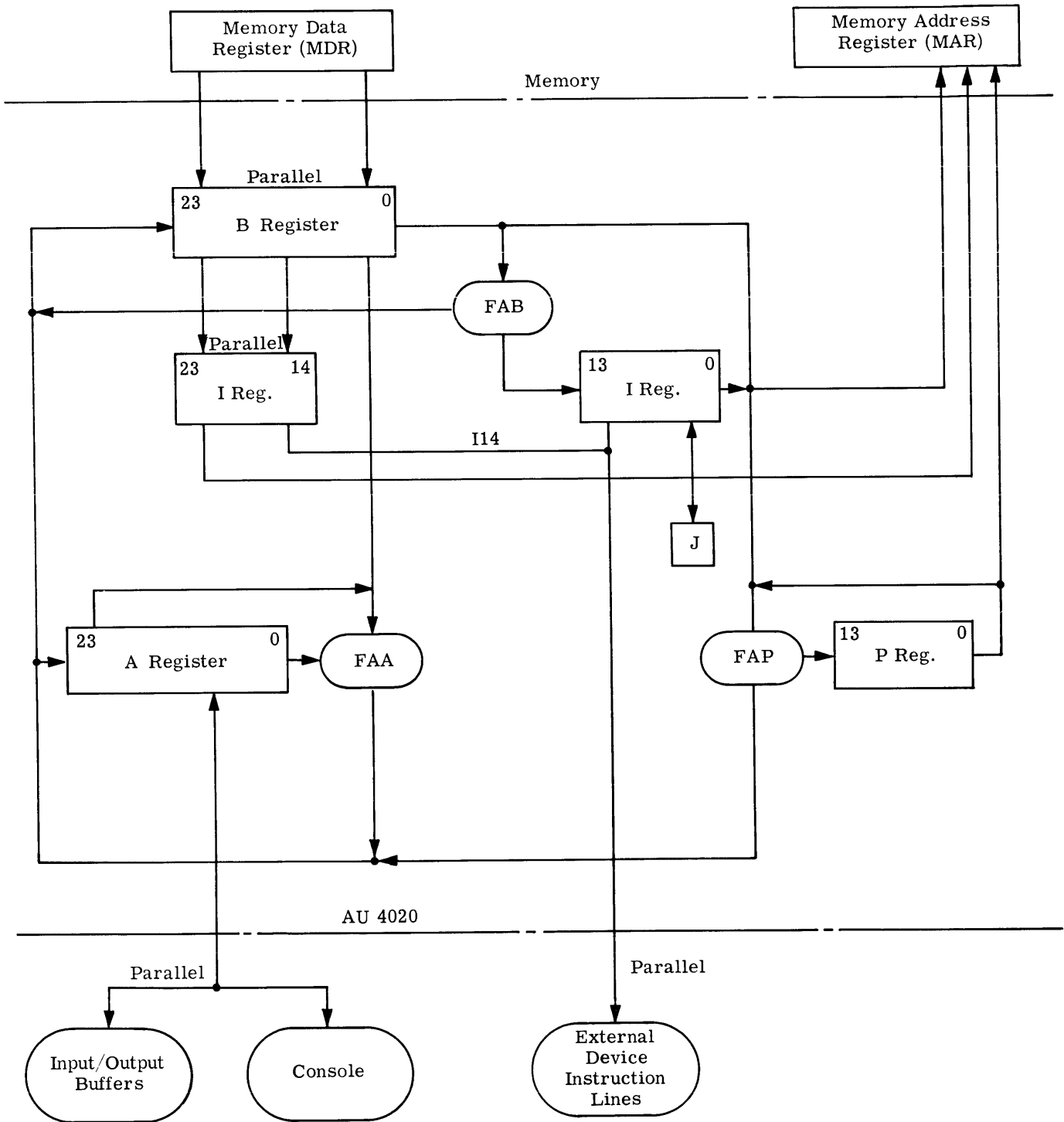


Figure III-1. Simplified Block Diagram of AU 4020

INSTRUCTION FORMATS

The GE/PAC 4000 instruction formats are illustrated below:

	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FULL OPERAND Hardware	ØP				X	*	Y																	
GEN 1	0 0 0 1 0 1				X	G										K								
GEN 2	0 1 0 1 0 1				X	S			D															
GEN 3	1 0 0 1 0 1				X	G										K								

ØP	Instruction Octal	G	Micro-coded GEN 1 Subcommand
X	Index Word Address	K	Bit Position of A-Register or Length of Shift
*	Relative Addressing Indicator	S	GEN 2 Subcommand
Y	Operand Address (a few instructions will use Y as the Operand; others will use Y as the address of the Operand)	D	I/O Device Address

Bit position 23 of the Full Operand formats above is used to designate either a hardware or quasi-implemented instruction.

These formats will assist in clarifying the following discussions on programming features.

Figure III-2.

The full-operand type of instruction is used to perform arithmetic operations, logical operations, index control operations, and data transfers to and from memory. Data transferred to memory may be internal with the AU or may come from input/output equipment. Data transferred from memory may go to the AU internal or be transferred through the AU to the input/output equipment. In GE/PAC 4040 systems, full-operand type instructions having bits 23 = 1 are implemented as quasi-instructions. In GE/PAC 4050/4060 systems, only those full-operand instructions having bits 23-21 = 111 are quasi-instructions.

Quasi-instruction types are extended-function commands that provide operations not wired into the hardware of the AU. They are implemented by the use of packaged subroutines. The defined operation is analogous to what may be called a Save I and Branch instruction.

What is meant by Save I and Branch may be best seen by analyzing what takes place when a quasi instruction is implemented. The quasi instruction is transferred from memory to B register, thence to the I register to be decoded. The operand, bits 12-0, are stored in core cell 002 or, if desired, are modified, then stored (Save I). The next instruction is addressed from bits 23-18 which selects one core cell from locations 040-077. The instruction contained in one of these cells "branches" the sequential program into a subroutine. Quasi instructions have the same instruction format as do full-operand instructions but are distinguished from them by the 1 bit placed in position 23. Since bits 23-18 are used to address the next instruction, the 1 bit in position

23 insures that the core cell addressed is located between 040 and 077. Examples of quasi instructions employed by this system are: Multiply, Divide, Floating Point Arithmetic, and those instructions involving the use of double-register lengths.

GEN 1 instructions are used for bit manipulations of the A register. By controlling the operation of Full Adder A, individual bits of the A register may be shifted in position to the right, masked by ones or zeros, tested for polarity, or counted for numbers of ones or zeros contained therein. Micro-coding of the instruction may be manipulated to perform many functions on the A register.

GEN 2 instructions are employed by the GE/PAC system to select modules and devices in the input/output equipment. The format is microcoded so as to have unique addressing bits necessary to select the devices and modules. GEN 2 instructions are only partially decoded in the AU and the selection portion of its bit configuration (bits 14-0) are never decoded in the AU but are instead transferred to the input/output equipment for device and module selection.

GEN 3 instructions are used for shifting the bits of the A register to the left and for shifting the combined A and Q registers in either direction. Microcoding of bits 14-5 selects the exact shift desired. In GE/PAC 4040 systems, GEN 3 instructions are quasi-instructions. In GE/PAC 4050/4060 systems, they are hardware implemented instructions.

All four of the instruction types may be modified. Modification is accomplished in two ways:

(1) indexing and (2) relative addressing. Indexing may be performed on all the instruction types, but relative addressing is limited to use on Full Operand and Quasi instructions.

Indexing is accomplished by taking the three bits 17 through 15 and addressing a core cell in location 001 through 007. The contents of the selected core cell is added to bits 13-0 of the instruction.

Relative addressing, as well as indexing, augments bits 13-0 of the instruction (the operand in the case of the Full Operand and Quasi instructions.) Modification takes place by adding the core cell address of the instruction to the instruction operand. The core cell address for most instructions is the bit information contained in the P register.

Instruction sequencing for the AU is as follows: The P register transfers information to the memory address register (MAR) and a core cell is addressed. The instruction contained therein is transferred, in parallel, via the memory data register (MDR) to AU buffer register B. B (23-14) is transferred, in parallel, to the I register for instruction decoding. B(13-0) is transferred, in serial, through Full Adder B to I (13-0). If desired, the contents of I are modified at this point.

Depending on the instruction decoded, the contents of the I register, modified or unmodified, will address memory and fetch data for arithmetic and logical operations, transfer data to or from memory, transfer data to or from the input/output equipment, or select modules and devices within the input/output equipment.

PROGRAMMING INSTRUCTIONS

The GE/PAC 4000 System has over 100 instructions used to control the operation of the computer and its various input/output devices. This section of the Programming Techniques Manual describes each of these instructions in detail and includes illustrations of their usage.

Each instruction is represented within the central processor by a specific pattern of 24 binary digits (bits), which when brought into the I register is decoded to control the required sequence of internal operations, but for programming convenience each instruction has been assigned a three-letter mnemonic code. Many of the instructions require, and include in their bit patterns, the specification of one or more operands such as the memory location (address) of a piece of data, a constant indicating the number of places a number is to be shifted within a register, or

which index location is to be used to modify an instruction. Each different kind of operand has been assigned a specific letter for convenience in describing the instructions.

The instruction descriptions which follow consist of two parts. The heading is a brief symbolic description made up of the mnemonic code, the letters specifying the required operands, the execution time in microseconds, and the octal representation of the binary command as it exists in the computer. The execution time given includes the time required to extract the instruction itself from memory, place it in the I register, and decode it.

Below the heading description is a more detailed description of the effect produced by the execution of the instruction. These detailed descriptions make use of the following conventions and terminology.

Single-letter abbreviations refer to registers or, in the case of the letter Y, to an operand address. For example, A refers to the A register.

Single letters in parentheses preceded by the letter C refer to the contents of the register specified by the letter within the parentheses. For example, C(A) is to be read as "the contents of the A register."

Subscripts are used to refer to only part of a register or of the contents of a register. For example, I₀₋₁₃ should be read as "bits 0 through 13 of the I register." C(I)₀₋₁₃ should be read as "the contents of bits 0 through 13 of the I register." A_{S, 22-18} is equivalent to A₂₃₋₁₈.

The word "cleared", when used in reference to a register or part of a register, means that the contents of the specified register or part of a register are reset to zero.

In all instructions involving the extraction of a word from storage, the word in storage remains unchanged. Likewise, in all instructions involving the transfer of information from one register to another or to storage, the contents of the register from which the information is transferred are unchanged unless the instruction description specifically states otherwise.

Unless the instruction description specifically states otherwise, all instructions may be automatically modified by indexing or by having the relative bit (14) set to give an effective memory address Z.

The following table summarizes the use of the various letter designations for registers in the instruction descriptions:

<u>Letter</u>	<u>Designation</u>	<u>No. Bits</u>	<u>Bit Positions</u>
A	Primary Arithmetic Register	24	s, 22-0
Q	Auxiliary Arithmetic Word	24	s, 22-0
J	Counter	5	4-0
N	Peripheral Register	7	6-0
X†	Index Word Address	3	13-0
I	Instruction Register	24	23-0
Y	Operand Address or Address of Operand Address		
K	Operand		
D	Address of Input/Output Device		
P	Program Address Register	14	13-0
C	Converter Register	12	14
*	Relative Addressing Indicator	1	14
TSTF	Test flip-flop	1	
PAIF	Permit-automatic interrupt flip-flop	1	
OVRF	Overflow flip-flop	1	

†Actually designates specific memory location rather than a register.

INTERNAL EFFECT INSTRUCTIONS

Internal effect instructions are those that control the operation of all of the internal registers and components of the GE/PAC 4000 Process Computer System.

1. DATA TRANSFER AND ARITHMETIC INSTRUCTIONS

Data transfer and arithmetic instructions facilitate the movement of data between core storage and the registers in the arithmetic unit. These instructions require an operand address to specify the particular place in core storage that contains the data that is to be used as the operand of this instruction. The octal operation code shown in the heading of each instruction is only two octal digits and is representative of bits positions 23 through 18 of the instruction, see I register format page.

A. Data Transfer Instruction Definitions

		MICROSECONDS	OCTAL CODE
LDA	Y	16	00
LOAD A. C(Y) Replace C(A). Y is not changed.			
STA	Y	16	32
STORE A. C(A) Replace C(Y). A is not changed.			
DLD	Y	125	41
DOUBLE LENGTH LOAD. The (C(Y) and C(Y+1) replace C(A) and C(Q). Y and Y+1 are unchanged.			
DST	Y	132	63
DOUBLE LENGTH STORE. C(A) and C(Q) replace C(Y) and C(Y+1). A and Q are unchanged.			

LDQ	Y	132	42
LOAD Q. C(Y) Replace C(Q). Y is not changed.			
STQ	Y	132	44
STORE Q. C(Q) Replace C(Y). Q is not changed.			
LDI	Y	162	52
LOAD A INDIRECT. C(Y) ₀₋₁₃ refer to a memory location Z. The contents of this memory location Z replaces the contents of A. Y and Z are not changed. Indexing and relative addressing if used are completed prior to fetching (Y). If C(Y) ₁₈ is a 1, Y + C(Y) ₀₋₁₃ is the address of Z.			
STI	Y	153	53
STORE A INDIRECT. C(Y) ₀₋₁₃ refer to a memory location Z. C(A) are stored in memory location Z. The C(A) are not changed. Indexing and relative addressing, if used, are completed prior to fetching (Y). If C(Y) ₁₈ is a 1, Y + C(Y) ₀₋₁₃ is the address of Z.			
OOM	Y	256	62
OPERATE ON MEMORY. Memory location Y functions as the A register for the instruction in the first location following OOM.			

B. Fixed-Point Arithmetic Instruction Definitions

The capacity of the A register may be exceeded in the execution of add and subtract instructions, resulting in a condition known as "overflow". When this happens, the overflow indicator is turned on, the high order (most significant) bit of the result is lost, and the sign of the result is reversed. This overflow condition may be sensed by the program and the result corrected. (Sensing is described under JNO instruction.)

ADD	Y	16	11
ADD. C(Y) are added algebraically to C(A). The result is placed in A. C(Y) are unchanged.			
SUB	Y	16	31
SUBTRACT. C(Y) are algebraically subtracted from C(A). The result is placed in A. C(Y) are unchanged.			
DAD	Y	284	51
DOUBLE LENGTH ADD. C(Y) and C(Y+1) ₂₂₋₀ are algebraically added to C(A) and C(Q) ₂₂₋₀ . The result is placed in A and Q ₂₂₋₀ . The sign of Q is set to zero. C(Y) and C(Y+1) are unchanged.			
DSU	Y	298	61
DOUBLE LENGTH SUBTRACT. C(Y) and C(Y+1) ₂₂₋₀ are algebraically subtracted from C(A) and C(Q) ₂₂₋₀ . The result is placed in A and Q ₂₂₋₀ . The sign of Q is set to zero. C(Y) and C(Y+1) are unchanged.			

MPY	Y	1951	55
-----	---	------	----

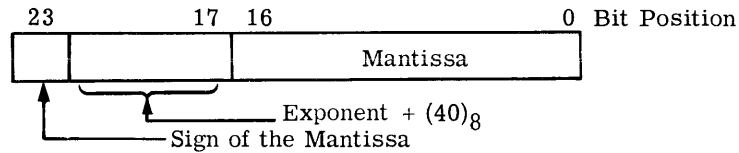
MULTIPLY. C(Y) are algebraically multiplied by C(Q). The result is placed in A and Q₂₂₋₀ with the most significant half in A. The sign of Q is set to zero. If C(A) are not set to zero before the MPY command is given, C(A) are added algebraically to the least significant half of the product. Thus, with proper scaling, it is possible to form the value $ab + c$. C(Y) are unchanged.

DVD	Y	5350	65
-----	---	------	----

DIVIDE. C(A) and C(Q)₂₂₋₀ are algebraically divided by C(Y). The quotient is placed in Q; the remainder is placed in A. The sign of the remainder is the sign of the divisor. The magnitude of the divisor must be greater than the magnitude of A. If not, the overflow indicator is turned ON. C(Y) are unchanged. The sign of the A register applies to the whole dividend.

C. Floating-Point Arithmetic Instruction Definitions

Single-word (24 bit) floating-point arithmetic instructions are provided with the following data format:



The overflow flip-flop (OVRF) is set if arithmetic overflow occurs (the magnitude of the exponent exceeds 5 bits, i. e., numbers larger or smaller than 2^{31}).

FLO	K	460	7400
-----	---	-----	------

FLOAT-FIXED NUMBER. C(A) are converted from a fixed-point number with scale factor K to a normalized floating-point number.

FIX	K	408	7402
-----	---	-----	------

FIX-FLOATING NUMBER. C(A) are converted from a floating-point number to a fixed-point number with a scale factor of K.

FAD	Y	1060	70
-----	---	------	----

FLOATING ADD. C(Y) are added to C(A). The normalized result is placed in A. C(Y) are unchanged.

FSU	Y	1090	71
-----	---	------	----

FLOATING SUBTRACT. C(Y) are subtracted from C(A). The normalized result is placed in A. C(Y) are unchanged.

FMP	Y	1523	72
-----	---	------	----

FLOATING MULTIPLY. C(Y) are multiplied by C(A). The normalized result is placed in A. C(Y) are unchanged.

FDV	Y	2993	73
-----	---	------	----

FLOATING DIVIDE. C(A) are divided by the C(Y). The normalized quotient is placed in A. C(Y) are unchanged.

2. REGISTER MANIPULATION INSTRUCTIONS

The register manipulation instructions are used to transfer information from one register to another, change the contents of a particular register, or otherwise cause actions in the registers that do not involve operands from core storage. Since no operand address need be specified, bits 23 through 18 and 13 through 0 can specify the operation code. The octal code in the heading of these instructions is given as an 8-digit octal number equivalent to the 24 binary bits of the instruction. Instructions with the operand K may be index modified.

Register Manipulation Instruction Definitions

LDK	K	97	4000K K K K
LOAD A WITH K. C(A) ₁₃₋₀ are replaced by K. C(A) ₂₃₋₁₄ are set to zero.			
AKA	K	127	6000K K K K
ADD K TO A. K is added to the C(A). The OVRF flip-flop will be set if overflow occurs.			
SKA	K	157	5000K K K K
SUBTRACT K FROM A. K is subtracted from C(A). The OVRF flip-flop will be set if overflow occurs.			
MAQ		176	45004330
MOVE A TO Q. C(A) replace C(Q). Zeros replace C(A).			
LDZ		16	05000000
LOAD ZERO INTO A. A is loaded with zeros.			
LDO	K	16	0500300K
LOAD ONE INTO BIT K OF A. All other bits are set to zero.			
LMO		16	05060000
LOAD MINUS ONE INTO A. A is loaded with "1's".			
ADO	K	16	0500700K
ADD ONE TO BIT K. Plus one is added algebraically to bit K. The overflow Flip-Flop is not affected by this instruction.			
CPL		16	05010000
COMPLEMENT A. Each bit in A is inverted; that is, each "1" is replaced by a zero and each zero is replaced by "1".			
NEG		16	05013000
NEGATE A. The 2's complement (negative value) of C(A) replaces C(A).			

NOP	Y	16	2620000
-----	---	----	---------

NO OPERATION. No operation is performed.

3. LOGICAL INSTRUCTIONS

Logical instructions are used to extract, compare, combine, or otherwise logically operate on information. Like the arithmetic instruction, they require data from core storage in the form of comparison constants, extraction mask, etc. They must then include an operand address within the instruction.

A. Logical Word Instruction Definitions

ORA	Y	16	21
-----	---	----	----

OR Y INTO A. Each bit of Y is examined. If there is a "1" bit in Y in a given position, a "1" bit is placed in A in that position. C(Y) and the other bit positions of A are unchanged.

ANA	Y	16	20
-----	---	----	----

AND Y TO A. Corresponding bits of A and Y are compared. If the corresponding positions in both A and Y contain a "1", a "1" is placed in that position of A. If either contain a zero, a zero is placed in that position of A.

ERA	Y	16	10
-----	---	----	----

EXCLUSIVE OR TO A. Corresponding bits of A and Y are compared. If the corresponding positions in A and Y are alike, a zero is placed in that position of A. If they are unlike, that position is set to a "1".

B. Logical Bit Instruction Definitions

SBK	K	16	0504600K
-----	---	----	----------

SET BIT K. Bit K of A is set to one. All other bits remain unchanged.

RBK	K	16	0504500K
-----	---	----	----------

RESET BIT K. Bit K of A is set to zero. All other bits remain unchanged.

CBK	K	16	0504700K
-----	---	----	----------

CHANGE BIT K. Bit K of A is complemented. All other bits remain unchanged.

IBK	K	16	0500100K
-----	---	----	----------

ISOLATE BIT K. Bit K of A is unchanged and all other bits of A are set to zero.

LBM	K	16	0506300K
-----	---	----	----------

LOAD BIT MASK. Bit K of A is set to zero and all other bits of A are set to one.

4. SHIFT INSTRUCTIONS

The shift instructions shift the contents of the A register to the right or left serially (bit by bit) either alone or with C(Q).

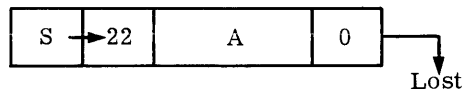
Shift instructions take advantage of a micro-coding technique in specifying the instruction. Bits 23 through 18 of shift instructions specify the standard shift code and bits 14 through 5 are assigned special significance to further define the exact type of shift to be accomplished.

The normal shift instructions have been assigned mnemonics as described in the following section, but for special operations the programmer may combine the micro-coding bits to specify a very special kind of shift instruction. These infrequently used special instructions have not been assigned mnemonics. Mnemonics may be assigned by a special DEF (Define A NEW OPERATION) pseudo instruction in the assembly (PAL) program.

Instruction Definitions

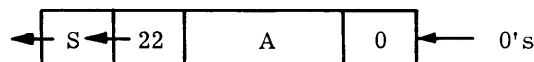
SRA	K	16	0501404K
-----	---	----	----------

SHIFT RIGHT ARITHMETIC. C(A)₂₂₋₀ are shifted right K places. If A is plus, zeros are inserted in the vacated positions. If A is minus, "1's" are inserted in the vacated positions. Bits shifted out of position 0 are lost. The sign of A is unchanged.



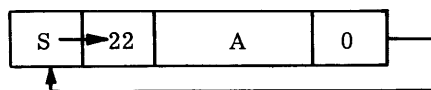
SLA	K	833	4500204K
-----	---	-----	----------

SHIFT LEFT ARITHMETIC. C(A)₂₃₋₀ are shifted left K places. Vacated positions of A are filled with zeros. If the original sign of A is positive, the overflow indicator will be turned on if a "one" bit is shifted into A_S. If the original sign of A is negative, the overflow indicator will be turned on if a "zero" bit is shifted into A_S.



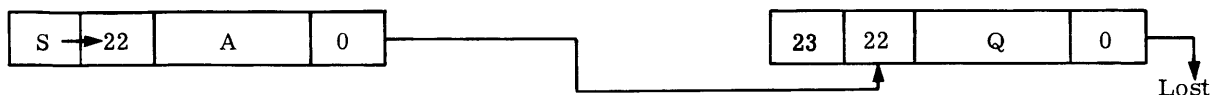
SRC	K	16	0500404K
-----	---	----	----------

SHIFT RIGHT CIRCULAR. C(A)₂₃₋₀ are shifted right K places in a circular fashion; that is, the bit shifted out of position 0 is inserted in position 23, replacing the bit shifted out of position 23.



DRA	K	694	4500440K
-----	---	-----	----------

DOUBLE RIGHT ARITHMETIC. C(A)₂₂₋₀ and C(Q)₂₂₋₀ together are shifted K places to the right. Bits shifted out of A₀ shift into Q₂₂. Bits shifted out of Q₀ are lost. If the sign of A is plus, zeros fill the vacated positions; if the sign of A is minus, "1's" fill the vacated positions. Q₂₃ is set to zero. The sign of A is unchanged. If shift is greater than 23 the timing is 1732 microseconds.



DLA

K

1004

4500644K

DOUBLE LEFT ARITHMETIC. $C(A)_{22-0}$ and $C(Q)_{22-0}$ together are shifted K places to the left. Bits shifted out of Q_{22} shift into A_0 . The vacated positions of Q are filled with zeros. If the original sign of A is positive, the overflow indicator will be turned on if a "one" bit is shifted into A_S . If the original sign of A is negative, the overflow indicator will be turned on if a "zero" bit is shifted into A_S . Q_{23} is set to zero. If shift is greater than 23 the timing is 2351 microseconds.



DRC

K

838

4500530K

DOUBLE RIGHT CIRCULAR. $C(A)_{23-0}$ and $C(Q)_{23-0}$ together are shifted K places to the right in a circular fashion. Bits shifted out of A_0 shift into Q_{23} and those from Q_0 shift into A_{23} . If shift is greater than 23 the timing is 2017 microseconds.



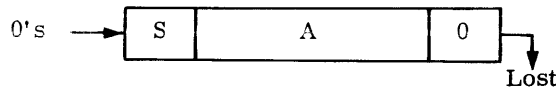
SRL

K

15

0500004K

SHIFT RIGHT LOGICAL. A_{23-0} are shifted K places to the right. Zeros are shifted in through the high-order part of the register. Bits shifted out of A_0 are lost.



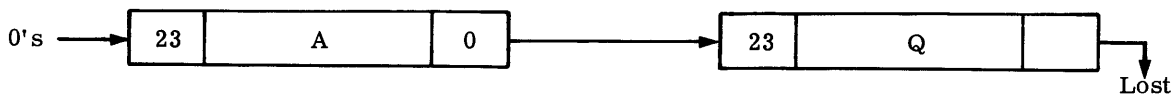
DRL

K

776

4500430K

DOUBLE RIGHT LOGICAL. A_{23-0} and Q_{23-0} are shifted K places to the right. Zeros are shifted in through the high-order part of the A register. A_0 shifts into Q_{23} . Bits shifted out of Q_0 are lost. If the shift is greater than 23 the timing is 1902 microseconds.



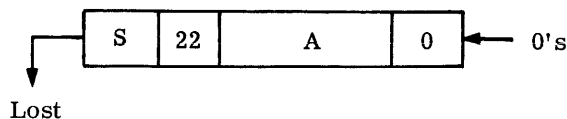
SLL

K

680

4500200K

SHIFT LEFT LOGICAL. A_{23-0} are shifted K places to the left. Zeros are shifted in through the low-order part of the register. Bits shifted out of the high-order of the register are lost. The overflow flip-flop is not set by this shift.



DLL

K

751

4500720K

DOUBLE LEFT LOGICAL. A_{23-0} and Q_{23-0} are shifted K places to the left. Zeros are shifted in through the low-order part of the Q register. Bits shifted out of the high-order part of the A register are lost. The overflow flip-flop is not set by this shift. If shift is greater than 33 the timing is 1846 microseconds.



5. BRANCH INSTRUCTIONS

Branch instructions are used to transfer control to instructions not directly in sequence. Unconditional branch instructions transfer control to the indicated instruction directly. Conditional instructions test some condition in the computer to determine which of two instructions specified to transfer control to. All test instructions covered in the next group set or reset the Test Flip-Flop (TSTF).

A. Unconditional Branch Instructions

The special "store the contents of the P register and branch unconditionally" instruction is defined in section 6.

BRU

Y

14

14

BRANCH UNCONDITIONALLY. Control is transferred to the instruction located at Y . Y becomes the address of the next instruction and is transferred from I_{13-0} to P_{13-0} .

B. Conditional Branch Instructions

A conditional branch instruction will transfer control to either the address specified or the next instruction. A jump instruction will transfer control to either the first or the second sequential instruction after the conditional branch instruction.

BTS

Y

14

34

BRANCH IF TSTF SET. Control is transferred to the instruction located at Y if the TSTF is set. If the TSTF is reset, control is transferred to the next instruction.

BTR

Y

14

30

BRANCH IF TSTF RESET. Control is transferred to the instruction located at Y if the TSTF is reset. If the TSTF is set, control is transferred to the next instruction.

JND

32

25040000

JUMP IF NO DEMAND. The demand button on the computer console sets the demand flip-flop. JND transfers control to the next location and resets the Demand Flip-Flop if set. If the flip-flop is reset, control is transferred to the second sequential location.

JNO

32

25060000

JUMP IF NO OVERFLOW. JNO tests the overflow flip-flop (OVRF). If set, the flip-flop is reset and control is transferred to the next instruction. If reset, control is transferred to the second sequential location.

JNP	32	25070000
-----	----	----------

JUMP IF NO PARITY ERROR. JNP tests the core parity flip-flop. If set, the flip-flop is reset and control is transferred to the next instruction. If reset, control is transferred to the second sequential location.

JNR	D	32	25060000
-----	---	----	----------

JUMP IF DEVICE D NOT READY. Control is transferred to the next sequential instruction if device D is ready. If device D is busy, control is transferred to the second sequential location.

JNE	D	32	25070000
-----	---	----	----------

JUMP IF DEVICE D NOT IN ERROR. Control is transferred to the next sequential instruction if an error on device D has occurred. If no error, control is transferred to the second sequential location.

6. TEST INSTRUCTIONS

Program test instructions cause the test flip-flop (TSTF) to be set and/or reset according to the instruction. Three types of instructions are available: Word tests, partial word tests and bit tests. The partial word tests make use of the J counter.

SET	16	05004637
-----	----	----------

SET TSTF. The test flip-flop TSTF is set.

RST	16	05004737
-----	----	----------

RESET TSTF. The test flip-flop TSTF is reset.

A. Word Tests.

RNZ	16	05004470
-----	----	----------

RESET TSTF IF A IS NONZERO. The TSTF is reset if any bit in the A register is a one. The TSTF is unchanged if all bits are zero.

SNZ	16	05004570
-----	----	----------

SET TSTF IF A IS NONZERO. The TSTF is set if any bit in the A register is a one. The TSTF is unchanged if all bits are zero.

TNZ	16	05004770
-----	----	----------

TEST A NONZERO. The TSTF is set if any bit in the A register is a one. The TSTF is reset if all bits are zero.

TZE	16	05004670
-----	----	----------

TEST A ZERO. The TSTF is set if all bits in the A register are zero. The TSTF is reset if any bit is a one.

TNM	16	05070770
-----	----	----------

TEST NOT MINUS ONE. The TSTF is set if all bits in the A register are ones. The TSTF is reset if any bit is a zero.

TZC		16	05064670
TEST ZERO AND COMPLEMENT. The TSTF is set if all bits in the A register are zeros. The TSTF is reset if any bit is a one. The contents of the A register is then replaced by its complemented value.			
B. Partial Word Tests.			
Counting of the most and least significant bits in the A register must be followed by a Load X with count (LXC) command before another GEN 1 instruction is executed.			
CLZ		16	05070137
COUNT LEAST SIGNIFICANT ZEROS. The number of zero bits to the rightmost one bit in the A register are counted. The count is placed in the J counter.			
CLO		16	05004137
COUNT LEAST SIGNIFICANT ONES. The number of one bits to the rightmost zero bit in the A register are counted. The count is placed in the J counter.			
CMO		16	05004237
COUNT MOST SIGNIFICANT ONES. The number of one bits to the leftmost zero bit in the A register are counted. The count is placed in the J counter.			
CMZ		16	05070237
COUNT MOST SIGNIFICANT ZEROS. The number of zero bits to the leftmost one bit in the A register are counted. The count is placed in the J counter.			
LXC	X	21	17
LOAD X WITH COUNT. The C(J) replace the C(X) ₄₋₀ . The C(X) ₂₃₋₅ are set to zero. X is a specified index word 1 to 7. This instruction must follow each count instruction before another GEN 1 instruction is executed.			
TSC	K	16	0500464K
TEST AND SHIFT CIRCULAR. The C(A) ₂₃₋₀ are shifted right circular K places. If all K bits shifted out of A ₀ are zero, the TSTF is set. If a one bit is shifted out of A ₀ the TSTF is reset.			
C. Bit Tests.			
Logical operations based upon true or false conditions may be programmed in a direct manner with the bit test instructions.			
TEV	K	16	0507070K
TEST BIT K EVEN. The TSTF is set if bit K in the A register is zero. The TSTF is reset if bit K is one.			
TOD	K	16	0500470K
TEST BIT K ODD. The TSTF is set if bit K in the A register is one. The TSTF is reset if bit K is zero.			
SEV	K	16	0507050K
SET TSTF IF BIT K IS EVEN. The TSTF is set if bit K in the A register is zero. The TSTF remains unchanged if bit K is one.			
SOD	K	16	0500450K
SET TSTF IF BIT K IS ODD. The TSTF is set if bit K in the A register is one. The TSTF remains unchanged if bit K is zero.			

REV	K	16	0507040K
RESET TSTF IF BIT K IS EVEN. The TSTF is reset if bit K in the A register is zero. The TSTF remains unchanged if bit K is one.			
ROD	K	16	0500440K
RESET TSTF IF BIT K IS ODD. The TSTF is reset if bit K in the A register is one. The TSTF remains unchanged if bit K is zero.			
TOR	K	16	0504570K
TEST ODD AND RESET BIT K. The TSTF is set if bit K in the A register is a one. Bit K is then set to zero in the A register. The TSTF is reset if bit K is a zero.			
TER	K	16	0504560K
TEST EVEN AND RESET BIT K. The TSTF is set if bit K in the A register is a zero. Bit K is then set to zero in the A register. The TSTF is reset if bit K is a one.			
TES	K	16	0504660K
TEST EVEN AND SET BIT K. The TSTF is set if bit K in the A register is a zero. Bit K is then set to one in the A register. The TSTF is reset if bit K is a one.			
TOS	K	16	0504670K
TEST ODD AND SET BIT K. The TSTF is set if bit K in the A register is a one. The TSTF is reset if bit K is a zero. Bit K is then set to one in the A register.			

7. AUTOMATIC ADDRESS MODIFICATION AND LOOP CONTROL INSTRUCTIONS

Data and Instructions appear in storage as combinations of binary digits. This allows the arithmetic unit to perform arithmetic functions on instructions as well as data. When instructions are changed by arithmetic operations, the change is referred to as address modification. When automatic modification of instructions in the I register takes place using one of the address modification locations, it is referred to as automatic address modification.

The automatic address modification instructions operate in conjunction with core locations 0001 to 0007. These are called X locations 1 to 7. In all instructions, except INX, LXX, LXC, LDX, STX, and TXH, bits 15, 16 and 17 indicate whether or not automatic address modification is to take place. If bits 15, 16 and 17 are zero, no address modification will take place. If bits 15, 16 and 17 are non-zero, the instruction will be modified in the I register before it is executed. This modification consists of the addition of a portion of the contents of the X location to the instruction in the I register. Bit 14 is used to indicate relative addressing. If both bit 14 and bits 15, 16 and 17 are set, indexing will take place before relative addressing. When automatic address modification is called for, an additional 7 microseconds is added to the normal instruction execution time.

All seven X locations 1, 2, 3, 4, 5, 6 and 7 may be used in conjunction with INX, LDX, STX, TXH, and LXX instructions. All X locations may therefore be incremented and tested to accomplish counting or tallying. In the seven instructions bits 15, 16 and 17 are used to specify which of the seven X locations is to be used in the execution of the instruction. The DMT instruction may be used with the X locations or any other memory locations.

Instruction Definitions

LDX	Y, X	21	16
LOAD X LOCATION FROM Y. The C(Y) ₀₋₂₃ replace the C(X) ₀₋₂₃ . C(Y) are unchanged.			
STX	Y, X	21	06
STORE X LOCATION INTO Y. The C(X) ₀₋₂₃ replace the C(Y) ₀₋₂₃ . C(X) are unchanged.			

ABT	D	32	25030000
<p>ABORT DEVICE D OPERATION. If device D is in operation, ABT causes its operation to terminate at the earliest possible instant (normally within 100 ms). Device D's operation may or may not be completed. The JNR instruction is used to detect completion of operation termination. If device D is not in operation, ABT is ignored.</p>			
ACT	D	32	25010000
<p>ACTIVATE DEVICE D. If device D is not in operation, ACT turns its ready signal off for 8 microseconds. This will initiate an automatic priority interrupt. If device D is in operation, ACT is ignored.</p>			
IN	D	32	25050000
<p>INPUT FROM DEVICE D. Data from device D is transferred into A. The next operation of device D is initiated.</p>			
OPR	D	32	25020000
<p>OPERATE DEVICE D. Enables power at peripheral device D.</p>			
OUT	D	32	25040000
<p>OUTPUT TO DEVICE D. Data from the A register is transferred to device D. One operation of device D is initiated.</p>			
SEL	D	32	25000000
<p>SELECT DEVICE D. Connects device D input/output lines to its buffer register. SEL is not normally required to operate the standard devices connected to GE/PAC.</p>			
<h2>10. AUTOMATIC PROGRAM INTERRUPT INSTRUCTIONS</h2>			
<p>The GEN 2 instructions for controlling input and output also control certain computer actions.</p>			
IAI		32	25030000
<p>INHIBIT AUTOMATIC INTERRUPT. The PAIF is reset. Inhibitible interrupts are ignored until the next PAI instruction. Non-inhibitible interrupts are not affected.</p>			
PAI		32	25020000
<p>PERMIT AUTOMATIC INTERRUPT. The PAIF is set. Inhibitible interrupts are permitted.</p>			
SSA		32	25010000
<p>SET STALL ALARM. This instruction is used to periodically set a time delay device. When the time delay expires, an alarm condition prevails. The time delay device is manually adjustable for delays of from 1 to 5 seconds.</p>			
RCS		32	25050000
<p>READ CONSOLE SWITCHES. The contents of the console toggle switches replace the C(A). Switch down generates a one in A. Switch up generates a zero in A.</p>			

IV GE/PAC 4050/4060 SYSTEM

APPLICATION

In most process computer applications, less than 5 per cent of the instructions executed are multiply and divide arithmetic operations. Thus, the GE/PAC 4040 system was designed without the additional hardware for these instructions. These extended arithmetic operations are program implemented in the GE/PAC 4040 system by using a unique combination of hardware and programming to provide the quasi instruction.

The GE/PAC 4050/4060 System (Model AU 2 arithmetic and control unit with 5 and 2-microsecond memories) provides additional hardware implementation of the quasi instructions. The 4050/4060 System is used where greater computer power or faster arithmetic speeds are required.

Programs written for the GE/PAC 4040 System are completely compatible with the 4050 and 4060 Systems. The 4050/4060 System provides 13 additional instructions for faster input/output control, variable field arithmetic, and faster program linkage control.

FEATURES

1. The 4050/4060 Systems can use more than one core memory with both 5 microsecond, 2 microsecond or other speed memory in the same computer.
2. The normal operation times for most unmodified instructions in typical calculations is less than 12 microseconds or 6 microseconds for the 5 and 2-microsecond memories. Multiply time is 21.8 and 15 microseconds, respectively.
3. 2's complement arithmetic is employed.
4. 7 indexing address modification memory cells are provided.
5. Over 118 defined instructions.
6. 4 to 64K of core memory with direct and relative addressing, and with indirect addressing operations.
7. Four fixed-point modes of arithmetic operation; single, double, partial and variable register arithmetic.
8. Complete variable field operations.

9. Circular list techniques using real-time queuing list instructions.
10. High-speed input/output from memory under program control.
11. Powerful input/output controller instructions.
12. Memory fences for protection of memory.
13. Ability to treat each memory location as an accumulator.
14. Ability to store or load all working registers by a single command. This also permits block memory transfers.
15. Two formats for floating-point operation: 24-bit and 48-bit.
16. 6 input/output channels to arithmetic unit or to the accumulator.
17. Repeat instruction provides for single-instruction loop control.
18. Masked memory search capability.

DESCRIPTION

The Model AU 2 Arithmetic and Control Unit is comprised of data registers, and control flip-flops as shown in block diagram of Figure IV-1. The registers used are as follows: A, Q, B, I, H, P, J, and I/O Selector Hub. Names of the flip-flops and full adders employed are: Flip-Flops, Test (TSTF), Overflow (OVRF), Permit Automatic Interrupt (PAIF), Priority Interrupt First (PIIF), Execution (XECF), Demand (DEMF), and Carry--Full Adders "F", "N", "I", GEN 1, and Field. The parallel arithmetic unit in conjunction with the A and Q Registers provides high-speed arithmetic operations. The function of these elements is discussed below:

A Register - Accumulator for arithmetic operations and temporary storage for data coming from or going to the input/output equipment. 24 bits.

Q Register - Auxiliary accumulator for double-precision format. It holds the multiplier for multiplication. 24 bits.

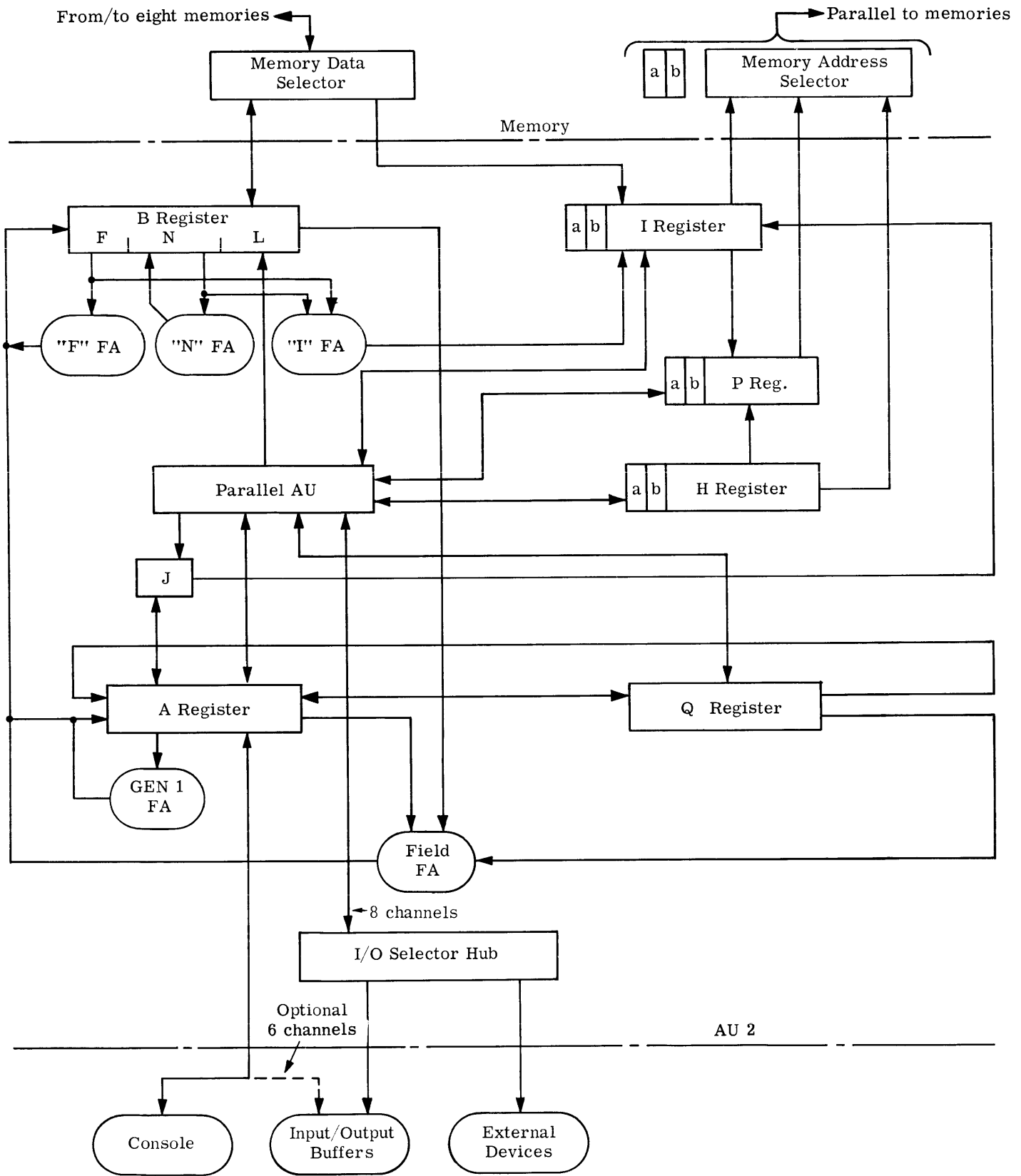


Figure IV-1. Simplified Block Diagram of AU 2

B Register - Memory buffer register for the arithmetic unit. 24 bits.

I Register - Instruction register for holding each command that controls the operation of the arithmetic unit and associated registers. 26 bits.

H Register - Holding register for repeated arithmetic operations and I/O addressing. 16 bits. See instruction RPT, LDR, and STR.

P Register - Place counter for controlling the location (place) of the next instruction to be executed. 16 bits.

J Counter - Used with logic commands to specify the bit to be operated on or to count one or zero bits. 5 bits.

I/O Selector Hub - Used to control input/output devices. From 8 to 24 bits. Provides memory to peripheral communication through the AU. See instructions LDB and STB.

"F" Field, "N" Field and "T" Field full adders - Used for linear stacking, circular lists and real

time queuing lists. See instructions IDL, ODL, ABL, AEL, RBL and REL.

GEN 1 full adder - Used for bit manipulations of the A register.

Field full adder - Provides variable field arithmetic for loading, adding, subtracting, testing and storing pieces of information contained within a data word (packed data). See instructions LDF, AFA, SKA, TFE, TFL, and STF.

The instruction format for the GE/PAC 4060 is identical to that of the GE/PAC 4040 shown in Figure III-2.

INSTRUCTIONS

All of the instructions given in Section III for the GE/PAC 4040 apply to the GE/PAC 4060. The execution times are given in the Instruction Summary in Section XIV. The following instructions apply only to the GE/PAC 4060 (4025 Arithmetic and Control Unit).

1. VARIABLE FIELD ARITHMETIC

Mnemonic	Field	Time in Microseconds	Octal Code
LDF	Y	17.7/10.9	23
LOAD FIELD. The specified field of location Y is loaded into the corresponding location of A. The field is specified by ones in the Q register.			
AFA	Y	17.7/10.9	03
ADD FIELD TO A. The specified field of location Y is added to the corresponding field in the A register. The field is specified by ones in the Q register.			
SFA	Y	17.7/10.9	22
SUBTRACT FIELD FROM A. The specified field of location Y is subtracted from the corresponding field in the A register. The field is specified by ones in the Q register.			
TFE	Y	17.7/10.9	02
TEST FIELD EQUAL. The specified field of location Y is compared with the corresponding field in the A register. If the fields are equal, the TSTF is set. If the fields are not equal, the TSTF is reset. The field is specified by ones in the Q register.			
TFL	Y	17.7/10.9	12
TEST FIELD LESS. The specified field of location Y is compared with the corresponding field in the A register. If the field in A is less than the field in Y, the TSTF is set. If greater or equal, the TSTF is reset. The field is specified by ones in the Q register.			

Mnemonic	Field	Time in Microseconds	Octal Code
STF	Y	22.7/12.5	13

STORE FIELD. The specified field in the A register is stored in the corresponding field in location Y. The field is specified by ones in the Q register.

2. INPUT/OUTPUT CONTROL AND LIST COMMANDS

IDL	Y	60.9/37.1	67
-----	---	-----------	----

INPUT FROM DEVICE TO LIST. The list control word located at Y is examined. If the list is full, program control is transferred to the first sequential location. If the list is not full, the list control word Y is updated (Refer to AEL instruction for details). Data is transferred from the device specified by the device address in location Y-1 and appended to list Y as the new ending item. The next operation of the device is initiated and control is transferred to the second sequential location.

ODL	Y	60.9/37.1	66
-----	---	-----------	----

OUTPUT TO DEVICE FROM LIST. The list control word located at Y is examined. If the list is empty, control is transferred to the first sequential location. If the list is not empty, the list control word Y is updated. The beginning item of the list is removed and transferred to the device specified by the device address in location Y-1. The next operation of the device is initiated and control is transferred to the second sequential location.*

LDB	Y	10/32	37
-----	---	-------	----

LOAD HIGH-SPEED I/O BUFFER. C(Y) is transferred to an external buffer register previously selected and tested for ready. When this instruction is used with the repeat mode of operation (RPT command), data transfers can be made up to a 500KC rate with the GE/PAC 4060 and up to 200KC with the GE/PAC 4050.

STB	Y	10/32	36
-----	---	-------	----

STORE HIGH-SPEED I/O BUFFER. The contents of an external buffer register are stored in location Y. When this instruction is used with the repeat mode of operation (RPT command), data transfers can be made up to a 625KC rate with the GE/PAC 4060 and up to 200KC with the GE/PAC 4050.

3. PROGRAM SUBROUTINE LINKAGE INSTRUCTIONS

Real-time programming requires frequent loading and storing of arithmetic and index registers. The following instructions provide for loading and storing A, Q, X2, X3, X4, X5, X6, and X7 registers automatically.

LDR	Y	75/24	64
-----	---	-------	----

LOAD REGISTERS. The contents of memory locations Y, Y+1, Y+2, ... Y+7 are loaded into A, Q, X2, X3, X4, X5, X6, and X7 respectively. The contents of memory locations Y thru Y+7 are unchanged.

STR	Y	75/24	54
-----	---	-------	----

STORE REGISTERS. The contents of A, Q, X2, X3, X4, X5, X6 and X7 are stored in memory locations Y, Y+1, Y+2, ..., Y+7, respectively. The contents of A, Q, X2, X3, X4, X5, X6 and X7 are unchanged.

*Note: When used as an interrupt instruction, P is not changed, but if list is full (for an input) or empty (for an output), an "echo" is sent to the API module.

4. REPEAT MODE OF OPERATION

One-word loop operation, formerly required by programming technique, has been implemented into the capabilities of the GE/PAC 4000 hardware. This operation is provided in the following manner:

- A. The instruction to be repeated is executed the number of times as specified in an index register.
- B. Each time the instruction to be repeated is executed, the effective operand of that instruction is incremented by one.
- C. After the instruction is repeated, the loop counter is incremented, and the results of the execution are tested. If the proper test condition exists, the loop is exited.

This mode allows for loop operation (eliminating the instruction fetch time of the repeat instruction required. For example, to add 1000 numbers would require 1005 memory cycles, not 2000 or 3000 memory cycles as other computers might require without this feature.

Mnemonic	Field	Time in Microseconds	Octal Code
RPT	Y, X	25/10	27

REPEAT INSTRUCTION IN LOCATION Y. The repeat instruction executes the instruction located at Y as many times as specified by the contents of the X register plus one. The instruction located at Y must be indexed by the same index as the RPT instruction. The operand address of the instruction in Y is decremented by the contents of the X register to determine the first effective address of the RPT'ed instruction. The effective address is then incremented each time the object instruction is repeated. The last executed entry is that specified by the effective (non-indexed) address in location Y. The repeat function terminates when the TSTF changes; otherwise the RPT function terminates after the specified number of repeats.

V DATA MANIPULATION

GENERAL

The "bit" is the elemental unit of data. A larger and sometimes more convenient unit is the "word" which (in GE/PAC) consists of an ordered set of 24 "bits". GE/PAC is word organized with respect to addressing and arithmetic. It is partially bit organized with respect to logic.

Data manipulation is accomplished in one or more of the following registers.

- A - Register
- Q - Register (word location 10g in GE/PAC 4040)
- Addressed memory location Z

The group of "General" instructions provide the means to load or store each register. "Arithmetic," "Logic," and "Test" instructions implement data manipulation operations.

The A Register, being the primary working register, is affected by a majority of the data manipulation operations. The Q Register is used as an extension of the A Register or as an auxiliary register. Few operations affect it. The "Operate on Memory" instruction provides a means of applying all of the A Register operations to an addressed memory location.

DATA REPRESENTATION

Data representation is provided for two types of data:

1. Numeric variables
2. Logic variables

NUMERIC VARIABLES

Numeric variables are represented by sequences of digits. The GE/PAC word, with bits interpreted as digits, provides binary representation of any real number with a precision of 23 binary digits.

There are three common binary representations for negative numbers:

1. Sign plus absolute value
2. "one's" complement
3. "two's" complement

In each case, the left-most bit is interpreted as the sign of the number - - - -0 meaning +, and 1 meaning -

"Two's Complement" representation is used in GE/PAC for fixed point arithmetic.

"Sign plus absolute value" representation is used in GE/PAC for floating point arithmetic.

The integral and fractional portions of the number are separated by a "binary point". The binary point has no analogy in hardware. In the case of fixed point arithmetic, the binary point is supplied mentally by the programmer and is referred to as the "scale" of the number. In the case of floating-point arithmetic, the binary point is defined by an exponent within the floating-point number representation. The mantissa is always a normalized fraction. Arithmetic data formats are depicted in Figure V-1. Figure V-2 gives the number range of fixed and floating point variables.

Note on Numeric Data Representation¹

Any real number may be represented in binary notation as:

$$+ \dots + X_n 2^n + X_{n-1} 2^{n-1} + \dots + X_1 2 + X_0 + X_{-1} 2^{-1} + \dots + X_{-n} 2^{-n} + \dots$$

where the X_i are the "coefficients of the powers of 2" and may assume the values 0 and 1. If a one-to-one correspondence between these X_i and bits in computer words be made, then numbers may be represented by sequences of bits. GE/PAC with a 24-bit word provides sequence lengths of 24 and 48 bits and can approximate any real number with a precision of 23 (46) bits.

The two's complement of a number $S_{n+23} \dots X_{n+1} X_n$ is numerically equal to the number

$$\left[(1_{n+24} 0_{n+23} \dots 0) - (0_{n+24} S_{n+23} X_{n+22} \dots X_n) \right]$$

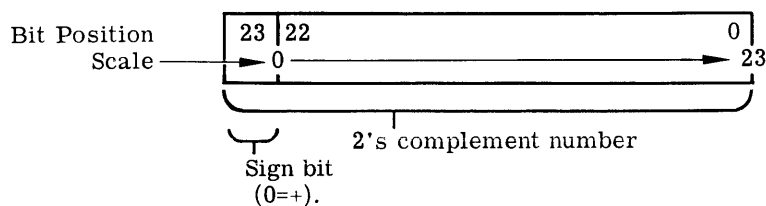
Thus $-7_{10} = 10000_2 - 00111_2 = 1001_2$,

$-(-1)_{10} = 10000_2 - 01111_2 = 0001_2$,

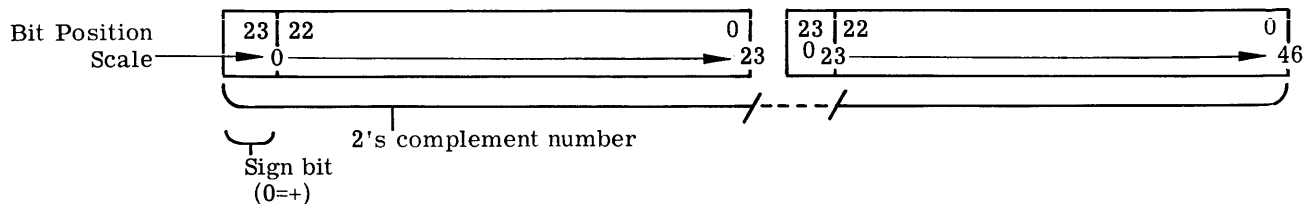
and $-0_{10} = 10000_2 - 00000_2 = 0000_2$

NOTE 1: The omission of this section from the programmers reading plan will not be detrimental to his understanding of real-time GE/PAC programming.

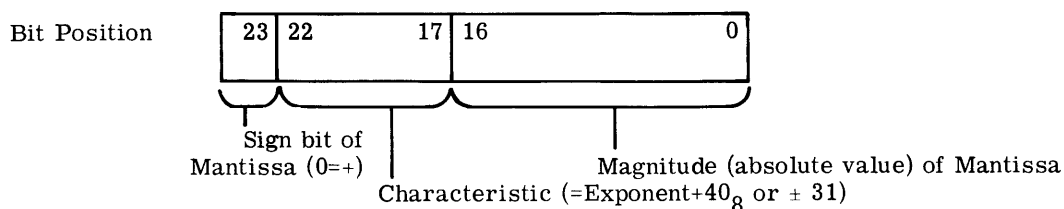
Single Word Fixed Point



Double Word Fixed Point



Single Word Floating Point



Double Word Floating Point

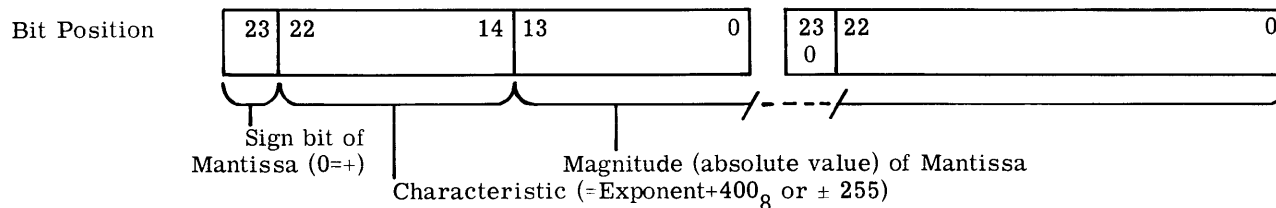


Figure V-1. Arithmetic Data Formats

The reader will discover that an alternate method to negate a number is to form the one's complement and add one.

$$\text{Thus } -7_{10} = \overline{0111}_2 + 0001_2 = 1000_2 + 0001_2 = 1001_2,$$

$$-(-1)_{10} = \overline{1111}_2 + 0001_2 = 0000_2 + 0001_2 = 0001_2,$$

$$\text{and } -0_{10} = \overline{0000}_2 + 0001_2 = 1111_2 + 0001_2 = 0000_2.$$

This latter method is used in GE/PAC since:

1. The former method requires an additional bit in the A Register.
2. The latter method treats subtraction as a special case of addition.

Another way to interpret a 2's complement number is to think of the "sign bit" as being the coefficient of a negative multiplier. All other multipliers are positive.

$$\text{Thus } -7 = 1001_2 = -8_{10} + 0_{10} + 0_{10} + 0_{10} + 1_{10},$$

$$1_{10} = 0001_2 = +0_{10} + 0_{10} + 0_{10} + 1_{10},$$

$$\text{and } -2_{10} = 1110_2 = -8_{10} + 4_{10} + 2_{10} + 0_{10}.$$

LOGIC VARIABLES

Yes/no, true/false, on/off, and set reset conditions are represented by Logic Variables and obey the laws of Boolean algebra. If a one-to-one correspondence between Logic Variables and bits be made.

FIXED POINT NUMBER RANGE

Format	Minimum Number	Maximum Number	Increment
<u>Single Word</u>			
As found in memory	40000000 ₈	37777777 ₈	00000001 ₈
Decimal equivalent	-2^s	$2^s - 1$	2^{s-23}
<u>Double Word</u>			
As found in memory	40000000 00000000 ₈	37777777 37777777 ₈	00000000 00000001 ₈
Decimal equivalent	-2^s	$2^s - 1$	2^{s-47}

Notes

- a. s = scale of fixed point number.
- b. $2^s = 10^{0.30s}$, approximately.
- c. Although the negative endpoint, -2^s , can be represented, its negative, 2^s , cannot. Therefore -2^s is not normally considered to be part of the range of a numeric variable.

FLOATING POINT NUMBER RANGE

Format	Minimum Number	Maximum Number	Increment
<u>Single Word</u>			
As found in memory	77777777 ₈	37777777 ₈	0c000001 ₈
Approximate decimal equivalent	-0.00000×10^9	$+0.99999 \times 10^9$	2^{c-49}
<u>Double Word</u>			
As found in memory	77777777 37777777 ₈	37777777 37777777 ₈	0cc00000 00000001 ₈
Approximate decimal equivalent	$-0.999,999$ $999,99 \times 10^{77}$	$+0.999,999$ $999,99 \times 10^{77}$	2^{cc-69}

Notes

- a. c = characteristic of floating number. Range of c is $0 \leq c \leq 63$ (cc with range $0 \leq cc \leq 511$, double word).
- b. The number "zero" is arbitrarily defined to be the number 00000000₈ (00000000, 00000000₈ for double word format) as it would occur in memory, and has the approximate decimal equivalent $+0.0 \times 10^{-9}$ ($+0.0 \times 10^{-77}$, double word).
- c. The number "minus zero", which is representable in "sign plus absolute value" notation can never result from a GE/PAC floating point operation and should be considered illegal.
- d. Note that increment is not a representable floating point number. The minimum positive representable number is 00200000₈ and has the approximate decimal equivalent $+0.5 \times 10^{-9}$ (00020000, 00000000₈ and $+0.5 \times 10^{-77}$, double word).

then a set of 24 such variables may be represented by a GE/PAC computer word. In such a representation, 1 usually means true and 0 means false.

GE/PAC provides Word Logical Instructions to operate on sets of bits, and Bit Logical Instructions to operate on individual bits.

ARITHMETIC OPERATIONS

ARITHMETIC OVERFLOW

The result of an arithmetic operation (addition, subtraction, multiplication, or division) upon two 24 bit numbers may exceed 24 bits. Any further computation involving the result is meaningless.

This fact is the primary source of grief in the programming of numerical problems. A condition called "arithmetic overflow" occurs when the magnitude of the resulting number is too large.

GE/PAC provides arithmetic overflow detection for all of its arithmetic instructions except ADO.

"Arithmetic underflow" occurs when a result is too small in magnitude to affect the least significant bit of the register in which the operation is performed. In case underflow occurs, the result is replaced by "0" but no underflow detection is provided.

If "overflow" occurred during the execution of an ADD, SUB, AKA, SKA, MPY, DVD, SLA, DLA, FIX, FLO, FAD, FSU, FMP, or FDV, the overflow flip-flop (OVRF) will be set. If "overflow" did not occur, the OVRF will be left unchanged. It is very inefficient to test overflow by the JNO instruction after each arithmetic instruction. A feature of GE/PAC is that this test need be given only once -- at the end of the computation -- to provide valid arithmetic overflow detection.

It should be emphasized that a "carry" out of the A Register and arithmetic overflow are unrelated in two's complement fixed point arithmetic -- such a carry is neither a necessary nor sufficient condition for arithmetic overflow. Each arithmetic operation has a different criterion for determining whether or not overflow occurred. These criteria are:

ADD

If (A) and (Y) have like signs, and the sign of the Sum differs from that of its arguments, the OVRF is set; otherwise the OVRF is unaffected.

SUB

If (A) and the one's complement of (Y) have like signs and the sign of the difference differs from that of its arguments, the OVRF is set; otherwise the OVRF is unaffected.

MPY	REGISTER			ARITHMETIC OVERFLOW
	Y	Q	A	
	M	M	≥ 0	yes
	M	M	< 0	no
value	(Y)	(Q)	(A)	no

M in the above table is the octal number 40000000. It is an illegal number in the sense that this maximum negative number is its own negative. That is, its negative is not positive in the accepted sense -- namely that the so-called sign bit be a 0.

If arithmetic overflow occurs as a result of MPY, the OVRF will be set; otherwise, the OVRF is unchanged.

DVD

If the magnitude of (Y) exceeds that of (A), the OVRF is unaffected.

If the magnitude of (A) exceeds that of (Y), the OVRF is set.

If the magnitude of (A) equals that of (Y) and the sign of the resulting remainder is the same as the sign of the divisor, the OVRF is unaffected; if these signs differ, the OVRF is set.

SLA and DLA

If (A) is positive: If any "1" bit is shifted into A₂₃, the OVRF is set; otherwise the OVRF is not changed.

If (A) is negative: If any "0" bit is shifted into A₂₃, the OVRF is set; otherwise the OVRF is not changed.

Floating-Point Instructions

Arithmetic overflow occurs when the magnitude of the exponent of the result is too large for the exponent field of the floating-point format. Exponent "overflow" and "underflow" are detected and will set the OVRF. If Mantissa "underflow" occurs, the result will be replaced by "0".

FIXED-POINT ARITHMETIC OPERATIONS

The term "scaling" is used to describe the analysis process associated with the programming of numerical problems using fixed-point arithmetic. Fixed-point arithmetic operations are subject to the following rules:

Addition and Subtraction

Variables to be added (subtracted) must have like scales. If scales differ, each variable is scaled by multiplying it by an appropriate power of two so that the resulting scales are the same. The scale of the result is that of its addends.

Multiplication

Variables to be multiplied may have different scales. The scale of the result is equal to the sum of the scales of its factors.

Division

Variables may have different scales. The scale of the quotient is equal that of the dividend diminished by the scale of the divisor. The scale of the remainder is equal to the scale of the dividend diminished by 23.

Scaling may be accomplished in one or both of two ways:

- a. Mental multiplication by a power of 2
- b. Actual multiplication by a power of 2 as implemented by a shift (SRA, SLA)

Most precise results are obtained when scaling is planned so that the number of significant bits in the working registers A and Q is maximized at all times.

FLOATING-POINT ARITHMETIC OPERATIONS

The purpose of floating-point arithmetic is to minimize or eliminate the need for scaling. "Floating-Point Scaling" consists of analyzing all equations comprising a problem, and reformulating if necessary to insure that all variables (including results and partial results) are representable by the floating-point format.

The programmer would write FIX, FLO, FAD, FSU, FMP, or FDV when using single-word floating-point arithmetic within a MONITOR system program. Double-length floating-point is available when required via normal subroutine linkage.²

<u>Single word</u>		<u>Double word</u>	
FAD	Y, X	SPB	FAD2
		LDA	Y, X

LOGIC AND BIT LOGIC OPERATIONS

GE/PAC has the conventional logical instructions (CPL, ORA, ANA, ERA) which address memory may operate on the addressed 24-bit words. In addition, a GEN 1 class of instructions allows the programmer to address individual bits of a word on the A Register and perform the same operations on the addressed bits. Thus:

SBK	K	Sets the Kth bit of A to a "1"
CBK	K	Changes the Kth bit of A
TOD	K	Tests the Kth bit of A

(and many others) are provided. Instructions such as CPL and shifting instructions are a natural by-product. An appreciation of the value of the operations to process control is obtained from an example: A large share of a control program consists of decision making based upon the status of true/false bit-logic variables. These conditions are normally read into the computer via a Digital Input Scanner.

Assume that one step in a start-up process consists of determining that one and only one of two pumps is "on" with its valve being open and also that a main valve is open.

A = Motor on, pump #1
 B = Valve open, pump #1
 C = Motor on, pump #2
 D = Valve open, pump #2
 E = Main valve open

The programming technique does not depend upon the arrangement of the variables; hence all may be assumed to fall in the same word without loss of generality. The step may be expressed in the form of a logic equation:

$$\text{If } [A*(\bar{B}\bar{C}\bar{D}) + \bar{A}*(\bar{B}C\bar{D})] * E = 1, \text{ GO TO OK}$$

GE/PAC coding for this equation is given in Figure V-3.

<u>OP</u>	<u>ADDR</u>	<u>REMARKS</u>
LDA	GRØUP 00	
TØD	BITA B	IF (A-1) 02, 01, --
BTR	*+5	
REV	BITB 01	T = A*B* \bar{C} * \bar{D}
RØD	BITC	
RØD	BITD	
BRU	*+5	GØ TØ 03
SET	02	T = 1* \bar{B} *C*D
RØD	BITB	
REV	BITC	
REV	BITD	
REV	BITE 03	T = T*E
BTS	ØK B	IF (T-1) 04, ØK, --
BRU	ALARM 04	

* USE OF GE/PAC BIT LOGIC INSTRUCTIONS
 * TO EVALUATE LOGIC EQUATION
 * IF (A*B*-C*-D + -A*-B*C*D) * E = 1, GO TO OK

Figure V-3.

NOTE 2: In non-MONITOR applications, if double-length floating-point is used exclusively, quasi instruction subroutine linkage may be used.

It is possible for a clever programmer to produce a shorter program using GE/PAC's conventional logical instructions (CPL, ORA, ANA, ERA) if rigid assumptions with respect to the location and arrangement of the bit variables are made. In this latter case, all five variables must be assumed to fall in sequence in the same digital word, GE/PAC coding for the equation is given in Figure V-4.

OP	ADDRESS	REMARKS
LDA	GROUP	EABCD-----
SRL	1	OEABCD-----
ERA	GROUP	1-010----- IF OK
ERA	MASK1	
ANA	MASK2	
TZE		
BTS	OK	
BRU	ALARM	
*		BIT A MUST BE BIT 22
*		BIT B MUST BE BIT 21
*		BIT C MUST BE BIT 20
*		BIT D MUST BE BIT 19
*		BIT E MUST BE BIT 23
MASK 1		
CON	/44000000	100100000...
MASK2		
CON	/56000000	101110000...
*		USE OF GE/PAC CONVENTIONAL LOGIC
*		INSTRUCTIONS TO EVALUATE
*		LOGIC EQUATION
*		IF (A*B*-C*-D + -A*-B*C*D) * E = 1,
		GO TO OK

Figure V-4.

A comparison of the two methods explains the value of bit logic.

Bit Logical Method Advantages:

a. Permits a natural and straightforward algorithmic evaluation of logic equations. Therefore:

1. Suited for use by junior programmers without any loss in quality of coding.
2. Suited for use in a compiler.
3. Reduces programming time.

b. Permits symbolic representation of "bit variables". Minimizes the need for programming dictation of the specific arrangement of inputs to the Digital Input Scanner.

Disadvantages:

- a. Requires more memory and execution time.

Conventional Logical Method Advantages:

- a. Requires less memory and execution time if optimally coded.

Disadvantages:

a. Requires ingenuity in choice of specific technique and bit arrangement. Therefore:

1. Quality of coding is dependent upon the caliber of the programmer. Non-optimum coding can require more memory and execution time than produced by the bit logic method.

2. Not suited for use in a compiler.

3. Increased programming time.

b. Symbolic representation of contact closures is very awkward. Requires restrictions on hardware grouping of variables.

c. A bit arrangement that is optimum for one test is probably not optimum for another test in the same program involving the same set of variables.

LISTS AND THEIR USAGE

DEFINITIONS

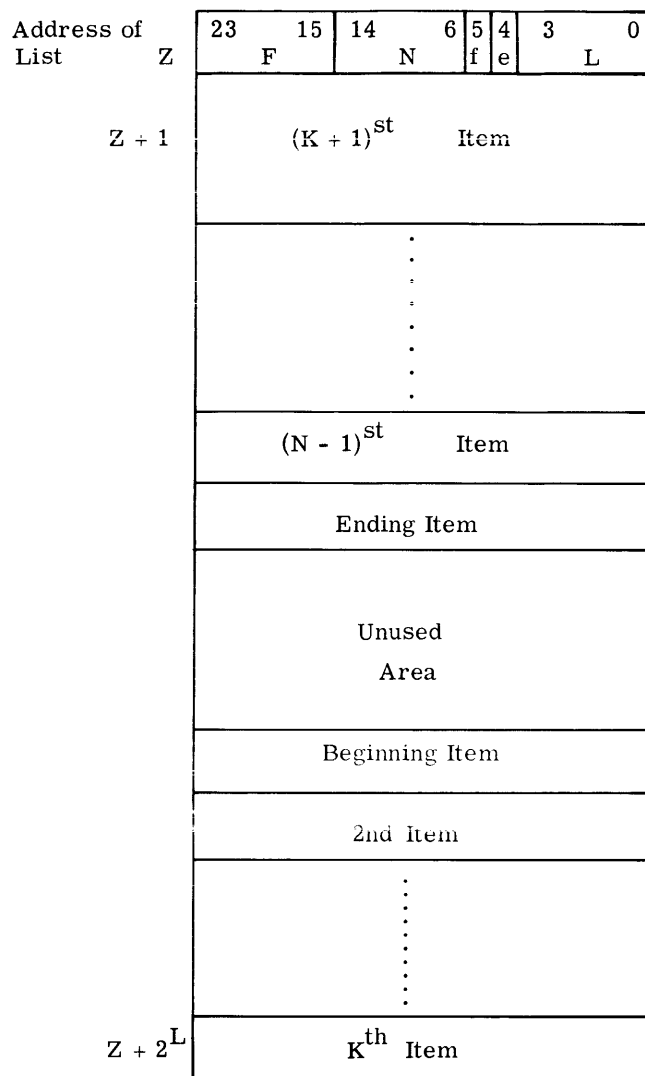
A "List" consists of an ordered set of "items" contained within a fixed-length block of memory ($2^L + 1$ words where L is an integer, $1 \leq L \leq 9$).

The first word of the block is a "list control word" and serves as the address or label of the list. The maximum size of any given list is 2^L "items". The physical ordering of a list is circular in nature, that is, the address 2^L relative to the list control word is followed sequentially by the relative address 1. The terms "beginning item" and "ending item" are arbitrarily attached to the first and last members of the set of items currently forming the list. Figure V-5 pictures list addressing details. Instructions are provided for appending additional items to the beginning or end of the list and for removing the beginning or ending items.

These instructions are:

ABL	Z	Append item to beginning of List Z
AEL	Z	Append item to end of list Z
RBL	Z	Remove beginning item from list Z
REL	Z	Remove ending item from list Z.

An item, when appended, becomes the new beginning (ending) item. The removal of an item forces the adjacent item to be the new beginning (ending) item. The current beginning and ending items are the only items addressable by program ---- note this contrast to normal table indexing techniques which allow addressing of any entry in a table.



LIST CONTROL WORD

- L Specifies maximum size of list (2^L items where $1 \leq L \leq 9$).
- F Specifies location of "beginning item" ($0 \leq F \leq 2^L - 1$).
- N Specifies number of items currently in list ($0 \leq N \leq 2^L$).

Address of Beginning Item
 $Z + 1 + (F + 1) \text{ MOD } 2^L$

Address of Ending Item
 $Z + 1 + (F + N) \text{ MOD } 2^L$

The list is empty if $N = 0$ and $e = 1$

The list is full if $N = 0$ and $f = 1$

Figure V-5. GE/PAC List Addressing Details

THE QUEUING LIST AND ITS USAGE

Webster's definition of a "queue", a "waiting line, as of persons before a ticket window", describes the Queuing List quite satisfactorily.

A typical usage of the queuing list is for channeling of information to and through Input/Output devices. Suppose a request for the "on demand print out" of the current value of some analog input be made. The scan command word and appropriate control information would be appended to a queue controlling the operation of the scanner. When available, the raw "count" value of this input will be appended to a queue awaiting conversion from binary to engineering units.

Another queue is a waiting line to the binary to BCD conversion routine. The resulting ordered set of BCD characters is appended to the output driver queue controlling a typewriter.

The instruction pair AEL, RBL are used for queuing. Just as it is possible for a "Very Important Person" to go to the "head of the line", high priority items may be appended at the head (beginning) of the list so that it would be the next item to be removed. Thus AEL and RBL provide normal queuing; ABL is used to place high priority items at the "head" of the list.

THE STACKING LIST AND ITS USAGE

The "stack" or "push down list" is so named because the item most recently appended to the list is always removed prior to the removal of less recently appended items. The phrase "last in - first out" is sometimes used to describe the operation of a stack.

A typical usage of a stack is as a Common temporary storage block for programs that are subject to Program Interrupt. Each program by appending all of its intermediate results in the stack could be "clobbered" by a higher priority program. When all higher priority programs were completed, this clobbered program would be restored to continue from its point of interruption.

The use of the stack requires that the relative priorities of all programs having data within the stack do not change. Further, use of the stack on GE/PAC 4020 is relatively slow. When stacking of data is not satisfactory for a given program, one of the following alternates must be used:

1. The program is assigned unique locations in COMMON for its intermediate results.
2. The program is executed in the Interrupt Inhibited mode.
3. The program is restarted from its initial conditions.

The instruction pair AEL, REL are for stacking.

GEN 1 INSTRUCTIONS¹

GE/PAC 4000 contains the ability to execute many more instructions than those listed with mnemonics in the PAL assembly program. The rules for making these instructions are discussed below and involve microcoding. Microcoding is the ability to implement instructions by the proper use of different bits in the instruction word. There are 1024 unique GEN 1 instruction octal words. Thirty-two of these have been determined to be useful enough to justify assigning them mnemonics and are listed in the beginning of this manual. Additional microcoded octal instructions may be defined in the PAL assembly program by using the DEF pseudo instruction as detailed in the PAL Assembly Manual.

GEN 1 instructions are microcoded and involve serial shifting of the A Register, affect the J Counter, and optionally affect the Test Flip-Flop (TSTF). The instruction format is divided into microcoded categories as defined below:

- 05 = GEN 1 instruction octal
- X = Index word indicator
- C_C = Operation control
- α_C = Alpha control
- β_C = Beta control
- T_C = Test control
- Δ_C = Shift length control
- k = Shift length or bit position designator

23 --- 18	17 - 15	14 13	12 11	10 9	8 - 6	5	4 -- 0
05	X	C _C	α _C	β _C	T _C	Δ _C	k

A GEN 1 instruction is indicated by octal 05 in bit positions 23 through 18 inclusive. The index word indicator designates the index word used to modify the instruction. The α_C, β_C, Δ_C, and k determine two 24-bit operands. These operands are developed, based on the contents of the I Register prior to the instruction. The C_C then combines the operands and places the result into the A Register. T_C either counts the significant zeros of the α operand, or tests the α operand and affects the TSTF. k specifies the number of shift positions or the bit position to be tested in the α operand.

The GEN 1 instruction is executed in the following manner: Instruction fetch and indexing occur during timing states 1 and 2. Since Indexing Address Modification affects 13 through 0 of the instructions, indexing can change the meaning of the GEN 1 instruction. Implementation of timing state 3 is as follows:

1. The complemented value of k, (I₄₋₀) is placed in the J counter.
2. The control fields determine inputs to the serial adder, to the test flip-flop, and to the shifting control.
3. The contents of the A register are ringshifted bit-by-bit through the adder, the output of the adder being shifted into A₂₃ of the A register.
4. The J counter is incremented one for each one-bit shift. The final contents of this counter is 37₈ if Δ_C = 1, is 27₈-k if Δ_C = 0, k ≤ 23₁₀ and is 37₈ if Δ_C = 0, k = 24₁₀.

Figure V-6 presents the rules for obtaining all possible α and β operands. The α operand is the modified, and perhaps shifted, contents of the A register. The β operand provides a means for further modifying one bit of the α operand.

Figure V-7 states rules for combining the α and β operands.

Figure V-8 specifies possible testing actions.

To explain the usage of these tables, several examples are given:

Examples	α _C	β _C	C _C	Δ _C	T _C	K
1. Load zero	00	00	00	0	000	00
2. Load "minus one" and set TSTF	00	00	11	0	110	00
3. Shift right 5 and fill with "ones"	00	00	11	1	000	05
4. Shift circular 21 and set TSTF if any of the right 21 bits had been a 1. Reset the TSTF if all of these bits had been zero.	01	00	00	1	111	25 ₈
5. Test bit k: Set TSTF if bit k is even. Reset TSTF if bit k is odd. Reset bit k	01	01	01	0	110	k

NOTE 1: The omission of this section from the programmers "reading plan" will not be detrimental to his understanding of real time GE/PAC programming.

		OPERAND α			
$\Delta_c \backslash \alpha_c$		00	01	10	11
$\Delta_c = 0$		$0_{23}, \dots, 0_k, \dots, 0_0$	$\overset{23}{a}_{23}, \dots, \overset{k}{a}_k, \dots, \overset{0}{a}_0$	$\overline{\overset{23}{a}_{23}}, \dots, \overline{\overset{k}{a}_k}, \dots, \overline{\overset{0}{a}_0}$	$\overset{23}{a}_{23}, \dots, \overset{23}{a}_{23-k}, \dots, \overset{23}{a}_0$
$\Delta_c = 1$					
$K \leq 23$		$0_{23}, \dots, 0_{24-k}, \overset{23}{a}_{23-k}, \dots, \overset{k}{a}_0$	$\overset{k-1}{a}_{23}, \dots, \overset{0}{a}_{24-k}, \overset{23}{a}_{23-k}, \dots, \overset{k}{a}_0$	$\overline{\overset{k-1}{a}_{23}}, \dots, \overline{\overset{0}{a}_{24-k}}, \overset{23}{a}_{23-k}, \dots, \overset{k}{a}_0$	$\overset{23}{a}_{23}, \dots, \overset{23}{a}_{23-k}, \overset{22}{a}_{22-k}, \overset{k}{a}_0$
$K > 23$		$0_{23}, \dots, 0_0$	$\overset{23}{a}_{23}, \dots, \overset{0}{a}_0$	$\overline{\overset{23}{a}_{23}}, \dots, \overline{\overset{0}{a}_0}$	$\overset{23}{a}_{23}, \dots, \overset{23}{a}_0$

		OPERAND β			
$\Delta_c \backslash \beta_c$		00	01	10	11
$\Delta_c = 0$					
$K \leq 23$		$0_{23}, \dots, 0_k, \dots, 0_0$	$0_{23}, \dots, \dots, \overset{k}{a}_k, \dots, 0_0$	$0_{23}, \dots, \overset{k}{a}_k, \dots, 0_0$	$0_{23}, \dots, 1_k, \dots, 0_0$
$K > 23$		$0_{23}, \dots, 0_0$	$0_{23}, \dots, 0_0$	$0_{23}, \dots, 0_0$	$0_{23}, \dots, 0_0$
$\Delta_c = 1$		$0_{23}, \dots, 0_0$	$0_{23}, \dots, 0_0$	$0_{23}, \dots, 0_0$	$0_{23}, \dots, 0_0$

NOTES: a_i^k means that the contents of bit position i of α is a_k (ie $\alpha_i = a_k$).

$\overline{a_i^k}$ means that $\alpha_i = \overline{a_k}$.

Figure V-6.

		OPERATION			
$\Delta_c \backslash C_c$		00	10	01	11
$\Delta_c = 0$		$\alpha + \beta \longrightarrow A$	$\alpha \text{ ERA } \beta \longrightarrow A$	$\alpha + \beta + 1 \longrightarrow A$	$\overline{(\alpha \text{ ERA } \beta)} \longrightarrow A$
$\Delta_c = 0; \beta_c = 0$		$\alpha \longrightarrow A$	$\alpha \longrightarrow A$	$\alpha + 1 \longrightarrow A$	$\overline{\alpha} \longrightarrow A$
$\Delta_c = 1; k \leq 23$		$\alpha \longrightarrow A$	$\alpha \longrightarrow A$	$\alpha + 1 \longrightarrow A$	$\alpha \text{ ERA } \gamma \longrightarrow A$
$\Delta_c = 1; k > 23$		$\alpha \longrightarrow A$	$\alpha \longrightarrow A$	$\alpha + 1 \longrightarrow A$	$\overline{\alpha} \longrightarrow A$

NOTES: ERA means exclusive OR

$\overline{\alpha}$ means ones complement of α

γ means $1_{23}, \dots, 1_{24-k}, \dots, 0_0$

$+$ means arithmetic addition

$+1$ means $+0000001_8$

Figure V-7.

TC	Test Action on Operand α
000	No test.
001	No test, count least significant ones (K must be equal to 37_8).
010	No test, count most significant ones (K must be equal to 37_8).
011	Not meaningful.
100	Reset TSTF if "False", otherwise TSTF unchanged.
101	Set TSTF if "False", otherwise TSTF unchanged.
110	Set TSTF if "True", reset TSTF if False.
111	Reset TSTF if "True", set TSTF if False.

Where the "True" criteria is defined as:

If $\Delta_c = 0$, "True" means $\alpha_k = 0$.
 If $\Delta_c = 1$, "True" means $\alpha_{k-1} = \alpha_{k-2} = \dots = \alpha_0 = 0$.

Figure V-8.

PROCEDURE: Each count instruction must be immediately followed by its associated LXC. These instructions must be bracketed by IAI and PAI in a permitted program.

```
IAI
CLZ
LXC X
PAI
```

Non-Inhibitable Program Interrupt Action Routines must save and restore the J Counter if the Action Routine contains any GEN 1 instruction.

The J Counter is saved by the coding:

```
LXC X COUNT → X.
```

The J Counter can be restored by the following coding:

```
LDO O, X
CLZ
```

Note that the A Register is destroyed by this operation.

Figures V-9 and V-10 present the flow chart and coding for a typical application which uses a count instruction. A simple executive routine examines bits in a computer word, transferring program control to a subroutine as a control bit is set.

The reader will observe that this technique can be used for either of the following purposes:

1. As a Demand Routine ECP
2. As a "Change of State" ECP Associated with Program Interrupt and/or contact status groups read in via the Digital Input Scanner. (DIS)

A NOTE ON THE COUNT INSTRUCTIONS

The instructions CMO and CMZ are used to count the number of most significant ones (zeros) of the data word in the A Register. CLO and CLZ count the number of least significant ones (zeros) in the A Register.

These instructions leave the determined "count" value in the J Counter; an LXC instruction must be executed to transfer it to an Index word. These instructions are not protected from Program Interrupt. Moreover, Quasi instructions must not occur between a count instruction and its associated LXC. Certain quasi instructions will use the J counter without restoring it. These considerations dictate the following STANDARD GE/PAC PROGRAMMING

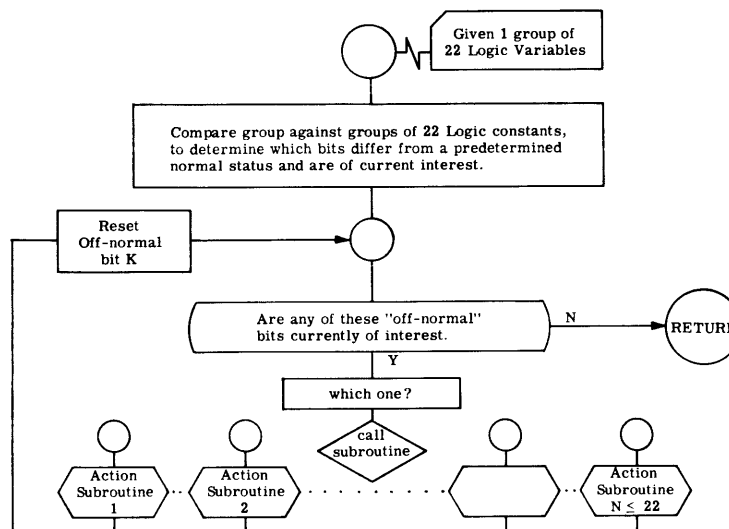


Figure V-9. Application of Count Instruction

LOCATION	OP CODE	ADDRESS		REMARKS
ACTA	LDA	GROUP	000	GROUP OF 22 LOGIC VARIABLES DETERMINE OFF-NORMAL VARIABLES DETERMINE OFF-NORM OF INTEREST
	ERA	NORMAL		
	ANA	INTRST		
ACTB	TZE		001	B IF (ANY OFFNORMAL OF
	BTS	ACTC		X INTEREST) --, 003, 002
	IAI		002	DETERMINE WHICH ONE K
	CLZ			
	LXC	3		
	PAI			
	SPB	ACTION, 3		CALL ACTIONF (K)
	RBK	0, 3		RESET OFFNORMAL BIT K
	BRU	ACTB		GO TO 001
ACTC	LPR	SX	003	RETURN
NORMAL	BSS	1		
INTRST	BSS	1		
ACTION	BRU	ACT1		
	BRU	ACT2		
	BRU	ACTN		
SX	BSS	1		

Figure V-10. Application of Count Instruction

VI PROGRAM CONTROL

INTRODUCTION

A real-time process is characterized by the occurrence of many events, some continuous, others random in nature. Events may occur simultaneously. A digital computer is a serial device; that is, it performs its program operations serially, one by one. As a consequence, real-time programs are distinctively different from their non-real time counterparts.

The most distinctive characteristic feature of a real-time program is the seemingly disorganized way in which it is executed.

1. Programs need not be completed without the occurrence of several interruptions. Thus, the most critical programs are executed first while other programs wait momentarily.
2. Internal data transfer is rarely accomplished in one operation.
3. Intercommunication between programs becomes significant and time consuming. Information is constantly being stored and retrieved from small tables. Table storage and retrieval rates are asynchronous.
4. If a program is likely to be interrupted prior to completion, care must be taken to insure that its working storage is not destroyed.
5. Subroutines common to several programs must receive special attention. Temporary constants used by a subroutine with respect to a given program must be preserved until execution of the subroutine is completed.

A second distinctive feature is that the program is an active element in the process system. It must keep real time, and must initiate actions at specified real times and after specified time delays.

A third distinctive feature is the requirement that it should never wait upon I/O equipment; there is always some time-critical function to be implemented. In an off-line computer system, lost time would only cost money. In on-line operation, lost time could mean something undone, perhaps endangering system security.

The preceding characteristic features indicate that the so-called "housekeeping functions" necessary in every computer program become extremely

important in the real-time program. The program capabilities of a process control computer must first provide efficient housekeeping (Program Control) capabilities, then provide arithmetic capabilities.

PROGRAM CONTROL

A computer program consists of an ordered sequence of instructions to the computer. These instructions are placed in memory cells having sequential addresses so that the ordering of the addresses serves to order the instructions.

The term "Program Control" is used in describing the instruction sequencing process and is associated with a location in memory. Program Control is specified by the contents of the Program Counter. Program Control normally is transferred from one location to the next sequential location as instructions are executed. Program Control can be transferred to an arbitrary location by "branch" instructions. It can be made to conditionally skip a location by "jump" instructions.

Nominally, the Program Counter contains the address within the program of the instruction which is to be executed next. However, there are many exceptions to this statement. Most modern process computers have "execute" and "program interrupt" functions which complicate the instruction sequencing process. It is paradoxical that this complication is one of programming description and not one of hardware; these functions are actually easier to provide than they are to describe.

In GE/PAC, the effect upon Program Control by the following functions is similar:

1. Execute (XEC) instruction
2. "Quasi" subroutine linkage instructions
3. Program interrupt

All three are "execute" actions which command the computer to obtain its next instruction from some location other than that specified by the Program Counter. One instruction, out of normal sequence, is executed. If this object (inserted) instruction is not a branch (jump, XEC, Quasi), Program Control is transferred to the next sequential location (with respect to the Program Counter) in the normal sequence.

The location of the object instruction is determined by the specific action: XEC uses its operand address. Quasi's use their OP codes. Program interrupt requires an address source external to the computer.

An XEC or Quasi action inhibits normal incrementing of the Program Counter during the execution of the Initiating instruction. The execution of the inserted instruction is normal. A program interrupt action inhibits normal incrementing of the Program Counter during the execution of the inserted instruction; a branch in the interrupt location may transfer Program Control.

MEMORY ADDRESSABILITY

The GE/PAC method of addressing core memory is the outgrowth of General Electric's recognition of two addressing requirements for process computers.

The first, and more immediate, requirement is to be able to transfer a program from bulk storage (Drum Memory) to core memory and instantly relocate all of its memory addressing instructions. The term "dynamic relocatability" was coined to describe this requirement. General Electric's solution is GE/PAC Relative Addressing which modifies the Y operand address of the instruction by its location in memory as the instruction is executed. It allows the on-line executive control program to instantly relocate a program from drum to any available place in core. More efficient use of both core and drum is possible.

The second addressing requirement is efficient program addressability to at least 64,000 words of random-access core memory. This requirement is medium range, but has extremely long-range implications. Real time program memory requirements have steadily increased over the past few years. Ideally all memory should be random access, but economic considerations have dictated the use of small working core memories with large bulk auxiliary memories. General Electric's solution combines GE/PAC Relative Addressing with certain indirect addressing operations to provide a theoretical maximum program addressability of one million words core memory.

Today's GE/PAC limitations to core memories of 64,000 words or less is a hardware limitation only -- the inevitable breakthrough in random access memory technology will not obsolete GE/PAC software.

Memory addressability by a memory addressing instruction is an "area addressability". A given instruction can generally address: 1 -- the area of 16384 words at the beginning of core 2 -- the area of 16384 words extending 8192 words either side of the instruction. Indexing address modification allows the instruction to address any memory location. Indirect addressing instructions also allow addressing of any memory location. A rigorous definition of these areas follows.

Every GE/PAC memory addressing instruction can address memory in the following ways: Y is the instruction's operand address; * is the instruction's present location address; Z is the instruction's effective operand address and T_U and T_M depend on the memory size.

Memory Size	T_U	T_M
8K	8K	8K
12K ¹	12K	16K
16K	16K	16K
24K ¹	24K	32K
32K	32K	32K
48K ¹	48K	64K
64K	64K	64K

1. Directly, relative to location 00:

LDA Y Address range is $0 \leq Z < 16K$
where $Z = Y$, and $0 \leq Y < 16K$

2. Directly, relative to instruction's location:

LDA Y Address range $*-8K \leq Z < *+8K$
where $Z = *+Y$, and $-8K \leq Y < 8K$

3. Memory addressing instructions which can be index address modified may address any memory location -- (X) is the contents of index word:
 $0 < (X) < 16K$ in GE/PAC 4040; $0 \leq (X) < 64K$ in GE/PAC 4050 and 4060.

LDA Y, X Address range is $0 \leq Z < T_U$
where $Z = [Y+(X)] \text{ MODULO } T_U$
 T_M and $0 \leq Y < 16K$,
or $Z = [*+Y+(X)] \text{ MODULO } T_M$
and $-8K \leq Y < 8K$

NOTE 1: In the event that the memory size is not a power of 2 (i. e., $T_U = T_M$), programs may erroneously attempt to address the non-existent memory area $T_U \leq Z < T_M$.

NOTE 2: Definition of MODULO arithmetic: Given integers A, B, and C where $A \geq 0$, $B \geq 0$, and $C > 0$:

$(A + B) \text{ MODULO } C =$
A + B if $0 \leq A + B < C$;
otherwise it is equal to $A + B - n * C$ where n is an integer chosen so that $0 \leq A + B - n * C < C$

$(A - B) \text{ MODULO } C =$
A - B if $0 \leq A - B < C$;
otherwise it is equal to $A - B \pm n * C$ where n is an integer chosen so that $0 < A - B \pm n * C < C$

4. In addition, the indirect addressing instructions LDI, STI, LDP and LPR address any memory location (Z) is the contents of memory location Z):

LDI Y,X Address range is $0 \leq W < T_U$
 where $W = \lfloor (Z) \rfloor \text{ MODULO } T_M$
 or $W = \lfloor (Z) + Z \rfloor \text{ MODULO } T_M$
 and Z is defined in one of the
 3 ways above

ADDRESS MODIFICATION

RELATIVE ADDRESS MODIFICATION

The GE/PAC instruction format allocates a one bit field I_* to specify Relative Address modification. This bit is called the Relative Addressing Indicator and is designated by *. Relative Address Modification may occur in any Full Operand and Quasi instruction. It may not occur in any GEN 1 or GEN 2 instruction; this bit position, I_{14} , is used for other purposes in these instructions.

When * is zero in a Full Operand or Quasi instruction, no relative modification occurs. When * is 1, an addend address is added to the signed 13 bit operand address to form a positive modified operand address. The normal addend address¹ is that of the location that the instruction occupied in memory. An exception occurs when the instruction being executed is the "object instruction" of a Quasi instruction or of a Program Interrupt; in these cases, the addend address is obtained from the Program Counter. Since this address is meaningless, Branch Vectors for Quasi's and Program Interrupts must not use relative addressing.

INDEXING ADDRESS MODIFICATION

The GE/PAC instruction format allocates a three bit field I_x to indicate Indexing Address Modification. The field specifies seven consecutive core memory locations for use as Index Words. The content of the field is called the "X word indicator."

When Indexing Address Modification is specified (the field is non-zero), the 14 bit² address in the specified Index Word is added to the 14 bit operand address² of the instruction in the I register to form the positive effective operand address.¹

<u>X-Word Indicator</u>	<u>Effect Upon Address Modification</u>
000	No Modification
001	Indexing Modification Using X Word 1
010	Indexing Modification Using X Word 2
011	Indexing Modification Using X Word 3
100	Indexing Modification Using X Word 4
101	Indexing Modification Using X Word 5
110	Indexing Modification Using X Word 6
111	Indexing Modification Using X Word 7

Indexing address modification occurs as defined above whenever the X-Word Indicator is non-zero for all instructions except the six Indexing Control instructions. The Indexing Control instructions provide the means to load, store, increment, and test individual index words without using the A register. The usage of X-word 1 and X-word 2 is restricted. X-word 1 is used automatically by the arithmetic unit for Subroutine linkage, Quasi instruction linkage, and Program Interrupt linkage. X-word 2 is used automatically for Quasi instruction linkage. As a general practice, these two X-words should not be used within a program.

If the X-word Indicator in an Indexing Control instruction is zero, the instruction completely changes its meaning.

STX	($06X_8$)	becomes DMT 060_8
INX	($24X_8$)	LDX ($16X_8$), L XK ($07X_8$), and
LXC	($17X_8$)	store zeros into location zero (0).

STX, LXC, LDX, L XK, INX and TXH are illegal for X=0 by programming convention in the assembly program.

NOTE 1: The address is 14 bits in GE/PAC 4040 and 16 bits in GE/PAC 4050 and 4060.

NOTE 2: Relative address modification, if required, occurs prior to indexing address modification. In this case the resulting modified positive operand address is added to the address in the index word to obtain the effective positive operand address.

THE PURPOSE AND NATURE OF INDEXING

The fundamental purpose of Indexing is to accomplish address modification of Memory Addressing instructions. A memory address is always positive. Indexing address modification is implemented within the arithmetic unit by the addition of the index and the positive Operand Address Y^2 of the instruction.

Indeed, memory addressing instructions must have positive operand addresses. For example, the instruction LDA -1, X which will be correctly executed on many computers, is an illegal GE/PAC statement.³ The GE/PAC Assembly program will tag all negative absolute addresses as being possible errors.

An index is in essence an address and therefore should be considered as positive. Moreover, since the instruction TXH presumes both the index X and the test value K to be positive, it is dangerous to interpret an index as negative.

A secondary purpose of indexing is to permit the modification of GEN1 and GEN2 instructions. Since a portion of the operation code extends into the operand address field, indexing permits modification of the instruction within its lower 14 bits (I_{13-0}). Modification is MODULO 16K.

SUBROUTINE LINKAGE

The subroutine linkage instruction SPB is an ideal example of the benefit of a system approach to process computer design. SPB is much more than a mere subroutine linkage instruction. The specifications of quasi instruction linkage and program interrupt are intimately related to SPB.

SPB saves the status¹ (interrupt, test, and overflow flip-flops and the Place counter) of a main program, inhibits interrupt, and transfers program control as directed by the operand address. Return from a subroutine to the main program is implemented by the instruction LDP and/or LPR which restores part or all of the saved status.

Subroutine linkage to a subroutine located either absolutely within the COMMON area or within (assembled with) the same subprogram may be accomplished directly by the SPB instruction. Subroutine linkage to a subroutine located without (separately assembled disjoint subprogram) should be accomplished indirectly via a request to the system Executive Control Program.

In a real time process system program, X-word 1 should always be saved at the beginning of a subroutine used, that is, the first instruction executed after any SPB should be an STX. If Program Interrupts are not used and the subroutine contains no Quasi Instruction, then this rule may be violated.

Subroutines in the COMMON region must operate in the Program Interrupt inhibited mode. Subroutines within a subprogram may optionally operate in the Program Interrupt permitted mode.

NOTE 1: SPB also saves the trapping mode status if the memory protection option is present (option available only in GE/PAC 4050 and 4060).

NOTE 2: If relative addressing is specified, this is the positive address (*+Y).

NOTE 3: The following example is cited:

Assume that the indicated index word contains 3101_{10} . The arithmetic unit will interpret the address "-1" as being 16383_{10} . The effective operand address is then $(19484_{10}) \text{ MODULO memory size}$. If the memory size is less than or equal to 16K the effective operand address is 3100_{10} . If the memory size exceeds 16K, the effective operand address will be 19484_{10} .

If a subroutine Calling Sequence contains parameters, then X-word 2 may be used by the subroutine to access these parameters provided the access is made prior to the execution of any Quasi or PAI instruction.

	Calling Sequence		Subroutine		
P	SPB	SBR	SBR	STX 02, 1	Store P+1 in X2.
P+1	LDX	Y1, 3	STA	SA	Save A register.
P+2	LDA	T1, 3	STX	SX3, 3	Save X3 in SX3.
P+3		(Error Return)	XEC	0, 2	1st Parameter
P+4		(Normal Return)	XEC	1, 2	2nd Parameter
			INX	3, 2	Set Normal Return
			STX	SX, 2	
			-	}	Execute Subroutine
			-		
			-		
			LDX	SX3, 3	Load X3.
			LPR	SX	Return to P+4.

PROGRAM SWITCHES

THE COMPUTED GO-TO SWITCH

This section describes the GE/PAC coding necessary to implement the FORTRAN computed GO-TO many way switch.

GO-TO ($N_1, N_2, N_3 \dots N_n$), I

The N_i are program labels (locations). The value I is a fixed point integral variable and may be assigned or computed by the object program.

Typical coding:

```

LDA  I1      I = I1 / I2
DVD  I2
STQ  I
.
.
LDX  I, 3    Go-To (N1, N2, N3, ... NN), I
BRU  *, 3
BRU  N1
BRU  N2
BRU  N3
.
BRU  NN

```

THE ASSIGNED GO-TO SWITCH

The ASSIGNED GO-TO requires careful study because it is a ready source of program error in a relative addressing computer.

GO-TO I, ($N_1, N_2, N_3 \dots N_n$)

In the previous section I was an index value; here it is one of the n labels N_k .

The natural way of coding this switch on a non-relative machine is as follows:

```

A  LDA  N+K
   STA  I
.
.
N  BRU  I (or XEC I)
   BRU  N1
   BRU  N2
.
.
   BRU  NN

```

The above coding will not work on a relative addressing computer. Why? The instruction BRU NK is assembled relative to N+K; its operand address is NK - (N+K). If executed in location I it will transfer Program Control to location NK - (N+K) + I instead of location NK.

The ASSIGNED GO-TO Program switch should be implemented by the computed switch technique. Thus:

```

A  LXX  K, 2 } or { LDK  K
   STX  I, 2 }   STA  I
.
N  LDX  I, 2
   BRU  *, 2   or XEC *, 2
   BRU  N1
   BRU  N2
.
.
   BRU  NN

```

TESTS AND THE CONDITIONAL BRANCH BITS

GE/PAC has two conditional branch instructions BTS and BTR which conditionally branch on the status (set/reset) of a test flip-flop TSTF. With the exception of the test for arithmetic overflow, BTS or BTR is used for all conditional branches based upon internal effect tests. The TSTF serves as a memory element to remember the result of a previous test and will retain this status until changed by program. Since its status is not destroyed by BTS or BTR, many branches may be made upon the result of a single test. The status of TSTF is saved by SPB and can be restored by LPR.

There are two types of testing instructions available to the programmer.

1. The first type generally has the letter T as the first character of its mnemonic. It places the result of the test (true/false) into the TSTF. Examples are TEV, TNZ, TOD, TOR, TSC, TXH, and TZE.

2. The second type generally has the letter S or R as the first character of its mnemonic. It affects (sets/resets) the TSTF only if the test is true. Examples are REV, RNZ, ROD, SEV, SNZ, and SOD.

Type T tests are ordinarily used for conventional decision making. Type S or R tests are most useful in the evaluation of logic equations (refer to Section V - LOGIC AND BIT LOGIC OPERATIONS, for an example).

Three Program Control Instructions (SET, RST, LPR) are available to set the TSTF to a predetermined status.

INDEX AND LOOP CONTROL

Two GE/PAC instructions, INX and DMT, are used for Indexing control. TXH and DMT are used for End of Loop testing. Figure VI-1 gives three common loop control examples. The first two examples describe "backwards loop control"; the third example describes "forward loop control." Note that the Loop Counter Word I in example 1 may be either an index word or an arbitrary memory location.

QUASI INSTRUCTIONS (or Extended Function Commands)

A quasi instruction is a special two-address instruction from the hardware point of view. Its octal specifies a fixed address in a branch (SPB) vector leading to a subroutine. Its Y address is a normal operand address. The computer computes the effective operand address from the Y, *, and X fields of the instruction and saves this address in index word two. It then executes the SPB instruction in the location specified by the instruction octal (40 to 77).

Webster states that "Quasi" used as a prefix means: "that which resembles or is used as" an instruction. The quasi instruction is well named, for floating add (FAD), which is implemented by subroutine, is used by the programmer in the same way as the hardware implemented FAD instruction. Several benefits are evident.

1. A richer effective instruction repertoire permits easier coding.
2. Upward program compatibility is possible.
3. This technique reduces memory requirements of programs when compared to the use of conventional subroutine techniques.

Not all functions are best implemented as quasi's. Square root, which normally uses the A Register as its operand, does not require a two-address subroutine link. Therefore to implement it by a quasi wastes the Y operand address.

Figure VI-2 presents an elementary example of a "Store Zero into Memory" Quasi instruction. Assume STZ is arbitrarily assigned 63g as its Quasi octal. Note that inhibitable program interrupts are inhibited during the execution of the STZ subroutine (this is true by convention of all Quasi Instruction Subroutines). Note also that the first instruction of the subroutine is a STX to conform with standard subroutine convention (Section VI - SUBROUTINE LINKAGE) so that Program Control is not lost in case a non-inhibitable program interrupt executes a SPB.

Instructions in the Quasi branch vector must not use relative addressing (see Section VI - ADDRESS MODIFICATION).

XEC AND ITS USAGE

In an Execute Instruction (XEC) the address portion of the instruction specifies an object instruction to be executed, but does not set the location counter to the location of the object instruction as would a branch instruction. Thus, in effect, an XEC calls a one-instruction subroutine and specifies immediate return to the main routine.

The use of the XEC operation arises directly from the fact that the object instruction does not imply its own successor (unless it be a branch). The XEC simplifies modification of non-indexable instructions (such as index control instructions) as well as providing the ability to effectively "modify" programs (remotely) which, for some reason, may not be directly modified. Effective use may also be made of the XEC in the case of subroutine calling sequences where the calling sequence to the subroutine may include several parameters specified in actual machine instructions which the subroutine treats as second-order, one-word, subroutines.

This instruction also appeared in the GE 412 system. It completes a set of four program control operations:

1. Program Control is retained by the Main Program (Normal Instruction Sequencing).
2. Program Control is given to another program (Branching).
3. Program Control is usurped by another program (Program Interrupt).
4. Program Control is lent to another (one instruction) program (execute).

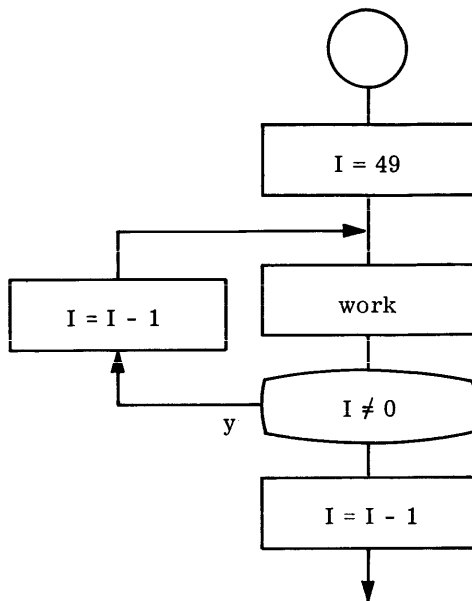
FORTRAN

Example 1

```

00 I = 49
01 .
.
.
.
I = I - 1
IF (I) 02, 01, 01
02
03
    
```

FLOW CHART



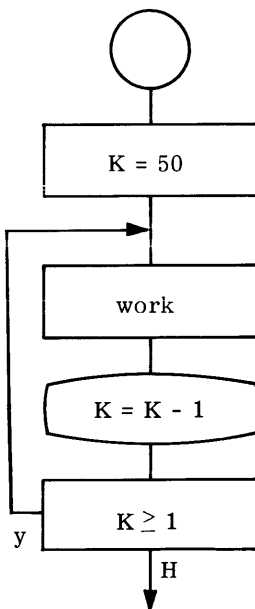
SYMBOLIC PROGRAM

Location	OP Code	Address
P01	LDA	D49
	STA	I
	.	.
	.	.
	.	.
	DMT	I
	BTS	P01
		CONTINUE

Example 2

```

10 K = 50
11 .
.
.
.
K = K - 1
12 IF (K) 12, 12, 11
13
    
```

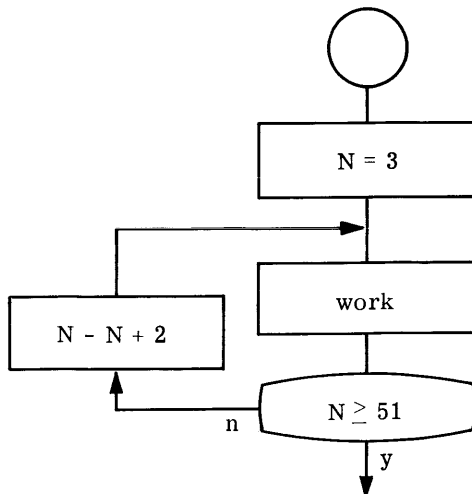


P11	LXK	50, K
	.	.
	.	.
	.	.
	.	.
	INX	-1, K
	TXH	1, K
	BTS	P11
		CONTINUE

Example 3

```

20 DO 22 N = 3, 51, 2
.
.
.
.
22 CONTINUE
    
```



P21	LXK	3, N
	TXH	51, N
	BTS	*+3
	INX	+2, N
	BRU	P21
		CONTINUE

Figure VI-1. Several Loop Control Examples

*	SOURCE	OBJECT						
	.	.						
PLACE	STZ /700	63000700	QUASI-LINKAGE INSTRUCTION					
	.	.						
*	ORG /63							
	SPB STZ		PORTION OF QUASI BRANCH VECTOR					
	.	.						
*	ORG STZ		STZ QUASI- SUBROUTINE					
	STX SX, 1		(X1) = PLACE + 1					
	STA SA		SAVE A REGISTER					
	LDZ							
	STA 0, 2		(X2) = /700					
	LDA SA							
	LPR SX		RETURN TO PLACE + 1					
SX	BSS 1							
SA	BSS 1							

Figure VI-2. Example of Quasi Instruction Linkage and Subroutine

Several examples of the use of XEC follows:

1. Indexing of non-indexable instructions

```

.           .           .
.           .           .
.           .           .
XEC KL, J   XECK1, J   XEC
.           .           .
.           .           .
KL LXX 3, K  K1 INX 2, K  KT TXH 25, K
LXX 7, K     INX 3, K    TXH 28, K

```

2. Program Switches - (Described in Section VI - PROGRAM SWITCHES.)

3. Double Indexing of Arrays

```

A = TABLE (I, J)   XEC LT, I
                    STA A
                    .
                    .
                    .
                    LT LDA TABLE, J
                    LDA TABLE+JM, J
                    LDA TABLE+2JM, J
                    LDA TABLE+3JM, J

```

4. Execution of "manufactured" instructions. If it is necessary to modify an instruction (other than by indexing) it must be placed in COMMON and be executed via XEC. The programmer should be extremely careful when manufacturing memory addressing instructions (refer to Section VI - RELATIVE, ABSOLUTE AND COMMON SYMBOLS).

5. In cases where several programs are identical except for one instruction, XEC can be used to reduce overall storage requirements by combining the programs:

	LXK	I, J	PROG	STX	SX, 1
	SPB	PROG		.	
	.	.		.	
	.	.		.	
	.	.		.	
	.	.		.	
	.	.		XEC,	INS, J
	.	.		.	
	.	.		.	
	.	.		.	
	.	.		LPR	SX
	.	.	INS	ADD	NUM
	.	.		SUB	NUM
	.	.		DVD	NUM
	.	.		.	
	.	.		.	

PROGRAM INTERRUPT

Program interrupt is the only known satisfactory hardware method to synchronize a computer program with the outside world.

An occurrence of an event in the outside world may be sensed and a signal (indicating event true/false) connected to the GE/PAC Automatic Program Interrupt Module. Two types of event detection are provided by the GE/PAC API module.

1. The event has occurred (was and/or is true) at some time in the past and a Program Interrupt to acknowledge the occurrence has not yet occurred. This is a momentary signal.

2. The event is now occurring (was and is true) and a Program Interrupt to acknowledge the occurrence has not yet occurred. This is a continuous type signal.

In either case, the GE/PAC API module will remember the event until a program interrupt can be made to acknowledge its occurrence. The GE/PAC API will then forget the event.

Type 1 event detection is normally used to provide event recording, pulse accumulation, and time. Type 2 event detection is designed for use in controlling Interrupt driven I/O equipment.

Two classes of Program Interrupt (with respect to Program Control) are provided in GE/PAC systems:

INHIBITABLE INTERRUPTS

The program has control over whether or not such interrupts may occur. Execution of an IAI or SPB instruction places the computer in the "Program Interrupt inhibited mode." Inhibitible interrupts will be delayed until such time as a PAI instruction is executed to place the computer in the "Program Interrupt permitted mode." The instructions LDP and LPR will either inhibit or permit interrupt depending upon bit 21 of the referenced memory location.

NONINHIBITABLE INTERRUPTS

The program has no control over the occurrence of such interrupts. A Program Interrupt acknowledging the occurrence of the event will be made within 1 to 82 microseconds of the occurrence of the event.

In order to insure program integrity during program-to-subroutine communication it is necessary to place a program restriction upon the SPB instruction to guarantee the program execution of one additional instruction (a STX) prior to permitting additional interrupts. A similar restriction upon the BRU instruction is necessary to simplify FORTRAN program-to-subprogram communication. These restrictions unfortunately make it possible to "hang up" the program by locking out all interrupts if either a SPB * or BRU * is executed. BTS * and XEC * are other noninterruptable program stops. None of these stops, however, are normally acceptable in a real time program.

Program Interrupt may not occur more frequently than every other instruction; that is, the instruction succeeding the interrupt-inserted instruction will always be determined by the Program Counter.

If an LDP or LPR which is to permit interrupt is executed, one additional instruction will be executed before an inhibitible interrupt may occur. A non-inhibitible interrupt may occur immediately following the LDP or LPR.

The program sequence PAI, IAI will not permit any requested Program Interrupts to occur. The sequence PAI, NOP, IAI would only allow the highest priority requested interrupt to occur, etc.

The program sequence LXX N-1, 6 PAI, NOP, IAI, DMT 6, BTS *-4 will guarantee the servicing of N requested Program Interrupts.

A Program Interrupt is accomplished by the execution of one instruction out of normal Program Sequence. Sequential memory locations 100g, 101g, . . . are reserved for this purpose corresponding to Interrupts #0, #1, . . . Although the only theoretical limitation to the number of interrupts is memory

size, available "off the shelf" hardware is limited to a maximum of 32 interrupts.

Only eight GE/PAC instructions should be used within the "Interrupt Control Table" (locations 100g, . . .). Other instructions will cause "program bugs" that could be extremely difficult to find. Relative addressing must not be used in these Interrupt Control Instructions. Permissible GE/PAC instructions are DMT, SPB, NOP, BRU, IDL, ODL, STB, LDB.

RELATIVE, ABSOLUTE AND COMMON SYMBOLS

This section concerns itself with the treatment given symbolic labels by the GE/PAC Assembly Program during the assembly process.

A programmer may specify an operand address, of an instruction which references memory, to be assembled absolute or relative. An absolute address is the true location of the referenced memory cell. A relative address is the difference between the address of the referenced memory cell and the address of the memory cell of the instruction. The result is negative when the referenced address is smaller than the instruction's address. The operand address is designated absolute (0) or relative (1) in bit fourteen of the instruction.

An operand as written by the programmer can consist of one or a combination of the following terms:

An absolute label	}	absolute terms
An integer		
A relative label	}	relative terms
The * (present location)		

Combinations are formed using add (+), subtract (-), multiply (*), and divide (/) operators (Reference Section 1.3, Process Assembler Language Manual).

An operand is assembled relative if any relative term (a relative label or the asterisk) appears in the combination of terms. An operand is assembled absolute if no relative term appears.

A label usually represents an address in memory. To reference an address absolute, the label associated with that address must be specified as absolute. Otherwise, it is assumed to be relative by the assembler. A Label specified as Common to more than one program is assembled as an absolute label in all programs in which it is referenced.

A label is specified to be Common by placing an asterisk in column seven of the line of coding where the label appears in the location field. Each common label and its value is available through the assembler to all other programs in a system.

Two additional methods to specify a label absolute are as follows:

1. Place a dash in column seven of the line of coding where the label appears in the location field.

Example: ALPHA-LDA BETA ALPHA will be referenced absolute.

2. Use the EQL Pseudo-Op to equate the label to an absolute operand.

Example: GAMMA EQL ALPHA+2 both ALPHA and 2 are absolute, therefore the operand is absolute and GAMMA will be considered absolute.

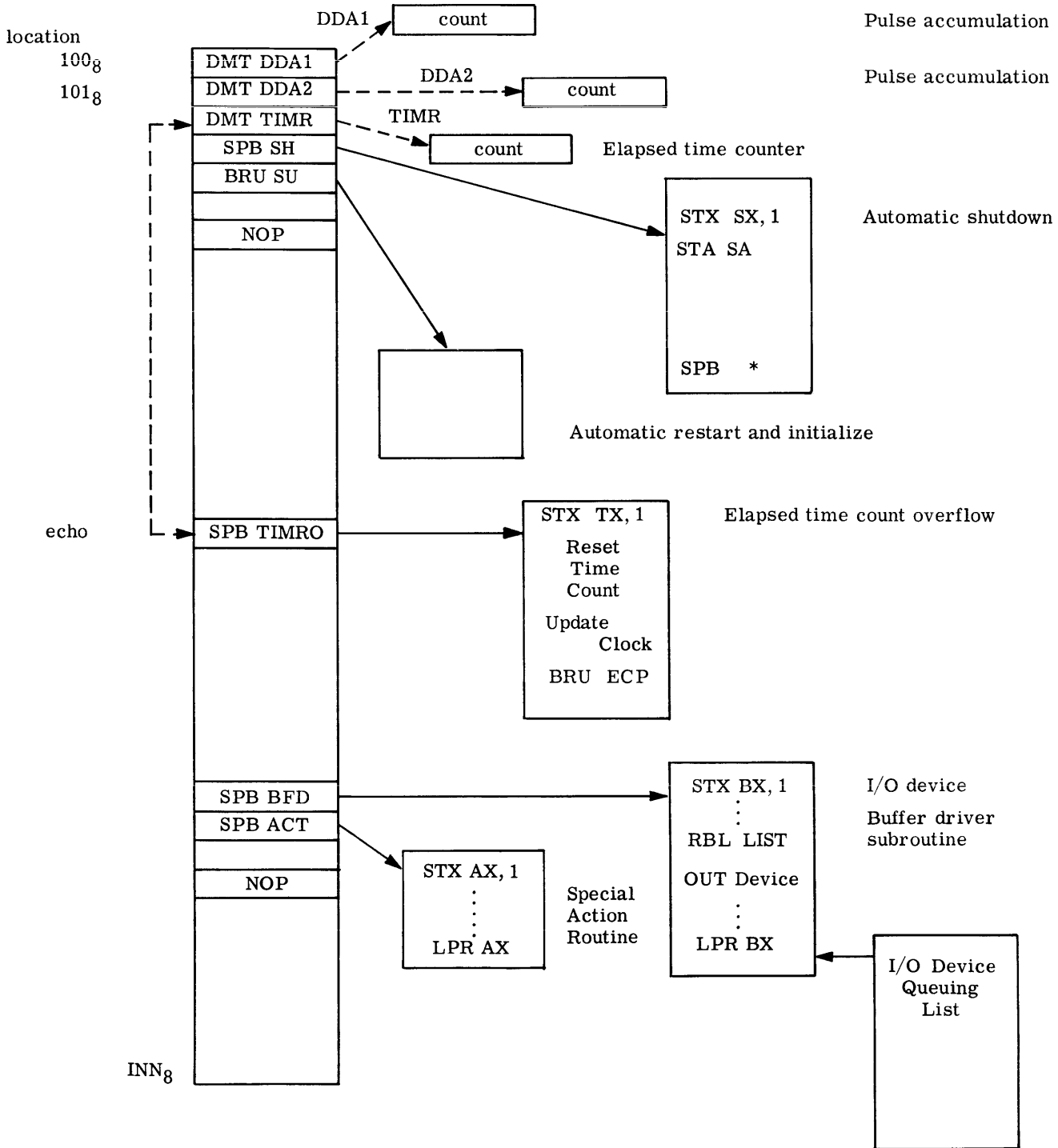


Figure VI-3. Typical Program Interrupt Usage

VII INPUT AND OUTPUT CONTROL

GEN 2 INSTRUCTIONS

GEN 2 instructions are microcoded instructions used for control of Input/Output Equipments and for certain computer actions.

Instruction format is:

23	18	17	15	14	12	11	0
25		X		S			D

where:

- 25 is the instruction octal
- X is the X word indicator
- S is the instruction secondary-octal
- D is the device address

Device address D = 0000₈ is a fictitious address referring to the computer itself.

The following instructions are provided with D = 0000:

Program Control

25010000	SSA	Set Stall Alarm
25020000	PAI	Permit Automatic Program Interrupt
25030000	IAI	Inhibit Automatic Program Interrupt
25040000	JND	Jump If No Demand
25050000	RCS	Read Console Switches

Validity Tests

25060000	JNO	Jump if No Overflow
25070000	JNP	Jump if No Core Parity

Device addresses D = 11DD₈ refer to peripheral devices. Device address D = 2400₈ refers to the GE/PAC drum. Other device addresses may be defined as required.

The generic instructions

2500DDDD	SEL	Select device D; S = 0
2501DDDD	ACT	Activate device D; S = 1
2502DDDD	OPR	Operate device D; S = 2
2503DDDD	ABT	Abort device D's operation; S = 3

2504DDDD	OUT	Output from A Register and initiate operation of device D; S = 4
2505DDDD	IN	Input to A Register and initiate operation of device D; S = 5
2506DDDD	JNR	Jump if not ready; S = 6
2507DDDD	JNE	Jump if no error; S = 7

are recognized by the assembly program. Specific meanings for each action are determined by the specific requirements of each given device D. The user is referred to specific descriptions of the various devices for detailed information.

Since Indexing Address Modification affects bits 13 through 0 of the instruction, indexing can change the meaning of the GEN 2 instruction.

The device address is a function, not of the device, of the physical location of the device within the GE/PAC cabinetry. Therefore, it is extremely important for each system program to be designed to permit device addresses to be changed without costly reprogramming or major reassemblies. A recommended GE/PAC technique is to list all device addresses in a table, assigning each table entry a Common symbolic label. This Device Address Table must be located absolutely. Each program using input/output devices will refer symbolically to the appropriate Device Address as indicated in Figure VI-1. A recommended alternate technique is to use the EQL Pseudo Op. Figure VI-2 gives details.

I/O COMMUNICATION

GE/PAC 4040 provides up to 24 independent communication paths between the A Register and Input/Output devices external to the computer. Each path is specified by the most significant six bits of a GEN 2 instruction's Device Address and links the arithmetic unit with a "module". The term module refers to the circuitry which controls the operation of a device (s) as directed by the S portion of the GEN 2 instructions. Module is also used to describe the associated cabinetry. Examples of modules are Peripheral Buffers, Analog Input Scanners, and Digital Input Scanners.

The term "device" is used to designate an electronic or mechanical unit that is controlled indirectly by the computer via the GEN 2 instruction. The device is the interface between the computer and the external world. Examples of devices are paper tape readers and punches, analog-input sensors, and digital-input sensors.

The Select instruction: SEL should never be required in any normal system program. If the necessary switching and data transmission between the arithmetic unit and device cannot be accomplished

within 24 microseconds, then a SEL must be given prior to the operation of the device. SEL is not needed for the operation of any device described in this manual.

RDR	ORG *CON 0, 1100	CONSOLE PAPER TAPE READER
TYP1	*CON 0, 1103	TYPEWRITER NO. 1
RDR2	*CON 0, 1105	PAPER TAPE READER NO. 2
DIS	*CON 0, 4400	DIGITAL INPUT SCANNER
AIS	*CON 0, 2100	ANALOG INPUT SCANNER
MCO	*CON 0, 4300	MULTIPLE CONTACT OUTPUT CONTROL
TCO	*CON 0, 4600	TIMED CONTACT OUTPUT CONTROL
DRUM	*CON 0, 2400	DRUM

TYPE	LDA TYP1, X LDA CHARACTER OUT 0, X	

Figure VII-1. Recommended Device Address Addressing Technique

RDR	EQL /1100	CONSOLE PAPER TAPE READER
TYP1	EQL /1103	TYPEWRITER NO. 1
RDR2	EQL /1105	PAPER TAPE READER NO. 2
DIS	EQL /4400	DIGITAL INPUT SCANNER
AIS	EQL /2100	ANALOG INPUT SCANNER
MCO	EQL /4100	MULTIPLE CONTACT OUTPUT CONTROL
TCO	EQL /4300	TIMED CONTACT OUTPUT CONTROL
DRUM	EQL /2400	DRUM

TYPE	LDA CHARACTER OUT TYP1	

Figure VII-2. Recommended Device Address Addressing Technique

I/O CONTROLLER INSTRUCTIONS

The GE/PAC 4050/4060 I/O Controller instruction ODL combines the functions performed by RBL and OUT to transfer data directly from a list in memory to an output device without using the A register. If the B register is substituted for the A register in the description of RBL and OUT, the composite function is identical to that performed by ODL. Similarly, IDL combines the IN and AEL functions without affecting the A register. These instructions, when used in a device's non-inhibitible location(s), perform hardware block buffering of data between the list and the device. Figure VII-3 illustrates format details.

Output block buffering is accomplished in the following manner. The Output Program assembles output data into a list by AEL'ing each output item into a separate word of the list. The data format is the same as the OUT instruction. The device's non-inhibitible interrupt is then ACT'ivated. Whenever the output device is ready to receive data, it requests a non-inhibitible "data interrupt".

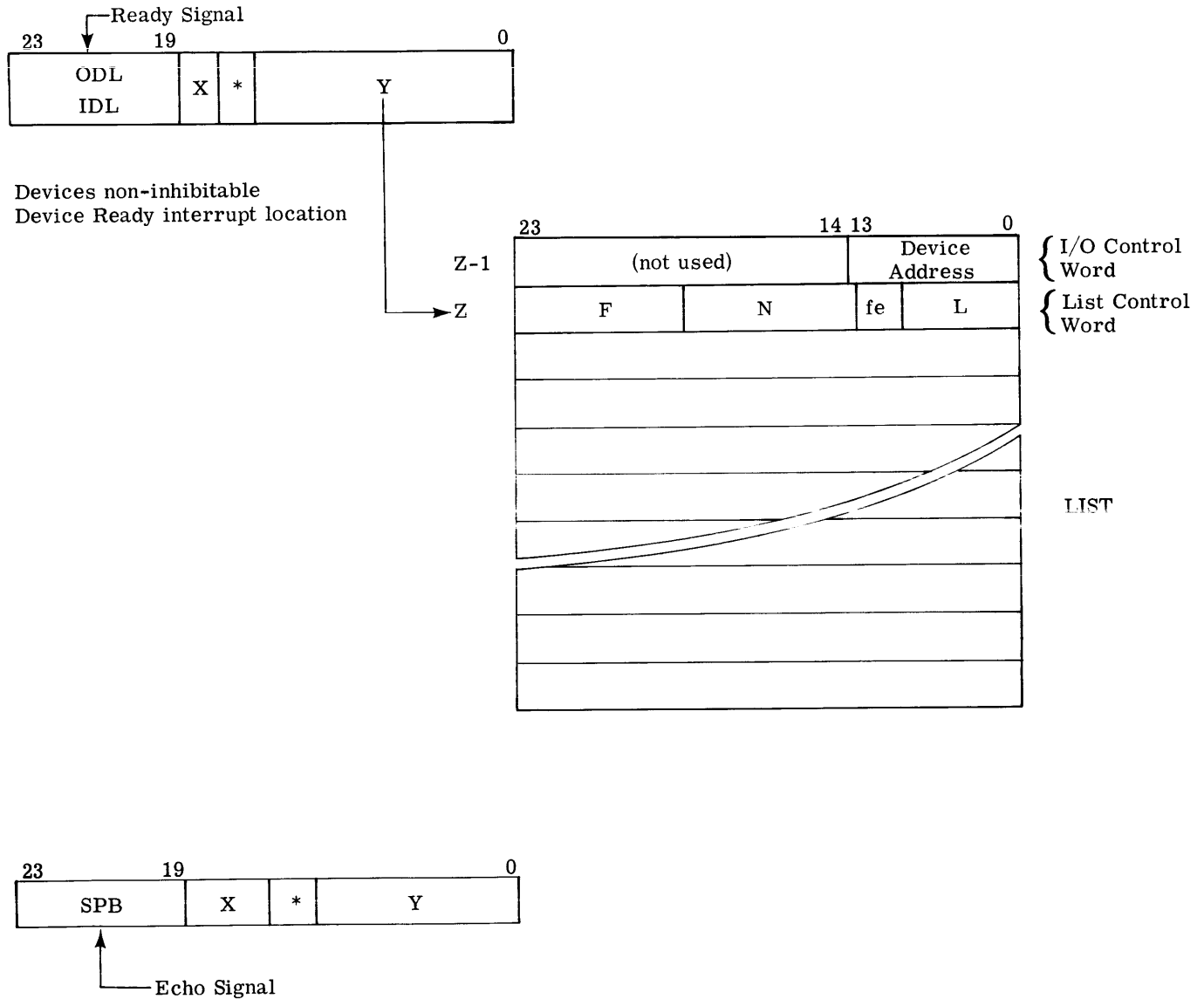
The ODL instruction in the device's interrupt location is executed when the interrupt request is serviced. The Program Counter is unaffected. If the list is not empty, data is removed from the list, transferred to the device's buffer register, and a device operation initiated. If the list is empty, a

"list-empty" echo signal is transmitted to the Program Interrupt Module. This signal requests an inhibitible Program Interrupt. An SPB instruction in this echo interrupt location will transfer Program Control back to the Output Program; additional data can be assembled into the output list, if required. When a hardware malfunction occurs the "data interrupt" does not occur. This condition can be sensed by periodically executing the appropriate JNE

instruction or by wiring the hardware-generated error signal to still another inhibitible program interrupt.

Input block buffering is accomplished in an analogous manner using the IDL instruction. Input parity errors are interpreted as hardware malfunctions.

The I/O list must be preceded by an I/O control word as shown in Figure VII-3.



VIII DATA COMMUNICATION

BULK STORAGE DRUM MEMORY (4500)

GENERAL

The GE/PAC drum is available for use as bulk storage in various sizes from 16K to 256K words. Transfer is by word between any core address and any drum address. Transfer may be specified in blocks of N words ($0 \leq N < 16K$).

Write Protect Pinboard located on the drum cabinet provides selective drum memory protection in blocks of 4K words. Figure VIII-1 indicates details.

A Program Load option is available (Section X) to provide "pushbutton" transfer of 18 drum words

(drum addresses 00₈ through 21₈) to core locations 00₈ through 21₈.

INSTRUCTIONS

Four instructions provide "Program Control". (DRUM EQL 2400₈ is the device address of the drum).

ØUT DRUM

ØUT causes the drum controller to fetch the contents Y of location zero. Y is the address of a block of three consecutive drum control words. The Drum Controller will perform a drum operation as specified by these control words.

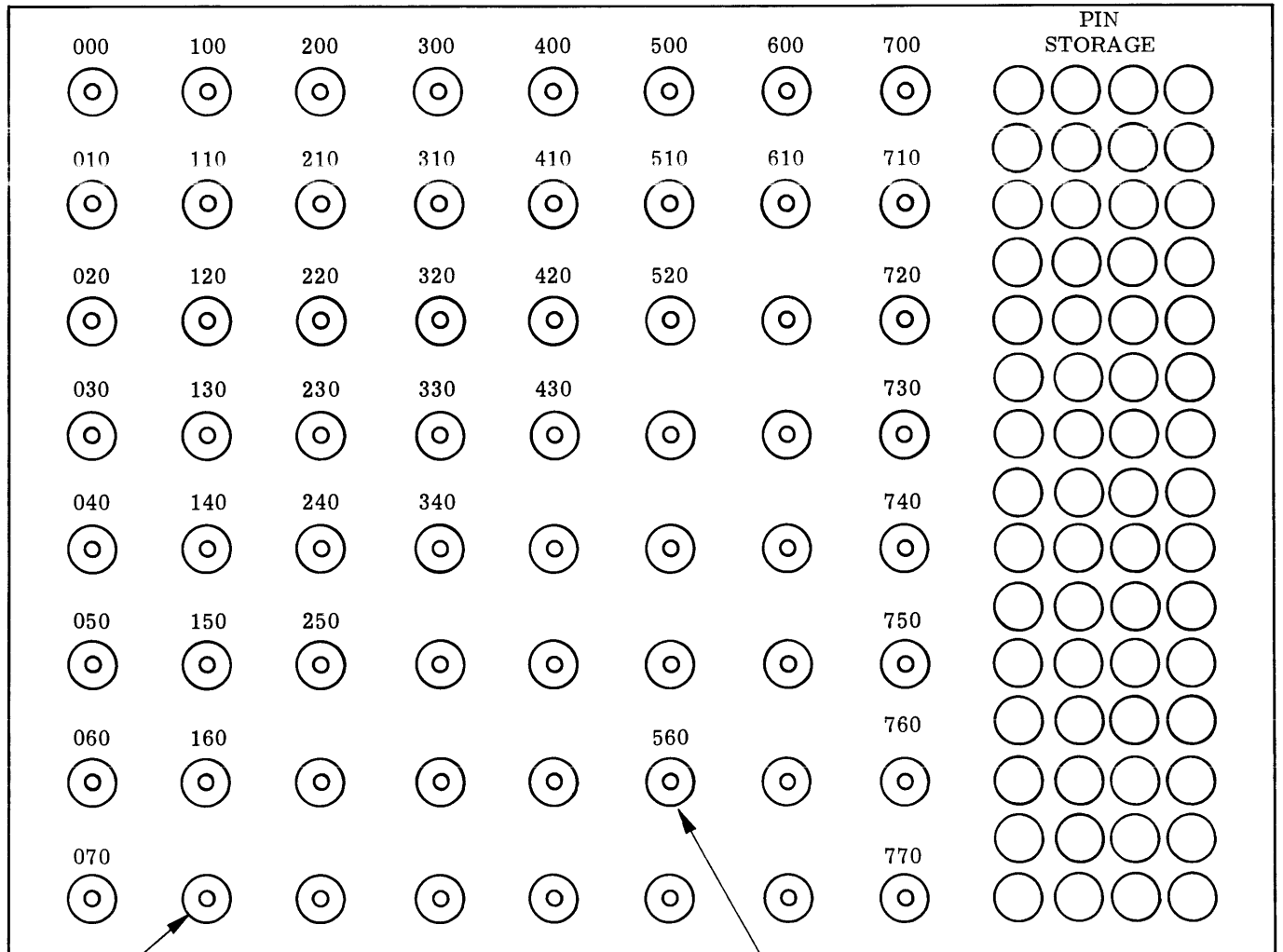
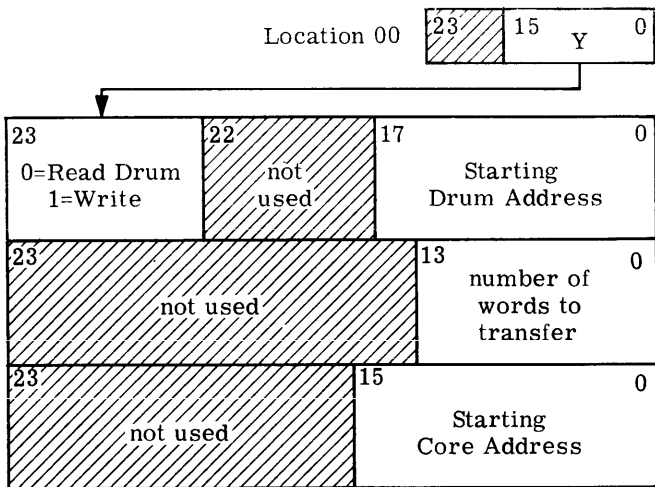


Figure VIII-1. Drum Write Protect Pinboard



A parity check is made on each word as the transfer is made; the "drum error flip-flop" is set in case a parity error is detected.

The drum error flip-flop is reset by each new ØUT instruction. If a parity error occurs during a drum operation, the block transfer will halt uncompleted. The faulty word will be written on drum if the transfer was core to drum; it will not be written on core if the transfer was drum to core.

If a drum transfer operation is in process when ØUT is executed, the new ØUT is ignored.

JNE DRUM

JNE will transfer Program Control to the second sequential location if the drum error flip-flop is reset; it will transfer Program Control to the first sequential location if the flip-flop is set (a parity error causing a drum operation abort has occurred). JNE resets the error flip-flop. This flip-flop may also be reset by either:

- Pushing the "d-c power" switch on the computer console to the initialize position, or
- Pushing the clear error switch on the system console located on the front of the drum cabinet.

JNR DRUM

JNR transfers program control to the second sequential location if a drum transfer is in progress; program control is transferred to the first sequential location if the drum is not in operation. The JNR signal (test line 1) may optionally be connected to the Automatic Program Interrupt Module to initiate a program interrupt when the signal changes from "not ready" to "ready".

ABT DRUM

ABT aborts any current drum operation following the completion of the current word transfer. The JNR signal will indicate "ready" (within 100 ms) when the drum operation is terminated. ABT is ignored if the drum is not operating.

ACT DRUM

ACT forces the "drum ready" signal to be not ready for 8 microseconds. If the drum is not in operation and if the "drum ready" signal is connected to a program interrupt input, a program interrupt is requested. If the drum is in operation, the interrupt is not requested.

INTERRUPT OPERATION

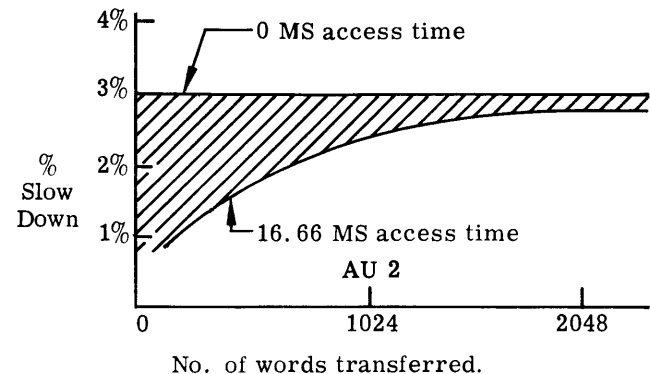
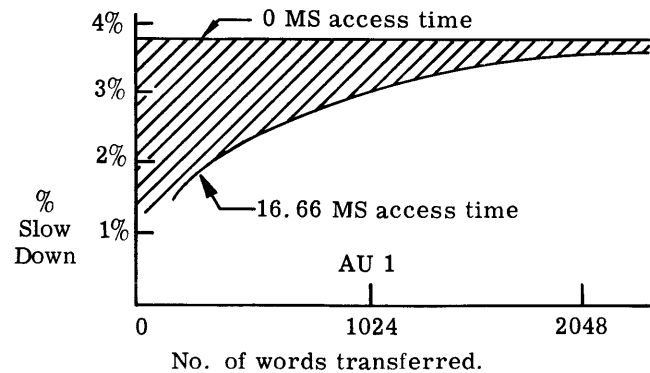
One interrupt operation is recommended. Program Control is accomplished by means of a SPB instruction in an inhibitible interrupt location. The JNR (test line 1) signal is used as both the change and level inputs to a two input program interrupt. ACT is used to conditionally request this program interrupt.

TIMING

The time required to effect the transfer includes drum access time plus actual transfer time:

access time: variable - 0 to 16.66 MS
transfer time: 0.0650MS per word.

When a drum transfer operation is in progress, the arithmetic units will be slowed down by small percentages as shown in the following tables.



Arithmetic Unit Slowdown during Drum Operation

IX PROCESS COMMUNICATION

DIGITAL INPUT SCANNER (4400)

The Digital Input Scanner (DIS) is a solid state device used to read groups of 21 logic variables into the A register at electronic speed. The DIS is composed of the Digital Input Controller and the Signal Conditioning and Switching Units. The logic variables normally indicate the status (true/false, yes/no, set/reset, open/closed, on/off) of various process system control devices, such as relays, valves, and motors. Indication of the status of various components within the GE/PAC computer system is another typical application of logic variables.

23	22	21	20	19	2	1	0
0	0	0	L ₂₀	L ₁₉	L ₂	L ₁	L ₀

Digital Input (A Register) Format

Certain inputs may be subject to possible short circuit conditions. In this event, the project engineer may elect to provide "fused" protection of the input circuits. One fuse protects 20 logic inputs in a group; the 21st input (L₂₀) is used to indicate the status (good/blown) of the fuse. Indication of a good fuse (L₂₀ = 1) does not necessarily imply that the corresponding data is valid. L₂₀ = 0 implies bad data.

The DIC Controller is packaged in standard configurations for selecting a maximum of 8, 24 and 64 groups. The signal conditioning and switching for each is packaged separately and can be ordered with any number of groups providing the number of groups is less than the maximum provided in the Controller.

Number of Inputs	Group Number K
1 to 8 groups, 21 inputs each	00 ₈ thru 07 ₈
1 to 24 groups, 21 inputs each	00 ₈ thru 27 ₈
1 to 64 groups, 21 inputs each	00 ₈ thru 77 ₈

An option provides change detection on a group and would provide a "true" signal if any of the 21 variables of the group changed state. The signal may be connected to a Program Interrupt input or it may be connected as a DIS input to be treated as a logic variable itself.

Fig. IX-1 presents two coding examples of the use of the DIS. Example 1 asks a "digital question". Further information on the evaluation of Logic equations may be found in Section V - LOGIC AND BIT LOGIC OPERATIONS. Example 2 is a "scan for digital alarm" routine. Seven consecutive groups are scanned. Selected points are to be alarmed (printed in red) when they change from normal to off-normal and alarmed again (printed in black) when they return to normal. Coding for the print program is not shown.

Operation of the DIS requires one instruction:

IN DIS +K
 IN replaces (A₂₀₋₀) by the Kth group of 21 logic variables. Bits A₂₃, A₂₂, and A₂₁ are set to "0". DIS is the device address of the Digital Input Scanner and must be defined by an EQL Pseudo Op. The range of the group number K is 0 ≤ K ≤ 77₈.

Values of K greater than 77₈ will modify the device address --- an indeterminate program bug.

ANALOG INPUT SCANNER (GE/PAC 4100)

GENERAL

Physical Quantities such as temperatures, rates of flow, weights, pressures, etc. are transduced into voltages by Analog Input Devices. The Analog Input Scanner (AIS) is a module used to control the selection and conversion of these voltages into numeric "count" values.

The functional relation between the numeric count value and its corresponding physical quantity is one of several forms depending upon the specific Input Sensor Device utilized.¹

The AIS operates at less than electronic speed, one operation normally requiring 25 milliseconds. The AIS is not normally directly used by a functional program. Analog values are read by queueing requests to a scan control program.

NOTE 1: Additional information may be found in the General Electric Co. Technical Information Series Report No. R62 PC-1 titled "A Programmer's Handbook of Process Sensors".

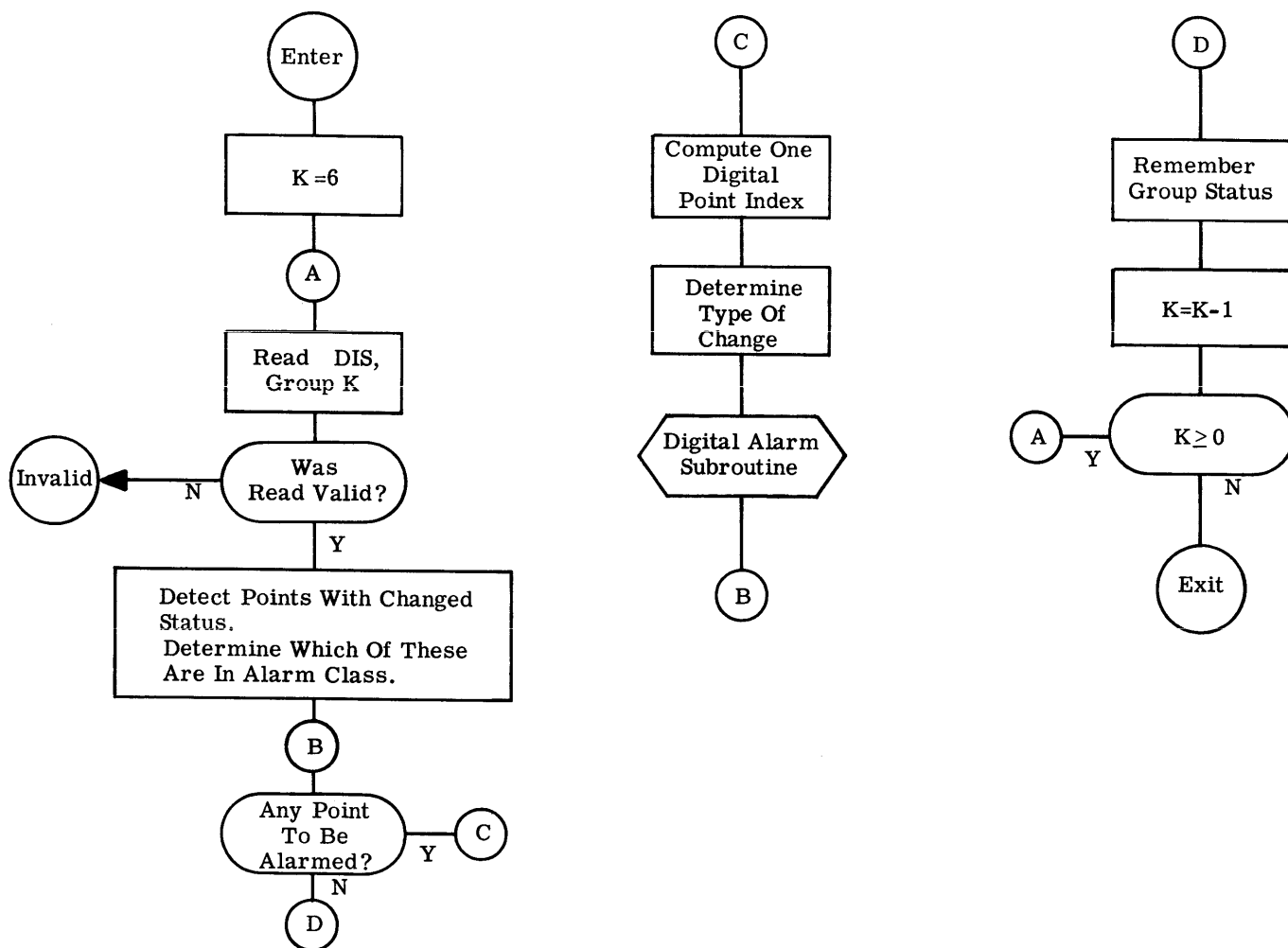


Figure IX-1. Scan For Digital Alarm (Example 2)

The Analog to Digital Converter (A/D Converter), which measures the input voltage and generates an equivalent numeric count value, is the heart of the AIS. Two different types of A/D Converters are available in GE/PAC Scanners:

Successive Approximation A/D Converter, GE/PAC 4130

Integrating A/D Converter, GE/PAC 4135

Each AIS module must have one of the above types; it may not have both. If both types are required in a given application, two GE/PAC scanner modules must be supplied. Refer to paragraph 7.3.1 for description of each type.

The scanner is operated by outputting a Scan Command Word (Figure IX-2) from the Arithmetic Unit (OUT instruction) to the Scan Command Register T. After the conversion is completed, the data (count value) may be read into the computer (IN instruction). Detailed operating instructions are given on page IX:4.

There are two modes of operation as seen from the program control viewpoint:

1. Addressing of inputs individually
2. Addressing of inputs in sets of N (N=2, 4)

Any AIS can be used in the first mode. The second mode is limited to use with scanners having the "Automatic Group Advance" option so that one Scan Command Word controls two or four inputs.

Each Scan Command Word (SCW) uniquely defines a single analog input when the AIS is used in the "single input mode". A "0" in bit 18 of the Scan Command Word indicates this mode. Each group may consist of a maximum of 256 inputs.

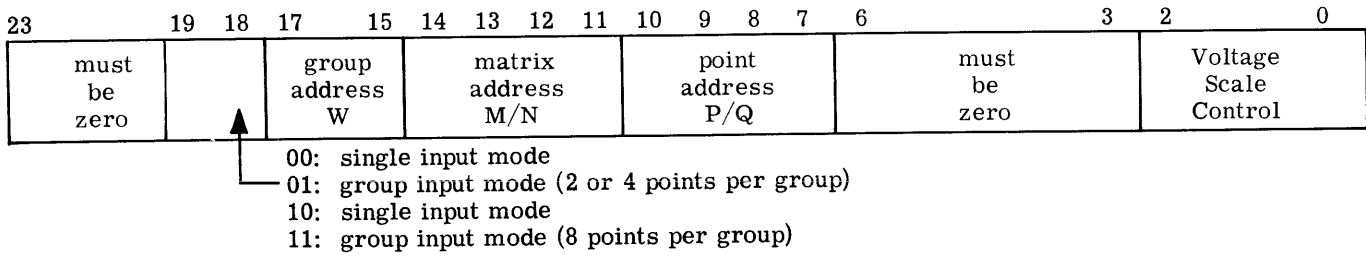
Each Scan Command Word (SCW) implicitly specifies a set of N-W analog inputs when the AIS is used in the "group input mode". A "1" in bit 18 of the Scan Command Word indicates this mode. The initial OUT instruction will result in the conversion of the addressed input (Group W). Succeeding IN instructions will increment the Group Address and cause conversion of the inputs in Groups W+1, W+2, .. N-1.

```

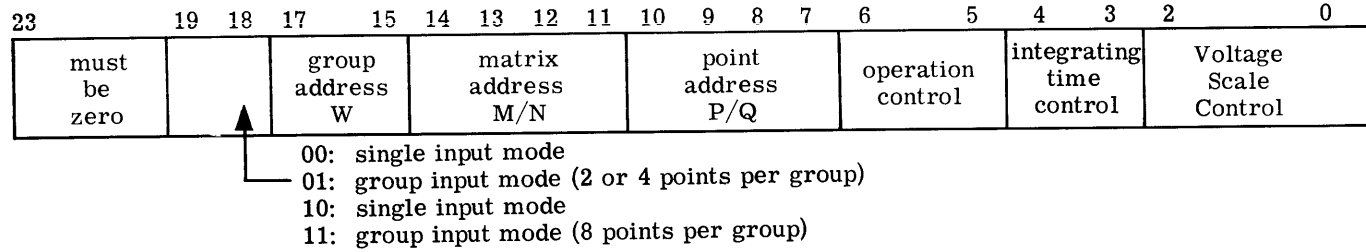
DIS          CON  /DDDD          DEVICE ADDRESS
LDX          DIS, 4
IN           3, 4
TEV         20
BTS         INVALID          B  IF (VALID READING) --, INVALID, 010
TOD         5
REV         18                010 B  IF (BIT5 * BIT 18) --, 020,
BTS         (TRUE)
           (FALSE)                020
*
*          SCAN FOR DIGITAL ALARM ROUTINE
*
*          EXAMPLE 1
*          FIGURE
K           EQL
L           EQL
I           EQL
DIS         EQL (DEVICE ADDRESS)
DSA         LXX 6, K            000    K = 6
           LXX 132, L          005    L = 6 * 22
DSAA        IN  DIS, K
           TEV 20
           BTS INVALID          B  IF (VALID READING) --, INVALID, 010
           STA TEMP
           ERA DISVAL, K
           ANA ALARM, K
DSAB        TZE DSAD          015    B  SAVE GROUP,
           BTS DSAD          020    DETERMINE WHICH ONE
           IAI
           CLZ
           LXC I
           PAI
           RBK 0, I
           STA TEMP1
           LDA L
           ADD I
           STA J
           LDA TEMP
           ERA NORMAL, K
           IBK I
           STA COLOR
           SPB
           LDX J, 3
           LDA COLOR
           LDA TEMP1
           BRU DSAB
           LDA TEMP          030    REMEMBER GROUP STATUS
           STA DISVAL, K
           INX -22, L
           DMT K
           BTS DSAA
           (EXIT)          040    L = L - 22
           BSS 1
           BSS 1
           BSS 7
           BSS 7
           BSS 7
           BSS 7
           X 1 = ALARM, 0 - DO NOT ALARM
*
*          EXAMPLE OF DIGITAL QUESTION
*          EXAMPLE 2
*          FIGURE

```

SCAN COMMAND WORD (Successive Approximation Converter)



SCAN COMMAND WORD (Integrating Converter)



C Register Format Converted value

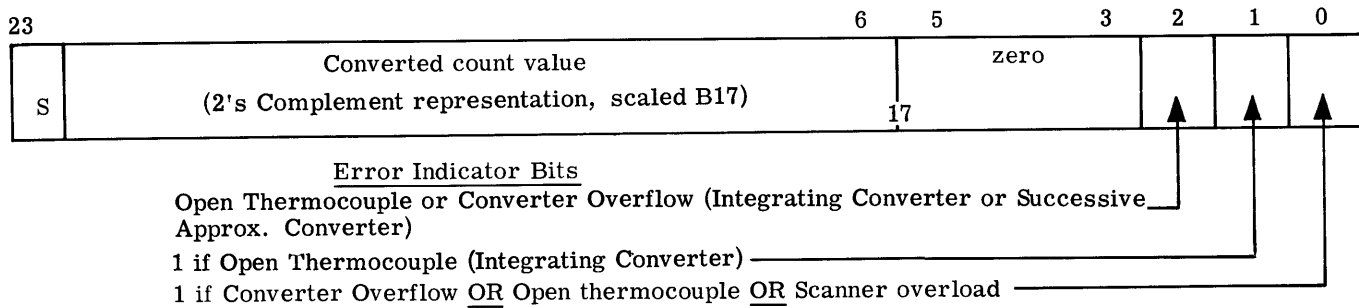


Figure IX-2. AIS Control and Data Formats

The group input mode of operation allows a higher scanning rate than does the single input mode. Scanning rates are summarized in the following table:

Successive Approximation A/D Converter (Hg Relays)

	Single Input Mode	Group Input Mode 2 pts/group	Group Input Mode 4 pts/group
Free-Running Scanner	50	88	148
4040 Programmed Control	45	80	127
4060 Programmed Control	49	87	140

Integrating A/D Converter (Hg Relays)

	Single Input Mode	Group Input Mode 2 pts/group	Group Input Mode 4 pts/group
Free-Running	25	N/A	N/A
4040 Programmed Control	-	N/A	N/A
4060 Programmed Control	24	N/A	N/A

In order to permit successful operation of the AIS in the group input mode, all points to be so used must be so arranged that inputs with like Matrix and Point addresses (SCW₁₄₋₇) have the same Voltage Scale. Points to be used in the single input mode can be arbitrarily "grouped".

Analog to Digital Converters

The Successive Approximation A/D Converter successively generates the sign and each of 12 bits of the count value by comparing the input voltage to known signed voltages. The conversion operation is accomplished in approximately 700 microseconds. The resulting numeric count value represents a voltage value which occurred some time during the 700 microsecond period. The count value is placed right justified into C₂₃₋₆.

The Integrating or "Voltage to Frequency" Converter generates a sequence of pulses whose instantaneous signed pulse rate is directly proportional to

the instantaneous signed input voltage. A counter circuit accumulates the number of pulses occurring during a known period of time (Integrating Time) and places it right justified into C₂₃₋₆. It is an "integrated average" over the specified time period.

The Successive Approximation converter is considerably faster than the Integrating converter. It measures instantaneous voltage and is sensitive to noise errors. Therefore, every input signal is usually "filtered" through an individual resistance-capacitance filter circuit. Unfortunately these circuits frequently introduce another type of error called "common mode". The advantage of the Integrating converter is that it tends to filter the input signal; filter circuits are not always necessary.

SPECIFICATION OF THE SCAN COMMAND WORD

This section describes the various fields which make up the Scan Command Word.

Sensor Address SCW 16-7

Bits 16 thru 7 uniquely define a single analog input.

Bits 16 and 15 specify the input's Group Address.

Operation Mode SCW 18

Bit 18 specifies the mode of Operation for single or group inputs.

Voltage Scale Control SCW 2-0

Bits 2-0 specify the voltage range (-FSV volts to + FSV volts) within which the input voltage V, is presumed to lie. The scanner generates (in the C register) a signed numeric count value proportional to the input voltage when the voltage lies within this range. If the voltage is not within the specified range, a signed maximum count value is generated and the converter overflow bit(s) is set. To obtain maximum precision, the voltage scale should be chosen so that the magnitude of the input voltage falls between 1/2 FSV and FSV.

The measured numeric voltage is equal to:

Successive Approximation Converter:

$$\text{numeric voltage} = \frac{\text{FSV}}{4000} * \text{numeric count value}$$

Integrating Converter:

$$\text{numeric voltage} = \frac{\text{FSV}}{\text{Full Scale Counts}} * \text{numeric count value}$$

Operation Control SCW 6-5

Scanners with the Successive Approximation Converter only measure DC voltages. Scanners with

the Integrating Converter measure either AC or DC voltages and can be used to measure the pulse rate of non-continuous voltages. AC measurements are restricted to multiples and submultiples of 60 cycles (50 cycles when using a 50 cycle converter).

Integrating Time Control SCW 4-3

This field is applicable only to scanners with integrating converters. Refer to paragraph 7.2.1 for a discussion of its usage.

Unused Fields SCW 23-19, SCW 17

These fields are reserved for future use with automatic scanning functions. In specific applications, program information may be placed in the fields.

Converted Count Value C23-6

The 2's complement representation of the converted numeric count value is placed in C₂₃₋₆. The AIS is fully compatible with GE/PAC 2's complement fixed point arithmetic.

Error Indicator Bits C2-0

Bits C₂₋₀ are error indicators associated with the preceding scan operation.

<u>C₂₋₀</u>	<u>Last Scan Operation</u>
------------------------	----------------------------

000	No error. Count value is valid.
001	Scanner Overload error. Count value is meaningless.

This is a rare hardware failure within the point selection circuitry of the Common Scanner Control (GE/PAC 4100). An Alarm message should be printed stating "Scanner Overload" and giving the point address. The AIS may or may not be usable for reading other analog inputs.

010	Undefined (hardware error)
011	Integrating A/D Converter: An open or high resistance thermocouple was detected during the last scan operation. Count value is meaningless. An alarm message should be printed.
100	Undefined (hardware error)
101	Converter Overflow: Count value is maximum, but meaningless.
110	Undefined (hardware error)
111	Successive approximation A/D Converter: Converter Overflow error. If the input device was a thermocouple, this may be indicative of an open or high resistance thermocouple. Count value is maximum value, but meaningless.

Integrating A/D Converter:
Both Converter Overflow and Open Thermocouple errors. Count value is meaningless.

Instructions

Program control over the AIS is accomplished through the use of the following instructions:

ØUT AIS

ØUT transfers a Scan Command Word from the A Register to the AIS's Scanner Command Register T, resets the Scanner Ready and Data Ready signals, and initiates one complete scan operation (connection of a specified input point to the scanners voltage measuring circuitry, signal amplification if required,

and conversion to an integral "count" value). The time required to effect the operation depends upon the A/D Converter. The Successive Approximation Converter requires approximately 25 milliseconds; the time for the Integrating Converter depends on the Integrating Time specified (see Figure IX-3). When the operation is completed, the Data Ready signal is set. This signal may be used to request a Program Interrupt. ØUT may be executed at any time; if the AIS is in operation, this previous operation will be aborted. The (A) is preserved by this instruction.

Voltage Scale Control

SCW 02-00	FULL SCALE VOLTAGE (FSV)			
	Successive Approximation Converter, Model 4130			VIDAR Model 4135 Integrating Converter
	Model 4120 (REDCOR) Low-Level Amplifier (Obsolete)	Model 4121 (PRESTON) Low-Level Amplifier	Model 4127 (PRESTON) High-Level Amplifier	
0	80 mv	10 mv	80 mv	10 mv
1	40 mv	20 mv	160 mv	20 mv
2	20 mv	40 mv	320 mv	40 mv
3	--	80 mv	640 mv	80 mv
4	10 mv	160 mv	2.5 V	160 mv
5	--	10 V	10 V	320 mv
6	--	10 V	10 V	640 mv
7	--	10 V	10 V	1 V

Operation Control for Integrating Converter

SCW 06-05	Mode of Operation
0	Voltage, DC
1	Voltage, AC
2	Voltage, DC with Open Thermocouple detection.
3	Count pulses per unit Integrating Time.

Integrating Time Control for Integrating Converter

SCW 04-05	Integrating Time (MS)		Full Scale Counts at 1000 KC Count Rate		Operation Time (MS) Single Input Mode	
	60 cycle	50 cycle	60 cycle	50 cycle	60 cycle	50 cycle
0	16.67	20	1:16,667	1:20,000	35.2	38.5
1	33.33	40	1:33,333	1:40,000	51.9	58.5
2	100.	100	1:65,535	1:65,535	118.5	118.5
3	1000.	1000	1:65,535	1:65,535	1018.5	1018.5

Figure IX-3

ØPR AIS

ØPR causes the AIS to go through one partial scan operation. In the single input mode, a re-conversion is made on the previously specified input point. The Data Ready signal will be set in approximately 750 μ s with the Successive Approximation Converter, and the Integrating Time with the Integrating Converter.

In the group input mode, the Group Address is reset to zero and re-conversion occurs. The Successive Approximation Converter will set the Data Ready Signal for the input of Group 0 in 3 MS. The Integrating Converter will set the signal in the specified integrating time plus 2 MS.

IN AIS

IN transfers the count value and error indicator bits from the Scanner Converter Register C into the A register. The contents of the C register is not destroyed. In the group input mode, the Data Ready signal is reset, and conversion of the input in the next group is initiated. The Data Ready signal for this next input will occur in 1.7 MS with the successive approximation converter.

The execution of IN to transfer the count from the last group will set the Scanner Ready signal. This signal may be used to effect a Program Interrupt. Any additional IN will reread the count from the last group and initiate a re-conversion of the input in the last group.

JNR AIS

JNR will transfer Program Control to the second sequential location if the Data Ready signal is reset. If the signal is set, Program Control is transferred to the 1st sequential location. Normally this flip-flop condition is wired into the automatic priority interrupt as discussed following.

JNE AIS

JNE will transfer Program Control to the second sequential location if no error occurred during some preceding scan operation; otherwise Program Control is transferred to the first sequential location. The error flip-flop is reset by JNE.

ACT AIS

ACT forces the Scanner Ready and JNR signals to be set for 16 μ s. If the Scanner is ready and if the Scanner Ready (JNR) signal is a Program Interrupt Input, this Program Interrupt is requested; otherwise the interrupt is not requested.

INTERRUPT OPERATION

Single Interrupt Operation

Hardware Required:

One 2-input Program Interrupt
Change Input: JNR Signal
Level Input: JNR Signal

Program Required:

SPB instruction in interrupt location
1 I/O Buffer Driver Program
1 Scan Command Buffer List
1 Data Buffer List

Three Interrupt Operation (GE/PAC 4050/4060 only)

Hardware Required:

GE/PAC 4025 Arithmetic Unit
One 2-input Program Interrupt
Change Input: Scanner Ready Signal
Level Input: Scanner Ready Signal
One 2-input Program Interrupt
Change Input: Data Ready Signal
Level Input: Data Ready Signal
One 1-input Program Interrupt
Change Input: ØDL List-Empty Echo Signal

Program Required:

ØDL instruction in Scanner Ready interrupt location.
IDL instruction in Data Ready interrupt location.
1 Scan Command Buffer List
1 Data Buffer List
SPB instruction in Echo interrupt location.

ANALOG INPUT POINT ADDRESSING SCHEMES

Each input is subject to 3 different identification schemes:

System Address (e. g., BF101 point identification)

This address is assigned by the customer and/or the system analyst.

Termination (Scan Command) Address (WMNPQ)

This address is assigned by the system engineer and refers to the physical location of the point's termination in the AIS Termination Rack.

Point Index

This address is assigned by the Programming Analyst. It should ideally bear some simply functional relationship to the System Address.

ACCURACY IN MEASUREMENT OF ANALOG QUANTITIES

Measurement of an analog quantity involves four separate steps each one contributing to measurement inaccuracies:

Conversion of the analog quantity to an analog voltage by a transducer. Equations defining this relationship for standard transducers are usually accurate to no more than 1/2%. Additional accuracy can be obtained by individual calibration of each sensor.

Transmission of the analog voltage from the sensor to the AIS. Errors are introduced by voltage drops and electromagnetic inductive effects (noise).

Conversion of the analog voltage to a numeric voltage by the AIS and read in of this numeric value into the computer's memory. The conversion process is subject to several errors which can be corrected by program. Details are given below.

Conversion of the corrected numeric voltage to a numeric quantity corresponding to the original analog quantity. The computer program uses the inverse to the equation referred to in second paragraph under Accuracy in Measurement of Analog Quantities.

The GE/PAC scanner can be programmed to convert an input voltage into a numeric count value that differs from the true voltage value by no more than 0.001 (full-scale voltage).

Inaccuracies in measurement of input voltages by a GE/PAC scanner are determined by the algebraic sum of three types of errors. These errors are Gain Errors (errors in the slope of the input-output relationship), Offset Errors (the degree by which the intercept does not go through zero), and Repeatability Errors (noise). This total error, which is no more than 0.005 (full-scale voltage), is subject to reduction by programming techniques.

Offset errors can be reduced by making a weighted average measurement¹ on a short circuit (zero input) and reducing all other readings by this measured "offset" value. The equation for this would be:

True Reading = Actual Reading - Offset Reading
where signed Offset and Actual Readings are made on the same voltage scale.

Repeatability Errors can be reduced by "digital" filtering. Simply stated, this is the process of averaging¹ a number of readings to minimize the effects of noise in the system.²

The Offset Correction Term both will vary as a function of time and temperature. Therefore, measurements must be made to determine these quantities often enough so that any changes between measurements will be small. Experience indicates an interval of approximately five minutes would be reasonable under normal environmental conditions. If the machine is located in an environment where temperature changes are very rapid (0.5°/min.), shorter intervals would be desirable.

The degree of improvement in each of these error terms is quite dependent upon several external effects, but generally speaking, improvements in the Repeatability Errors will be 2 or 3 to 1. The Offset Error improvement is more dependent upon the scale used, but on more sensitive scales the improvement may be as high as 5:1 also.

A special type of scanner measurement error occurs whenever the scan operation results in a converter overflow (the input voltage magnitude exceeds the specified Full-Scale Volts). The resulting numeric count value is meaningless. Furthermore, the voltage value measured in the next scan operation may be in error because of residual effects from the preceding voltage overload. The magnitude of this error is a function of this voltage overload, but never should exceed 0.01 (full-scale voltage).

If this error is unacceptable, then a "dummy" scan (preferably of an offset value) may be inserted following the converter overflow to absorb the residual voltage effects.

MULTIPLE OUTPUT DISTRIBUTOR (4300)

GENERAL

The Multiple Output Distributor (MOD) consists of a Multiple Output Controller and "Output Functions". The Multiple Output Controller (MOC) is a module used for outputting data from the arithmetic unit to Output Functions within the MOD which remember the data and control the operation of the following types of output devices:

1. Display Output Devices
2. Binary Output Devices
3. Analog Output Devices

The MOD is operated by outputting a Command Word (figure below) from the computer to the MOC's Command Register. The MOC will then transfer the data to the memory of the Output Function specified by the Group Address of the Command Word. The

NOTE 1: A suggested weighted average is $7/32 * (\text{sum of last four readings}) + 1/8 * (\text{new reading})$ when a reading is made once each minute.

NOTE 2: Additional information may be found in the General Electric Co. Technical Information Series Report No. RG1-OC-1 titled "Signal Filtering in Digital Control Computer Systems".

Program Control is accomplished by use of a SPB instruction in an inhibitable interrupt location. This interrupt is a type 2 two-input interrupt (see Section VI). The Alternate Ready signal is its change input; its level input is the JNE signal. This interrupt will be requested if:

1. The Alternate Ready signal occurs, and
2. No MØD error occurred.

Error detection is provided by use of a SPB instruction in an inhibitable type 1 one-input interrupt location (see Section VI). The JNE signal is the interrupt input. This error interrupt will be requested whenever a MØD error is detected.

OUTPUT FUNCTIONS

Three standard Output Functions are described below. The memory elements in each are bi-stable latched relays (see figure below). The relay requires no more than 4MS to switch from one position to the other. Small permanent magnets will then hold the switch in that position after electric power is removed from the relay.

Note also that the Function Group Address does not uniquely specify an Output Function ---- that is, several Output Functions could have the same Output Function Group Address.

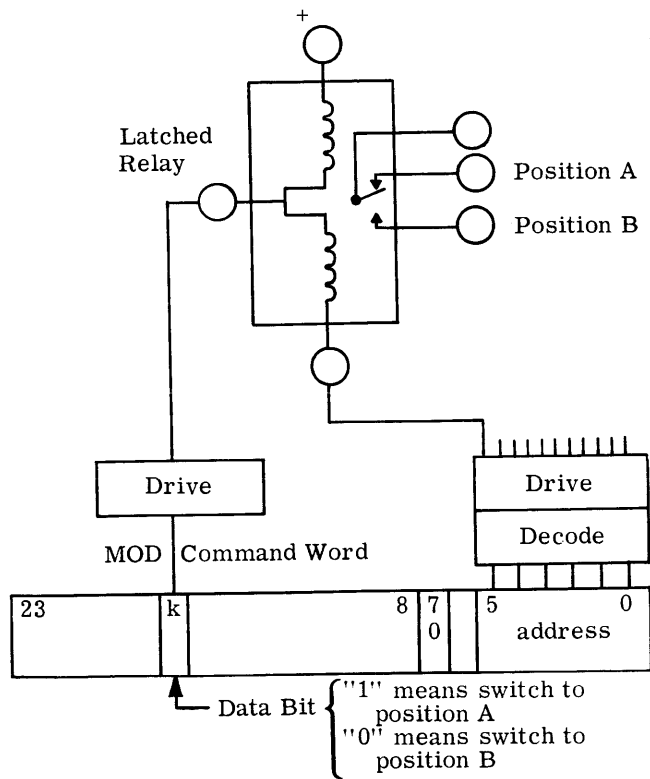


Figure IX-4

Binary Output Function (4367)

The Binary Output Function provides control for 8 logic (binary) outputs corresponding to 8 data bits of the MØD Command Word.

Display Output Function (4360)

The Display Output Function provides the control for one BCD display digit (including 4 bit BCD to decimal conversion if required). It requires 4 data bits of a MØD Command Word.

Analog Output Function

The Analog Output Function provides control for conversion from a 8 or 10-bit positive binary integer in a MØD Command Word to an analog voltage output.

The Output Function is not generally the final output actuating device. That is, there will normally be one or more levels of electromechanical or pneumatic control between the Output Function and the actuating device. The MOD's Ready Signal can be used neither to indicate the actual time of the output action nor as proof that the output will actually occur. If reliability considerations require, a set of contacts may be attached to the actuating device. Its status may then be read by the Digital Input Scanner.

PROGRAMMING USAGE

From the Program Control viewpoint, all outputs are initiated by a MOD Driver Subroutine, never by the originating subprogram. Further, each output bears one of two relationships to the originating program:

1. The originating program need not be notified when the output operation is initiated - that is, program continuation is independent of the output.
2. The originating program must be notified when the output operation is initiated - that is, program continuation is dependent upon the output.

Bit 6 of the MOD Command Word is used by the originating program to specify to the MØD and its Driver Subroutines which relationship applies to that output.

Figure IX-4 is the flow chart and coding showing MOD usage with GE/PAC 4040 and GE/PAC 4060.

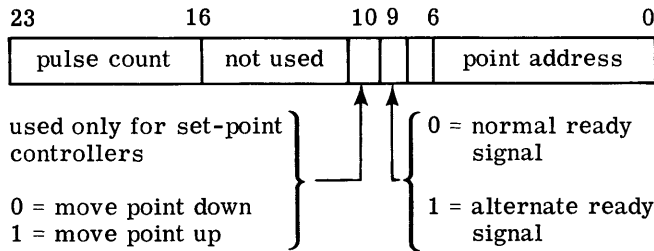
TIMED OUTPUT CONTROLLER (4302)

GENERAL

The Timed Output Controller is used to select and then time contact closures. The controller will select and energize one of a maximum of 128 relays and then hold it energized for a period of time specified by the control word. The holding period is the specified number of counts of a clock pulse.

The pulse frequency is determined by the application, but must fall within the range of 20 cps to 20KC.

An optional capability of the TOC permits its use to operate GE/MAC and other set-point controllers requiring polarized pulse train inputs. In this usage, the energized relay gates a pulse train to the set-point controller, the number of control pulses being equal to the specified number of clock pulses. The TOC is operated by outputting a command word (figure below) from the computer to the TOC's command register.



Overload detection circuitry will abort the output operation and set a TOC Overload Indication if the TOC attempts to select more than one relay at a time.

The TOC is designed to be usable with zero, one, or three Program Interrupts. The Arithmetic Unit (4025) requires three interrupts for most efficient usage, but three interrupts will not increase the performance with Arithmetic Unit 4020 over that afforded by using one interrupt.

INSTRUCTIONS

OUT TOC

OUT transfers a command word from the A register to the TOC command register and initiates the operation of the TOC. The contents of the A register are unchanged. Bit 9 of the Command Word specifies which of the two "ready" signals will be reset at the beginning of the operation and set at the completion of the designated holding period.

OUT may be executed at any time. If the TOC is in operation, however, the OUT will be ignored.

JNR TOC

JNR will transfer Program Control to the second sequential location if the normal ready signal is reset (Bit 6=0). If it is set (Bit 6=1), Program Control is transferred to the first sequential location. In the case of a 1 interrupt TØC, the Normal Ready signal is used as the interrupt input. In the case of a 3 interrupt TØC, each Ready signal is a separate interrupt input.

ACT TØC

ACT forces the Normal Ready signal to become "reset" for 8 us. If this signal is a Program Interrupt Input and if the TOC is not in operation, a Program Interrupt will be requested.

JNE TØC

JNE will transfer Program Control to the second sequential location if a TOC overload did not occur during a preceding output operation. Otherwise, Program Control is transferred to the first sequential location and the MØD Overload Indicator is reset.

INTERRUPT OPERATION (4040)

Single interrupt operation is recommended. The Normal Ready signal is used as the input to an inhibitible one-input program interrupt (type 1 interrupt, see Section VI). Interrupt is conditionally requested by the ACT instruction. Bit 9 of the command word must be reset to a "0" just prior to outputting via the ØUT instruction.

INTERRUPT OPERATION (4060)

Three interrupt operation is recommended.

Data output is accomplished by a ØDL instruction in a non-inhibitible interrupt location. This interrupt is a type 2 two-input interrupt (see Section VI). Its change input is the Normal Ready signal; its level input is the JNE signal. Interrupt is conditionally requested by the ACT instruction. Outputs will be successively buffered from an output list to the TØC until:

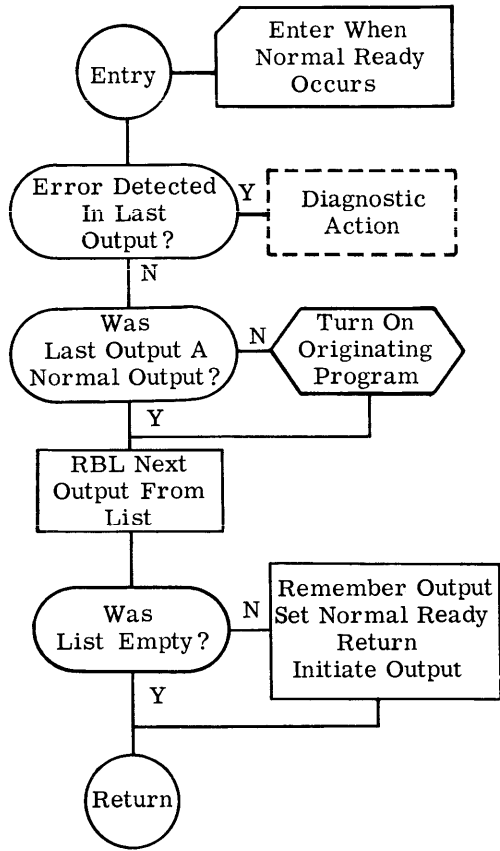
1. The list becomes empty, or
2. The Alternate Ready signal occurs, or
3. A TOC error occurs.

Program Control is accomplished by use of a SPB instruction in an inhibitible interrupt location. This interrupt is a type 2 two-input interrupt (see Section VI). The Alternate Ready signal is its change input; its level input is the JNE signal. This interrupt will be requested if:

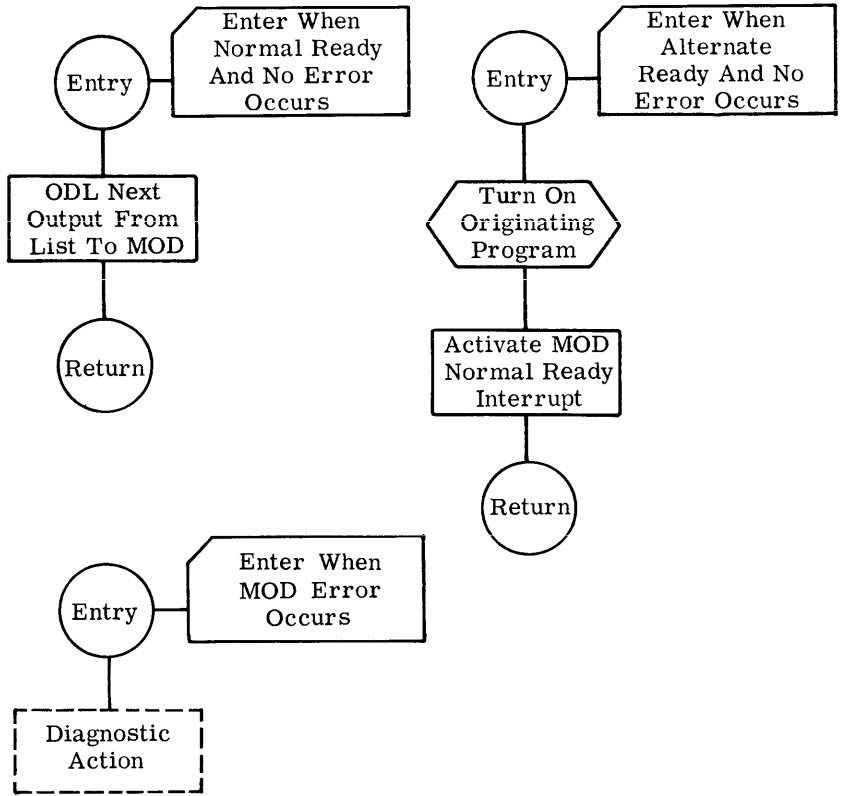
1. The Alternate Ready signal occurs, and
2. No TOC error occurred.

Error detection is provided by use of a SPB instruction in an inhibitible type 1 one-input interrupt location (see Section VI). The JNE signal is the interrupt input. This error interrupt will be requested whenever a TOC error is detected.

GE/PAC 4040



GE/PAC 4060



MODLST

F	N			L
23	7	6	5	0
		0		
		0		
		0		
		1		
Program Number				
		0		

Output Buffer List
Model 4020 and 4025

Figure IX-4. Multiple Output Distributor Drive Routines

PROGRAMMING USAGE

From the Program Control viewpoint, all outputs are initiated by a TOC Driver Subroutine, never by the originating subprogram. Further, each output bears one of two relationships to the originating program:

1. The originating program need not be notified when the output operation is initiated - that is, program continuation is independent of the output.

2. The originating program must be notified when the output operation is initiated - that is, program continuation is dependent upon the output.

Bit 9 of the TOC Command Word is used by the originating program to specify to the TOC and its Driver Subroutines which relationship applies to that output.

* INTERRUPT BRANCH VECTOR LOCATION				
	SPB	MODDRV		INHIBITABLE - IF NORMAL READY SIGNAL
* MOD DRIVER ROUTINE--				
MODDRV	STX	MODX, 1	00	
MOD	EQL			DEVICE ADDRESS.
	STA	SAVEA		
	JNE	MOD		B IF (ERROR DETECTED IN LAST OUTPUT)--,
	BRU	DIAGNOSTIC		X 10, DIAGNOSTIC ROUTINE.
	LDA	MODCOM	10	B IF (LAST OUTPUT A NORMAL OUTPUT)--,
	TEV	6		X 20, 30.
	BTS	MOD20		
	RBL	MODLST	20	REQUEST TURNON OF ORIGINATING PROGRAM
	NOP			
	STA	*+3		
	LDZ			
	SPB	TPNC01		CALL MONITOR-TURN-PROGRAM-ON-SBR (
	BSS	1		X 0, PROGRAM-NUMBER).
MOD20	RBL	MODLST	30	FETCH NEXT OUTPUT COMMAND WORD.
	BRU	MOD40		B IF (LIST EMPTY) --, 40, 50.
	STA	MODCOM	40	REMEMBER OUTPUT COMMAND WORD.
	RBK	6		FORCE NORMAL READY SIGNAL.
	OUT	MOD		INITIATE OUTPUT OPERATION.
MOD40	LDA	SAVEA	50	RETURN TO INTERRUPTED PROGRAM.
	LPR	MODX		
MODLST	CON	0,000000L		
	BSS	L		
MODX	BSS	1		
SAVEA	BSS	1		
MODCOM	BSS	1		

* MULTIPLE OUTPUT DISTRIBUTOR DRIVE ROUTINE FOR GE/PAC 4040				

* INTERRUPT BRANCH VECTOR LOCATIONS				
	ODL	MODLST		UNINHIBITABLE - IF NORMRDY AND NO ERROR
	SPB	MODDRV		INHIBITABLE --- IF ALTNRDY AND NO ERROR
	SPB	DIAGNOSTIC		INHIBITABLE --- IF MOD ERROR.
* MOD DRIVER ROUTINE --				
MODDRV	STX	MODX, 1	100	
MOD	EQL			DEVICE ADDRESS.
	STA	SAVEA		
	RBL	MODLST	120	REQUEST TURNON OF ORIGINATING PROGRAM
	NOP			
	STA	*+3		
	LDZ			
	SPZ	TPNC01		CALL MONITOR-TURN-PROGRAM-ON-SBR (
	BSS	1		X 0, PROGRAM-NUMBER).
	ACT	MOD	130	ACTIVATE NORMAL INTERRUPT.
	LDA	SAVEA	150	RETURN
	LPR	MODX		
MODLST	CON	0,000000L		
	BSS	L		
MODX	BSS	1		
SAVEA	BSS	1		

* MULTIPLE OUTPUT DISTRIBUTOR DRIVE ROUTINES FOR GE/PAC 4060				

X MAN COMMUNICATION

PERIPHERAL BUFFER (Model 4201)

GENERAL DESCRIPTION

The Peripheral Buffer is a module used by the computer to control the operation of peripheral devices. Control information flow is from the I Register to the module. Data information flow is by character between the arithmetic unit and the module.

Peripherals available for use with the Buffer are:

Paper-Tape Reader, Input
7-Bit Binary
Photoelectric

Paper-Tape Punch, Output
7-Bit Binary

Fixed-Carriage Typer, Output
7-Bit BCD

Long-Carriage Typer, Output
7-Bit BCD

Card Reader, Input
12-Bit Binary

Input/Output Typer
7-Bit BCD

A maximum of eight peripherals can be attached to any peripheral buffer. One of these must be a paper-tape reader.

The Peripheral Buffer can be connected to any channel; several buffers can share the same channel. However, in order to permit the use of a standard Load Program, the Buffer to be used for console peripherals will have device addresses 11DD₈.

Each Peripheral Buffer provides eight device addresses:

Octals DD00 through DD07.

Console peripherals have standard addresses. Other peripherals are assigned addresses by the requisition engineer.

<u>Console Devices</u>	<u>Standard Addresses</u>
Paper-Tape Reader	1100 ₈
Paper-Tape Punch	1101 ₈
Fixed-Carriage Typer	1102 ₈

The normal operation of a peripheral device can be described in three statements:

● Operation Initiation

The operation cycle is initiated by an IN (OUT, IDL, ODL) instruction. The following actions occur:

1. One character of data is transferred (see IN (OUT) below),
2. The Peripheral Buffer becomes unavailable (the PBA Signal is reset),
3. The device becomes not ready (the device's Ready Signal is reset),
4. The device's "Deadman Timer" begins timing, and
5. Mechanical operation of the device begins.

● Peripheral Buffer Availability

The device requires the use of the Peripheral Buffer only during the first portion of its operation cycle. The device will then release the Peripheral Buffer and allow it to become available (the PBA Signal is set). At this time the Buffer can be used to initiate the operation of any peripheral that may be ready.

● Operation Completion

At the completion of the device's operation cycle, the following actions occur:

1. The device becomes ready (the device's Ready Signal is set) and
2. The device's Deadman Timer is turned off.

Each peripheral has associated with it an independent Deadman Timer which will detect and indicate the failure of the device to complete its operation cycle. If a device fails to turn off its Deadman Timer within 4 to 8 seconds after an operation initiation, the device's Deadman Error Indicator will be set. If the failure occurs prior to Peripheral Buffer Availability, the Peripheral Buffer will remain unavailable until the execution of a JNE instruction releases it. If the failure occurs after the PBA signal is set, the Buffer can be used to operate other peripheral devices. In either case, the device's Ready Signal will remain reset until the proper JNE instruction is executed.

More than 90 percent of all peripheral failures will occur prior to Peripheral Buffer Availability. These errors may be due to:

1. The attempted operation of a non-operable device. A device is non-operable if it has a mechanical defect, lacks a-c power, lacks d-c power, is non-existent, or has been switched off-line. A non-operable device will normally appear ready (Device Ready Signal is set) until an attempt is made to operate it.

2. The attempted outputting of an illegal character to a typewriter (an illegal character will not necessarily cause a failure on certain types of typewriters).

INSTRUCTIONS

JNR 25X6DD00

JNR will transfer Program control to the second sequential location if some peripheral device attached to the addressed peripheral buffer is in operation. If all of these devices are ready, Program Control will be transferred to the first sequential location. The test implicitly indicates whether or not the last initiated operation (IN, OUT, IDL, ODL) is completed.

The JNR signal is the logical AND of the individual device ready signals for all eight devices.

The JNR signal may optionally be connected to the Automatic Program Interrupt Module so as to initiate an interrupt whenever the signal changes from "not ready" to "ready". It is connected to the "Peripheral Ready Light" on the computer console so that the light is on when any device is Not Ready.

OUT 2504DD0D

The OUT instruction may be executed at any time to transfer the rightmost seven bits¹ of the A Register to the N Register within the Peripheral Buffer, and initiate one operation of the addressed device. The previous content of the N Register is lost. The contents of the A Register is not changed. Although OUT may be executed at any time, no action will occur if either the Peripheral Buffer or the addressed device is in operation. If the output device is a paper-tape punch, odd parity will be generated on seven bits and be punched as a bit in an eighth channel.

Figure X-1 lists the standard typewriter character set and typewriter codes. Figure X-2 lists the standard punched paper-tape format.

IN 2505DD0D

The IN instruction is executed to transfer the current contents of the addressed device's Read Mechanism to the rightmost seven bits¹ of the A Register and initiate one operation of the addressed device. The remaining bits of the A Register are set to zeros by this instruction.

Although IN may be executed at any time, the information transferred will be meaningless if either the addressed peripheral buffer or device is in operation; the operation of the addressed Buffer and device will not be affected.

If the input device is a paper-tape reader, these seven data bits will be checked for odd parity. Even parity will set the Peripheral Buffer Error Indicator and will light the "Peripheral Error" light on the computer console. The light can be turned off by depressing the "clear" switch. IN should be immediately followed by a JNE instruction to check for correct parity.

JNE PERIPHERAL 2507DDED

JNE transfers Program Control to the second sequential location if the error indicator(s) specified by E is reset. If the specified error indicator(s) is set, Program Control is transferred to the first sequential location and an action specified by E is taken.

<u>E</u>	<u>Error Indicator(s)</u>	<u>Action</u>
000	None	None
001	Addressed Device's Input Demand Indicator	Reset Input Demand Indicator
010	Addressed Device's Deadman Error Indicator	None
011	Addressed Device's Deadman Error Indicator	Reset Device's Deadman Error Indicator
100	Parity Error Flip-flop	None
101	Parity Error Flip-flop	Reset Parity Error Flip-flop
110	Any Deadman Error Indicator in the Addressed Peripheral Buffer or the Parity Error Flip-flop.	None
111	Addressed Device's Deadman Error Indicator or the Parity Error Flip-flop.	Reset Deadman Error Indicator and reset Parity Error Flip-flop

NOTE 1: If the addressed device is a Card Punch (Card Reader), 12 bits of data are transferred.

CHARACTERS AND CODES

<u>CHARACTER</u>	<u>OCTAL CODE</u>	<u>NOTES</u>	<u>CHARACTER</u>	<u>OCTAL CODE</u>	<u>NOTES</u>
0	00		space	20	
1	01		.	33	
2	02		,	73	
3	03		+	60	OL
4	04		-	52	
5	05		*	54	OL
6	06		/	61	
7	07		=	75	OL
8	10		(35	OL
9	11)	55	OL
A	21		\$	53	
B	22		"	76	OL
C	23		>	16	F
D	24		<	36	F
E	25		[12	F
F	26]	34	F
G	27		:	15	F
H	30		;	56	F
I	31		,	57	F
J	41		#	13	F
K	42		@	14	F
L	43		&	32	F
M	44		\	37	F
N	45		↑	40	F
O	46		←	72	F
P	47		%	74	F
Q	50		?	17	F
R	51		!	77	PR
S	62		TAB	140	
T	63		CR	100	
U	64		TAB	141 to 157	OT, PR
V	65		CR	101 to 137	OT, PR
W	66		Black	160	OT, S, PR
X	67		Red	161	OT, S, PR
Y	70		No Action	177	R, PR
Z	71		Punch On	162	R, PR
			Punch ON	164	R, PR
			Control Mode Shift	1XX	IT
			Delete	177	P, PR
			Stop		P, R, PR

Notes:

Characters are applicable to all standard GE/PAC I/O devices unless noted otherwise.

F These characters not available on long-carriage typewriter.

OT Output Typewriters only.

R Tape Reproduction Devices only.

IT Input Typewriters only.

Holding this key down while typing a character generates the octal 1XX which is program interpreted as one of 64 control actions (e.g., typing a # in this mode generates 113₈ in the input typers buffer register).

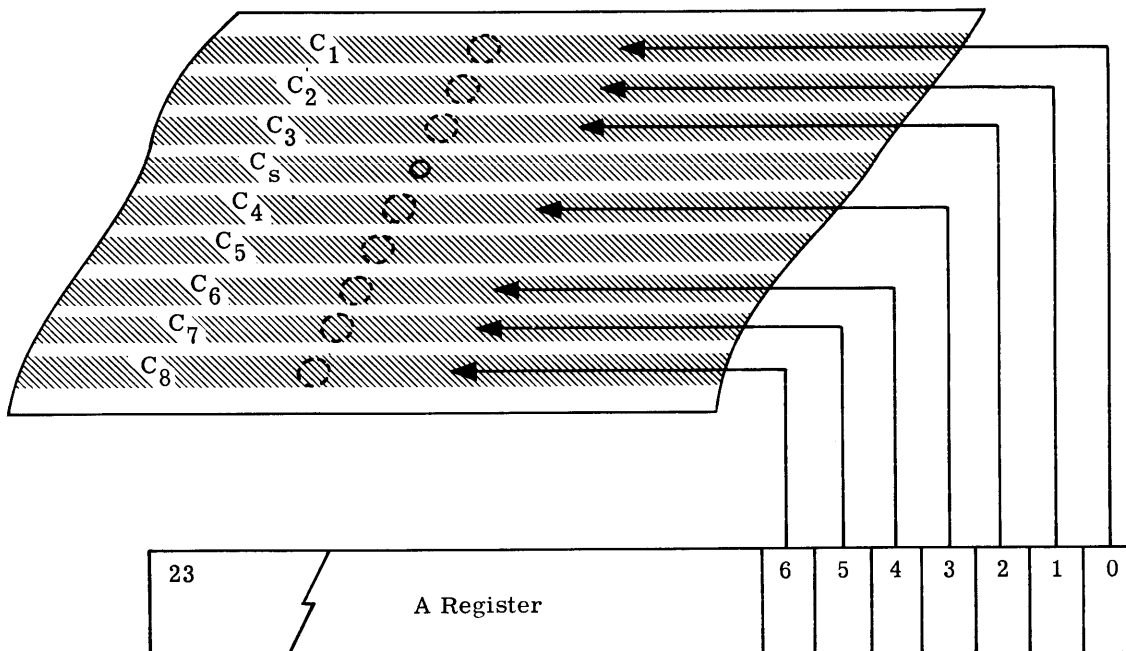
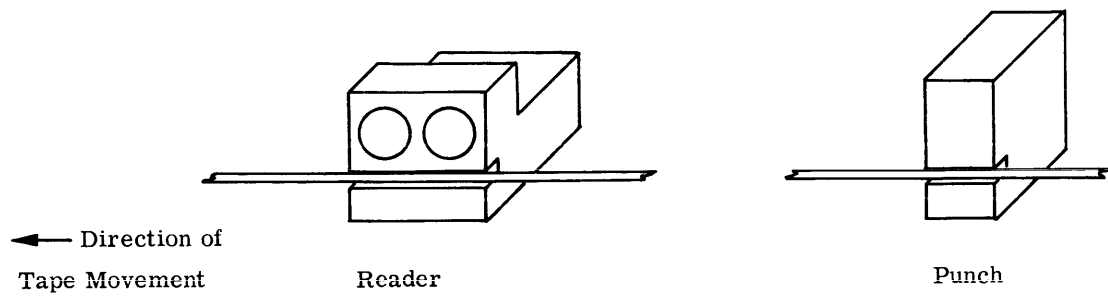
P Tape Preparation Devices only.

PR Illegal printer character.

S Optional on Selectric Typewriter.

OL Optional on long-carriage typewriter.

Figure X-1. Alpha-numeric Characters and Symbols



Paper tape shown moving through a punch and reader provides a frame of reference for the enlarged segment of tape shown below it. Format details are:

1. The paper tape has nine "channels" C_1 through C_8 and C_S running lengthwise along the tape.
2. Channel C_S is the "sprocket hole" channel. A "hole" in this channel defines a "data frame". That is, the hole indicates the presence of data in corresponding "punch positions" of the other eight channels (dotted circles in the sketch). Data frames need not be equally spaced.
3. A hole in a punch position defines a "1" data bit; no hole defines a "0" data bit. Data holes are twice the size of sprocket holes.
4. The IN (OUT) instruction transmits data between a data frame and bits A_{6-0} of the A Register as indicated.
5. Channel C_5 is the "parity channel". A hole is punched as required in order to guarantee an odd number of holes in each data frame.

Figure X-2. GE/PAC Punched Paper-Tape Format

ACT PERIPHERAL 2501DD0D

ACT forces the addressed device's Ready Signal to be reset for 8 μ s. If the device is ready and if this Ready Signal is a Program Interrupt Input, the device's Program Interrupt is requested; otherwise, the interrupt is not requested.

If all peripheral devices are ready and if the JNR signal is a Program Interrupt Input, the JNR Program Interrupt is requested; otherwise, the interrupt is not requested.

INTERRUPT OPERATION

Single Interrupt Operation (one peripheral at a time)

The Peripheral Buffer with the JNR Signal connected to the API Module and one buffer driver program (IDL (ODL) instruction) will operate one device at a time. Interrupt is conditionally requested by the ACT instruction.

GE/PAC 4040

Hardware Required:

- One 2-input Program Interrupt
- Change Input: JNR Signal (Test Line 1)

Program Required:

- SPB instruction in interrupt location
- One I/O Buffer Driver Program
- One I/O Buffer List

GE/PAC 4050/4060

Hardware Required:

- One 2-input Program Interrupt
- Change Input: JNR Signal (Test Line 1)
- Level Input: PBA·PBE Signal

Program Required:

- IDL (\emptyset DL) instruction in non-inhibitible interrupt location
- One I/O Buffer List

Multi-interrupt Operation (time-shared operation)

The GE/PAC Peripheral Buffer with individual Device Ready Signals connected to the API Module and individual buffer driver programs (individual IDL (\emptyset DL) instructions in model 4025) for each device provides the functional equivalent of an individual hardware controller for each peripheral. Each controller would buffer characters between a list in memory and its associated device and operate it semi-independently of all other devices. Each peripheral requires the use of the Peripheral Buffer for only the first part of its complete cycle; the Buffer can then be used to initiate the operation of some other peripheral. This mode allows a greater data throughput rate (e. g., 15 characters can be typed by each of two 15 cps typewriters in approximately 1-1/30 second). Hardware requirements are: one type 2 API input per peripheral device and additional core memory to accommodate individual buffer drivers and character tables. ACT conditionally requests the addressed device's Program Interrupt.

GE/PAC 4040

Hardware Required (per peripheral)

- One 2-input Program Interrupt
- Change Input: Device's Ready Signal (RDY2_m)

Program Required (per peripheral)

- SPB instruction in interrupt location
- One I/O Buffer Driver Program
- One I/O Buffer List

GE/PAC 4050/4060

Hardware Required (per peripheral)

- One 2-input Program Interrupt
- Change Input: Device's Ready Signal (RDY2_m)
- Level Input: PBA Signal for output devices
PBA·PBE Signal for input devices

Program Required (per peripheral)

- IDL (\emptyset DL) instruction in device's non-inhibitible interrupt location
- One I/O Buffer List

Peripheral Device	Time lapse between OUT (IN) instruction and Ready Signals. Peripheral Buffer Available Signal, ms	Device Ready Signal, ms
Typewriter, Fixed Carriage Normal Character, LC Normal Character, UC Carriage Return	 30 95	 40 - 65 65 - 130 up to 800
Typewriter, Long Carriage Normal Character Carriage Return	 40	 65 - 100 up to 800
PT Reader	4	10
PT Punch	4	9
Card Reader Normal Character Card feed		
Input Typewriter, Selectric	Dependent upon	Operator

NOTE: All times are approximate

Figure X-3. Peripheral Operation Timing

Type Size Characters per Inch	Number of characters in print line				
	IBM 11" Carriage	Selectric 15-1/2" Carriage	IBM 12" Carriage	MODEL 20" Carriage	B 30" Carriage
10	85	130	88	167	265
12	102	156	106	201	318
14	-	-	124	234	371

NOTE: The vertical line spacing on all typewriters is 6 lines per inch.

Figure X-4. Typewriter Format Data

Error Detection

The recommended method for detecting a device failure or a parity error is to periodically execute a JNE instruction (2507DD60) placed within a frequently entered program. However, a Peripheral Buffer Error (PBE) Signal is available for optional use as a Program Interrupt input (the PBE signal is equivalent to the JNE signal with E=6).

Hardware Required:

One 1-input Program Interrupt:

Change Input: PBE Signal

OPERATORS CONSOLE

The operator communicates with the digital computer through the operator's console. This console is of special design to meet the needs of the process control computer application. Figure X-5 and 6 show typical desk-top consoles for power plant monitoring applications. Figure X-7 shows a sequence monitoring console for plant start-up and shut-down on the left, deviation indicators in the center, and operator demand functions on the right. Six trend recorders are at the top, and they can be selected to trend record any sensor or calculated value. Figure X-8 shows an electric utility power dispatcher's console for controlling power plant generating units. Figure X-9 and 10 show typical operator's consoles for steel and chemical applications.

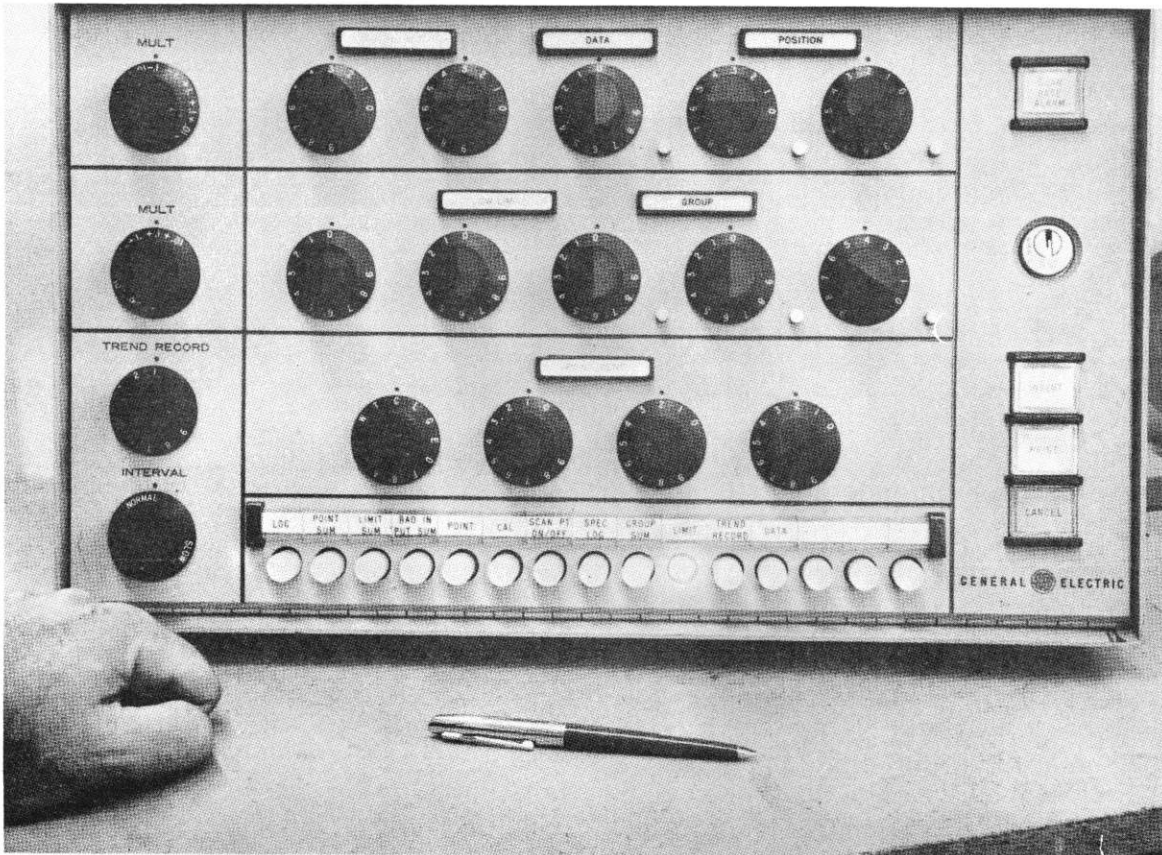


Figure X-5. Operator's Console

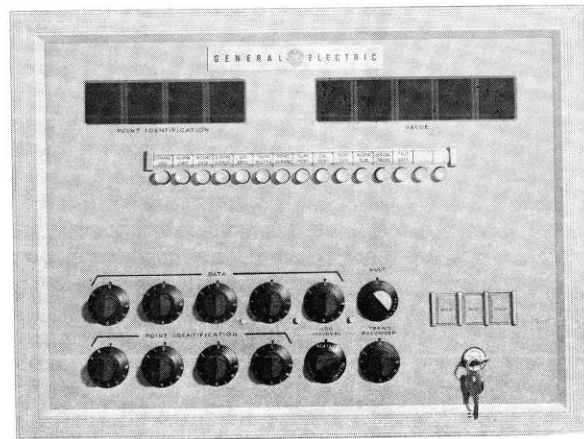


Figure X-6. Operator's Console

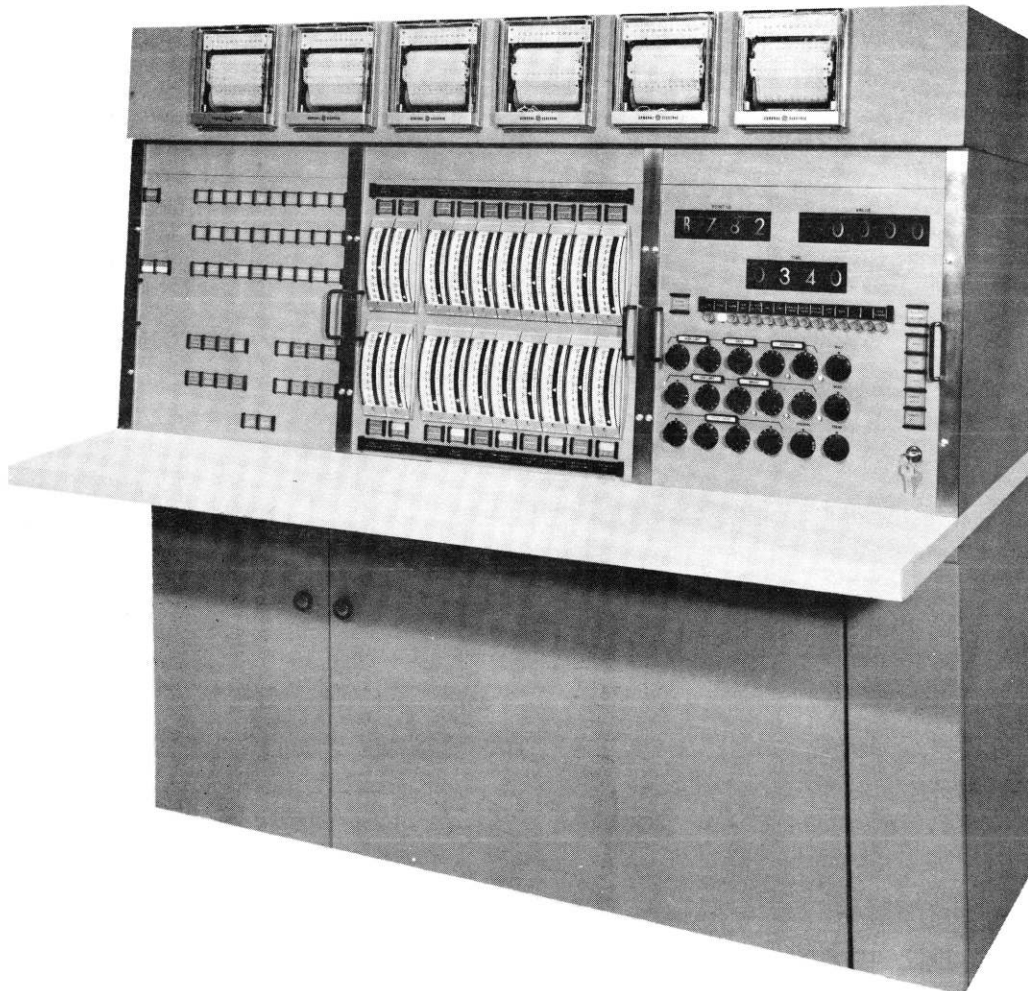


Figure X-7. Operator's Console



Figure X-8. Digital Dispatching Console

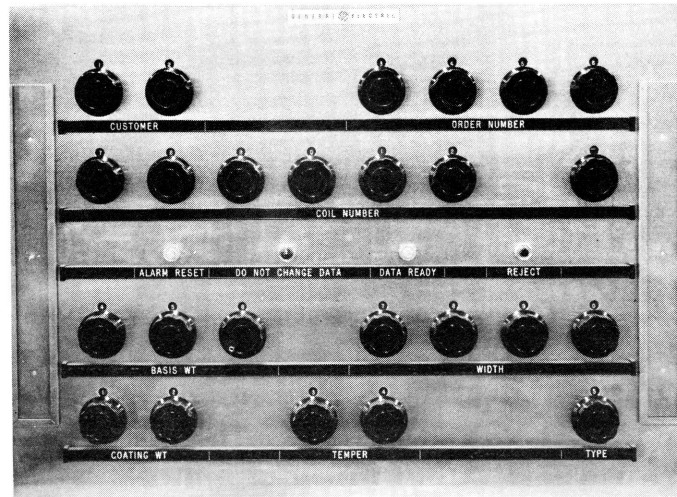


Figure X-9. Operator's Console

XI GE/PAC PROGRAMMER'S CONSOLE

The GE/PAC programmer's console is an integral part of the central processor, as shown in Figure XI-1. This console is one means for the programmer and maintenance man to communicate with the computer in actual machine language. Plant operating personnel are not required to, nor expected to, know how to use the programmer's console in process control applications. Once the programming has been completed, little use will be made of the programmer's console except for the displaying of various registers for maintenance personnel.

The underlying philosophy toward the use of consoles in the GE/PAC system is that each module of the system may have its associated console. More specifically, the arithmetic unit shall have a programming console which will contain displays and functions only related to the arithmetic unit. System displays and functions will be included on a separate system console.

DESCRIPTION OF GE/PAC 4040 CONSOLE

Twenty-four lights and console switches are used to display and enter data or commands into the computer. These lights and switches are divided into groups of three to represent 8 octal digits for programming convenience. Two modes of operation are possible: 1) Automatic, and 2) Manual by the Man/Auto/Console/off switch located in the lower center portion of the console. During automatic operation, the console switches can only be read into the A Register by the programmed instruction Read Console Switches (RCS) (25050000)₈. If a console switch is down, a "1" is set in that bit position in the A Register; if up, the contents of A are not changed. The console switches are sometimes referred to as "break-point" switches during automatic operation. The "break-point" refers to a decision that can be made as the result of a switch or switches being set (down). On-line routines that make use of the console switches are normally called for by the "Demand" pushbutton. This sets the demand flip-flop (D EMF), and the programmed instruction. Jump if No Demand (JND) (25040000)₈ is interrogated periodically to determine if the switches should be read and deciphered.

In the automatic mode all console switches and buttons except the following are disabled:

1. Clear Alarm
2. Power Off
3. Demand

4. Save P
5. Save I
6. Program Switches
7. Register Select Switch

The Console Off position allows the console to be disabled when in the automatic (running) mode of operation. A removable handle is provided to lockout the console in the automatic mode.

Manual operation of the console enables all console switches and buttons. Manual operation normally involves changes to the instruction register and memory locations. The section "Operation of Console" explains this step-by-step procedure.

REGISTER DISPLAYS

The selector switch in the lower left hand corner allows the following registers to be displayed:

A - Accumulator	23 - 0 bits positions
B - Buffer	23 - 0 bits positions
I - Instruction	23 - 0 bits positions
P - Place Counter	13 - 0 address bits
	14 relative address bit
	23 demand FF
	22 Overflow FF
	21 PAI
	20 TEST FF (TSTF)
	18 Periph ready FF
J - Counter	4 - 0 bit positions
	23 - S ₁ control sequence light
	22 - S ₂ control sequence light
	21 - S ₃ control sequence light
	20 - C ₁ control sequence light
	19 - C ₂ control sequence light
	18 - C ₃ control sequence light
AUX - Auxiliary Switch	For other peripheral registers

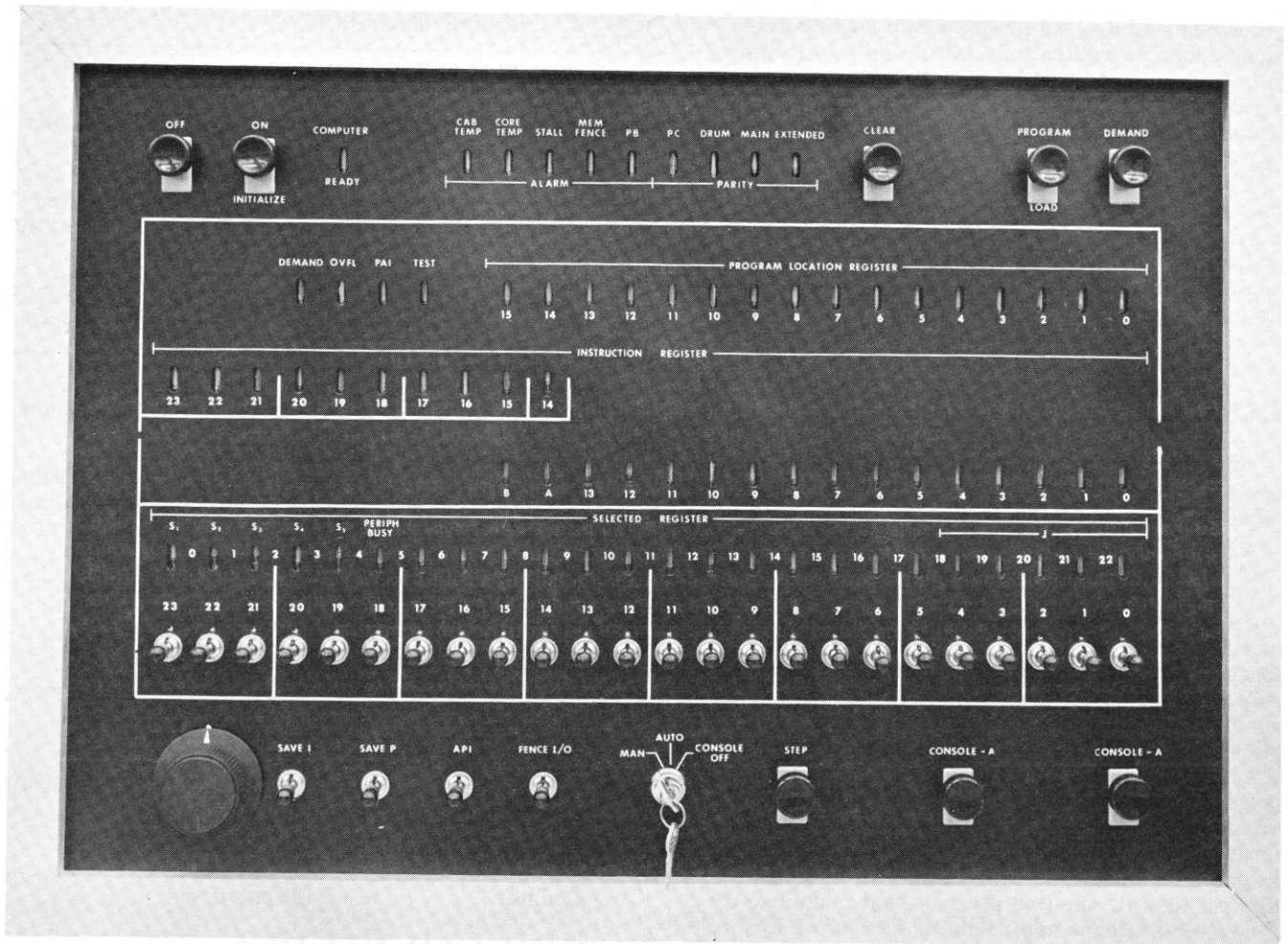


Figure XI-1. Programmer's Console - GE/PAC 4050 and GE/PAC 4060

ALARM LIGHTS

The following Alarm lights are displayed on the console:

1. Core Parity
2. Peripheral Error

The previous lights may be cleared by the CLEAR pushbutton.

3. Core temperature
4. Cabinet Temperature
5. Stall Alarm
6. Computer Ready

OPERATION OF GE/PAC 4040 CONSOLE

In the manual mode of operation the following operations are performed to change registers and memory locations:

1. Clear A - the contents of the A Register is set to all zeros.
2. Console \rightarrow A - the console switch settings are "or-ed" into the A Register.
3. Console \rightarrow B - the console switch settings are set in the B Register.
4. Step - the computer is stepped one instruction.
5. Save I - the save I switch inhibits the changing of the instruction portion (bits 23 through 14) of the I Register.
6. Save P - the save P switch inhibits the changing of the Place counter.
7. API Lockout - the automatic priority interrupt lockout switch disables all interrupts from the API module. This function is disabled when operating in the Console Off mode.

Steps to manually store a word into core memory:

1. Set console switches to (320YYYYY)₈
2. Press CONSOLE \rightarrow B
3. Set console switches to data to be stored
4. Press CLEAR A
5. Press CONSOLE \rightarrow A
6. Press STEP

Steps to manually display a word from core memory:

1. Set console switches to (000YYYYY)₈
2. Press CONSOLE \rightarrow B
3. Press STEP

USE OF PROGRAM LOAD OPTION

The Program Load Option provides a simple means of rapidly transferring a program from either paper tape or drum into GE/PAC's core memory.

The Program Load Routine (Figure XI-2) is electrically written into core locations 00₈ thru 33₈ when the "Program Load" pushbutton on the Computer Console is depressed (the button is disabled unless the computer is in the Manual Mode of operation).

The Routine is a simple paper tape loader designed to accept GE/PAC bi-octal format paper tape input media. This loader will accept only core starting address control frames and data control frames. It will operate incorrectly if other types of control frames are present on the tape. The Routine also contains instructions to transfer drum locations 00₈ thru 21₈ to core locations 00₈ thru 21₈ and to transfer Program Control, on demand, to the new program beginning at location 05₈.

STEP-BY-STEP OPERATING INSTRUCTIONS

Normal Operation

1. Rotate Keyswitch to "Manual" position.
2. Push "API Lockout" switch to "Inhibit" position. (All program interrupts (inhibitible and non-inhibitible) are inhibited.)
3. Push "Off/On" switch to "initialize" position. (The B, P, API registers and the DEMF are all reset.)
4. Push Program Load pushbutton.
5. Select program source media:
 - a. Drum: Push "Demand" button.
 - b. Paper Tape: Do not push "Demand" button.
6. Rotate Keyswitch to "Auto" position and push "Step" button. (If Drum was selected (step 5) a drum transfer will occur. The Program will wait for step 7.)
7. Push "Demand" button. (Program will transfer Program Control to Location 05₈. If paper-tape was selected, it will be read; otherwise, action depends upon the program in drum location 05₈ to 21₈.)

```

PROGRAM LOAD ROUTINE
100000000 100000000
* GE/PAC PROGRAM LOAD ROUTINE (MAY 11, 1964)
ORG 00
DRUM EQL /2400
RDR EQL /1100
PRGLOD LDA /12 00 DRUM POINTER WORD
00000 00000012 00000012 JND B IF (DEMAND) --, 04, 02
00001 25040000 25040000 OUT DRUM 02 INITIATE DRUM TRANSFER
00002 25042400 24042400 NOP
00003 05000040 05000040 SPB DEMAND 04 B IF (DEMAND) --, 04, 05
00004 33000027 33000027 READER SPB IN 05 READ CONTROL FRAME
00005 33000022 33000022 STA 04 X4 = CONTROL FRAME
00006 32000004 32000004 * C READ-PACK 4 FRAMES INTO 24BIT WORD
00007 07300003 07300003 LXX 3, 3 K = 3
00010 05000000 05000000 LDZ A = 0000
00011 05004062 05004062 PACK * SRC 18 11 SHIFT CHARACTER LEFT 6 PLACES
00012 32000000 32000000 STA 00 TEMP = A
00013 33000022 33000022 SPB IN READ DATA FRAME
00014 21000000 21000000 ORA 00 B A = 000C + TEMP
00015 06000003 06000003 DMT 03 K = K-1
00016 34000011 34000011 BTS PACK IF (K) 17, 11, 11
00017 26200001 26200001 INX 1, 2 17 INCREMENT STORE ADDRESS
00020 04400032 04400032 XEC CODE, 4 STA (DATA, STARTING ADDRESS), X4
00021 14000005 14000005 BRU READER GO TO 05
* C READ SUBROUTINE
00022 25061100 25061100 IN * INR RDR 22 B IF (RDR READY) --, 22, 25
00023 14000025 14000025 BRU INA
00024 14000022 14000022 BRU IN
00025 25051100 25051100 INA * IN RDR 25 A = 000C
00026 25071150 25071150 JNE RDR+/50 26 B IF (INPUT PARITY) --, 30, 27
* C DEMAND SUBROUTINE
00027 25040000 25040000 DEMAND * JND 27 B IF (DEMAND) --, 27, 30
00030 14100000 14100000 BRU 0, 1 30 RETURN
00031 14000027 14000027 BRU DEMAND
*
00032 32237777 32277745 CODE * STA *-/33, 2 (DATA WORD CODE = 0)
00033 32000002 32000002 STA 02 (BIN-START-ADDRESS CODE = 1)
*00000000 *00000000 END

```

Figure XI-2

Error Detection

1. Paper Tape Input Media. Reader stops (or fails to start).

a. "Error" light is "on". A Reader Ready error occurred - no recovery is possible. Check:

AC Power to PT Reader

Light on PT Reader

Positioning of paper tape in Reader

Return to step -- Normal Operation.

b. "Error" light is "off". An input parity error occurred. Recovery may be made as follows: The last character read is in the A Register and

should match the Data Frame on the PT to the left (as one faces the reader) of the photocell read heads. The character may be corrected by changing the contents of the A Register. Push the "Demand" button to continue reading.

2. Drum Input Media. Before performing step 7 above, check the following lights on the system console located on the front of the drum cabinet.

a. Composite Parity Error light.

b. Busy light (Position 2, bit 23 of display).

c. Drum Write light position 2, bit 21 of display). If any of these lights are "on", there can be no recovery - Return to step -- Normal Operation.

XII OCTAL DECIMAL CONVERSION TABLE

Octal-Decimal Integer Conversion Table

Octal	10000	20000	30000	40000	50000	60000	70000
Decimal	4096	8192	12288	16384	20480	24576	28672

Octal	0000 to 0377
Decimal	0000 to 0255

Octal	0	1	2	3	4	5	6	7
0000	0000	0001	0002	0003	0004	0005	0006	0007
0010	0008	0009	0010	0011	0012	0013	0014	0015
0020	0016	0017	0018	0019	0020	0021	0022	0023
0030	0024	0025	0026	0027	0028	0029	0030	0031
0040	0032	0033	0034	0035	0036	0037	0038	0039
0050	0040	0041	0042	0043	0044	0045	0046	0047
0060	0048	0049	0050	0051	0052	0053	0054	0055
0070	0056	0057	0058	0059	0060	0061	0062	0063
0100	0064	0065	0066	0067	0068	0069	0070	0071
0110	0072	0073	0074	0075	0076	0077	0078	0079
0120	0080	0081	0082	0083	0084	0085	0086	0087
0130	0088	0089	0090	0091	0092	0093	0094	0095
0140	0096	0097	0098	0099	0100	0101	0102	0103
0150	0104	0105	0106	0107	0108	0109	0110	0111
0160	0112	0113	0114	0115	0116	0117	0118	0119
0170	0120	0121	0122	0123	0124	0125	0126	0127
0200	0128	0129	0130	0131	0132	0133	0134	0135
0210	0136	0137	0138	0139	0140	0141	0142	0143
0220	0144	0145	0146	0147	0148	0149	0150	0151
0230	0152	0153	0154	0155	0156	0157	0158	0159
0240	0160	0161	0162	0163	0164	0165	0166	0167
0250	0168	0169	0170	0171	0172	0173	0174	0175
0260	0176	0177	0178	0179	0180	0181	0182	0183
0270	0184	0185	0186	0187	0188	0189	0190	0191
0300	0192	0193	0194	0195	0196	0197	0198	0199
0310	0200	0201	0202	0203	0204	0205	0206	0207
0320	0208	0209	0210	0211	0212	0213	0214	0215
0330	0216	0217	0218	0219	0220	0221	0222	0223
0340	0224	0225	0226	0227	0228	0229	0230	0231
0350	0232	0233	0234	0235	0236	0237	0238	0239
0360	0240	0241	0242	0243	0244	0245	0246	0247
0370	0248	0249	0250	0251	0252	0253	0254	0255

Octal	1000 to 1377
Decimal	0512 to 0767

Octal	0	1	2	3	4	5	6	7
1000	0512	0513	0514	0515	0516	0517	0518	0519
1010	0520	0521	0522	0523	0524	0525	0526	0527
1020	0528	0529	0530	0531	0532	0533	0534	0535
1030	0536	0537	0538	0539	0540	0541	0542	0543
1040	0544	0545	0546	0547	0548	0549	0550	0551
1050	0552	0553	0554	0555	0556	0557	0558	0559
1060	0560	0561	0562	0563	0564	0565	0566	0567
1070	0568	0569	0570	0571	0572	0573	0574	0575
1100	0576	0577	0578	0579	0580	0581	0582	0583
1110	0584	0585	0586	0587	0588	0589	0590	0591
1120	0592	0593	0594	0595	0596	0597	0598	0599
1130	0600	0601	0602	0603	0604	0605	0606	0607
1140	0608	0609	0610	0611	0612	0613	0614	0615
1150	0616	0617	0618	0619	0620	0621	0622	0623
1160	0624	0625	0626	0627	0628	0629	0630	0631
1170	0632	0633	0634	0635	0636	0637	0638	0639
1200	0640	0641	0642	0643	0644	0645	0646	0647
1210	0648	0649	0650	0651	0652	0653	0654	0655
1220	0656	0657	0658	0659	0660	0661	0662	0663
1230	0664	0665	0666	0667	0668	0669	0670	0671
1240	0672	0673	0674	0675	0676	0677	0678	0679
1250	0680	0681	0682	0683	0684	0685	0686	0687
1260	0688	0689	0690	0691	0692	0693	0694	0695
1270	0696	0697	0698	0699	0700	0701	0702	0703
1300	0704	0705	0706	0707	0708	0709	0710	0711
1310	0712	0713	0714	0715	0716	0717	0718	0719
1320	0720	0721	0722	0723	0724	0725	0726	0727
1330	0728	0729	0730	0731	0732	0733	0734	0735
1340	0736	0737	0738	0739	0740	0741	0742	0743
1350	0744	0745	0746	0747	0748	0749	0750	0751
1360	0752	0753	0754	0755	0756	0757	0758	0759
1370	0760	0761	0762	0763	0764	0765	0766	0767

Octal	0400 to 0777
Decimal	0256 to 0511

Octal	0	1	2	3	4	5	6	7
0400	0256	0257	0258	0259	0260	0261	0262	0263
0410	0264	0265	0266	0267	0268	0269	0270	0271
0420	0272	0273	0274	0275	0276	0277	0278	0279
0430	0280	0281	0282	0283	0284	0285	0286	0287
0440	0288	0289	0290	0291	0292	0293	0294	0295
0450	0296	0297	0298	0299	0300	0301	0302	0303
0460	0304	0305	0306	0307	0308	0309	0310	0311
0470	0312	0313	0314	0315	0316	0317	0318	0319
0500	0320	0321	0322	0323	0324	0325	0326	0327
0510	0328	0329	0330	0331	0332	0333	0334	0335
0520	0336	0337	0338	0339	0340	0341	0342	0343
0530	0344	0345	0346	0347	0348	0349	0350	0351
0540	0352	0353	0354	0355	0356	0357	0358	0359
0550	0360	0361	0362	0363	0364	0365	0366	0367
0560	0368	0369	0370	0371	0372	0373	0374	0375
0570	0376	0377	0378	0379	0380	0381	0382	0383
0600	0384	0385	0386	0387	0388	0389	0390	0391
0610	0392	0393	0394	0395	0396	0397	0398	0399
0620	0400	0401	0402	0403	0404	0405	0406	0407
0630	0408	0409	0410	0411	0412	0413	0414	0415
0640	0416	0417	0418	0419	0420	0421	0422	0423
0650	0424	0425	0426	0427	0428	0429	0430	0431
0660	0432	0433	0434	0435	0436	0437	0438	0439
0670	0440	0441	0442	0443	0444	0445	0446	0447
0700	0448	0449	0450	0451	0452	0453	0454	0455
0710	0456	0457	0458	0459	0460	0461	0462	0463
0720	0464	0465	0466	0467	0468	0469	0470	0471
0730	0472	0473	0474	0475	0476	0477	0478	0479
0740	0480	0481	0482	0483	0484	0485	0486	0487
0750	0488	0489	0490	0491	0492	0493	0494	0495
0760	0496	0497	0498	0499	0500	0501	0502	0503
0770	0504	0505	0506	0507	0508	0509	0510	0511

Octal	1400 to 1777
Decimal	0768 to 1023

Octal	0	1	2	3	4	5	6	7
1400	0768	0769	0770	0771	0772	0773	0774	0775
1410	0776	0777	0778	0779	0780	0781	0782	0783
1420	0784	0785	0786	0787	0788	0789	0790	0791
1430	0792	0793	0794	0795	0796	0797	0798	0799
1440	0800	0801	0802	0803	0804	0805	0806	0807
1450	0808	0809	0810	0811	0812	0813	0814	0815
1460	0816	0817	0818	0819	0820	0821	0822	0823
1470	0824	0825	0826	0827	0828	0829	0830	0831
1500	0832	0833	0834	0835	0836	0837	0838	0839
1510	0840	0841	0842	0843	0844	0845	0846	0847
1520	0848	0849	0850	0851	0852	0853	0854	0855
1530	0856	0857	0858	0859	0860	0861	0862	0863
1540	0864	0865	0866	0867	0868	0869	0870	0871
1550	0872	0873	0874	0875	0876	0877	0878	0879
1560	0880	0881	0882	0883	0884	0885	0886	0887
1570	0888	0889	0890	0891	0892	0893	0894	0895
1600	0896	0897	0898	0899	0900	0901	0902	0903
1610	0904	0905	0906	0907	0908	0909	0910	0911
1620	0912	0913	0914	0915	0916	0917	0918	0919
1630	0920	0921	0922	0923	0924	0925	0926	0927
1640	0928	0929	0930	0931	0932	0933	0934	0935
1650	0936	0937	0938	0939	0940	0941	0942	0943
1660	0944	0945	0946	0947	0948	0949	0950	0951
1670	0952	0953	0954	0955	0956	0957	0958	0959
1700	0960	0961	0962	0963	0964	0965	0966	0967
1710	0968	0969	0970	0971	0972	0973	0974	0975
1720	0976	0977	0978	0979	0980	0981	0982	0983
1730	0984	0985	0986	0987	0988	0989	0990	0991
1740	0992	0993	0994	0995	0996	0997	0998	0999
1750	1000	1001	1002	1003	1004	1005	1006	1007
1760	1008	1009	1010	1011	1012	1013	1014	1015
1770	1016	1017	1018	1019	1020	1021	1022	1023

Octal-Decimal Integer Conversion Table

Octal	10000	20000	30000	40000	50000	60000	70000
Decimal	4096	8192	12288	16384	20480	24576	28672

Octal	2000 to 2377
Decimal	1024 to 1279

Octal	3000 to 3377
Decimal	1356 to 1791

Octal	0	1	2	3	4	5	6	7
2000	1024	1025	1026	1027	1028	1029	1030	1031
2010	1032	1033	1034	1035	1036	1037	1038	1039
2020	1040	1041	1042	1043	1044	1045	1046	1047
2030	1048	1049	1050	1051	1052	1053	1054	1055
2040	1056	1057	1058	1059	1060	1061	1062	1063
2050	1064	1065	1066	1067	1068	1069	1070	1071
2060	1072	1073	1074	1075	1076	1077	1078	1079
2070	1080	1081	1082	1083	1084	1085	1086	1087
2100	1088	1089	1090	1091	1092	1093	1094	1095
2110	1096	1097	1098	1099	1100	1101	1102	1103
2120	1104	1105	1106	1107	1108	1109	1110	1111
2130	1112	1113	1114	1115	1116	1117	1118	1119
2140	1120	1121	1122	1123	1124	1125	1126	1127
2150	1128	1129	1130	1131	1132	1133	1134	1135
2160	1136	1137	1138	1139	1140	1141	1142	1143
2170	1144	1145	1146	1147	1148	1149	1150	1151
2200	1152	1153	1154	1155	1156	1157	1158	1159
2210	1160	1161	1162	1163	1164	1165	1166	1167
2220	1168	1169	1170	1171	1172	1173	1174	1175
2230	1176	1177	1178	1179	1180	1181	1182	1183
2240	1184	1185	1186	1187	1188	1189	1190	1191
2250	1192	1193	1194	1195	1196	1197	1198	1199
2260	1200	1201	1202	1203	1204	1205	1206	1207
2270	1208	1209	1210	1211	1212	1213	1214	1215
2300	1216	1217	1218	1219	1220	1221	1222	1223
2310	1224	1225	1226	1227	1228	1229	1230	1231
2320	1232	1233	1234	1235	1236	1237	1238	1239
2330	1240	1241	1242	1243	1244	1245	1246	1247
2340	1248	1249	1250	1251	1252	1253	1254	1255
2350	1256	1257	1258	1259	1260	1261	1262	1263
2360	1264	1265	1266	1267	1268	1269	1270	1271
2370	1272	1273	1274	1275	1276	1277	1278	1279

Octal	0	1	2	3	4	5	6	7
3000	1536	1537	1538	1539	1540	1541	1542	1543
3010	1544	1545	1546	1547	1548	1549	1550	1551
3020	1552	1553	1554	1555	1556	1557	1558	1559
3030	1560	1561	1562	1563	1564	1565	1566	1567
3040	1568	1569	1570	1571	1572	1573	1574	1575
3050	1576	1577	1578	1579	1580	1581	1582	1583
3060	1584	1585	1586	1587	1588	1589	1590	1591
3070	1592	1593	1594	1595	1596	1597	1598	1599
3100	1600	1601	1602	1603	1604	1605	1606	1607
3110	1608	1609	1610	1611	1612	1613	1614	1615
3120	1616	1617	1618	1619	1620	1621	1622	1623
3130	1624	1625	1626	1627	1628	1629	1630	1631
3140	1632	1633	1634	1635	1636	1637	1638	1639
3150	1640	1641	1642	1643	1644	1645	1646	1647
3160	1648	1649	1650	1651	1652	1653	1654	1655
3170	1656	1657	1658	1659	1660	1661	1662	1663
3200	1664	1665	1666	1667	1668	1669	1670	1671
3210	1672	1673	1674	1675	1676	1677	1678	1679
3220	1680	1681	1682	1683	1684	1685	1686	1687
3230	1688	1689	1690	1691	1692	1693	1694	1695
3240	1696	1697	1698	1699	1700	1701	1702	1703
3250	1704	1705	1706	1707	1708	1709	1710	1711
3260	1712	1713	1714	1715	1716	1717	1718	1719
3270	1720	1721	1722	1723	1724	1725	1726	1727
3300	1728	1729	1730	1731	1732	1733	1734	1735
3310	1736	1737	1738	1739	1740	1741	1742	1743
3320	1744	1745	1746	1747	1748	1749	1750	1751
3330	1752	1753	1754	1755	1756	1757	1758	1759
3340	1760	1761	1762	1763	1764	1765	1766	1767
3350	1768	1769	1770	1771	1772	1773	1774	1775
3360	1776	1777	1778	1779	1780	1781	1782	1783
3370	1784	1785	1786	1787	1788	1789	1790	1791

Octal	2400 to 2777
Decimal	1280 to 1535

Octal	3400 to 3777
Decimal	1792 to 2047

Octal	0	1	2	3	4	5	6	7
2400	1280	1281	1282	1283	1284	1285	1286	1287
2410	1288	1289	1290	1291	1292	1293	1294	1295
2420	1296	1297	1298	1299	1300	1301	1302	1303
2430	1304	1305	1306	1307	1308	1309	1310	1311
2440	1312	1313	1314	1315	1316	1317	1318	1319
2450	1320	1321	1322	1323	1324	1325	1326	1327
2460	1328	1329	1330	1331	1332	1333	1334	1335
2470	1336	1337	1338	1339	1340	1341	1342	1343
2500	1344	1345	1346	1347	1348	1349	1350	1351
2510	1352	1353	1354	1355	1356	1357	1358	1359
2520	1360	1361	1362	1363	1364	1365	1366	1367
2530	1368	1369	1370	1371	1372	1373	1374	1375
2540	1376	1377	1378	1379	1380	1381	1382	1383
2550	1384	1385	1386	1387	1388	1389	1390	1391
2560	1392	1393	1394	1395	1396	1397	1398	1399
2570	1400	1401	1402	1403	1404	1405	1406	1407
2600	1408	1409	1410	1411	1412	1413	1414	1415
2610	1416	1417	1418	1419	1420	1421	1422	1423
2620	1424	1425	1426	1427	1428	1429	1430	1431
2630	1432	1433	1434	1435	1436	1437	1438	1439
2640	1440	1441	1442	1443	1444	1445	1446	1447
2650	1448	1449	1450	1451	1452	1453	1454	1455
2660	1456	1457	1458	1459	1460	1461	1462	1463
2670	1464	1465	1466	1467	1468	1469	1470	1471
2700	1472	1473	1474	1475	1476	1477	1478	1479
2710	1480	1481	1482	1483	1484	1485	1486	1487
2720	1488	1489	1490	1491	1492	1493	1494	1495
2730	1496	1497	1498	1499	1500	1501	1502	1503
2740	1504	1505	1506	1507	1508	1509	1510	1511
2750	1512	1513	1514	1515	1516	1517	1518	1519
2760	1520	1521	1522	1523	1524	1525	1526	1527
2770	1528	1529	1530	1531	1532	1533	1534	1535

Octal	0	1	2	3	4	5	6	7
3400	1792	1793	1794	1795	1796	1797	1798	1799
3410	1800	1801	1802	1803	1804	1805	1806	1807
3420	1808	1809	1810	1811	1812	1813	1814	1815
3430	1816	1817	1818	1819	1820	1821	1822	1823
3440	1824	1825	1826	1827	1828	1829	1830	1831
3450	1832	1833	1834	1835	1836	1837	1838	1839
3460	1840	1841	1842	1843	1844	1845	1846	1847
3470	1848	1849	1850	1851	1852	1853	1854	1855
3500	1856	1857	1858	1859	1860	1861	1862	1863
3510	1864	1865	1866	1867	1868	1869	1870	1871
3520	1872	1873	1874	1875	1876	1877	1878	1879
3530	1880	1881	1882	1883	1884	1885	1886	1887
3540	1888	1889	1890	1891	1892	1893	1894	1895
3550	1896	1897	1898	1899	1900	1901	1902	1903
3560	1904	1905	1906	1907	1908	1909	1910	1911
3570	1912	1913	1914	1915	1916	1917	1918	1919
3600	1920	1921	1922	1923	1924	1925	1926	1927
3610	1928	1929	1930	1931	1932	1933	1934	1935
3620	1936	1937	1938	1939	1940	1941	1942	1943
3630	1944	1945	1946	1947	1948	1949	1950	1951
3640	1952	1953	1954	1955	1956	1957	1958	1959
3650	1960	1961	1962	1963	1964	1965	1966	1967
3660	1968	1969	1970	1971	1972	1973	1974	1975
3670	1976	1977	1978	1979	1980	1981	1982	1983
3700	1984	1985	1986	1987	1988	1989	1990	1991
3710	1992	1993	1994	1995	1996	1997	1998	1999
3720	2000	2001	2002	2003	2004	2005	2006	2007
3730	2008	2009	2010	2011	2012	2013	2014	2015
3740	2016	2017	2018	2019	2020	2021	2022	2023
3750	2024	2025	2026	2027	2028	2029	2030	2031
3760	2032	2033	2034	2035	2036	2037	2038	2039
3770	2040	2041						

Octal-Decimal Integer Conversion Table

Octal	10000	20000	30000	40000	50000	60000	70000
Decimal	4096	8192	12288	16384	20480	24576	28672

Octal	4000 to 4377
Decimal	2048 to 2303

Octal	5000 to 5377
Decimal	2560 to 2815

Octal	0	1	2	3	4	5	6	7
4000	2048	2049	2050	2051	2052	2053	2054	2055
4010	2056	2057	2058	2059	2060	2061	2062	2063
4020	2064	2065	2066	2067	2068	2069	2070	2071
4030	2072	2073	2074	2075	2076	2077	2078	2079
4040	2080	2081	2082	2083	2084	2085	2086	2087
4050	2088	2089	2090	2091	2092	2093	2094	2095
4060	2096	2097	2098	2099	2100	2101	2102	2103
4070	2104	2105	2106	2107	2108	2109	2110	2111
4100	2112	2113	2114	2115	2116	2117	2118	2119
4110	2120	2121	2122	2123	2124	2125	2126	2127
4120	2128	2129	2130	2131	2132	2133	2134	2135
4130	2136	2137	2138	2139	2140	2141	2142	2143
4140	2144	2145	2146	2147	2148	2149	2150	2151
4150	2152	2153	2154	2155	2156	2157	2158	2159
4160	2160	2161	2162	2163	2164	2165	2166	2167
4170	2168	2169	2170	2171	2172	2173	2174	2175
4200	2176	2177	2178	2179	2180	2181	2182	2183
4210	2184	2185	2186	2187	2188	2189	2190	2191
4220	2192	2193	2194	2195	2196	2197	2198	2199
4230	2200	2201	2202	2203	2204	2205	2206	2207
4240	2208	2209	2210	2211	2212	2213	2214	2215
4250	2216	2217	2218	2219	2220	2221	2222	2223
4260	2224	2225	2226	2227	2228	2229	2230	2231
4270	2232	2233	2234	2235	2236	2237	2238	2239
4300	2240	2241	2242	2243	2244	2245	2246	2247
4310	2248	2249	2250	2251	2252	2253	2254	2255
4320	2256	2257	2258	2259	2260	2261	2262	2263
4330	2264	2265	2266	2267	2268	2269	2270	2271
4340	2272	2273	2274	2275	2276	2277	2278	2279
4350	2280	2281	2282	2283	2284	2285	2286	2287
4360	2288	2289	2290	2291	2292	2293	2294	2295
4370	2296	2297	2298	2299	2300	2301	2302	2303

Octal	0	1	2	3	4	5	6	7
5000	2560	2561	2562	2563	2564	2565	2566	2567
5010	2568	2569	2570	2571	2572	2573	2574	2575
5020	2576	2577	2578	2579	2580	2581	2582	2583
5030	2584	2585	2586	2587	2588	2589	2590	2591
5040	2592	2593	2594	2595	2596	2597	2598	2599
5050	2600	2601	2602	2603	2604	2605	2606	2607
5060	2608	2609	2610	2611	2612	2613	2614	2615
5070	2616	2617	2618	2619	2620	2621	2622	2623
5100	2624	2625	2626	2627	2628	2629	2630	2631
5110	2632	2633	2634	2635	2636	2637	2638	2639
5120	2640	2641	2642	2643	2644	2645	2646	2647
5130	2648	2649	2650	2651	2652	2653	2654	2655
5140	2656	2657	2658	2659	2660	2661	2662	2663
5150	2664	2665	2666	2667	2668	2669	2670	2671
5160	2672	2673	2674	2675	2676	2677	2678	2679
5170	2680	2681	2682	2683	2684	2685	2686	2687
5200	2688	2689	2690	2691	2692	2693	2694	2695
5210	2696	2697	2698	2699	2700	2701	2702	2703
5220	2704	2705	2706	2707	2708	2709	2710	2711
5230	2712	2713	2714	2715	2716	2717	2718	2719
5240	2720	2721	2722	2723	2724	2725	2726	2727
5250	2728	2729	2730	2731	2732	2733	2734	2735
5260	2736	2737	2738	2739	2740	2741	2742	2743
5270	2744	2745	2746	2747	2748	2749	2750	2751
5300	2752	2753	2754	2755	2756	2757	2758	2759
5310	2760	2761	2762	2763	2764	2765	2766	2767
5320	2768	2769	2770	2771	2772	2773	2774	2775
5330	2776	2777	2778	2779	2780	2781	2782	2783
5340	2784	2785	2786	2787	2788	2789	2790	2791
5350	2792	2793	2794	2795	2796	2797	2798	2799
5360	2800	2801	2802	2803	2804	2805	2806	2807
5370	2808	2809	2810	2811	2812	2813	2814	2815

Octal	4400 to 4777
Decimal	2304 to 2559

Octal	5400 to 5777
Decimal	2816 to 3071

Octal	0	1	2	3	4	5	6	7
4400	2304	2305	2306	2307	2308	2309	2310	2311
4410	2312	2313	2314	2315	2316	2317	2318	2319
4420	2320	2321	2322	2323	2324	2325	2326	2327
4430	2328	2329	2330	2331	2332	2333	2334	2335
4440	2336	2337	2338	2339	2340	2341	2342	2343
4450	2344	2345	2346	2347	2348	2349	2350	2351
4460	2352	2353	2354	2355	2356	2357	2358	2359
4470	2360	2361	2362	2363	2364	2365	2366	2367
4500	2368	2369	2370	2371	2372	2373	2374	2375
4510	2376	2377	2378	2379	2380	2381	2382	2383
4520	2384	2385	2386	2387	2388	2389	2390	2391
4530	2392	2393	2394	2395	2396	2397	2398	2399
4540	2400	2401	2402	2403	2404	2405	2406	2407
4550	2408	2409	2410	2411	2412	2413	2414	2415
4560	2416	2417	2418	2419	2420	2421	2422	2423
4570	2424	2425	2426	2427	2428	2429	2430	2431
4600	2432	2433	2434	2435	2436	2437	2438	2439
4610	2440	2441	2442	2443	2444	2445	2446	2447
4620	2448	2449	2450	2451	2452	2453	2454	2455
4630	2456	2457	2458	2459	2460	2461	2462	2463
4640	2464	2465	2466	2467	2468	2469	2470	2471
4650	2472	2473	2474	2475	2476	2477	2478	2479
4660	2480	2481	2482	2483	2484	2485	2486	2487
4670	2488	2489	2490	2491	2492	2493	2494	2495
4700	2496	2497	2498	2499	2500	2501	2502	2503
4710	2504	2505	2506	2507	2508	2509	2510	2511
4720	2512	2513	2514	2515	2516	2517	2518	2519
4730	2520	2521	2522	2523	2524	2525	2526	2527
4740	2528	2529	2530	2531	2532	2533	2534	2535
4750	2536	2537	2538	2539	2540	2541	2542	2543
4760	2544	2545	2546	2547	2548	2549	2550	2551
4770	2552	2553	2554	2555	2556	2557	2558	2559

Octal	0	1	2	3	4	5	6	7
5400	2816	2817	2818	2819	2820	2821	2822	2823
5410	2824	2825	2826	2827	2828	2829	2830	2831
5420	2832	2833	2834	2835	2836	2837	2838	2839
5430	2840	2841	2842	2843	2844	2845	2846	2847
5440	2848	2849	2850	2851	2852	2853	2854	2855
5450	2856	2857	2858	2859	2860	2861	2862	2863
5460	2864	2865	2866	2867	2868	2869	2870	2871
5470	2872	2873	2874	2875	2876	2877	2878	2879
5500	2880	2881	2882	2883	2884	2885	2886	2887
5510	2888	2889	2890	2891	2892	2893	2894	2895
5520	2896	2897	2898	2899	2900	2901	2902	2903
5530	2904	2905	2906	2907	2908	2909	2910	2911
5540	2912	2913	2914	2915	2916	2917	2918	2919
5550	2920	2921	2922	2923	2924	2925	2926	2927
5560	2928	2929	2930	2931	2932	2933	2934	2935
5570	2936	2937	2938	2939	2940	2941	2942	2943
5600	2944	2945	2946	2947	2948	2949	2950	2951
5610	2952	2953	2954	2955	2956	2957	2958	2959
5620	2960	2961	2962	2963	2964	2965	2966	2967
5630	2968	2969	2970	2971	2972	2973	2974	2975
5640	2976	2977	2978	2979	2980	2981	2982	2983
5650	2984	2985	2986	2987	2988	2989	2990	2991
5660	2992	2993	2994	2995	2996	2997	2998	2999
5670	3000	3001	3002	3003	3004	3005	3006	3007
5700	3008	3009	3010	3011	3012	3013	3014	3015
5710	3016	3017	3018	3019	3020	3021	3022	3023
5720	3024	3025	3026	3027	3028	3029	3030	3031
5730	3032	3033	3034	3035	3036	3037	3038	3039
5740	3040	3041	3042	3043	3044	3045	3046	3047
5750	3048	3049	3050	3051	3052	3053	3054	3055
5760	3056	3057	3058	3059	3060	3061	3062	3063
5770	3064	3065	3066	3067	3068	3069	3070	3071

Octal-Decimal Integer Conversion Table

Octal	10000	20000	30000	40000	50000	60000	70000
Decimal	4096	8192	12288	16384	20480	24576	28672

Octal	6000 to 6377
Decimal	3072 to 3327

Octal	7000 to 7377
Decimal	3584 to 3839

Octal	0	1	2	3	4	5	6	7
6000	3072	3073	3074	3075	3076	3077	3078	3079
6010	3080	3081	3082	3083	3084	3085	3086	3087
6020	3088	3089	3090	3091	3092	3093	3094	3095
6030	3096	3097	3098	3099	3100	3101	3102	3103
6040	3104	3105	3106	3107	3108	3109	3110	3111
6050	3112	3113	3114	3115	3116	3117	3118	3119
6060	3120	3121	3122	3123	3124	3125	3126	3127
6070	3128	3129	3130	3131	3132	3133	3134	3135
6100	3136	3137	3138	3139	3140	3141	3142	3143
6110	3144	3145	3146	3147	3148	3149	3150	3151
6120	3152	3153	3154	3155	3156	3157	3158	3159
6130	3160	3161	3162	3163	3164	3165	3166	3167
6140	3168	3169	3170	3171	3172	3173	3174	3175
6150	3176	3177	3178	3179	3180	3181	3182	3183
6160	3184	3185	3186	3187	3188	3189	3190	3191
6170	3192	3193	3194	3195	3196	3197	3198	3199
6200	3200	3201	3202	3203	3204	3205	3206	3207
6210	3208	3209	3210	3211	3212	3213	3214	3215
6220	3216	3217	3218	3219	3220	3221	3222	3223
6230	3224	3225	3226	3227	3228	3229	3230	3231
6240	3232	3233	3234	3235	3236	3237	3238	3239
6250	3240	3241	3242	3243	3244	3245	3246	3247
6260	3248	3249	3250	3251	3252	3253	3254	3255
6270	3256	3257	3258	3259	3260	3261	3262	3263
6300	3264	3265	3266	3267	3268	3269	3270	3271
6310	3272	3273	3274	3275	3276	3277	3278	3279
6320	3280	3281	3282	3283	3284	3285	3286	3287
6330	3288	3289	3290	3291	3292	3293	3294	3295
6340	3296	3297	3298	3299	3300	3301	3302	3303
6350	3304	3305	3306	3307	3308	3309	3310	3311
6360	3312	3313	3314	3315	3316	3317	3318	3319
6370	3320	3321	3322	3323	3324	3325	3326	3327

Octal	0	1	2	3	4	5	6	7
7000	3584	3585	3586	3587	3588	3589	3590	3591
7010	3592	3593	3594	3595	3596	3597	3598	3599
7020	3600	3601	3602	3603	3604	3605	3606	3607
7030	3608	3609	3610	3611	3612	3613	3614	3615
7040	3616	3617	3618	3619	3620	3621	3622	3623
7050	3624	3625	3626	3627	3628	3629	3630	3631
7060	3632	3633	3634	3635	3636	3637	3638	3639
7070	3640	3641	3642	3643	3644	3645	3646	3647
7100	3648	3649	3650	3651	3652	3653	3654	3655
7110	3656	3657	3658	3659	3660	3661	3662	3663
7120	3664	3665	3666	3667	3668	3669	3670	3671
7130	3672	3673	3674	3675	3676	3677	3678	3679
7140	3680	3681	3682	3683	3684	3685	3686	3687
7150	3688	3689	3690	3691	3692	3693	3694	3695
7160	3696	3697	3698	3699	3700	3701	3702	3703
7170	3704	3705	3706	3707	3708	3709	3710	3711
7200	3712	3713	3714	3715	3716	3717	3718	3719
7210	3720	3721	3722	3723	3724	3725	3726	3727
7220	3728	3729	3730	3731	3732	3733	3734	3735
7230	3736	3737	3738	3739	3740	3741	3742	3743
7240	3744	3745	3746	3747	3748	3749	3750	3751
7250	3752	3753	3754	3755	3756	3757	3758	3759
7260	3760	3761	3762	3763	3764	3765	3766	3767
7270	3768	3769	3770	3771	3772	3773	3774	3775
7300	3776	3777	3778	3779	3780	3781	3782	3783
7310	3784	3785	3786	3787	3788	3789	3790	3791
7320	3792	3793	3794	3795	3796	3797	3798	3799
7330	3800	3801	3802	3803	3804	3805	3806	3807
7340	3808	3809	3810	3811	3812	3813	3814	3815
7350	3816	3817	3818	3819	3820	3821	3822	3823
7360	3824	3825	3826	3827	3828	3829	3830	3831
7370	3832	3833	3834	3835	3836	3837	3838	3839

Octal	6400 to 6777
Decimal	3328 to 3583

Octal	7400 to 7777
Decimal	3840 to 4095

Octal	0	1	2	3	4	5	6	7
6400	3328	3329	3330	3331	3332	3333	3334	3335
6410	3336	3337	3338	3339	3340	3341	3342	3343
6420	3344	3345	3346	3347	3348	3349	3350	3351
6430	3352	3353	3354	3355	3356	3357	3358	3359
6440	3360	3361	3362	3363	3364	3365	3366	3367
6450	3368	3369	3370	3371	3372	3373	3374	3375
6460	3376	3377	3378	3379	3380	3381	3382	3383
6470	3384	3385	3386	3387	3388	3389	3390	3391
6500	3392	3393	3394	3395	3396	3397	3398	3399
6510	3400	3401	3402	3403	3404	3405	3406	3407
6520	3408	3409	3410	3411	3412	3413	3414	3415
6530	3416	3417	3418	3419	3420	3421	3422	3423
6540	3424	3425	3426	3427	3428	3429	3430	3431
6550	3432	3433	3434	3435	3436	3437	3438	3439
6560	3440	3441	3442	3443	3444	3445	3446	3447
6570	3448	3449	3450	3451	3452	3453	3454	3455
6600	3456	3457	3458	3459	3460	3461	3462	3463
6610	3464	3465	3466	3467	3468	3469	3470	3471
6620	3472	3473	3474	3475	3476	3477	3478	3479
6630	3480	3481	3482	3483	3484	3485	3486	3487
6640	3488	3489	3490	3491	3492	3493	3494	3495
6650	3496	3497	3498	3499	3500	3501	3502	3503
6660	3504	3505	3506	3507	3508	3509	3510	3511
6670	3512	3513	3514	3515	3516	3517	3518	3519
6700	3520	3521	3522	3523	3524	3525	3526	3527
6710	3528	3529	3530	3531	3532	3533	3534	3535
6720	3536	3537	3538	3539	3540	3541	3542	3543
6730	3544	3545	3546	3547	3548	3549	3550	3551
6740	3552	3553	3554	3555	3556	3557	3558	3559
6750	3560	3561	3562	3563	3564	3565	3566	3567
6760	3568	3569	3570	3571	3572	3573	3574	3575
6770	3576	3577	3578	3579	3580	3581	3582	3583

Octal	0	1	2	3	4	5	6	7
7400	3840	3841	3842	3843	3844	3845	3846	3847
7410	3848	3849	3850	3851	3852	3853	3854	3855
7420	3856	3857	3858	3859	3860	3861	3862	3863
7430	3864	3865	3866	3867	3868	3869	3870	3871
7440	3872	3873	3874	3875	3876	3877	3878	3879
7450	3880	3881	3882	3883	3884	3885	3886	3887
7460	3888	3889	3890	3891	3892	3893	3894	3895
7470	3896	3897	3898	3899	3900	3901	3902	3903
7500	3904	3905	3906	3907	3908	3909	3910	3911
7510	3912	3913	3914	3915	3916	3917	3918	3919
7520	3920	3921	3922	3923	3924	3925	3926	3927
7530	3928	3929	3930	3931	3932	3933	3934	3935
7540	3936	3937	3938	3939	3940	3941	3942	3943
7550	3944	3945	3946	3947	3948	3949	3950	3951
7560	3952	3953	3954	3955	3956	3957	3958	3959
7570	3960	3961	3962	3963	3964	3965	3966	3967
7600	3968	3969	3970	3971	3972	3973	3974	3975
7610	3976	3977	3978	3979	3980	3981	3982	3983
7620	3984	3985	3986	3987	3988	3989	3990	3991
7630	3992	3993	3994	3995	3996	3997	3998	3999
7640	4000	4001	4002	4003	4004	4005	4006	4007
7650	4008	4009	4010	4011	4012	4013	4014	4015
7660	4016	4017	4018	4019	4020	4021	4022	4023
7670	4024	4025	4026	4027	4028	4029	4030	4031
7700	4032	4033	4034	4035	4036	4037	4038	4039
7710	4040	4041	4042	4043	4044	4045	4046	4047
7720	4048	4049	4050	4051	4052	4053	4054	4055
7730	4056	4057	4058	4059	4060	4061	4062	4063
7740	4064	4065	4066	4067	4068	4069	4070	4071
7750	4072	4073	4074	4075	4076	4077	4078	4079
7760	4080	4081	4082	4083	4084	4085	4086	4087
7770	4088	4089						

Octal-Decimal Fraction Conversion Table

OCTAL	DECIMAL	OCTAL	DECIMAL	OCTAL	DECIMAL	OCTAL	DECIMAL
.000	.000000	.100	.125000	.200	.250000	.300	.375000
.001	.001953	.101	.126953	.201	.251953	.301	.376953
.002	.003906	.102	.128906	.202	.253906	.302	.378906
.003	.005859	.103	.130859	.203	.255859	.303	.380859
.004	.007812	.104	.132812	.204	.257812	.304	.382812
.005	.009765	.105	.134765	.205	.259765	.305	.384765
.006	.011718	.106	.136718	.206	.261718	.306	.386718
.007	.013671	.107	.138671	.207	.263671	.307	.388671
.010	.015625	.110	.140625	.210	.265625	.310	.390625
.011	.017578	.111	.142578	.211	.267578	.311	.392578
.012	.019531	.112	.144531	.212	.269531	.312	.394531
.013	.021484	.113	.146484	.213	.271484	.313	.396484
.014	.023437	.114	.148437	.214	.273437	.314	.398437
.015	.025390	.115	.150390	.215	.275390	.315	.400390
.016	.027343	.116	.152343	.216	.277343	.316	.402343
.017	.029296	.117	.154296	.217	.279296	.317	.404296
.020	.031250	.120	.156250	.220	.281250	.320	.406250
.021	.033203	.121	.158203	.221	.283203	.321	.408203
.022	.035156	.122	.160156	.222	.285156	.322	.410156
.023	.037109	.123	.162109	.223	.287109	.323	.412109
.024	.039062	.124	.164062	.224	.289062	.324	.414062
.025	.041015	.125	.166015	.225	.291015	.325	.416015
.026	.042968	.126	.167968	.226	.292968	.326	.417968
.027	.044921	.127	.169921	.227	.294921	.327	.419921
.030	.046875	.130	.171875	.230	.296875	.330	.421875
.031	.048828	.131	.173828	.231	.298828	.331	.423828
.032	.050781	.132	.175781	.232	.300781	.332	.425781
.033	.052734	.133	.177734	.233	.302734	.333	.427734
.034	.054687	.134	.179687	.234	.304687	.334	.429687
.035	.056640	.135	.181640	.235	.306640	.335	.431640
.036	.058593	.136	.183593	.236	.308593	.336	.433593
.037	.060546	.137	.185546	.237	.310546	.337	.435546
.040	.062500	.140	.187500	.240	.312500	.340	.437500
.041	.064453	.141	.189453	.241	.314453	.341	.439453
.042	.066406	.142	.191406	.242	.316406	.342	.441406
.043	.068359	.143	.193359	.243	.318359	.343	.443359
.044	.070312	.144	.195312	.244	.320312	.344	.445312
.045	.072265	.145	.197265	.245	.322265	.345	.447265
.046	.074218	.146	.199218	.246	.324218	.346	.449218
.047	.076171	.147	.201171	.247	.326171	.347	.451171
.050	.078125	.150	.203125	.250	.328125	.350	.453125
.051	.080078	.151	.205078	.251	.330078	.351	.455078
.052	.082031	.152	.207031	.252	.332031	.352	.457031
.053	.083984	.153	.208984	.253	.333984	.353	.458984
.054	.085937	.154	.210937	.254	.335937	.354	.460937
.055	.087890	.155	.212890	.255	.337890	.355	.462890
.056	.089843	.156	.214843	.256	.339843	.356	.464843
.057	.091796	.157	.216796	.257	.341796	.357	.466796
.060	.093750	.160	.218750	.260	.343750	.360	.468750
.061	.095703	.161	.220703	.261	.345703	.361	.470703
.062	.097656	.162	.222656	.262	.347656	.362	.472656
.063	.099609	.163	.224609	.263	.349609	.363	.474609
.064	.101562	.164	.226562	.264	.351562	.364	.476562
.065	.103515	.165	.228515	.265	.353515	.365	.478515
.066	.105468	.166	.230468	.266	.355468	.366	.480468
.067	.107421	.167	.232421	.267	.357421	.367	.482421
.070	.109375	.170	.234375	.270	.359375	.370	.484375
.071	.111328	.171	.236328	.271	.361328	.371	.486328
.072	.113281	.172	.238281	.272	.363281	.372	.488281
.073	.115234	.173	.240234	.273	.365234	.373	.490234
.074	.117187	.174	.242187	.274	.367187	.374	.492187
.075	.119140	.175	.244140	.275	.369140	.375	.494140
.076	.121093	.176	.246093	.276	.371093	.376	.496093
.077	.123046	.177	.248046	.277	.373046	.377	.498046

Octal-Decimal Fraction Conversion Table

OCTAL	DECIMAL	OCTAL	DECIMAL	OCTAL	DECIMAL	OCTAL	DECIMAL
.000000	.000000	.000100	.000244	.000200	.000488	.000300	.000732
.000001	.000003	.000101	.000247	.000201	.000492	.000301	.000736
.000002	.000007	.000102	.000251	.000202	.000495	.000302	.000740
.000003	.000011	.000103	.000255	.000203	.000499	.000303	.000743
.000004	.000015	.000104	.000259	.000204	.000503	.000304	.000747
.000005	.000019	.000105	.000263	.000205	.000507	.000305	.000751
.000006	.000022	.000106	.000267	.000206	.000511	.000306	.000755
.000007	.000026	.000107	.000270	.000207	.000514	.000307	.000759
.000010	.000030	.000110	.000274	.000210	.000518	.000310	.000762
.000011	.000034	.000111	.000278	.000211	.000522	.000311	.000766
.000012	.000038	.000112	.000282	.000212	.000526	.000312	.000770
.000013	.000041	.000113	.000286	.000213	.000530	.000313	.000774
.000014	.000045	.000114	.000289	.000214	.000534	.000314	.000778
.000015	.000049	.000115	.000293	.000215	.000537	.000315	.000782
.000016	.000053	.000116	.000297	.000216	.000541	.000316	.000785
.000017	.000057	.000117	.000301	.000217	.000545	.000317	.000789
.000020	.000061	.000120	.000305	.000220	.000549	.000320	.000793
.000021	.000064	.000121	.000308	.000221	.000553	.000321	.000797
.000022	.000068	.000122	.000312	.000222	.000556	.000322	.000801
.000023	.000072	.000123	.000316	.000223	.000560	.000323	.000805
.000024	.000076	.000124	.000320	.000224	.000564	.000324	.000808
.000025	.000080	.000125	.000324	.000225	.000568	.000325	.000812
.000026	.000083	.000126	.000328	.000226	.000572	.000326	.000816
.000027	.000087	.000127	.000331	.000227	.000576	.000327	.000820
.000030	.000091	.000130	.000335	.000230	.000579	.000330	.000823
.000031	.000095	.000131	.000339	.000231	.000583	.000331	.000827
.000032	.000099	.000132	.000343	.000232	.000587	.000332	.000831
.000033	.000102	.000133	.000347	.000233	.000591	.000333	.000835
.000034	.000106	.000134	.000350	.000234	.000595	.000334	.000839
.000035	.000110	.000135	.000354	.000235	.000598	.000335	.000843
.000036	.000114	.000136	.000358	.000236	.000602	.000336	.000846
.000037	.000118	.000137	.000362	.000237	.000606	.000337	.000850
.000040	.000122	.000140	.000366	.000240	.000610	.000340	.000854
.000041	.000125	.000141	.000370	.000241	.000614	.000341	.000858
.000042	.000129	.000142	.000373	.000242	.000617	.000342	.000862
.000043	.000133	.000143	.000377	.000243	.000621	.000343	.000865
.000044	.000137	.000144	.000381	.000244	.000625	.000344	.000869
.000045	.000141	.000145	.000385	.000245	.000629	.000345	.000873
.000046	.000144	.000146	.000389	.000246	.000633	.000346	.000877
.000047	.000148	.000147	.000392	.000247	.000637	.000347	.000881
.000050	.000152	.000150	.000396	.000250	.000640	.000350	.000885
.000051	.000156	.000151	.000400	.000251	.000644	.000351	.000888
.000052	.000160	.000152	.000404	.000252	.000648	.000352	.000892
.000053	.000164	.000153	.000408	.000253	.000652	.000353	.000896
.000054	.000167	.000154	.000411	.000254	.000656	.000354	.000900
.000055	.000171	.000155	.000415	.000255	.000659	.000355	.000904
.000056	.000175	.000156	.000419	.000256	.000663	.000356	.000907
.000057	.000179	.000157	.000423	.000257	.000667	.000357	.000911
.000060	.000183	.000160	.000427	.000260	.000671	.000360	.000915
.000061	.000186	.000161	.000431	.000261	.000675	.000361	.000919
.000062	.000190	.000162	.000434	.000262	.000679	.000362	.000923
.000063	.000194	.000163	.000438	.000263	.000682	.000363	.000926
.000064	.000198	.000164	.000442	.000264	.000686	.000364	.000930
.000065	.000202	.000165	.000446	.000265	.000690	.000365	.000934
.000066	.000205	.000166	.000450	.000266	.000694	.000366	.000938
.000067	.000209	.000167	.000453	.000267	.000698	.000367	.000942
.000070	.000213	.000170	.000457	.000270	.000701	.000370	.000946
.000071	.000217	.000171	.000461	.000271	.000705	.000371	.000949
.000072	.000221	.000172	.000465	.000272	.000709	.000372	.000953
.000073	.000225	.000173	.000469	.000273	.000713	.000373	.000957
.000074	.000228	.000174	.000473	.000274	.000717	.000374	.000961
.000075	.000232	.000175	.000476	.000275	.000720	.000375	.000965
.000076	.000236	.000176	.000480	.000276	.000724	.000376	.000968
.000077	.000240	.000177	.000484	.000277	.000728	.000377	.000972

Octal-Decimal Fraction Conversion Table

OCTAL	DECIMAL	OCTAL	DECIMAL	OCTAL	DECIMAL	OCTAL	DECIMAL
.000400	.000976	.000500	.001220	.000600	.001464	.000700	.001708
.000401	.000980	.000501	.001224	.000601	.001468	.000701	.001712
.000402	.000984	.000502	.001228	.000602	.001472	.000702	.001716
.000403	.000988	.000503	.001232	.000603	.001476	.000703	.001720
.000404	.000991	.000504	.001235	.000604	.001480	.000704	.001724
.000405	.000995	.000505	.001239	.000605	.001483	.000705	.001728
.000406	.000999	.000506	.001243	.000606	.001487	.000706	.001731
.000407	.001003	.000507	.001247	.000607	.001491	.000707	.001735
.000410	.001007	.000510	.001251	.000610	.001495	.000710	.001739
.000411	.001010	.000511	.001255	.000611	.001499	.000711	.001743
.000412	.001014	.000512	.001258	.000612	.001502	.000712	.001747
.000413	.001018	.000513	.001262	.000613	.001506	.000713	.001750
.000414	.001022	.000514	.001266	.000614	.001510	.000714	.001754
.000415	.001026	.000515	.001270	.000615	.001514	.000715	.001758
.000416	.001029	.000516	.001274	.000616	.001518	.000716	.001762
.000417	.001033	.000517	.001277	.000617	.001522	.000717	.001766
.000420	.001037	.000520	.001281	.000620	.001525	.000720	.001770
.000421	.001041	.000521	.001285	.000621	.001529	.000721	.001773
.000422	.001045	.000522	.001289	.000622	.001533	.000722	.001777
.000423	.001049	.000523	.001293	.000623	.001537	.000723	.001781
.000424	.001052	.000524	.001296	.000624	.001541	.000724	.001785
.000425	.001056	.000525	.001300	.000625	.001544	.000725	.001789
.000426	.001060	.000526	.001304	.000626	.001548	.000726	.001792
.000427	.001064	.000527	.001308	.000627	.001552	.000727	.001796
.000430	.001068	.000530	.001312	.000630	.001556	.000730	.001800
.000431	.001071	.000531	.001316	.000631	.001560	.000731	.001804
.000432	.001075	.000532	.001319	.000632	.001564	.000732	.001808
.000433	.001079	.000533	.001323	.000633	.001567	.000733	.001811
.000434	.001083	.000534	.001327	.000634	.001571	.000734	.001815
.000435	.001087	.000535	.001331	.000635	.001575	.000735	.001819
.000436	.001091	.000536	.001335	.000636	.001579	.000736	.001823
.000437	.001094	.000537	.001338	.000637	.001583	.000737	.001827
.000440	.001098	.000540	.001342	.000640	.001586	.000740	.001831
.000441	.001102	.000541	.001346	.000641	.001590	.000741	.001834
.000442	.001106	.000542	.001350	.000642	.001594	.000742	.001838
.000443	.001110	.000543	.001354	.000643	.001598	.000743	.001842
.000444	.001113	.000544	.001358	.000644	.001602	.000744	.001846
.000445	.001117	.000545	.001361	.000645	.001605	.000745	.001850
.000446	.001121	.000546	.001365	.000646	.001609	.000746	.001853
.000447	.001125	.000547	.001369	.000647	.001613	.000747	.001857
.000450	.001129	.000550	.001373	.000650	.001617	.000750	.001861
.000451	.001132	.000551	.001377	.000651	.001621	.000751	.001865
.000452	.001136	.000552	.001380	.000652	.001625	.000752	.001869
.000453	.001140	.000553	.001384	.000653	.001628	.000753	.001873
.000454	.001144	.000554	.001388	.000654	.001632	.000754	.001876
.000455	.001148	.000555	.001392	.000655	.001636	.000755	.001880
.000456	.001152	.000556	.001396	.000656	.001640	.000756	.001884
.000457	.001155	.000557	.001399	.000657	.001644	.000757	.001888
.000460	.001159	.000560	.001403	.000660	.001647	.000760	.001892
.000461	.001163	.000561	.001407	.000661	.001651	.000761	.001895
.000462	.001167	.000562	.001411	.000662	.001655	.000762	.001899
.000463	.001171	.000563	.001415	.000663	.001659	.000763	.001903
.000464	.001174	.000564	.001419	.000664	.001663	.000764	.001907
.000465	.001178	.000565	.001422	.000665	.001667	.000765	.001911
.000466	.001182	.000566	.001426	.000666	.001670	.000766	.001914
.000467	.001186	.000567	.001430	.000667	.001674	.000767	.001918
.000470	.001190	.000570	.001434	.000670	.001678	.000770	.001922
.000471	.001194	.000571	.001438	.000671	.001682	.000771	.001926
.000472	.001197	.000572	.001441	.000672	.001686	.000772	.001930
.000473	.001201	.000573	.001445	.000673	.001689	.000773	.001934
.000474	.001205	.000574	.001449	.000674	.001693	.000774	.001937
.000475	.001209	.000575	.001453	.000675	.001697	.000775	.001941
.000476	.001213	.000576	.001457	.000676	.001701	.000776	.001945
.000477	.001216	.000577	.001461	.000677	.001705	.000777	.001949

OCTAL-DECIMAL CONVERSION TABLE (cont)

Table of Powers of 2

2^n	n	2^{-n}
1	0	1.0
2	1	0.5
4	2	0.25
8	3	0.125
16	4	0.062 5
32	5	0.031 25
64	6	0.015 625
128	7	0.007 812 5
256	8	0.003 906 25
512	9	0.001 953 125
1 024	10	0.000 976 562 5
2 048	11	0.000 488 281 25
4 096	12	0.000 244 140 625
8 192	13	0.000 122 070 312 5
16 384	14	0.000 061 035 156 25
32 768	15	0.000 030 517 578 125
65 536	16	0.000 015 258 789 062 5
131 072	17	0.000 007 629 394 531 25
262 144	18	0.000 003 814 697 265 625
524 288	19	0.000 001 907 348 632 812 5
1 048 576	20	0.000 000 953 674 316 406 25
2 097 152	21	0.000 000 476 837 158 203 125
4 194 304	22	0.000 000 238 418 579 101 562 5
8 388 608	23	0.000 000 119 209 289 550 781 25
16 777 216	24	0.000 000 059 604 644 775 390 625
33 554 432	25	0.000 000 029 802 322 387 695 312 5
67 108 864	26	0.000 000 014 901 161 193 847 656 25
134 217 728	27	0.000 000 007 450 580 596 923 828 125
268 435 456	28	0.000 000 003 725 290 298 461 914 062 5
536 870 912	29	0.000 000 001 862 645 149 230 957 031 25
1 073 741 824	30	0.000 000 000 931 322 574 615 478 515 625
2 147 483 648	31	0.000 000 000 465 661 287 307 739 257 812 5
4 294 967 296	32	0.000 000 000 232 830 643 653 869 628 906 25
8 589 934 592	33	0.000 000 000 116 415 321 826 934 814 453 125
17 179 869 184	34	0.000 000 000 058 207 660 913 467 407 226 562 5
34 359 738 368	35	0.000 000 000 029 103 830 456 733 703 613 281 25
68 719 476 736	36	0.000 000 000 014 551 915 228 366 851 806 640 625
137 438 953 472	37	0.000 000 000 007 275 957 614 183 425 903 320 312 5
274 877 906 944	38	0.000 000 000 003 637 978 807 091 712 951 660 156 25
549 755 813 888	39	0.000 000 000 001 818 989 403 545 856 475 830 078 125
1 099 511 627 776	40	0.000 000 000 000 909 494 701 772 928 237 915 039 062 5

XIII FLOW CHART SYMBOLS

The use of flow charts is of great help in visualizing the flow of data and transformations to be made in a problem to be programmed. Flow charting an application before programming has several advantages:

1. It breaks the problem down into logical elements and subdivisions.
2. It points out areas of the problem which need further clarification, analysis, and definition.
3. It aids in coordinating the efforts of two or more programmer's working on the same application.
4. It aids in error-detection and error-isolation within a program.
5. It is a means of refreshing the programmer's concept of a program when he returns to a program which has remained static for some time.
6. It provides a common language between programmers not necessarily using the same computing equipment.

There are many different levels of detail and sophistication which may be shown in a flow chart. Usually, an initial "system" flow chart is drawn which breaks down a complex problem into relatively large logical segments. Each of the individual blocks within a flow chart may represent one or two instructions or as many as several thousand instructions. The blocks seldom refer to individual computer instructions such as ADD, SUB, STA. Instead, the blocks refer to logical decisions and functions which the computer is to perform upon the incoming data. Arrows show the direction of flow throughout the program.

Flow charting is a rather unique process. Seldom do two programmers obtain the exact same flow chart for a given problem, although both may be correct.

Special symbols not shown here may be used from time to time as the specific occasion demands; however, the meaning of any special symbol

should be clearly defined — preferably on the flow chart itself at the place where the symbol is first used.

Many mathematical symbols are used in flow charts. Some of the more common ones are:

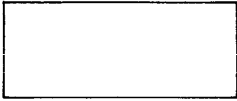

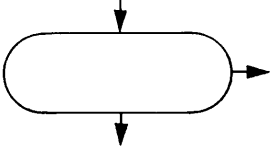
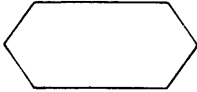


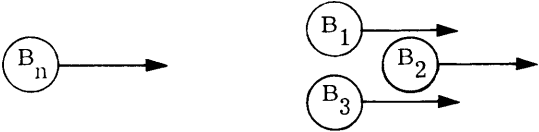
- = Equal to
- ≠ Not equal to
- > Greater than
- < Less than
- ≥ Greater than or equal to
- ≤ Less than or equal to
- Y Yes
- N No

→ "goes in to" e.g. "a + b → a" means that the sum of a and b is stored back in the same memory location that originally contained a.

Flow charts should be as neat and legible as possible. They are used to clarify the problem, not to cause confusion. Careless writing of numerics and alphabets is a common cause of errors in the interpretation of flow charts and program codings. In most cases, the hand-written coding sheets are handled by many people other than the person writing the program. Therefore, clarity is of the utmost importance. If this is doubted, a short time writing and debugging programs will convince any normal skeptic. The following conventions are recommended to avoid confusion.

<u>Numerics</u>	<u>Alphabets</u>
2	Z ("zee")
4, 9	
0	ō or ø ("oh")
1	I ("eye")
5	S or š ("ess")
7 (seven)	

Suggested Flow Chart Symbols
for Simplified Flow Charting

Symbol	Usage
	<p>Function or Operation Description</p>
	<p>Logic "Flow" (Follow the Arrows)</p>
	<p>Decision, Test, Comparison (2-or-3-way split)</p>
	<p>Subroutine</p>
	<p>Entrance, Exit, Stop</p>
	<p>Fixed Connector (Same symbol)</p>
	<p>Variable Connector (switch function)</p>

XIV GE/PAC INSTRUCTION SUMMARY

INSTRUCTIONS

EXECUTION TIMES (Microseconds)

<u>Page</u>	<u>Symbolic Code</u>	<u>Name</u>	<u>4040</u> <u>5 μs</u>	<u>4050</u> <u>5 μs</u>	<u>4060</u> <u>1.6 μs</u>
III:17	ABL	APPEND ITEM TO BEGINNING OF LIST	572	22.9	9.3
III:18	ABT	ABORT DEVICE D OPERATION	32	35	28.2
III:18	ACT	ACTIVATE DEVICE D	32	35	28.2
III:6	ADD	ADD	16	10	3.2
III:8	ADO	ADD ONE TO BIT K	16	17.7	10.9
III:17	AEL	APPEND ITEM TO END OF LIST	580	22.9	9.3
IV:3	AFA ¹	ADD FIELD TO A	---	17.7	10.9
III:8	AKA	ADD K TO A	127	5	1.6
III:9	ANA	AND TO A	16	10	3.2
III:12	BRU	BRANCH UNCONDITIONALLY	14	5	1.6
III:12	BTR	BRANCH IF TSTF RESET	14	5	1.6
III:12	BTS	BRANCH IF TSTF SET	14	5	1.6
III:9	CBK	CHANGE BIT	16	17.7	10.9
III:14	CLO	COUNT LEAST SIGNIFICANT ONES	16	17.7	10.9
III:14	CLZ	COUNT LEAST SIGNIFICANT ZEROS	16	17.7	10.9
III:14	CMO	COUNT MOST SIGNIFICANT ONES	16	17.7	10.9
III:14	CMZ	COUNT MOST SIGNIFICANT ZEROS	16	17.7	10.9
III:8	CPL	COMPLEMENT A	16	17.7	10.9
III:6	DAD	DOUBLE ADD	284	15	4.8
III:11	DLA	(SHIFT) DOUBLE LEFT ARITHMETIC	967	10-19.9	3.2-13.1
III:5	DLD	DOUBLE LENGTH LOAD	125	15	4.8
III:12	DLL	(SHIFT) DOUBLE LEFT LOGICAL	734	10-19.9	3.2-13.1
III:16	DMT	DECREMENT MEMORY AND TEST	21	15	4.8
III:10	DRA	(SHIFT) DOUBLE RIGHT ARITHMETIC	691	10-19.9	3.2-13.1
III:11	DRC	(SHIFT) DOUBLE RIGHT CIRCULAR	821	10-19.9	3.2-13.1
III:11	DRL	(SHIFT) DOUBLE RIGHT LOGICAL	797	10-19.9	3.2-13.1
III:5	DST	DOUBLE LENGTH STORE	132	15	4.8
III:6	DSU	DOUBLE SUBTRACT	298	15	4.8
III:7	DVD	DIVIDE	5344	33	26.2
III:9	ERA	EXCLUSIVE OR TO A	16	10	3.2

NOTE 1: GE/PAC 4050 and 4060

INSTRUCTIONS

EXECUTION TIMES (Microseconds)

<u>Page</u>	<u>Symbolic Code</u>	<u>Name</u>	<u>4040</u> <u>5 μs</u>	<u>4050</u> <u>5 μs</u>	<u>4060</u> <u>1.6 μs</u>
III:7	FAD	FLOATING ADD	947	100	70.0
III:7	FDV	FLOATING DIVIDE	2631	225	100.0
III:7	FIX	FIX FLOATING NUMBER	448	46	32.0
III:7	FLO	FLOAT FIXED NUMBER	566	46	32.0
III:7	FMP ²	FLOATING MULTIPLY	1460	185	100.0
III:7	FSU	FLOATING SUBTRACT	977	100	70.0
III:18	IAI	INHIBIT AUTOMATIC INTERRUPT	32	10	3.2
III:9	IBK	ISOLATE BIT K	16	17.7	10.9
IV:4	IDL ¹	INPUT FROM DEVICE TO LIST	---	62.9	39.1
III:18	IN	INPUT FROM DEVICE D	32	35	28.2
III:16	INX	INCREMENT X	21	15	4.8
III:12	JND	JUMP IF NO DEMAND	32	10	3.2
III:13	JNE	JUMP IF DEVICE D NOT IN ERROR	32	35	28.2
III:12	JNO	JUMP IF NO OVERFLOW	32	10	3.2
III:13	JNP	JUMP IF NO PARITY ERROR	32	10	3.2
III:13	JNR	JUMP IF DEVICE D NOT READY	32	35	28.2
III:9	LBM	LOAD BIT MASK	16	17.7	10.9
III:5	LDA	LOAD THE A REGISTER	16	10	3.2
IV:4	LDB ¹	LOAD HIGHSPEED I/O BUFFER	---	10	3.2
IV:3	LDF ¹	LOAD FIELD	---	17.7	10.9
III:6	LDI	LOAD INDIRECT	162	15	4.8
III:8	LDK	LOAD A WITH K	97	5	1.6
III:8	LDO	LOAD ONE INTO BIT K	16	17.7	10.9
III:16	LDP	LOAD PLACE	16	10	3.2
III:6	LDQ	LOAD THE Q REGISTER	132	10	3.2
IV:4	LDR ¹	LOAD REGISTERS	---	75	24.0
III:15	LDX	LOAD X WORD	21	15	4.8
III:8	LDZ	LOAD ZEROS INTO A	16	17.7	10.9
III:8	LMO	LOAD MINUS ONE	16	17.7	10.9
III:16	LPR	LOAD PLACE AND RESTORE	16	10	3.2
III:14	LXC	LOAD X WITH COUNT	21	15	4.8
III:16	LXK	LOAD X WITH K	21	15	4.8
III:8	MAQ	MOVE A TO Q	179	15	11.2
III:7	MPY ³	MULTIPLY	2010	21.8	15.0
III:8	NEG	NEGATE	16	17.7	10.9

NOTE 1: AU 2 (GE/PAC 4050 and 4060)

NOTE 2: FMP (4040 non-multiply step)

NOTE 3: MPY (4040 non-multiply step)

INSTRUCTIONS

EXECUTION TIMES (Microseconds)

<u>Page</u>	<u>Symbolic Code</u>	<u>Name</u>	<u>4040 5 μs</u>	<u>4050 5 μs</u>	<u>4060 1.6 μs</u>
III:9	NOP	NO OPERATION	16	15	4.8
III:6	OOM	OPERATE ON MEMORY	263	15	4.8
IV:4	ODL ¹	OUTPUT TO DEVICE FROM LIST	---	62.9	39.1
III:18	OPR	OPERATE DEVICE D	32	35	28.2
III:9	ORA	OR TO A	16	10	3.2
III:18	OUT	OUTPUT TO DEVICE D	32	35	28.2
III:18	PAI	PERMIT AUTOMATIC INTERRUPT	32	10	3.2
III:9	RBK	RESET BIT K	16	17.7	10.9
III:17	RBL	REMOVE BEGINNING ITEM FROM LIST	544	22.9	9.3
III:18	RCS	READ CONSOLE SWITCHES	32	10	3.2
III:17	REL	REMOVE ENDING ITEM FROM LIST	627	22.9	9.3
III:15	REV	RESET TSTF IF BIT K IS EVEN	16	17.7	10.9
III:13	RNZ	RESET TSTF IF A IS NONZERO	16	17.7	10.9
III:15	ROD	RESET TSTF IF BIT K IS ODD	16	17.7	10.9
IV:5	RPT ¹	REPEAT INSTRUCTION IN LOCATION Y	---	10	3.2
III:13	RST	RESET TSTF	16	17.7	10.9
III:9	SBK	SET BIT K	16	17.7	10.9
III:18	SEL	SELECT DEVICE D	32	35	28.2
III:13	SET	SET TSTF	16	17.7	10.9
III:14	SEV	SET TSTF IF BIT K IS EVEN	16	17.7	10.9
IV:3	SFA ¹	SUBTRACT FIELD FROM A	---	17.7	10.9
III:8	SKA	SUBTRACT K FROM A	141	5	1.6
III:10	SLA	SHIFT LEFT ARITHMETIC	879	10-18	3.2-11.0
III:11	SLL	SHIFT LEFT LOGICAL	698	10-18	3.2-11.0
III:13	SNZ	SET TSTF IF A IS NONZERO	16	17.7	10.9
III:14	SOD	SET TSTF IF BIT K IS ODD	16	17.7	10.9
III:16	SPB	SAVE PLACE AND BRANCH	16	10	3.2
III:10	SRA	SHIFT RIGHT ARITHMETIC	16	17.7	10.9
III:10	SRC	SHIFT RIGHT CIRCULAR	16	17.7	10.9
III:11	SRL	SHIFT RIGHT LOGICAL	15	17.7	10.9
III:18	SSA	SET STALL ALARM	32	10	3.2
III:5	STA	STORE CONTENTS OF A	14	10	3.2
IV:4	STB ¹	STORE HIGH SPEED I/O BUFFER	---	10	3.2
IV:4	STF ¹	STORE FIELD	---	22.7	12.5
III:6	STI	STORE INDIRECT	190	15	4.8
III:6	STQ	STORE CONTENTS OF Q	132	10	3.2

NOTE 1: AU 2 (GE/PAC 4050 and 4060)

INSTRUCTIONS

EXECUTION TIMES (Microseconds)

<u>Page</u>	<u>Symbolic Code</u>	<u>Name</u>	<u>4040</u> <u>5 μs</u>	<u>4050</u> <u>5 μs</u>	<u>4060</u> <u>1.6 μs</u>
IV:4	STR ¹	STORE REGISTERS	---	75	24.0
III:15	STX	STORE X	21	15	4.8
III:6	SUB	SUBTRACT	16	10	3.2
III:15	TER	TEST EVEN AND RESET BIT K	16	17.7	10.9
III:15	TES	TEST EVEN AND SET BIT K	16	17.7	10.9
III:14	TEV	TEST BIT K EVEN	16	17.7	10.9
IV:3	TFE ¹	TEST FIELD EQUAL	---	17.7	10.9
IV:3	TFL ¹	TEST FIELD LESS	---	17.7	10.9
III:13	TNM	TEST NOT MINUS ONE	16	17.7	10.9
III:13	TNZ	TEST A NONZERO	16	17.7	10.9
III:14	TOD	TEST BIT K ODD	16	17.7	10.9
III:15	TOR	TEST ODD AND RESET BIT K	16	17.7	10.9
III:15	TOS	TEST ODD AND SET BIT K	16	17.7	10.9
III:14	TSC	TEST AND SHIFT CIRCULAR	16	17.7	10.9
III:16	TXH	TEST X HIGH OR EQUAL	21	10	3.2
III:14	TZC	TEST ZERO AND COMPLEMENT	16	17.7	10.9
III:13	TZE	TEST A ZERO	16	17.7	10.9
III:16	XEC	EXECUTE	14	5	1.6

NOTE 1: AU 2 (GE/PAC 4050 and 4060)

Special Characteristics of GE/PAC Instruction

Mnemonic	Octal	Address Modification		May be Interrupted (2)		May Operate On Memory (OOM)		May be Repeated (RPT)	Occur in Quasi BRU Vector	
		Relative *	Index X	4040	4060	4040	4060	4060	4040	4060
ABL	57 X * Y	YES	YES	NO3	YES	NO	NO	NO	NO	YES
ABT	25 X 3 D	NO	YES	YES	YES	NO	NO	NO	YES	YES
ACT	25 X 1 D	NO	YES	YES	YES	NO	NO	NO	YES	YES
ADD	11 X * Y	YES	YES	YES	YES	YES	YES	YES	YES	YES
ADU	05x0700K	NO	YES	YES	YES	YES	YES	YES6	YES	YES
AEL	47 X * Y	YES	YES	NO3	YES	NO	NO	NO	NO	YES
AFA1	03 X * Y	YES	YES		YES		4	YES		YES
AKA	00 X 0 Y	NO	YES	NO3	NO	NO	NO	NO	NO	YES
ANA	20 X * Y	YES	YES	YES	YES	YES	YES	YES	YES	YES
BRU	14 X * Y	YES	YES	NO	NO	NO	NO	NO	YES	YES
BTR	30 X * Y	YES	YES	NO	NO	NO	NO	NO	YES	YES
BTS	34 X * Y	YES	YES	NO	NO	NO	NO	NO	YES	YES
CBK	05x4700K	NO	YES	YES	YES	YES	YES	YES6	YES	YES
CLO	05004137	NO	NO	YES	NO	YES	YES	NO	YES	YES
CLZ	05070137	NO	NO	YES	NO	YES	YES	NO	YES	YES
CMO	05004237	NO	NO	YES	NO	YES	YES	NO	YES	YES
CMZ	05070237	NO	NO	YES	NO	YES	YES	NO	YES	YES
CPL	05010000	NO	NO	YES	YES	YES	YES	NO	YES	YES
DAD	51 X * Y	YES	YES	NO3	YES	NO	4	NO	NO	YES
DLA	45 X * Y	NO	YES	NO3	YES	NO	4	YES6	NO	YES
DLD	41 X * Y	YES	YES	NO3	YES	NO	4	NO	NO	YES
DLL	45x0720K	NO	YES	NO3	YES	NO	4	YES6	NO	YES
DMT	06 0 * Y	YES	NO	YES	YES	YES	YES	NO	YES	YES
DRA	45x0440K	NO	YES	NO3	YES	NO	4	YES6	NO	YES
DRC	45x0530K	NO	YES	NO3	YES	NO	4	YES6	NO	YES
DRL	45x0430K	NO	YES	NO3	YES	NO	4	YES6	NO	YES
DST	63 X * Y	YES	YES	NO3	YES	NO	NO	NO	NO	YES
DSU	01 X * Y	YES	YES	NO3	YES	NO	4	NO	NO	YES
DVD	65 X * Y	YES	YES	NO3	YES	NO	NO	YES	NO	YES
ERA	10 X * Y	YES	YES	YES	YES	YES	YES	YES	YES	YES
FAD	70 X * Y	YES	YES	NO3	NO3	NO	NO	NO	NO	NO
FDV	73 X * Y	YES	YES	NO3	NO3	NO	NO	NO	NO	NO
FIX	74 X 0 K	NO	YES	NO3	NO3	NO	NO	NO	NO	NO
FLO	74 X 2 K	NO	YES	NO3	NO3	NO	NO	NO	NO	NO
FMP	72 X * Y	YES	YES	NO3	NO3	NO	NO	NO	NO	NO
FSU	71 X * Y	YES	YES	NO3	NO3	NO	NO	NO	NO	NO
IAI	25030000	NO	NO	NO	NO	NO	NO	NO	YES	YES
IBK	05x0100K	NO	YES	YES	YES	YES	YES	YES6	YES	YES
IDL1	67 X * Y	YES	YES		NO		NO	NO		YES
IN	25 X 5 D	NO	YES	YES	YES	NO	NO	NO	YES	YES
INX	26 X * K	YES	NO	YES	YES	YES	YES	NO	YES	YES
JND	25040000	NO	NO	YES	YES	NO	NO	NO	YES	YES
JNE	25 X 7 D	NO	YES	YES	YES	NO	NO	NO	YES	YES
JNO	25060000	NO	NO	YES	YES	NO	NO	NO	YES	YES
JNP	25070000	NO	NO	YES	YES	NO	NO	NO	YES	YES
JNR	25 X 6 D	NO	YES	YES	YES	NO	NO	NO	YES	YES
LBM	05x6300K	NO	YES	YES	YES	YES	YES	YES6	YES	YES
LDA	00 X * Y	YES	YES	YES	YES	YES	YES	YES	YES	YES
LDR1	37 X * Y	YES	YES		YES		NO	YES		YES
LDF1	23 X * Y	YES	YES		YES		4	YES		YES
LDI	52 X * Y	YES	YES	NO3	YES	NO	YES	NO	NO	YES
LDK	40 X * Y	YES	YES	NO3	NO	NO	NO	NO	NO	YES
LDO	05x0300K	NO	YES	YES	YES	YES	YES	YES6	YES	YES
LDP	15 X * K	YES	YES	NO	YES	YES	YES	YES	YES	YES
LDO	42 X * Y	YES	YES	NO3	YES	NO	YES	YES	NO	YES
LDR1	64 0 * Y	YES	NO		NO		NO	NO		YES
LDX	16 X * Y	YES	NO	YES	YES	YES	YES	NO		YES
LDZ	05000000	NO	NO	YES	YES	YES	YES	NO	YES	YES
LMO	05060000	NO	NO	YES	YES	YES	YES	NO	YES	YES
LPR	35 X * Y	YES	YES	NO	YES	YES	YES	YES	YES	YES
LXC	17x00000	NO	NO	YES	YES	YES	YES	NO	YES	YES
LXK	07 X * K	YES	NO	YES	YES	YES	YES	NO	YES	YES
MAQ	45004330	NO	NO	NO3	YES	NO	4	NO	NO	YES
MPY	55 X * Y	YES	YES	NO3	YES	NO	NO	NO	NO	YES
NEG	05013000	NO	NO	YES	YES	YES	YES	NO	YES	YES
NOP	26200000	NO	NO	YES	YES	YES	YES	NO	YES	YES
UDL1	56 X * Y	YES	YES		NO		NO	NO		YES
OOM	62 X * Y	YES	YES	NO3	NO	NO	NO	NO	NO	YES

Special Characteristics of GE/PAC Instruction

Mnemonic	Octal	Address Modification		May be Interrupted (2)		May Operate On Memory (OOM)		May be Repeated (RPT)	Occur in Quasi BRU Vector	
		Relative *	Index X	4040	4050/4060	4040	4050/4060	4060	4040	4050/4060
									4040	4050/4060
OPR	25 X 2 D	NO	YES	YES	YES	NO	NO	NO	YES	YES
URA	21 X * Y	YES	YES	YES	YES	YES	YES	YES	YES	YES
OUT	25 X 4 D	NO	YES	YES	YES	NO	NO	NO	YES	YES
PAI	25020000	NO	NO	NO3	NO	NO	NO	NO	YES	YES
RBK	05X4500K	NO	YES	YES	YES	YES	YES	YES6	YES	YES
RBL	56 X * Y	YES	YES	NO3	YES	NO	NO	NO	NO	YES
RCS	25050000	NO	NO	YES	YES	NO	NO	NO	YES	YES
REL	46 X * Y	YES	YES	NO	YES	NO	NO	NO	NO	YES
REV	05X7040K	NO	YES	YES	YES	YES	YES	YES	YES	YES
RNZ	05004470	NO	NO	YES	YES	YES	YES	NO	YES	YES
ROD	05X0440K	NO	YES	YES	YES	YES	YES	YES6	YES	YES
RPR	27 X * Y	YES	NO	-	NO	NO	NO	NO	YES	YES
RPT		YES	NO	-	5	-	-	-	-	-
RST	05004737	NO	NO	YES	YES	YES	YES	NO	YES	YES
SBK	05X4600K	NO	YES	YES	YES	YES	YES	YES6	YES	YES
SEL	25 X 0 D	NO	YES	YES	YES	NO	NO	NO	YES	YES
SET	05004637	NO	NO	YES	YES	YES	YES	NO	YES	YES
SEV	05X7050K	NO	YES	YES	YES	YES	YES	YES6	YES	YES
SFA1	22 X * Y	YES	YES		YES		4	YES		YES
SKA	50 X 0 Y	NO	YES	NO3	NO	NO	NO	NO	NO	YES
SLA	45X0204K	NO	YES	NO	YES	NO	4	YES6	NO	YES
SLL	45X0200K	NO	YES	NO	YES	NO	4	YES6	NO	YES
SNZ	05004570	NO	NO	YES	YES	YES	YES	NO	YES	YES
SOD	05X0450K	NO	YES	YES	YES	YES	YES	YES6	YES	YES
SPB	33 X * Y	YES	YES	NO	NO	NO	NO	NO	YES	YES
SRA	05X1404K	NO	YES	YES	YES	YES	YES	YES6	YES	YES
SRC	05X0404K	NO	YES	YES	YES	YES	YES	YES6	YES	YES
SRL	05X0004K	NO	YES	YES	YES	YES	YES	YES6	YES	YES
SSA	25010000	NO	NO	YES	YES	NO	NO	NO	YES	YES
STA	32 X * Y	YES	YES	YES	YES	NO	NO	YES	YES	YES
STB1	36 X * Y	YES	YES		YES		NO	YES		YES
STF1	13 X * Y	YES	YES		YES		4	NO		YES
STI	53 X * Y	YES	YES	NO3	YES	NO	NO	NO	NO	YES
STQ	44 X * Y	YES	YES	NO3	YES	NO	NO	YES	NO	YES
STR1	54 0 * Y	YES	NO		NO		NO	NO		YES
STX	06 X * Y	YES	NO	YES	YES	YES	YES	NO	YES	YES
SUB	31 X * Y	YES	YES	YES	YES	YES	YES	YES	YES	YES
TER	05X4560K	NO	YES	YES	YES	YES	YES	YES	YES	YES
TES	05X4660K	NO	YES	YES	YES	YES	YES	YES	YES	YES
TEV	05X7070K	NO	YES	YES	YES	YES	YES	YES6	YES	YES
TFE1	02 X * Y	YES	YES		YES		4	YES		YES
TFL1	12 X * Y	YES	YES		YES		4	YES		YES
TNM	05070770	NO	NO	YES	YES	YES	YES	NO	YES	YES
TNZ	05004770	NO	NO	YES	YES	YES	YES	NO	YES	YES
TOD	05X0470K	NO	YES	YES	YES	YES	YES	YES6	YES	YES
TOR	05X4570K	NO	YES	YES	YES	YES	YES	YES6	YES	YES
TOS	05X4670K	NO	YES	YES	YES	YES	YES	YES	YES	YES
TSC	05X0464K	NO	YES	YES	YES	YES	YES	YES6	YES	YES
TXH	24 X 0-K	NO	NO	NO	NO	NO	NO	NO	YES	YES
TZC	05064670	NO	NO	YES	YES	YES	YES	NO	YES	YES
TZE	05004670	NO	NO	YES	YES	YES	YES	NO	YES	YES
XEC	04 X * Y	YES	YES	NO7	NO7	YES	YES	YES	YES	YES

[ALL INSTRUCTIONS MAY BE XECED]

NOTES

1. 4025 INSTRUCTION ONLY
2. AN INSTRUCTION IS INTERRUPTABLE IF A NONINHIBITABLE PROGRAM INTERRUPT CAN OCCUR BETWEEN THE INSTRUCTION + ITS SUCCESSOR (REFER TO NOTE 3).
3. NONINHIBITABLE INTERRUPTS MAY OCCUR DURING THE EXECUTION OF THIS INSTRUCTION, NOT BETWEEN IT AND ITS SUCCESSOR
4. THIS INSTRUCTION MAY BE OOMED BUT DOES NOT YIELD THE EXPECTED RESULT
5. THE RPT OPERATION IS INTERRUPTABLE IF THE REPEATED INSTRUCTION IS INTERRUPTABLE
6. THIS INSTRUCTION MAY BE REPEATED NO MORE THAN 31-K TIMES
7. INTERRUPTS MAY NOT OCCUR BETWEEN THE EXECUTION OF XEC AND ITS OBJECT INSTRUCTION. IF THE OBJECT INSTRUCTION IS INTERRUPTABLE, AN INTERRUPT MAY OCCUR FOLLOWING ITS EXECUTION.

Progress Is Our Most Important Product

GENERAL  ELECTRIC