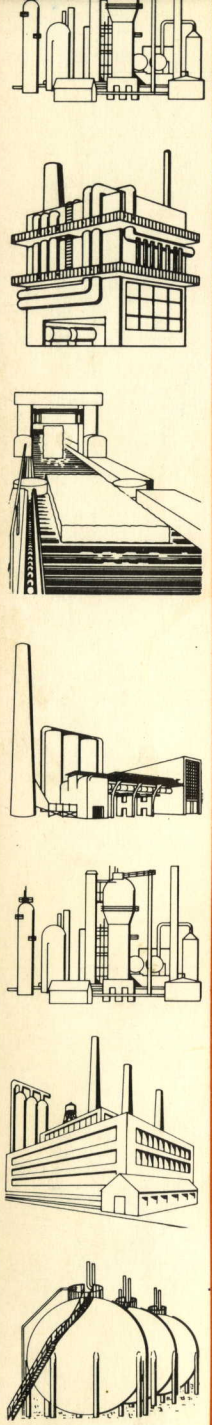
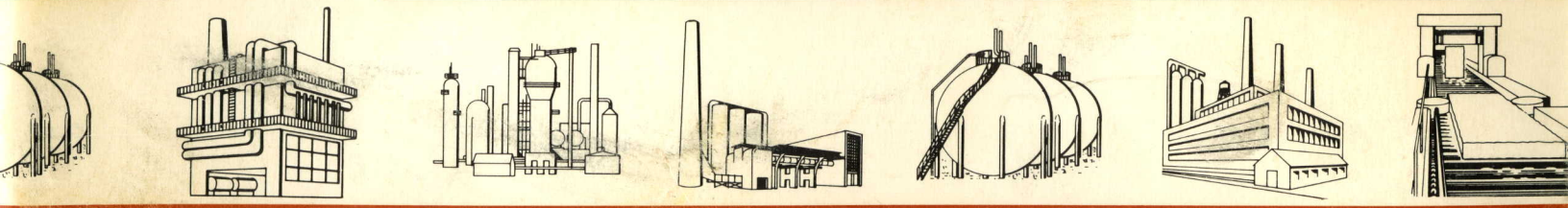
PCP 56 A
GET 2969 A



GE 412

PROGRAMMING MANUAL



PROCESS COMPUTER SECTION
INDUSTRY CONTROL DEPARTMENT
PHOENIX, ARIZONA

GENERAL ELECTRIC

GE 412

PROGRAMMING MANUAL

PROCESS COMPUTER SECTION
INDUSTRY CONTROL DEPARTMENT
GENERAL ELECTRIC COMPANY
PHOENIX, ARIZONA

GENERAL ELECTRIC 412
DIGITAL COMPUTER SYSTEM

IN THE CONSTRUCTION OF THE EQUIPMENT DESCRIBED, GENERAL ELECTRIC COMPANY RESERVES THE RIGHT TO MODIFY THE DESIGN FOR REASONS OF IMPROVED PERFORMANCE AND OPERATIONAL FLEXIBILITY.

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

PREFACE

This manual, as the name implies, is devoted to an explanation of how to program for the GE 412 Process Computer System. Throughout the manual, computer hardware has been described whenever an understanding of the hardware was considered necessary or helpful to the programmer. Likewise, digital computer concepts have been dwelt upon to the extent believed necessary. The basic functions performed by a computer are used to illustrate the computer instructions described. This concept is shown in the matrix below.

Although programming has been presented in the order intended to be most helpful to the student, for example, with problems dispersed throughout the text, it is hoped the manual can satisfy the experienced programmer who wants only reference information.

Other publications on the GE 412 Process Computer System are: the "GE 412 System Manual" which presents an overall view of the computer system and of the services which General Electric provides to its customers; application manuals which explain in detail the various types of computer applications and a variety of publications on equipment operation.

	Basic GE 412 Functions								
GE 412 Instructions	Arranging and Editing of Data	Arithmetic Operations	Comparing, Combining Extracting Operations	Decision Making	Address Modification	Real-Time Operations	Reading in and Typing out Numbers	Subroutine Operations	Subroutine Operations
Data Transfer	X	X	X	X	X		X	X	X
Arithmetic		X		X	X		X	X	X
Logical			X	X					
Shifting			X				X	X	X
Branching				X	X		X	X	X
X Location					X			X	X
Elapsed Time						X		X	
Automatic Interrupt						X			
Peripheral Input-Output								X	
System Input-Output							X		

Figure 1. The GE 412 Process Control and Monitoring Computer System

I INTRODUCTION

A. PROCESS COMPUTING

A discussion of fundamental programming techniques for the GE 412 Process Control and Monitoring Computer System requires an understanding of basic digital computer concepts and of the functions that can be accomplished in process logging, monitoring, and control with a digital computer. The complex relationships between the variables in a process make it extremely desirable to have a fast and efficient means of coordinating the monitoring and the control of these variables. Process computing requires real-time scanning and controlling of these variables. This means that at the time a physical phenomenon takes place in a process, the computer must at the same time be able to sense it, correlate and evaluate its meaning with other sensings, and take some action to control or regulate the process.

The GE 412 System's major functions are to efficiently and quickly sense, correlate and evaluate, and control the operations of a process. The computer system can sense analog signals from process sensors such as thermocouples and pressure sensors by converting these into equivalent digital values. The computer can directly sense digital inputs of information such as valve positions, on-off status of equipment, manual switch positions, and accumulations from counters. The computer can correlate and evaluate digital values through the use of high speed digital computer techniques. Control values are developed through logic and computation, and dictated to the process as control outputs. These may be analog signals converted from digital control values and sent to process control devices, or they may be in the form of signals to open and close contacts which in turn control the on-off status of process equipment such as pumps and motors.

In addition to these three major functions, the GE 412 System may accomplish monitoring of process equipment for off-limit conditions, performance and efficiency calculations, periodic logs, and trend recording. Because of the "real-time" aspect of process computing, timing is of much greater consideration than it is in business or scientific computing where processing of data is done off-line or after the fact.

The GE 412 System is highly adaptable to industrial processes and other functions in industries such as electric and gas utilities, steel mills, cement, mining, glass making, chemical, petroleum, and petrochemical. This versatility is achieved by means of a fast and powerful computer having a wide array of peripheral input-output equipment facilitating fast, efficient communication between the computer and the process.

B. BASIC DIGITAL COMPUTER CONCEPTS

All digital computers consist of five basic components: input, output, storage, arithmetic, and control.

1. Input Component

The input component consists of one or more units that introduce information into the computer from an external source. It accepts information in a variety of forms and converts this information into a digital form ready for entry into the computer. Paper tape readers, punched card readers, analog-to-digital converters, and manual switches are typical input units used with computer systems.

2. Output Component

The output component consists of one or more units that accept digital information from the computer and convert that information to a form applicable to external use. Paper tape punches, electric typewriters, visual displays, and digital-to-analog converters are typical output units used with computer systems.

3. Storage Component

The storage component consists of one or more units that store or remember information within the computer system. These units usually store information electronically, taking advantage of magnetic or other electronic principles to remember bistable conditions. Binary 1's and 0's can be indicated as bistable conditions and stored as such. A fixed number of binary digits make up a storage word. The storage unit is divided into thousands of addressable positions, each capable of storing one word. By using special combinations of the bits within a word, many types of information may be represented such as: data, constants, special alphabetic codes, and computer instructions. Computer instructions are explained in the control component discussion.

Magnetic drum, magnetic tape, and magnetic core are storage units used in computer systems.

4. Arithmetic Component

The arithmetic component performs arithmetic and logical decision-making functions within the computer system. The numbers used in these arithmetic and comparison operations may also be stored in the storage unit until used, and the results of arithmetic operations may also be stored back into the storage unit. Digital computers carry out all arithmetic operations by controlled additions. Subtraction is done by complementary arithmetic; multiplication is done by adding and shifting; and divide is done by subtracting and shifting. The basic circuit in the arithmetic component is, therefore, a binary adder.

5. Control Component

The control component automatically controls the operation of the other four components of the computer system. The step by step description of what the computer is to do is specified by a logical sequence of computer instructions, called a stored program. This program is stored in the storage unit of the computer system along with the required data and constants. Each instruction is contained in a storage word and is a specially coded group of bits which specify two things: (1) the operation to perform, e.g., add, subtract, load, and (2) the storage location of the data or the constant to use as the operand of the operation code. Therefore, if the computer is to do a sequence of 100 operations, at least 100 locations in the storage unit are required to store the program. The step by step execution of each instruction in the sequence is controlled by the control component. The program is executed in a definite cycle which is a repetition of the following three steps.

a. An instruction is extracted from memory and decoded in the control component.

b. The instruction is executed during which time data may be extracted from or put into storage.

c. Advance the instruction counter indicating which is the next institution to be executed.

Figure 2 shows the relationship between the five major computer components. The dotted lines represent control lines and the solid lines represent information flow.

C. COMPUTER LANGUAGE

The binary number system is the basic language or form of information representation in all digital computer systems. This language uses only two (binary) digits, zero and one, and is used primarily because it is very easy to represent physically. Many physical mediums can be used to represent binary numbers such as: an electric switch open or closed, plus or minus polarity, current or no current, a spot on a magnetizable surface (drum, disk, tape) magnetized in one direction or another, and a position on a punched card or paper tape with a hole or no hole. All of these mediums have just two stable, mutually exclusive states, and can thus readily be represented by zero or one.

1. Number Systems

All number systems can be represented in the same pattern and can then be related one to another. The following describes the common number systems used in digital computers.

a. The Pattern of Numbers

$$N = A_n r^n + A_{n-1} r^{n-1} + \ldots + A_0 r^0$$

where:

N is the number

A is an admissible symbol $\left[0 \leq A \leq (r - 1) \right]$ and is an integer,

r is the radix or base (total number of admissible marks),

n is the position of the symbol with reference to the point separating the integer from the fractional parts.

b. The Decimal Number System

The formula may seem formidable, but witness the formation of the number 4999 in the decimal system and note the pattern:

$$4999 = 4000 + 900 + 90 + 9 \text{ or}$$

$$4999 = (4 \times 10^3) + (9 \times 10^2) + (9 \times 10^1) + (9 \times 10^0)$$

Thus, all numbers are formed by stating the coefficients (symbols) of the powers (positions). Other number systems develop numbers

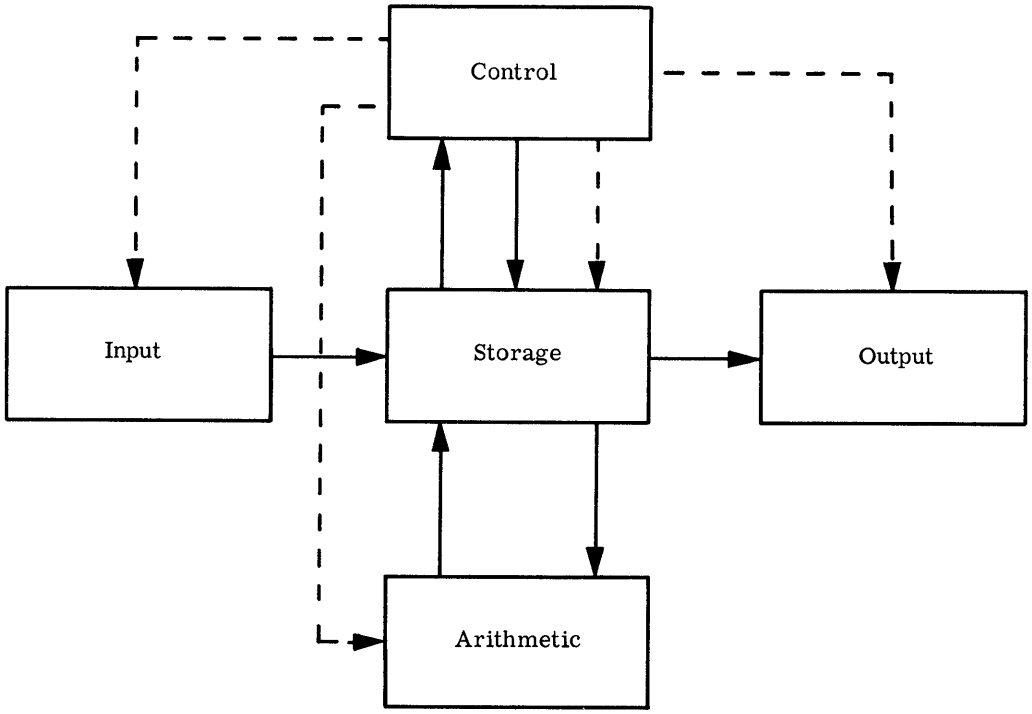


Figure 2. Five Basic Components of Digital Computers

in the same manner using radices other than 10.

c. The Binary Number System - The GE 412 Computer's Language

The binary number system has only two admissible symbols, 0 and 1, and therefore the radix is two. For example, the decimal number N represented in the binary number system as 11010 is:

$$N = (1 \times 2^4) + (1 \times 2^3) + (0 \times 2^2) + (1 \times 2^1) + (0 \times 2^0)$$

$$= (1 \times 16) + (1 \times 8) + (0 \times 4) + (1 \times 2) + (0 \times 1)$$

$$= 16 + 8 + 2 = 26 \text{ (decimal equivalent).}$$

To raise 11010 (binary) to the next number greater (i. e. binary counting), raise the extreme right-hand symbol (0) to the next higher admissible symbol (1). To raise the result, 11011, to the next number greater, move left to the first zero [in position 2^2 in this example, raise this to the next higher symbol (1)], setting everything at the right to 0. Therefore, the next number greater than 11011 becomes 11100.

The use of the binary number system to represent bistable (2 position steady state) devices is contingent upon being able to convert conveniently from one number system (decimal) to another (binary). This has proved to be a relatively insignificant problem and well worth the necessary effort. A single binary digit, either 1 or 0, is commonly called a "bit". The binary number 11011 would therefore have 5 bits.

d. The Octal Number System

The octal number system contains the

admissible symbols 0, 1, 2, 3, 4, 5, 6, and 7. Therefore, the radix is eight. Applying the octal number 356 to the pattern of numbers formula to find the equivalent decimal result we find:

$$N = (3 \times 8^2) + (5 \times 8^1) + (6 \times 8^0)$$

$$= (3 \times 64) + (5 \times 8) + (6 \times 1)$$

$$= 192 + 40 + 6 = 238 \text{ (decimal)}.$$

Actually an octal digit (0-7) may conveniently refer to a group of three binary digits, since there are eight unique configurations of each group of three binary digits. For example, $(110\ 101\ 011)_2 = (653)_8$.

e. Coded Numbers and Letters

A binary coded number expresses each digit of a number in any system by the binary notation for each digit of that number. For example, consider the decimal number 127, which can be written as:

Straight Binary:	1111111 or		
Binary Coded Decimal:	0001	0010	0111
	(1)	(2)	(7)

Other forms of BCD representation are possible. Note that the straight binary representation of the decimal number 127 requires only seven positions, although the binary coded decimal (BCD) representation of the same number uses twelve positions. The advantage of BCD is that it is easily converted position by position from BCD to decimal or decimal to BCD.

Since alphabetic characters and special symbols as well as the decimal digits 0-9 are handled by the GE 412, a form of binary notation representing these characters had to be devised. Common practice calls for the inclusion of a fifth and sixth binary digit (in addition to the four bits required to represent decimal information) to represent each alphanumeric character. The binary representation of alphanumeric characters is listed in Appendix B.

2. Binary Arithmetic

a. Addition

Binary addition is really quite simple, once the rules are learned, because there are so few combinations possible with only two digits. The complete binary, single digit, addition table for A + B is:

A \ B	0	1
0	0	1
1	1	0*

* With a one bit carry.

The one bit carry is roughly equivalent, in decimal, to the sum 1 + 9 = 0 with a carry of 1 to the next higher order position. It may also be helpful to notice that in counting one place past 001 in the table of binary integers (which corresponds to the addition of 1 and 1) gives 010. We can now perform some examples of binary addition after noting that 1 + 1 + 1 = 1 with a 1 bit carry into the next position, which will sometimes occur.

EXAMPLES

Carries	1	111	11111111
	101010	10111000	111111111
	+001001	+ 101011	+ 1
Sum	110011	11100011	1000000000

b. Subtraction

The table for the binary subtraction of B from A is no more complicated than that for addition:

A \ B	0	1
0	0	1*
1	1	0

* With a one bit borrowed from the next higher order to the left.

EXAMPLES

	0		0 0001		
45	~~1~~01101	354	~~1~~0~~1~~~~1~~~~0~~0010	44	101100
-25	-11001	-170	-10101010	-34	-100010
20	10100	184	10111000	10	1010

To simplify the operation of the computer, subtraction is actually done by forming the complement of the subtrahend and then adding the complement to the minuend. This method can be illustrated by the following decimal example using the 10's complement of the subtrahend. Notice that the 10's complement of 126,944 = (1,000,000 - 126,944) = 873,056.

Standard Subtraction	10's Complement Subtraction
493,201	493,201
-126,944	+873,056
366,257	1,366,257
	-1,000,000
	366,257

This may seem to be an exceedingly difficult way of subtracting two numbers (as, indeed it is, in decimal). However, in binary, the 2's complement of the subtrahend is easily obtained merely by changing all zeros to ones, and ones to zeros, and then adding a one. Thus, in binary, notice that the 2's complement of 100010 = (1000000 - 100010) = (011110).

Standard Subtraction	2's Complement Subtraction
101100	101100
-100010	+011110
1010	1001010
	-1000000
	1010

Note that there is a 1 bit carried out of the high-order (left) end of the latter sum. This carry is used to form the correct sign of the result. In the GE 412, the sign of the number is designated by a bit in the high order position. A zero in this position designated a positive number; a one designates a number in its complement form. Further, all negative numbers are represented by the 2's complement of the equivalent positive number. This will be clearer if we now again do the previous example, and include the sign bits in the addition.

Example 1.		Example 2.	
44	0 Δ 101100	-5	1 Δ 111011
-34	+1 Δ 011110	+7	+0 Δ 000111
10	0 Δ 001010	2	0 Δ 000010

The carry out of the sign bit is disregarded. The "Δ" symbol is used to separate the sign from the number.

c. Multiplication

This is simply a process of repeated additions. We can multiply in binary by developing an appropriate multiplication table and following a process similar to decimal multiplication.

In binary we have:

A \ B	0	1
0	0	0
1	0	1

For example:

```
     101011
      x1011
     101011
    101011
   101011
  111011001
```

Actually in the GE 412, multiplication is handled as a series of additions of the multiplicand and shifts of the product as it is formed. Each bit of the multiplier is examined in turn; if it is a one, the multiplicand is added; if it is a zero, no addition takes place. In either case, the product that is being formed is shifted one position.

d. Division

This can be carried out in binary by the same process as decimal; i.e. repeated subtraction. For example:

```
          1001 = quotient
1011 / 1101101
       1011
        10101
         1011
         1010 = Remainder
```

As in subtraction, the 2's complement of the divisor is added repetitively to the dividend to effect a subtraction. Situations occurring during multiplication and division when the operands have unlike signs are handled automatically by the computer.

3. Scale Factors

The position of the decimal point in decimal arithmetic and the binary point in binary arithmetic, must be considered in every arithmetic operation, whether done on paper, with a desk calculator, or by a computer. Most digital computers, just as desk calculators, do not have the built-in capability of keeping track of the binary or decimal point as they perform arithmetic operations. These machines

are therefore called fixed point computers. Some scientific digital computers have this built-in capability to keep track of the point and these machines are called floating point computers. The GE 412 is a fixed point binary digital computer.

A method of keeping track of the binary point must therefore be devised for the programmer to use. Locating the binary point relative to one fixed position in the 20 bit words in the GE 412 is an advantageous method. The fixed position could be anywhere in the word, but once assigned, that position must be used consistantly. In this programming manual the position between the sign bit and bit position number 1 is assigned as this fixed position used to locate the actual position of the binary point as shown below. The programmer therefore uses this method to locate the actual binary point in all numbers in the GE 412.

S	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19

reference point

The binary scale factor, designated by "B", indicates the position of the actual binary point in a word with reference to this fixed position between the sign bit and bit position 1. For example, the binary equivalent of $(9.5)_{10}$, which is $(1001.1)_2$, is represented in a 20 bit computer word with a B7, as $(0\Delta0001001100000000000)_2$. Since the fixed position is between the sign and bit position 1, the actual binary point is 7 places to the right of this position. The number therefore has a binary scale factor of 7 (B7). Consider the following examples to further understand the positioning of binary points and their associated scale factors.

Number in Decimal	B	Number in Binary	Representation in Computer Word
63.75	9	111111.11	0Δ0001111111100000000
-63.75	9	-(111111.11)	1Δ1110000000100000000
496.	19	111110000.	0Δ0000000000111110000
0.125	-2	.001	0Δ1000000000000000000
0.5	0	.1	0Δ1000000000000000000
496.	21	111110000.	0Δ0000000000001111100

Notice in the above examples, it is possible for the actual binary point to be positioned outside of the 19 bits of a word. These are the cases where a number has either leading or trailing zeros.

During arithmetic operations, care must be taken in order to position the operands for correct results of the operation and to prevent overflow, which occurs when the result is too large to be contained in the arithmetic register at the scale factor used. The following rules apply for keeping track of scale factors during arithmetic operations.

Operation	Condition
Addition	The two operands must have equal B's.
Subtraction	The two operands must have equal B's.
Multiplication	The B of the product is equal to the sum of the B's of the multiplier and multiplicand.
Division	The B of the quotient is equal to the B of the dividend minus the B of the divisor.

The programmer must know the scale factors of all numbers used in any program at all times, and use the proper rules to keep track of the scale factors as numbers are used in arithmetic operations.

II. DESCRIPTION OF GE 412 COMPONENTS

The GE 412 Process Computer System shown in Figure 3, is specially designed for logging, monitoring, and control of industrial processes. The GE 412 System features high speed core storage, magnetic drum backup storage, priority program interrupt, over 100 basic computer instructions, and rugged construction for industrial environments.

A. CENTRAL PROCESSING UNIT

The GE 412 central processing unit is a single address, stored program, fixed point, binary digital computer. The central processing unit performs the computational (arithmetic), storage, and control functions for the GE 412 System. The units that make up the central processor are shown in Figure 4.

1. Storage Section

The storage section consists of a high speed magnetic core storage unit, a backup magnetic drum storage unit, two buffer registers, and the associated address selection circuitry. A group of 20 binary digits (bits) forms the basic unit of addressable information, called a word. A word may be used to store either data, constants, or computer instructions. When a word is stored in either storage unit, a parity bit is added, making it 21 bits in length. This parity bit is used to check the validity of information as it is transferred out of storage. The buffer registers associated with the storage units generate the parity bit as a word passes into the unit and check the parity bit as the word passes out of it.

a. High Speed Storage

The high speed storage unit (Figure 5) employs thousands of tiny magnetic cores, each of which can be magnetized to represent a binary digit. Units of 4,096 or 8,192 words capacity are available for model 412A systems. Units of 12,288 or 16,384 words capacity are also available for model 412B systems.

b. The Z Register

The Z register is a focal point for information flowing into or out of high speed storage. As illustrated in Figure 4, all information passing between high speed storage and any other unit must pass through the Z register. Thus the Z register forms a buffer storage location for one word, which allows asynchronous devices to share the use of high speed storage.

c. Automatic Address Modification Locations

Automatic address modification is achieved using three locations in high speed core storage, designated 00001, 00002, and 00003. Words in these locations contain the address modifiers. The contents of bit positions 7 through 19 of the words can be automatically added to the address of an instruction in the I register. Bit positions 5 and 6 of the instruction to be modified specify which one of the three automatic address modification locations is to be used. The address portion of the instruction and the contents of the selected modification location are sent through the adder where they are added. The changed address is then returned to the I register and the instruction is executed. One additional word time (20 microseconds) is required for automatic address modification.

d. Backup Bulk Storage

A magnetic drum backup bulk storage unit (Figure 6) is available in the GE 412 systems to increase the storage capacity in multiples of 8,192 words to a maximum of 57,344 words of backup storage in model 412A systems or in multiples of 8,192 words to a maximum of 172,032 words of backup storage in model 412B systems.

The magnetic drum is divided into many tracks with a read-write head associated with each track. In model 412A systems the drum rotates at 3,600 rpm and each track contains 128 words. In model 412B systems the drum rotates at 1,800 rpm and each track contains 256 words. Since the drum speed is asynchronous to central processor timing, a special drum buffer register is provided, through which the drum storage unit has access to high speed storage as required for reading or writing operations. When reading or writing operations are being performed, the drum and the central processor have access to the high speed unit on a time-sharing basis, with central processor operations proceeding at approximately 84% of normal speed. The backup drum storage may be used for storage of data, tables, subroutines, or inactive portions of the main program as desired. Words are transferred between drum and high speed storage in blocks of from one to eight drum tracks at a rate of over 7,500 words per second.

e. The W Register

The W register provides a buffer storage means for one word of information. This allows the asynchronous magnetic drum storage to operate

Figure 3. GE 412 Process Computer System

with the Z register which is synchronized to central processor timing. Information flowing between high speed storage and backup storage passes directly between the Z register and the drum buffer register.

2. Arithmetic Section

The arithmetic section performs addition, subtraction, multiplication, and division. It makes logical decisions concerning the magnitude of numbers, algebraic signs, and over accumulations. Three registers designated A, Q, and B, each 20 bits in length, are used in arithmetic operations. The A and Q registers can operate independently or together. When combined, they form a double size word 38 bits in length plus a sign.

a. The A Register

The A register serves as the accumulator for the central processor. The contents of the A register may be interrogated for positive values, negative values, zero, odd, or even numbers in order to effect program branches. The functions which the A register performs in the arithmetic process are the following:

Holds the augend during addition
Holds the sum after addition
Holds the minuend during subtraction
Holds the result after subtraction
Holds the most significant half of the product after multiplication
Holds the most significant half of the divident before division
Holds the quotient after division
Holds the most significant half of the double length word in the execution of all double length word operations
Holds the word on which extraction is performed during the execution of the extract instruction
Carries the word to be shifted during various shift instructions

The contents of the A register are displayed on the Programming and Maintenance Console at all times.

b. The Q Register

The Q register acts as an accumulator when combined with the A register to form a double length 38-bit word plus sign. This arrangement is used for all double word length instructions.

±	1	← A →	19	±	1	← Q →	19

Information flowing from storage into the Q register must pass through the A register. The functions which the Q register performs are the following:

Holds the least significant half of the double length word during the execution of double length instructions
Holds the least significant half of the result after multiplication
Holds the least significant half of the divident prior to division
Holds the remainder after division
Holds the least significant half of the information to be shifted during double shift instructions
Can be shifted right or left along with the N, M, and A registers in special shift instructions

The contents of the Q register may be displayed on the Programming and Maintenance Console at any time.

c. The B Register

The B register serves as a one word buffer storage means between the arithmetic and control sections and the Z register. All information flowing from high speed storage via the Z register to other internal registers passes through the B register. The B register is used to hold the operand of arithmetic operations after the operand has been accessed from high speed storage. The high speed storage unit and the Z register may then be used for other functions at the same time as the execution of instructions that do not require an operand from storage, such as shifting and branching instructions, or instructions that require more than one word time for execution, as do multiply and divide. This allows the transfer of information between high speed storage and backup storage to take place at the same time the central processor is executing instructions. The B register has the following functions during arithmetic operations:

Holds the addend for addition
Holds the subtrahend for subtraction
Holds the multiplicand for multiplication
Holds the divisor for division

The contents of the B register may be displayed on the console at any time.

3. Control Section

The control section governs the sequential execution of the individual instructions of the stored program. It consists of three registers for automatic

control, automatic priority program interrupt, and a programming and maintenance console for manual control.

a. The I Register

The I register is the instruction register. It holds the 20 bits of an instruction during the execution of that instruction.

0 – 4	5 6	7 – 19
Operation Code	x x	Operand Address

Bit position 0 through 4 indicate the operation which is to be performed. Bits 5 and 6 refer to the automatic address modification location to be used, if any. Bit positions 7 through 19 refer to the operand storage address in instructions that require an operand from memory; or, when an operand address is not required, these bits have various meanings, as indicated in the instruction repertoire.

The contents of the I register are displayed on the console at all times.

b. The P Register

The program address register (P register) is the location which controls the sequential execution of instructions. It holds the memory address of the next instruction to be executed. The P register is incremented by one before the execution of an instruction so that it normally contains the address of the next instruction in sequence. The 13 (model 412A) or 14 (model 412B) bits of the P register are displayed on the console at all times. The contents of this register may be stored in a specified automatic address modification location; and it may be loaded directly from the I register.

c. Automatic Priority Program Interrupt

The automatic priority program interrupt feature permits execution of functions according to their planned priority. It allows the computer to keep under constant surveillance critical points in a process without consuming valuable computer time by constantly scanning these points. This feature enables the computer to immediately recognize the random occurrence of critical conditions and promptly take whatever action may be necessary.

The basis of the automatic interrupt is the priority interrupt register. This is a 12 bit register divided into three priority groups, each group having 4 interrupt levels. Each interrupt source is associated with a bit in the interrupt register. The program can enable Group I, Groups I and II, and Groups I, II, and III to be interrupted. After enabling certain groups, the program can then permit or inhibit an automatic program interrupt. When an interrupt occurs, the contents of the P register are stored in core storage location $(0006)_8$ the contents of the A register are stored in location $(0007)_8$ and control is transferred to the instruction in location $(0007)_8$ plus the priority of the interrupt, (1 through 12). If the source of the interrupt was from priority level 2, then control is transferred to the instruction in location $(0011)_8$.

d. The Programming and Maintenance Console.

The programming and maintenance console shown in Figure 7 provides the indicating control center for the programmer and product service engineer. It permits manual control in contrast to automatic or program control. Manual control is used to initially load the program into memory, start program execution, monitor the progress of the program, and occasionally stop the program for maintenance and trouble shooting. The central processor's operating status can be seen from the display lights on the console panel. The lights show the contents of the A register, the I register, and the P register. Twenty switches on the console permit direct loading into the A register. The console has parity and overflow alarm lights and an automatic-manual lockout switch. The console has other indicating lights and a switch for selection and display of other registers in the computer.

4. Peripheral Section

The peripheral input-output devices such as the paper tape readers, paper tape punches, and typewriters operate much slower than the internal speed of the computer. The M and N registers are provided as one-character buffer registers to allow the slow input- output devices and the central processing unit to operate simultaneously. The contents of the M and N registers are transferred to and from the A register by shifting. These registers are 7 bits in length and store the binary-coded representation of one character as it is typed or punched, or as the character is read from the paper tape reader. This technique allows two peripheral devices and the central processing unit to operate simultaneously without loss of time or facility.

5. Special Real-Time Features

Process computer systems require special facilities not normally found in computer systems. These facilities allow the process computer system to operate efficiently on a real-time basis, being continuously aware of the actual time of day and elapsed time intervals.

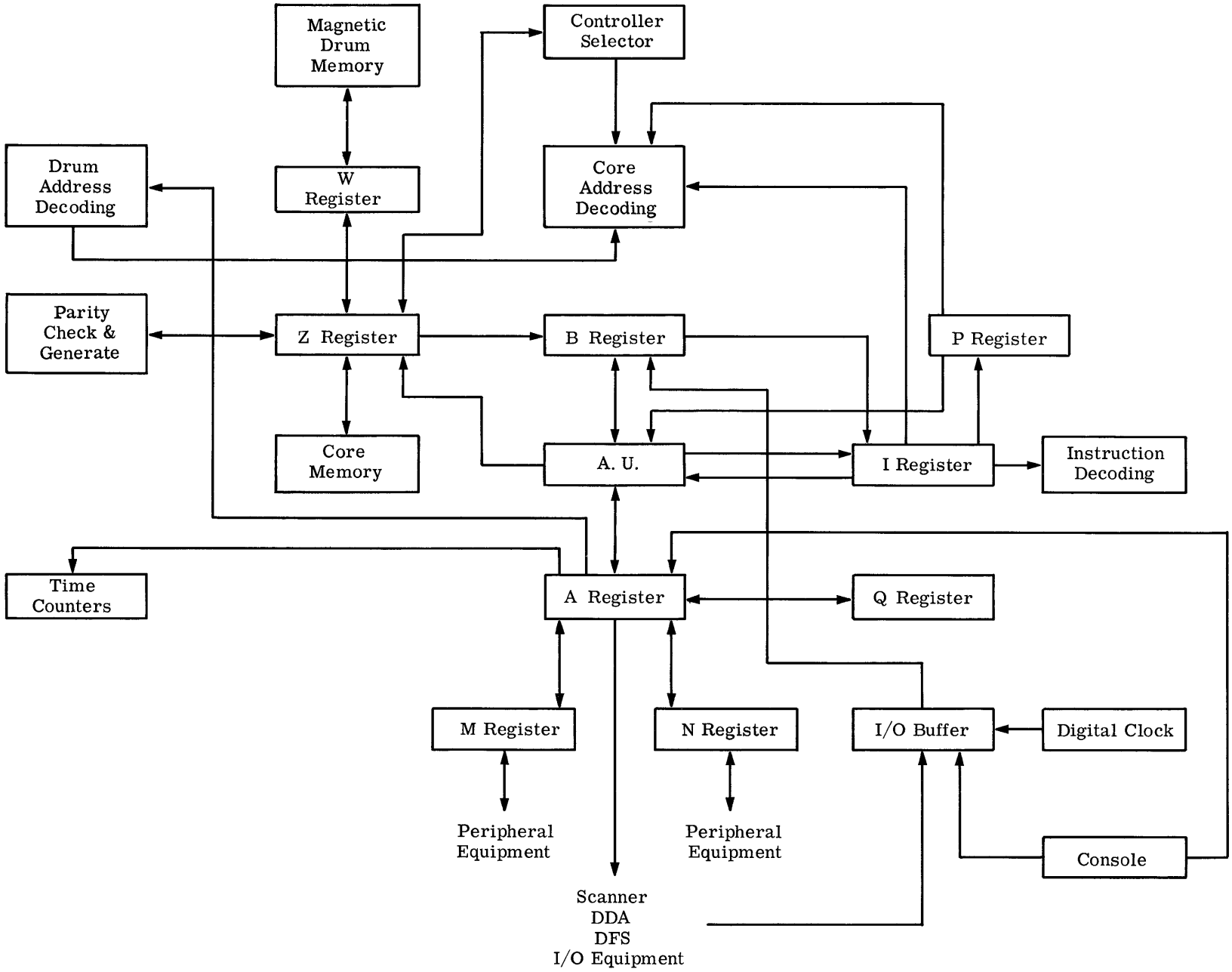


Figure 4. Information Flow in the GE 412

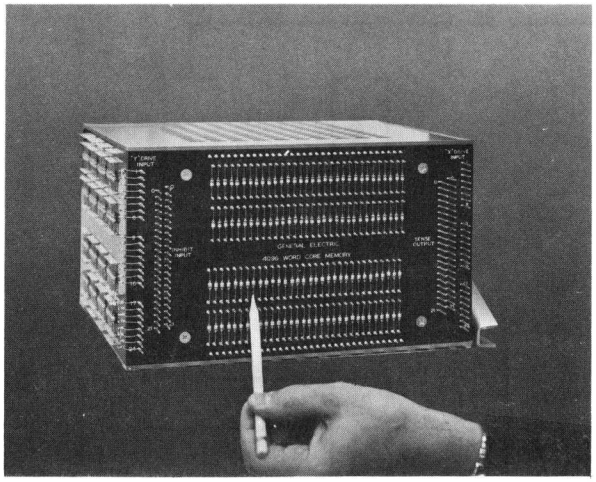


Figure 5. High Speed Storage Unit

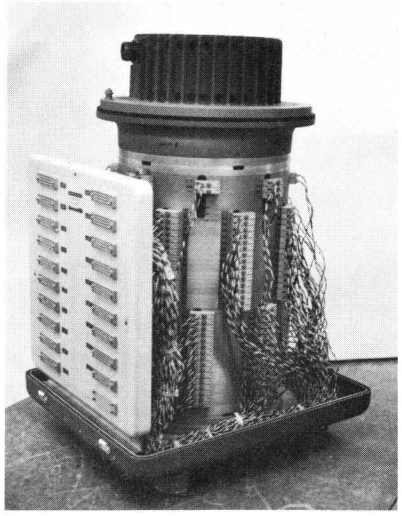


Figure 6. Backup Storage Unit

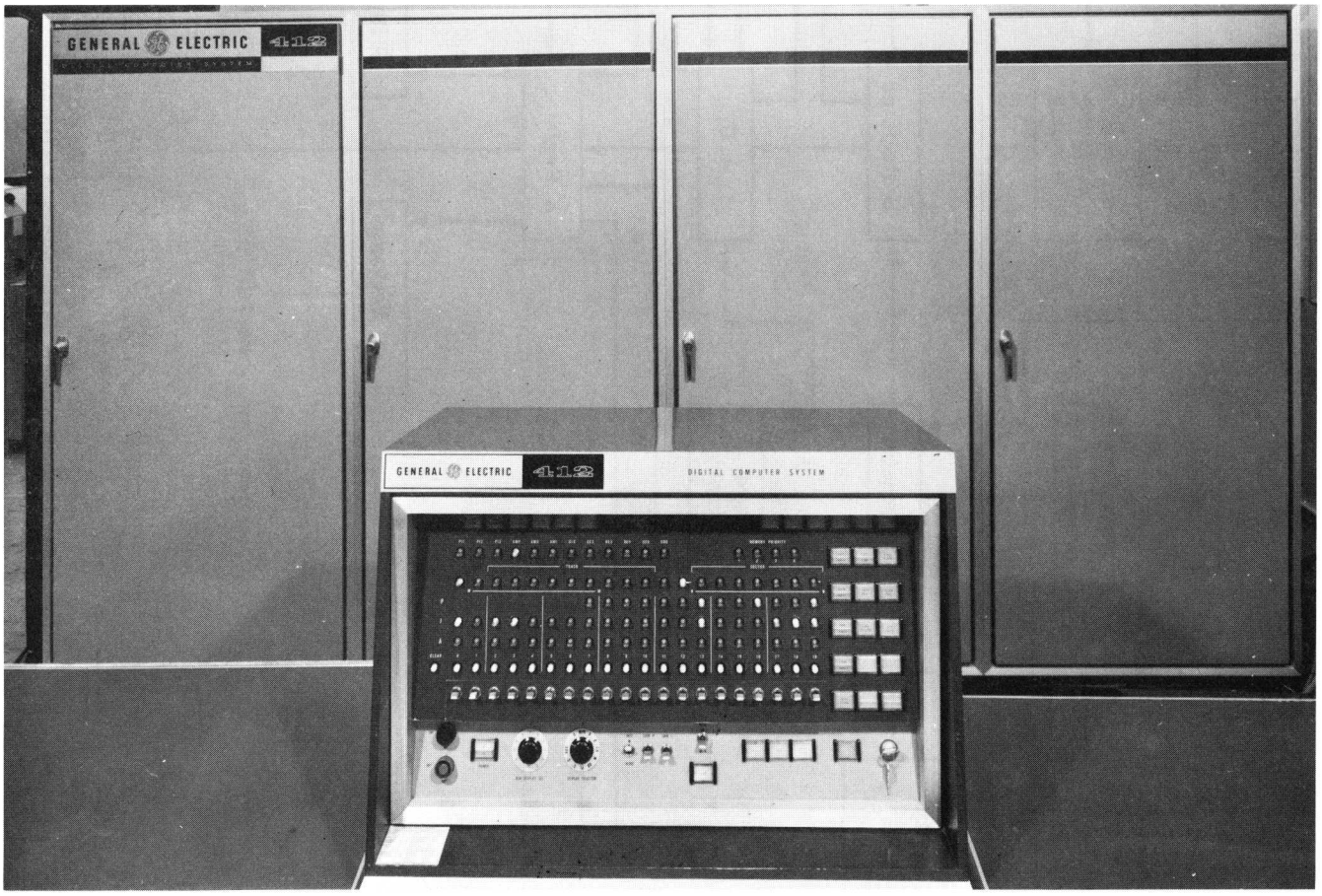


Figure 7. GE 412 System's Programming and Maintenance Console

a. Real-Time Clock

A 24-hour real-time solid state digital clock is provided as an integral part of the GE 412 System. Six binary-coded-decimal characters representing hours, minutes, and seconds may be transferred from the digital clock directly into the A register by a computer instruction. The digital clock derives its timing directly from a 60 cycle a-c source in the computer. Manual clock reset pushbuttons are provided to set time into the clock originally.

b. Elapsed Time Counters

The GE 412 System provides four elapsed time counters capable of counting elapsed time intervals under program control. Intervals from 3.2 milliseconds to over four hours can be timed by the elapsed time counters. Timing increments of 3.2 milliseconds, 16.7 milliseconds, one second, and one minute may be used by the counters. They are 8 bits in length, thus capable of counting up to 256 increments of time before overflowing. The overflow of a counter can be sensed by a branch instruction in the program or sent to the automatic priority program interrupt register to automatically interrupt the program. A counter is initiated by transferring a control word to the selected counter directly from the A register under program control. The control word specifies the interval to be timed, the interrupt status, and the time increment to be used. The four elapsed time counters are therefore under direct control of the program and afford great flexibility in elapsed time counting.

B. PERIPHERAL INPUT-OUTPUT EQUIPMENT

Peripheral input-output equipment consists of the digital devices used by the process operators, programmers, and product service engineers for communication with the central processing unit. They are used in loading, checking, and modifying the computer programs. Information flowing to and from the peripheral devices goes through the M and N registers.

1. Paper Tape Readers

Two types of readers are used for input with the GE 412 System. The high speed paper tape reader reads 8-channel punched paper tape at a rate of 100 characters per second. Computer instructions, data, constants, and other information may be read into the computer one character at a time from the reader through the M or N register. The low speed paper tape reader is identical to the high speed reader except that it reads at a rate of 20 characters per second.

2. Paper Tape Punches

Two types of punches are used for output with the GE 412 System. The high speed paper tape punch produces standard 8-channel paper tape codes. It is capable of punching at a rate of 100 characters per second. The low speed paper tape punch is identical to the high speed punch except that it operates at a rate of 20 characters per second.

3. Electric Typewriters

Several types of output typewriters are available for the GE 412 System. They include variations of standard IBM and Friden machines which operate at average rates of about 8 characters per second and the IBM Selectric typer which operates at about 15 characters per second. Variations include alphanumeric and numeric typesets and 12, 20, and 30-inch carriages.

C. SYSTEM INPUT-OUTPUT EQUIPMENT

Many types of equipment are provided as optional devices in the GE 412 System. The exact complement of equipment used in a particular system is governed by the requirements of the process.

1. System Operation Input-Output Equipment

The system operation input-output equipment is used mainly in communication between the GE 412 System and the operator or operators of the process.

a. Operator Console

The operator console provides a means for communication between the operator and the process via the computer. Various types of consoles are available, depending on the functional requirements of the system. See Figure 8.

b. Parallel Entry Column Printer

The parallel entry column printer permits line-by-line printout of 11 numeric characters per line under program control. A printing speed of five lines per second is possible. It is primarily used to print alarm values, but may be used to log or tabulate other selected values. The printer may be located as far as 50 feet from the central processor.

c. Serial Entry Column Printer

The serial entry column printer is used for the same functions as the parallel entry column printer. This printer prints 10 characters per line at a rate of one line per second.

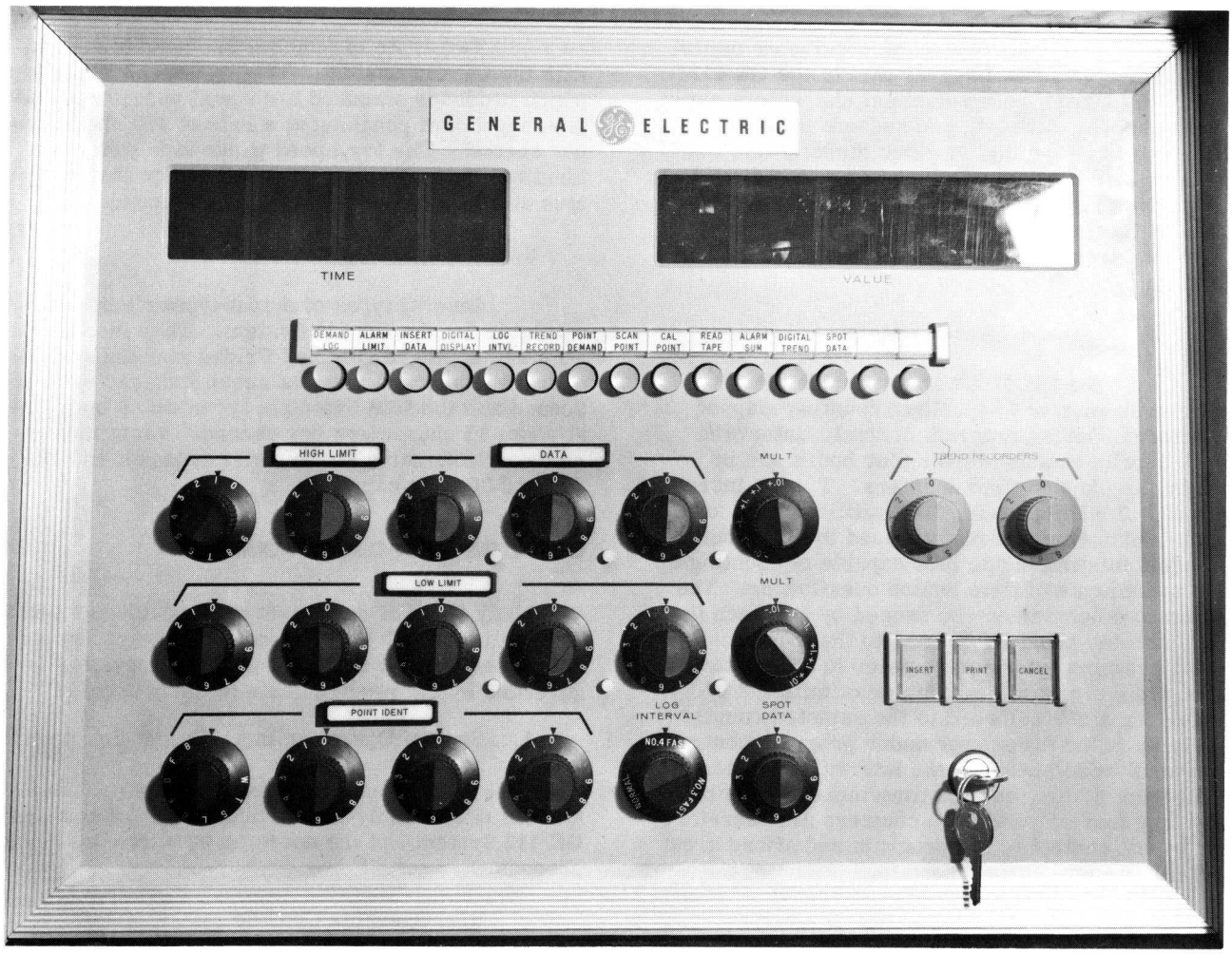


Figure 8. Typical Operator's Console, General Purpose Type

d. Stack Fed Card Reader

The stack fed card reader is an input device which reads alphanumeric information into the computer from Hollerith punched cards at a rate of 30 cards per minute. The reader is stack fed and has both feed-check and out-of-cards check.

e. Single Entry Card Reader

The single entry card reader reads alphanumeric information into the computer from Hollerith punched cards at an average rate of 9.5 characters per second. The cards are manually placed in the reader one at a time.

f. Trend Recorders

The GE 412 System may incorporate both graphic (analog) and tabular (digital) trend recorders. Under program control digital characters may be transmitted to tabular recorders or analog signals may be transmitted to drive graphic recorders. Graphic recorders range from one single-colored pen recorder to variable-colored multipen recorders.

2. Process Input-Output Equipment

Process input-output equipment is used for communication between the process and the GE 412 System.

a. The Scanner-Distributor

The scanner-distributor is the major communication device used for communication between the process and the computer. It mates the GE 412 System to process instruments and controllers for on-line process control computing. The exact employment of the scanner-distributor is dependent upon the requirements of the process, but flexibility and facility make it compatible with almost any process sensing or controlling equipment. The scanner-distributor has two basic modes of operation: analog-to-digital (input), and subcontrol (output).

Selection of a unique pair of contacts to sense the output analog signal from a sensor, or selection of a unique contact through which to distribute a voltage is accomplished with a mercury-wetted relay matrix multiplexer. The relay matrix is available in sizes from 96 to 1536 pairs of contacts. The position of the desired pair or individual contact in the relay matrix is specified by a control word transferred from the computer to the scanner-distributor. This matrix address is then decoded by the control circuits of the scanner-distributor and the specified pair of contacts or individual contact selected.

(1) Analog-to-Digital Mode. As an input device operating under program control, the scanner-distributor selects one of the many analog inputs from sensors to be scanned. It then converts this analog signal to binary digital form for use in the computer. The characteristics of this analog sensor signal such as matrix address, polarity, and range are specified by a command word supplied to the scanner-distributor by the computer. Once the scanner-distributor receives this command word, it executes the operation independently of the computer, and thus frees the computer to perform other tasks. The digital equivalent of the analog input signal is placed in the converter register. The contents of this register may then be read directly into the A register of the central processor.

The selection and digitizing of an analog sensor input may be accomplished at one of two rates selected under program control for each input. At the lower rate, which yields 12-bit conversion accuracy, it requires about 125 milliseconds. At the higher rate, yielding 10-bit conversion accuracy, it requires approximately 50 milliseconds. Thus the normal maximum rate of scanning is 20 points per second. If higher scanning rates are desired, they may be achieved through the use of multiple input channels, with a maximum rate of 114 points per second possible with 8 input channels.

If the full-size relay matrix does not provide sufficient input or output capability, up to five distinct scanner-distributors may be connected into the GE 412 system.

(2) The C Register. The C register or converter register is physically part of the analog-to-digital converter; it has two functional purposes. While functioning as an integral part of the analog-to-digital converter, the C register holds the "count" proportional to the input analog voltage. The count range is 0 to 4095 and is indicated in 12 bit positions of the register. In its other function, the C register serves as a 12-bit computer output register. Acting as an output buffer, it holds setup information for electronic or relay drivers for performing such operations as resetting analog outputs or updating visual displays.

(3) Subcontrol Mode. The scanner-distributor functions as a distributor when it allocates or distributes a selected voltage signal to a selected system or process device. The selected voltage to be distributed, the duration of distribution, and the desired relay matrix address is specified in the command word transferred to the scanner-distributor from the computer. Using subcontrol mode of operation, the scanner-distributor can activate many operations. It can activate the conversion of a binary number stored in the C register to an analog signal to be applied to a process controller or trend recorder. It can activate the read-in of digital switches, digital counters, special digital sensors, and similar devices. It can also open or close a relay in order to turn on or off equipment such as motors or pumps. Designated voltages may also be used to step a stepping-switch control device.

b. Output Distributor

In applications such as fully-automated hot strip steel mills, which require large numbers of system outputs (subcontrol functions) in addition to relatively large or fast input scanning requirements, it is advantageous to separate the two functions. This is possible through the use of the output distributor, which performs essentially the same functions as the subcontrol mode of operation of the scanner distributor. The output distributor has its own independent matrix of 128 or 256 relays and thus operates entirely independently of the scanner.

c. Fast Digital Scanner

The fast digital scanner is an input device which samples the open-closed status of sensor contacts selected at random in groups of 16. A maximum of 64 groups, or 1024 contacts may be scanned

by the fast digital scanner. It is used where it is desirable to sense the open-closed positions of contacts at electronic speeds. Typical applications are the scanning of two-condition elements such as high-low pressure or temperature sensors, open-closed valve positions, on-off status of motors or pumps, and brakes open or closed. If required, up to five separate and independent fast digital scanners may be used in one GE 412 system.

d. Digital Data Accumulator

The digital data accumulator is an input device used to temporarily accumulate digital sensor information before it is transferred into the computer for processing. Accumulators are available that count up to 65,535 pulses. Using the digital data accumulator it is possible to count such things as prime feet, damaged feet, off-gage feet, and off-color feet of steel being processed in a continuous strip, or kilowatt-hour counts, flow meter pulses, or counts from a coal weighing scale in steam electric plants. The accumulations are then read into the computer and used to up date cummulative totals. If required, up to five separate and independent digital data accumulators may be used in one GE 412 system.

e. Digital-to-Analog Converter

The digital-to-analog converter provides an analog equivalent to a digital value. Typical applications are the computation of an optimum set point for a process controller or regulator and the transmission of this set point in the form of an analog signal to the controller. The scanner-distributor and the C register are used in conjunction with the digital-to-analog converter to accomplish this type of output.

f. Thermocouple Reference Unit

A thermocouple cold reference junction is available for applications which use thermocouples as sensors. The reference is of a special GE floating design which provides an extremely accurate temperature measurement. The computer program can use this cold reference termperature in linearizing the output voltages from thermocouples to compute the actual temperature measured by the thermocouple.

III. PROGRAMMING FUNDAMENTALS

The GE 412 System has over 100 instructions used to control the operation of the computer and its various input/output devices. This section of the Programming 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 20 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 multiples of the computer word time (20 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 a memory location. 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, I_{7-19} should be read as "bits 7 through 19 of the I register." $C(I)_{7-19}$ should be read as "the contents of bits 7 through 19 of the I register." $A_{s,1-4}$ is equivalent to A_{0-4}.

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.

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

Letter	Designation	No. Bits	Bit Positions
A	Primary Arithmetic Register	20	s, 1-19 or 0-19
Q	Auxiliary Arithmetic Register	20	s, 1-19 or 0-19
M	Peripheral Register	7	1-7
N	Peripheral Register	7	1-7
H	Peripheral Register	44	0-43

Letter	Designation	No. Bits	Bit Positions
X*	Auto. Add. Modification Location	13/14	7-19/6-19
I	Instruction Register	20	0-19
P	Program Address Register	13/14	7-19/6-19
C	Converter Register	12	8-19

*Actually designates specific memory location rather than a register.

Appendix E summarizes the various instruction formats and letter designations used in specifying operands in the instruction definitions.

A. INTERNAL EFFECT INSTRUCTIONS

Internal effect instructions are those that control the operation of all of the internal registers and components of the GE 412 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 0 through 4 of the instruction, see I register format page 11.

a. Data Transfer Instruction Definitions

LDA	Y	2	00

LOAD A. C(Y) Replace C(A). Y is not changed.

STA	Y	2	03

STORE A. C(A) Replace C(Y). A is not changed.

DLD	Y	3	10

DOUBLE LENGTH LOAD. If Y is even, the (C(Y) and C(Y+1) replace C(A) and C(Q). If Y is odd, C(Y) replace C(Q) and C(A). Y and Y+1 are unchanged. If this instruction is automatically modified, the address after modification determines the result as indicated above.

DST	Y	3	13

DOUBLE LENGTH STORE. If Y is even, C(A) and C(Q) replace C(Y) and C(Y+1). If Y is odd, C(Q) replace C(Y). The contents of A and Q are unchanged. If this instruction is automatically modified, the address after modification determines the result as indicated above.

STO	Y	2	27

STORE OPERAND ADDRESS. $C(A)_{7-19}$ replace $C(Y)_{7-19}$. C(A) and $C(Y)_{s,}$ 1-6 are unchanged.

b. 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 BOV of Section III 5b.)

ADD	Y	2	01

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

SUB	Y	2	02

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

DAD	Y	3	11

DOUBLE LENGTH ADD. If Y is even, C(Y) and $C(Y+1)_{1-19}$ are algebraically added to C(A) and $C(Q)_{1-19}$. If Y is odd, C(Y) and $C(Y)_{1-19}$ are algebraically added to C(A) and $C(Q)_{1-19}$. The result is placed in A and Q_{1-19}. The result is placed in A and Q_{1-19}. The sign of Q is set to agree with the sign of A. C(Y) and C(Y+1) are unchanged. If this instruction is automatically modified, the address after modification determines the result as indicated above.

DSU	Y	3	12

DOUBLE LENGTH SUBTRACT. If Y is even, C(Y) and $C(Y+1)_{1-19}$ are algebraically subtracted from C(A) and $C(Q)_{1-19}$. If Y is odd, C(Y) and $C(Y)_{1-19}$ are algebraically subtracted from C(A) and $C(Q)_{1-19}$. The result is placed in A and Q_{1-19}. The sign of Q is set to agree with the sign of A. C(Y) and C(Y+1) are unchanged. If this instruction is automatically modified, the address after modification determines the result as indicated above.

MPY	Y	13-18	15

MULTIPLY. C(Y) are algebraically multiplied by C(Q). The result is placed in A and Q_{1-19} with the most significant half in A. The sign of Q is the same as the sign of A after multiplication. 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. The overflow indicator is turned off by this instruction. The execution time is determined by the bit pattern of the multiplier (the number in the Q register). The following rule-of-thumb will give the execution time for any specific case: Starting with a base time of 12-1/2 word times, add 1/4 word time for each 1 bit in the multiplier exclusive of the sign bit. Any fractional word time in the result should be taken as a full word time.

DVD	Y	25, 28	16

DIVIDE. C(A) and $C(Q)_{1-19}$ are algebraically divided by C(Y). The quotient is placed in A; the remainder is placed in Q. The sign of the remainder is the sign of the dividend. The overflow indicator is turned off at the beginning of the execution of this instruction. The magnitude of the divisor must be greater than the magnitude of A. If not, the overflow indicator is turned ON and control is immediately transferred to the next instruction in sequence. C(Y) are unchanged. Execution will require 25 word times if the dividend is positive, 28 word times if the dividend is negative.

c. Examples of Data Transfer and Arithmetic Instructions

(1) Sum the numbers stored in octal locations 11500 and 11501 that have equal scale factors and store the sum in location 03000. Start the program in octal location 02000.

Location of Instruction	Operation Code	Operand Address	Actual Form of Instruction in Octal	Remarks
02000	LDA	11500	0011500	Load C(11500) into A
02001	ADD	11501	0111501	Add C(11501) to A
02002	STA	03000	0303000	Store sum
02003	next instruction			

(2) Sum the double precision number in locations 12550 (upper half) and 12551 (lower half) to the double precision number in locations 12560 and 12561, store the sum in octal locations 12000 and 12001. The numbers have equal scale factors. Start program in octal location 17700.

Location of Instruction	Operation Code	Operand Address	Actual Form of Instruction in Octal	Remarks
17700	DLD	12550	1012550	Load A & Q with 1st number
17001	DAD	12560	1112560	Add to A & Q second number
17702	DST	12000	1312000	Store double length sum
17703	next instruction			

(3) Solve R = X + Y - Z starting the program in octal location 01006.

Location	Data	B(scale factor)
04000	X	5
04001	Y	5
04002	Z	5
04003	R	5

Location of Instruction	Operation Code	Operand Address	Actual Form of Instruction in Octal	Remarks
01006	LDA	04000	0004000	Load A with X
01007	ADD	04001	0104001	Add Y to X
01010	SUB	04002	0204002	Subtract Z from X + Y
01011	STA	04003	0304003	Store R
01012	next instruction			

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 0 through 4 and 7 through 19 specify the operation code. The octal code in the heading of these instructions is given as a 7 digit octal number equivalent to the 20 binary bits of the instruction. These instructions are not normally modified by use of an X location.

a. Register Manipulation Instruction Definitions

LQA	2	2504004

LOAD Q FROM A. C(A) replace C(Q). C(A) are unchanged.

LAQ	2	2504001

LOAD A FROM Q. C(A) replace C(A). C(Q) are unchanged.

XAQ	2	2504005

EXCHANGE A AND Q. C(A) and C(Q) are interchanged.

MAQ	2	2504006

MOVE A TO Q. C(A) replace C(Q). Zeros replace C(A).

LDZ	2	2504002

LOAD ZERO INTO A. A is loaded with zeros.

LDO	2	2504022

LOAD ONE INTO A. A is cleared, and a "1" is placed in A_{19}.

LMO	2	2504102

LOAD MINUS ONE INTO A. A is loaded with "1's".

ADO	2	2504032

ADD ONE. Plus one is added algebraically to A_{19}. If the capacity of A is exceeded, the overflow indicator is turned ON.

SBO	2	2504112

SUBTRACT ONE. One is subtracted algebraically from A_{19}. If the capacity of A is exceeded, the overflow indicator is turned ON.

CPL	2	2504512

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	2	2504532

NEGATE A. The 2's complement (negative value) of C(A) replaces C(A). If the capacity of A is exceeded, the overflow indicator will be turned ON.

CHS	2	2504040

CHANGE SIGN OF A. The sign of A is changed. Positions 1-19 of A are unchanged.

NOP	2	2504000

NO OPERATION. No operation is performed.

b. Examples of Use of Register Manipulation Instructions

(1) Solve R = (X·Y) + 1. Start program in octal location 05000.

Location	Data	B	Range
07007	X	8	1<X<200
07010	Y	11	1<Y<1000
07011	R	19	2<R<200,001

Location of Instruction	Operation Code	Operand Address	Actual Form of Instruction in Octal	Remarks
05000	LDA	07007	0007007	Load A with X(B8)
05001	MAQ		2504006	Move X to Q(B8)
05002	MPY	07010	1507010	(X·Y) in A and Q B19.
05003	ADO		2504032	(X·Y)+1 in A and Q B19.
05004	STA	07011	0307011	Store upper significant
05005	next instruction			half of (X·Y)+1

(2) Solve $R = \frac{X}{Y} + Z$. Start program in octal location 00500.

Location	Data	B	Range
01500	X	18	0<X<100,000
01700	Y	10	500<Y<1000
01777	Z	8	Z = 1 (constant)
02100	R		

Location of Instruction	Operation Code	Operand Address	Actual Form of Instruction in Octal	Remarks
00500	LDZ		2504002	Zero A and Q since
00501	MAQ		2504006	numerator is single length.
00502	LDA	01500	0001500	Load A with X B18
* 00503	DVD	01700	1601700	Divide X by Y, B8
00504	ADD	01777	0101777	Add Z to X/Y, B8
00505	STA	02100	0302100	Store result, B8
00506	next instruction			

*Note absolute form of the maximum size of X in the A register is less than the absolute form of minimum value of Y in storage and therefore the divide instruction will not cause overflow. $(100,000)_{10}=(303240)_8$ $(500)_{10}=(764)_8$

0Δ0110000110101000000
0Δ0111110100000000000

A Register = X B18
Core Storage = Y B10

Absolute binary form X< absolute binary form of Y.

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 Instruction Definitions

ORY	Y	2	23

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

EXT	Y	2	20

EXTRACT. Each bit of Y is examined. If there is a "1" bit in Y in a given position, a zero is placed in the corresponding position of A. If there is a zero in a given position of Y, the corresponding position in A is left unchanged. Y is unchanged.

ANA	Y	2	22

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	2	21

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. Examples of Logical Instructions

Extract only bits 10 through 19 out of location 11000 and combine them with the information in location 12150 bits 0 through 9. Store combination in location 12151.

Location	Data
11000	Bits 10 through 19 important
12150	Bits 0 through 9 important
12151	Combination
10000	Constant (0001777)
	1 bits in positions 10 through 19.

Location of Instruction	Operation Code	Operand Address	Actual Form of Instruction in Octal	Remarks
01500	LDA	11000	0011000	Load A with C(11000)
01501	ANA	10000	2210000	And out bits 10 through 19
01502	STA	12151	0312151	Store temporarily
01503	LDA	12150	0012150	Load A with C(12150)
01504	EXT	10000	2010000	Extract bit 0 through 9
01505	ORY	12151	2312151	Combine with bits 10 through
01506	next instruction			19 of location 12151

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); they shift C(A) and C(Q) serially to the right with C(M) and/or C(N); they shift C(A) serially to the right with both C(M) and/or C(N). A maximum number of 31 places can be shifted; and this number is specified by bits 15 through 19 of the shift instruction. The execution time for shift instructions vary between 2 and 10 word times, depending on the number of shifts specified. Two word times are required for a shift of 2 bit positions or less. One additional word time is required for each additional 4 bit positions shifted or fraction thereof.

Shift instructions take advantage of a micro-coding technique in specifying the instruction. Bits 0 through 4 of shift instructions specify the standard shift code and bits 7 through 14 are assigned special significance to further define the exact type of shift to be accomplished.

The format below indicates the significance of these bits for all right shift instructions. For example, the micro-coding for a shift A into N instruction is a one bit in position 11 and zeros in positions 7, 8, 9, 10, 12, 13, and 14.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
1	0	1	0	0	X	X	0	N→A	M→A	Q→A	A→N	A→M	A→Q	A→A	←		K		→

The significance of the micro-coding bits in shift left instructions is as follows:

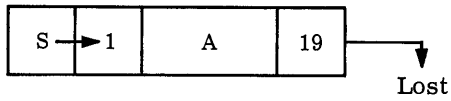
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
1	0	1	0	0	X	X	1			Q→A		A→A	Log-ical	Normal-ize	←		K		→

The normal shift instruction 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. The shift instructions involving the M and N registers are described in section (I. B. 1.). In the octal codes shown for all shift instructions the value of K is zero.

a. Instruction Definitions

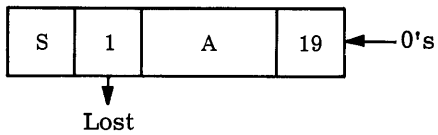
SRA	K	2+	2400000

SHIFT RIGHT A. $C(A)_{1-19}$ 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 19 are lost. The sign of A is unchanged.



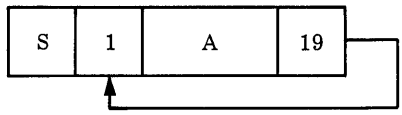
SLA	K	2+	2410000

SHIFT LEFT A. $C(A)_{1-19}$ are shifted left K places. Vacated positions of A are filled with zeros. If the sign of A is positive, the overflow indicator will be turned on if a "one" bit is shifted out of A. If the sign of A is negative, the overflow indicator will be turned on if a "zero" bit is shifted out of A. The sign of A is unchanged.



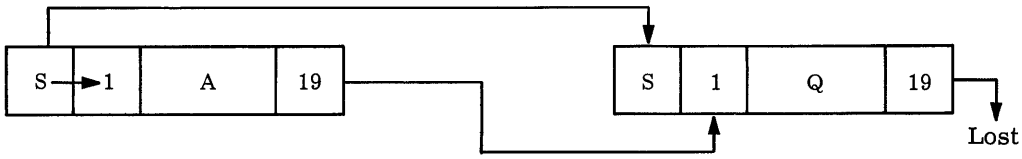
SCA	K	2+	2400040

SHIFT CIRCULAR A. $C(A)_{1-19}$ are shifted right K places in a circular fashion; that is, the bit shifted out of position 19 is inserted in position 1, replacing the bit shifted out of position 1. The sign of A is not affected.



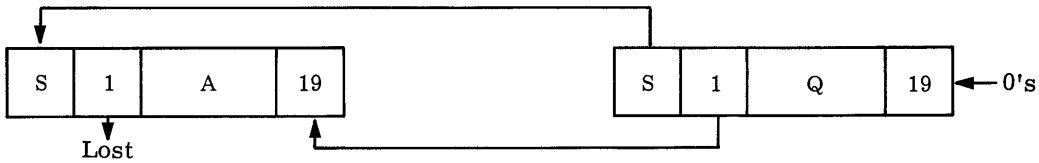
SRD	K	2+	2400100

SHIFT RIGHT DOUBLE. $C(A)_{1-19}$ and $C(Q)_{1-19}$ together are shifted K places to the right. Bits shifted out of A_{19} shift into Q_1. Bits shifted out of Q_{19} 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. The sign of Q is replaced by the sign of A. The sign of A is unchanged.



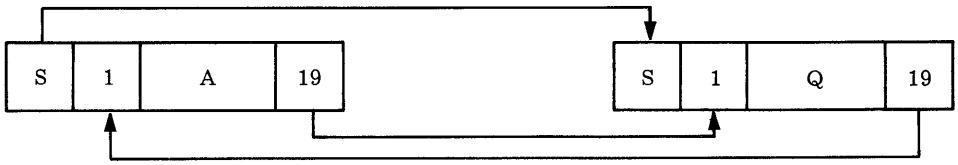
SLD	K	2+	2411000

SHIFT LEFT DOUBLE. $C(A)_{1-19}$ and $C(Q)_{1-19}$ together are shifted K places to the left. Bits shifted out of Q_1 shift into A_{19}. 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 out of A. If the original sign of A is negative, the overflow indicator will be turned on if a "zero" bit is shifted out of A. The sign of Q replaces the sign of A. The sign of Q is unchanged.



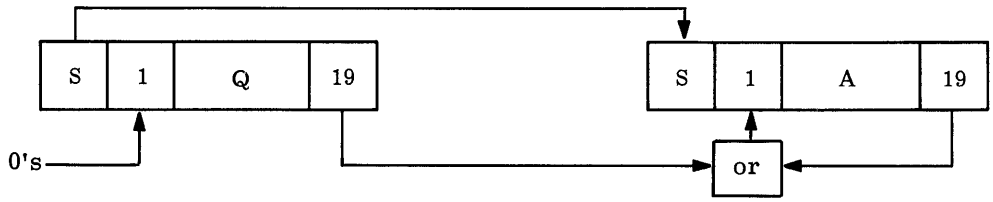
SCD	K	2+	2401100

SHIFT CIRCULAR DOUBLE. $C(A)_{1-19}$ and $C(Q)_{1-19}$ together are shifted K places to the right in a circular fashion. Bits shifted out of A_{19} shift into Q_1 and those from Q_{19} shift into A_1. The sign of A replaces the sign of Q. The sign of A is unchanged.



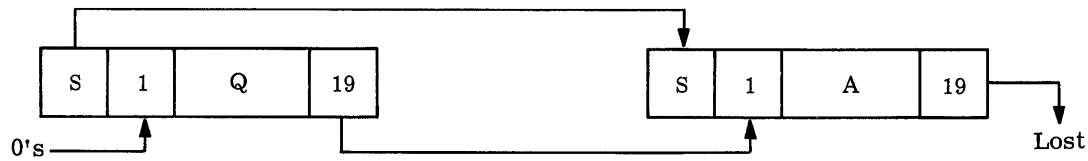
OQA	K	2+	2401040

OR Q INTO A. $C(A)_{1-19}$ and $C(Q)_{1-19}$ are each shifted K places to the right. Bits shifting out of A_{19} are combined in an "or" fashion with the bits shifting out of Q_{19} and the result placed in A_1. The sign of Q replaces the sign of A. The sign of Q is unchanged and zeros shift into the vacated positions of Q. Note that if a K of 19 is specified the $C(A)_{1-19}$ and the $C(Q)_{1-19}$ are combined in an "or" fashion.



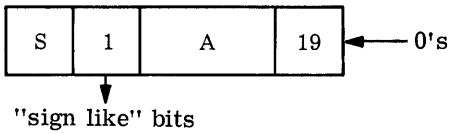
SQA	K	2+	240100

SHIFT Q TO A. $C(Q)_{1-19}$ and $C(A)_{1-19}$ are shifted K places to the right. Bits shifted out of Q_{19} shift into A_1. Bits shifted out of A_{19} are lost. The sign of Q replaces the sign of A. The sign of Q is unchanged. Zeros replace the vacated positions of Q.



NOR	K	2+	2410040

NORMALIZE A. A_{1-19} are shifted left K places or until $A_1 \neq A_0$. The value (K minus the number of places shifted) is placed in $C(00000)_{15-19}$. Zeros replace the vacated positions of A. The sign of A is unchanged. The $C(00000)_{0-14}$ are set to zero. This operation is a single-length arithmetic normalize and may be indexed.

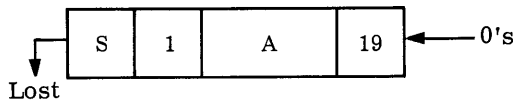


DNO	K	2+	2411040

DOUBLE LENGTH NORMALIZE. A_{1-19} and Q_{1-19} are shifted left (with Q_1 moving to A_{19}) K places or until $A_1 \neq A_0$. The value (K minus the number of places shifted) is placed in $C(00000)_{15-19}$. Zeros replace the vacated positions of Q. The signs of A & Q are unchanged. $C(00000)_{0-14}$ are set to zero. This operation is a double-length arithmetic normalize and may be indexed.

LLA	K	2+	2410100

LOGICAL LEFT A SHIFT. A_{0-19} 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.

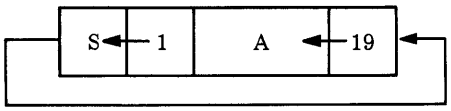


LLD	K	2+	2411100

LOGICAL LEFT DOUBLE SHIFT. A_{0-19} and Q_{1-19} 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. The sign of Q is unchanged.

LLC	K	2+	2410300

LOGICAL LEFT CIRCULAR SHIFT. A_{0-19} are shifted K places to the left in a circular fashion. A_0 shifts into A_{19}, A_1 shifts into A_0. The overflow flip-flop is not set by this shift.



b. Examples of Shift Instructions

(1) Solve R = (X·Y)/Z and start program in location 15555.

Location	Data	B	Range
17000	X	10	100<X<1000
17001	Y	12	1<Y<4000
17002	Z	10	100<Z<500
17003	R	16	0.2<R<40,000

Location of Instruction	Operation Code	Operand Address	Actual Form of Instruction in Octal	Remarks
15555	LDA	17000	0017000	Load X (B10) into A
15556	MAQ		2504006	Move it to Q (B10)
15557	MPY	17001	1517001	(X•Y) in A and Q (B22)
15560	SRD	$(4)_{10}$	2400104	(X•Y) in A and Q (B26)
15561	DVD	17002	1617002	(X•Y)/Z in A and Q (B16)
15562	STA	17003	0817003	Store R in core (B16)
15563	next instruction			

(2) Count the number of leading zeros in a number stored in location 02200 and store the count in location 02201. Assume a constant $(19)_{10}$ is stored in location 02300 at (B19).

Location of Instruction	Operation Code	Operand Address	Actual Form of Instruction in Octal	Remarks
00700	LDA	02200	0002200	Load number into A
00701	NOR	$K=(19)_{10}$	2410063	Normalize number
00702	LDA	02300	0002300	Load constant $(19)_{10}$ B19
00703	SUB	00000	0200000	Subtract C(00000)
00704	STA	02201	0302201	Store leading zero count.
00705	next instruction			

(3) Extract only bits 1 through 4 of the information stored in location 13130, and store at B19 in location 13131.

Location of Instruction	Operation Code	Operand Address	Actual Form of Instruction in Octal	Remarks
07777	LDA	13130	0013130	Load number into A
10000	LLC	$(5)_{10}$	2411105	Move bits 1-4 to 16-19
10001	LLA	$(16)_{10}$	2410110	Shift off top bits
10002	LLC	$(4)_{10}$	2411104	Shift important bits to 16-19
10003	STA	13131	0313131	Store extracted bits
10004	next instruction			

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. Conditions tested are the sign of the A register, zero in the A register, arithmetic overflow and others.

a. Unconditional Branch Instructions

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

BRU	Y	1	26

BRANCH UNCONDITIONALLY. Control is transferred to the instruction located at Y. Y becomes the address of the next instruction and is transferred from I_{7-19} to P_{7-19}. In model 412B, when $C(I)_{7-19}$ are transferred to P_{7-19}, P_6 is not disturbed. The BRU instruction will not be interrupted by automatic program interrupt.

JMP	Y	1	370

JUMP UNCONDITIONALLY. (Model 412B only.) Control is transferred to the instruction located at Y. Y (14-bit address) becomes the address of the next instruction and is transferred from I_{6-19} to P_{6-19}. This instruction may not be automatically modified because bit 6 is part of the address, Y. If bit 5 is set to specify modification by X-location 2, this instruction will then become the SPJ instruction.

b. Conditional Branch Instructions

A conditional branch instruction will transfer control to one of two instructions located relative to the location of the branch instruction itself. Control is transferred to either the first or the second sequential instruction after the conditional branch instruction. Each conditional branch instruction must specify a constant

J, which can equal 1 or 2, that is used to set bit 9 of the instruction to a 0 or 1 respectively. The presence or absence of the condition and bit 9 of the instruction determine the next instruction to be executed. With the branch instruction located at location L, the following table indicates the location of the instruction to which control is transferred.

J	Condition	Location of Next Instruction
1	Present	L + 1 Normal Branch
1	Absent	L + 2 Alternate Branch
2	Present	L + 2 Normal Branch
2	Absent	L + 1 Alternate Branch

BPL	J	2	J=1	2514001
			J=2	2516001

BRANCH ON PLUS. Branch to location L + J if the sign of A is plus. (Zero is considered to be plus.) Take the alternate branch if the sign of A is minus. A is unchanged by this instruction.

BZE	J	2	J=1	2514002
			J=2	2516002

BRANCH ON ZERO. Branch to location L + J if C(A) are zero. Take the alternate branch if C(A) are not zero. A is unchanged by this instruction.

BEV	J	2	J=1	2514000
			J=2	2516000

BRANCH ON EVEN. Branch to location L + J if A is even. A is even if $C(A)_{19}$ are zero. Take the alternate branch if A is odd. A is unchanged by this instruction.

BOV	J	2	J=1	2514003
			J=2	2516003

BRANCH ON OVERFLOW. Branch to location L + J if the overflow indicator is on. Take the alternate branch if the overflow indicator is off. This instruction turns the overflow indicator off.

BPC	J	2	J=1	2514004
			J=2	2516004

BRANCH ON PARITY, CORE. Branch to location L + J if the core parity error circuit and indicator are on. The core parity error circuit will then be reset but the indicator will remain on until reset manually from the console. Take the alternate branch if core parity error circuit is off. Depressing the console parity indicator to reset the indicator circuit resets the parity error circuit which is tested by the branch command.

BPD	J	2	J=1	2514005
			J=2	2516005

BRANCH ON PARITY, DRUM. Branch to location L + J if the drum parity error circuit and indicator are on. The drum parity error circuit will then be reset but the indicator will remain on until reset manually from the console. Take the alternate branch if drum parity error circuit is off. Depressing the console parity indicator to reset the indicator circuit resets the parity error circuit which is tested by the branch command.

---

c. Examples of Branch Instructions

(1) Find the larger of two numbers stored in locations 02000 and 02001 and store it in location 03000. The numbers have equal scale factors.

Location of Instruction	Operation Code	Operand Address	Actual Form of Instruction in Octal	Remarks
04001	LDA	02000	0002000	Find difference between
04002	SUB	02001	0202001	1st and 2nd
04003	BPL	1	2514001	Test sign of difference
04004	BRU	04007	2604007	If plus go to 04007
04005	LDA	02001	0002001	If minus load 2nd
04006	BRU	04010	2604010	Go to 04010 unconditionally
04007	LDA	02000	0002000	Load 1st
04010	STA	03000	0303000	Store larger of two
04011	next instruction			

(2) Given two numbers X and Y, compare them and if X=Y set Z=0, if X>Y set Z = X, if X<Y set Z=Y. X, Y and Z are stored in octal locations 04001, 04002, and 04003 respectively and have equal scale factors.

Location of Instruction	Operation Code	Operand Address	Actual Form of Instruction in Octal	Remarks
05077	LDA	04001	0004001	Load X into A
05100	SUB	04002	0204002	Subtract Y from X
05101	BZE	1	2514002	Test difference
05102	BRU	05110	2605110	If zero go to 05110
05103	BPL	2	2516001	
05104	BRU	05107	2605107	If minus go to 05107
05105	LDA	04001	0004001	If plus X>Y so load X
05106	BRU	05110	2605110	
05107	LDA	04002	0004002	Load Y since X<Y
05110	STA	04003	0304003	Store Z= to X or Y or 0
05111	next instruction			

6. Automatic Address Modification 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 three examples in section 6b depict the methods to accomplish these functions.

The automatic address modification instructions operate in conjunction with the first four core storage locations, locations 0000, 0001, 0002, and 0003. These are called X locations 0, 1, 2, and 3. In all instructions, except SPB, INX, LDX, STX, BXL, and BXH, bits 5 and 6 indicate whether or not automatic address modification is to take place. If bits 5 and 6 are zero, no address modification will take place. If bits 5 and 6 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 (as specified by bits 5 and 6) to the instruction in the I register. In model 412A, bits 7 through 19 of the specified X location are added to bits 7 through 19 of the I register, with any resulting carries out of bit position 7 being lost (ignored). In model 412B, bits 6 through 19 of the specified X location are added to bits 7 through 19 of the I register. In this addition in model 412B, bit 6 of the I register is considered to be a zero, and any resulting carry out of bit position 6 is lost (ignored). Note that X location 0 cannot be used for automatic address modification. Also, when automatic address modification is called for, one extra word time is added to the normal instruction execution time.

All four X locations (0, 1, 2, and 3) may be used in conjunction with SPB, INX, LDX, STX, BXH, and BXL instructions. All X locations may therefore be incremented and tested to accomplish counting or tallying. In these six instructions bits 5 and 6 are used to specify which of the four X locations is to be used in the execution of the instruction.

a. Instruction Definitions

LDX	Y, X	3	06

LOAD X LOCATION FROM Y. The $C(Y)_{0-19}$ replace the $C(X)_{0-19}$. C(Y) are unchanged. It should be noted that since only 13 bits are available for the specification of Y and bits 5 and 6 are needed to specify the X location to be used, Y is limited to the first 8,192 words of memory in all systems.

STX	Y, X	3	17

STORE X LOCATION INTO Y. The $C(X)_{0-19}$ replace the $C(Y)_{0-19}$. C(X) are unchanged. It should be noted that since only 13 bits are available for the specification of Y and bits 5 and 6 are needed to specify the X location used, Y is limited to the first 8,192 words of memory in all systems.

INX	K, X	3	14

INCREMENT X BY K. K, $C(I)_{7-19}$, are added absolutely to C(X), and the result replaces C(X). In model 412A, this addition involves only bits 7-19 of X, and any carry out of bit position 7 is lost. In model 412B, the addition involves bits 6-19 of X, and any carry out of bit position 6 is lost. Also, in model 412B, if K is greater than 4095 the number actually added to $C(X)_{6-19}$ will be 8192 greater than the specified K.

BXH	-K, X	3	05

BRANCH IF X IS HIGH OR EQUAL. If $C(X)_{7-19}$ are larger than or equal to K, the computer takes the next sequential instruction; if $C(X)_{7-19}$ are less than K, the computer skips the next instruction and executes the second sequential instruction. X is not changed. This instruction cannot be automatically modified since bits 5 and 6 are used to identify the particular X location (Note. K is required to be the 2's complement of the desired test value.)

BXL	-K, X	3	04

BRANCH IF X IS LOW. If $C(X)_{7-19}$ are less than K, the computer takes the next sequential instruction; if $C(X)_{7-19}$ are larger than or equal to K, the computer skips the next instruction and executes the second sequential instruction. X is unchanged. This instruction is not automatically modified since bits 5 and 6 are used to identify the particular X location. (Note. K is required to be the 2's complement of the desired test value.)

SPB	Y, X	2	07

STORE P AND BRANCH. In model 412A the address of this instruction replaces $C(X)_{7-19}$, and control is transferred to the instruction located at Y; that is, $C(I)_{7-19}$ replace $C(P)_{7-19}$. The $C(X)_{0-6}$ are set to zeros. In model 412B the address of this instruction replaces $C(X)_{6-19}$ and control is transferred to the instruction at Y; that is, $C(I)_{7-19}$ replace $C(P)_{7-19}$. Bit 6 of the P register is undisturbed. The $C(X)_{0-5}$ are set to zeros. The SPB instruction will not be interrupted by automatic program interrupt. Normally X locations 1, 2, or 3 is specified in using the SPB instruction, but X location 0 may be specified.

SPJ	Y	2	374

STORE P AND JUMP. (Model 412B only.) The 14-bit address of the SPJ instruction replaces the $C(X2)_{6-19}$, and control is transferred to the instruction located at Y. Y (14-bit address) becomes the address of the next instruction and is transferred from I_{6-19} to P_{6-19}. $C(X2)_{0-5}$ are set to zeroes. This instruction cannot be automatically modified and will not be interrupted by automatic program interrupt.

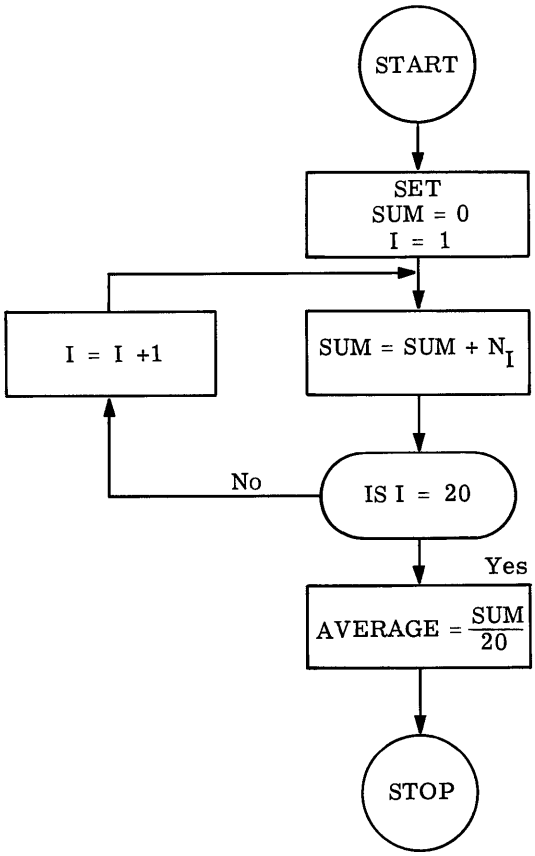
b. Examples of Address Modification

(1) A sequence of 20 numbers called N_i are stored in locations 02001 through 02024 with equal scale factors of B19. All of the numbers are less than $(100)_{10}$ and greater than zero. Average these numbers and store the average in location 02025 with a B19. Assume a constant $(20)_{10}$ at B19 is stored in location 02000.

(a) Solution without address modification.

Location of Instruction	Operation Code	Operand Address	Actual Form of Instruction in Octal	Remarks
01001	LDA	02001	0002001	Load N_1
01002	ADD	02002	0102002	add N_2
01003	ADD	02003	0102003	add N_3
01004	ADD	02004	0102004	add N_4
01005	ADD	02005	0102005	add N_5
01006	ADD	02006	0102006	add N_6
01007	ADD	02007	0102007	add N_7
01010	ADD	02010	0102010	add N_8
01011	ADD	02011	0102011	add N_9
01012	ADD	02012	0102012	add N_{10}
01013	ADD	02013	0102013	add N_{11}
01014	ADD	02014	0102014	add N_{12}
01015	ADD	02015	0102015	add N_{13}
01016	ADD	02016	0102016	add N_{14}
01017	ADD	02017	0102017	add N_{15}
01020	ADD	02020	0102020	add N_{16}
01021	ADD	02021	0102021	add N_{17}
01022	ADD	02022	0102022	add N_{18}
01023	ADD	02023	0102023	add N_{19}
01024	ADD	02024	0102024	add N_{20}
01025	MAQ		2504006	Move sum to B38 in A and Q
01026	DVD	02000	1602000	Divide by 20
01027	STA	02025	0302025	Store average at B19
01030	next instruction			

(b) The solution using address modification takes advantage of the fact that instructions are stored in storage just as data, that is, binary ones and zeros. One ADD instruction can therefore be used to add all 20 numbers, if after each time the ADD instruction is used, one is added to its address before it is used again.



Flow chart applies to examples b and c.

Location of Instruction	Operation Code	Operand Address	Actual Form of Instruction in Octal	Remarks
01001	LDZ		2504002	
01002	STA	02025	0302025	Set sum = 0
01003	LDA	01052	0001052	Reset modified address
01004	STA	01006	0301006	Set I = 1
01005	LDA	02025	0002025	
01006	(ADD	02001)	(0102001)	This address changes
01007	STA	02025	0302025	Sum = Sum + NI
01010	LDA	01006	0001006	A contains 0102001
01011	SUB	01050	0201050	Subtract 0102024
01012	BZE	1	2514002	IS I = 20
01013	BRU	01017	2601017	
01014	ADD	01051	0101051	I = I + 1 add 0102025
01015	STO	01006	2701006	Store modified address back
01016	BRU	01005	2601005	Go back to sum next N_i
01017	LDA	02025	0002025	Sum in A and Q B38
01020	MAQ		2504006	
01021	DVD	02000	1602000	Divide by $(20)_{10}$
01022	STA	02025	0302025	Store average, B19
01023	next instruction			
01050	ADD	02024	0102024	
01051	ADD	02025	0102025	Constants
01052	ADD	02001	0102001	

(c) Solution with automatic address modification uses the computer automatic ability to add the contents of an X location (00001, 00002, 00003 in core storage) to the address in the I register before it is executed.

Location of Instruction	Operation Code	Operand Address	X	Actual Form of Instruction in Octal	Remarks
01001	LDZ			2504002	set Sum = 0 in A
01002	LDX	01020	1	0621020	set I = 1
01003	ADD	02000	1	0122000	*
01004	BXH	(-20)	1	0537754	Is I = 20
01005	BRU	01010		2601010	
01006	INX	(1)	1	1420001	I = I + 1
01007	BRU	01003		2601003	repeat sum
01010	MAQ			2504006	Sum in A and Q (B38)
01011	DVD	02000		1602000	Divide by $(20)_{10}$
01012	STA	02025		0302025	Average (B19)
01013	next instruction				
01020	constant $(1)_{10}$			0000001	Constant

*The address of this ADD instruction is a base address (relative) that the contents of X location 0001 is added to. The first time it is executed it will be address 02001.

7. Real-Time Instructions

The GE 412 system has a solid state digital clock and four elapsed time counters that enable a program to be aware of real-time and to measure real-time intervals with precision. These time measuring components use a 60 cycle power source for a timing base and are as accurate as that source.

a. Real-Time Digital Clock

The real-time solid state digital clock provides a very accurate time base for the computer. It consists of a group of cascaded counters whose contents may be read directly into the A register of the computer. The binary-coded-decimal format of these counters as they are read into the A register is as follows:

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
Hours Tens		Hours Units				Minutes Tens			Minutes Units				Seconds Tens			Seconds Units			

The contents of the real-time digital clock counters may be read into the A register at anytime under program control.

RCL 2, 3 2510051

READ DIGITAL CLOCK. The 20 bits defining the time indicated by the read-time digital clock replace $C(A)_{0-19}$. This instruction should be preceded by a successful BCL instruction.

BCL	J	2	J=1 J=2	2514012 2516012

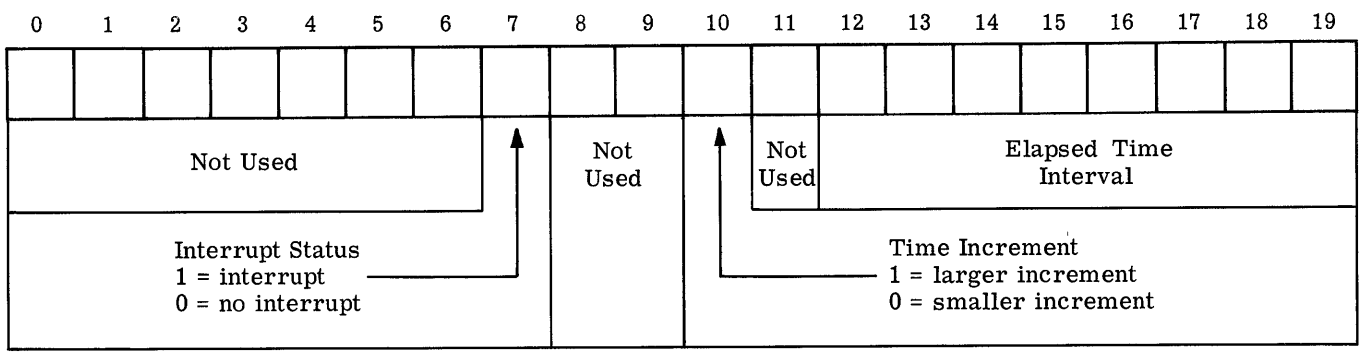
BRANCH ON CLOCK VALID. Branch to location L + J if a valid read-in can be obtained from the digital clock. Take the alternate branch if not. If the clock is being manually reset or if there has been a power failure a valid read may not be obtained from the clock. After a power failure the digital clock must be manually reset.

b. Elapsed Time Counters

Four elapsed time counters are provided in the GE 412 System to time elapsed time intervals. The counters are each 8 bits in length and are capable of counting up to 256 increments of time. There are four increments of time that can be used to count with the elapsed time counters. They are 3.2 milliseconds, 16.7 milliseconds, 1 second and 1 minute. Each of the elapsed time counters may be provided with increment pulses from two of the four incremental sources. The arrangement is as follows:

Counter	Incremental Source
1	3.2 milliseconds or 1 second
2	3.2 milliseconds or 1 minute
3	16.7 milliseconds or 1 second
4	16.7 milliseconds or 1 minute

The 3.2 millisecond incremental source is based on timing from a very accurate (0.1%) crystal oscillator in the computer. The other incremental sources are based on the 60 cycle power supply used by the real-time digital clock. The program initiates an elapsed time counter by transferring a control word to the selected counter. The format of the control word is as follows:



The control word specifies the desired elapsed time interval, the incremental time source, and the priority interrupt status. The desired elapsed time interval is specified in a true binary form, and the computer transforms this interval into its one's complement form as it is loaded into the elapsed time counter. The specified incremental time pulses are counted until the counter overflows. The elapsed time counters have a maximum probable positive error equal to the incremental source specified because the incremental time sources for the counters are asynchronous with computer timing. The elapsed time counter overflow condition may be fed directly into the interrupt register or may be branched on by the program. Branching on a specified counter overflow condition resets the overflow indicator. If the overflow condition has just caused automatic program interrupt, the overflow indicator must be reset by executing a BTC command or by initiating a new time interval count to reset the input to the interrupt register from that overflow indicator. Examples of using elapsed time counters will be given later in conjuction with illustrations of use of peripherals.

LTC	K	2	K=1	2500017
			K=2	2500020
			K=3	2500021
			K=4	2500022

LOAD TIME COUNTER K. $C(A)_{7,\ 10,\ 12\text{-}19}$ are loaded into elapsed time counter K, and elapsed time count is initiated.

				J=1	J=2
BTC	J, K	2	K=1	2514013	2516013
			K=2	2514014	2516014
			K=3	2514015	2516015
			K=4	2514016	2516016

BRANCH ON TIME COUNTER K OVERFLOW. Branch to location L + J if elapsed time counter K has completed its count (overflowed). Take the alternate branch if the counter has not. The overflow indicator for timer K is reset.

8. Magnetic Drum Information Transfer

Transfer of information between the high speed storage unit and the magnetic drum storage unit is initiated under direct program control, but once initiated proceeds independently of further program action. The magnetic drum is divided into tracks, and all transfers are effected in blocks of from 1 to 8 complete tracks of information. In model 412A systems each track contains 128 words and a transfer requires 16-2/3 milliseconds per track. In model 412B systems each track contains 256 words, and because the word-transfer rate is the same in both types of systems, each track transferred requires 33-1/3 milliseconds.

To effect a core/drum transfer, the program must transfer a drum command word from the A register into the drum command register. The drum command word specifies the direction of transfer, the number of consecutive tracks to be transferred, the beginning drum track address, and the beginning core address. The beginning core address is restricted to those core locations which are integer multiples of $(200)_8$. In model 412B systems the further restriction applies that only even integer multiples of $(200)_8$ may be used. Core location 00000 is a permissible beginning core address in either type of system. The format of the drum command word is given below.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
*	Number of Tracks −1			Beginning Drum Track Address									Beginning Core Address / $(200)_8$						See Text

The 9 bits (bits 4-12) designated for Beginning Drum Track Address are insufficient for addressing more than 512 tracks. This presents no problem in model 412A systems since the maximum number of tracks available is 448. Model 412B systems, however, may have up to 672 drum tracks. In model 412B systems having more than 512 drum tracks, bit 19 is used to effect the addressing of tracks 512 through 671. Bit 19 is available for such usage because only even multiples of $(200)_8$ are permissible as beginning core addresses.

If, during a drum/core transfer in either direction, a second transfer is initiated, the transfer in progress will be safely (without causing parity errors) aborted and the new transfer will begin immediately.

LDC	2		2500026

LOAD DRUM COMMAND REGISTER FROM A. The drum command register is loaded with a drum command word from C(A). The format of this word is shown above. Normal execution time is 2 word times, but if the instruction is given while a core-to-drum transfer is in progress, making use of the abort feature, the word at that time being written on drum will be completed, resulting in a total execution time of up to 9 word times.

BDC	J	2	J=1	2514020
			J=2	2516020

BRANCH ON DRUM OPERATION COMPLETE. Branch to location L + J if the previous drum transfer operation is complete. Take the alternate branch if the operation is not complete.

9. Automatic Program Interrupt Instructions.

Automatic program interrupt is one of the most powerful characteristics of the GE 412 System. It allows for pulses from external sources, (those from process sensing devices) and internal sources (such as completion signals from peripheral equipment or timer overflow impulses) to cause automatic interruption of the program currently operating in the computer. The actions demanded by the condition that caused interruption are then taken care of as quickly as possible and control is then returned to the program that was interrupted. The Program Interrupt Register is 12 bits in length and is divided into three groups of 4 levels (bits) each. A priority is assigned to each of the levels so that actions are accomplished in order of importance. The program can select group I (levels 1 through 4), groups I and II (levels 1 through 8), or groups I, II, and III (all levels) to cause program interruption. It may also inhibit all levels of interruption from causing program interruption. Interrupts (pulses) that occur in an inhibited state of operation are not lost, but are held in the interrupt register until such time as that group is enabled by the program and interruption will then occur.

When an interrupt occurs in an enabled group and interrupts are permitted, the following take place automatically in the order shown.

1. The instruction being executed is completed.

2. All further interrupts are inhibited until permitted by the program.

3. The contents of the P register (address of the next instruction normally executed) are stored in core storage location 00006. $C(00006)_{0-6}$ are unchanged in model 412A systems. In model 412B systems, $C(00006)_{0-5}$ are set to zeroes.

4. The contents of the A register are stored in location 00007.

5. The P register is set to location 00007 and the enabled portions of the interrupt register are examined bit by bit starting with the highest priority bit (level 1) and proceeding sequentially in descending order until the bit which caused the interrupt is reached. Before each bit is examined the P register is incremented by one. When the first interrupting bit is found, it alone is reset, and control is transferred to the instruction located at the address then in the P register. The computer then executes the program that begins at one of the 12 sequential locations starting in octal location 00010. For example, if level 2 caused interruption control is transferred to the instruction that is in location 00011, which is normally a BRU instruction that directs control to the program associated with level 2 interrupt.

Before permitting further interrupts, the program must transfer the address that indicates the restart point in the interrupted program (which is now in location 00006) to another location in core. It must by the same token save the contents of the A register and any other register that it will use and re-establish all conditions that prevailed at time of interruption. This is to ensure a proper re-entry into the interrupted program from the program that interrupted.

SAI	K	2	K=1	2500014
			K=2	2500015
			K=3	2500016

SELECT AUTOMATIC INTERRUPT GROUP K. Group K interrupt levels are selected to enable automatic program interrupts. Interrupts cannot actually occur, however, until permitted by the execution of a PAI instruction as described below.

PAI	2	2500012

PERMIT AUTOMATIC INTERRUPT. The previously selected priority levels are permitted to cause program interruption after the execution of the next instruction after the PAI.

IAI	2	2500013

INHIBIT AUTOMATIC INTERRUPT. All interrupts are inhibited or ignored. Interrupt conditions are not lost, but are stored in the interrupt register until such time as a PAI instruction is executed.

10. Other Internal Instructions

SSA	2	2500025

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 5 to 20 seconds.

RCS	2	2500024

READ CONSOLE SWITCHES. The A register is cleared and the bit pattern represented by the 20 A register toggle switches on the console is loaded into the A register. These switches are operated manually by the operator. A switch represents a 1 bit when it is depressed and a 0 bit when it is in its normal (raised) position.

XEC	Y	1+	34

EXECUTE THE INSTRUCTION AT Y. The instruction in memory location Y will be performed or "executed" as if it actually existed at the location of the XEC instruction, with a single exception. If the instruction at location Y is an SPB or an SPJ, the P register value stored will be that of the XEC instruction plus one. The XEC instruction may be indexed (modified by automatic program modification) and the C(Y) may be any valid instruction. The execution time will be one word time plus the normal execution time of the instruction at Y. Automatic program interrupt will not take place between the XEC instruction and the instruction at Y.

BRD	J	2	J=1	2514017
			J=2	2516017

BRANCH ON DEMAND. Branch to location L + J if the Demand pushbutton on the programming and maintenance console has been depressed. Take the alternate branch if the demand button has not been depressed. This instruction turns off the light behind the demand button.

B. EXTERNAL EFFECT INSTRUCTIONS

External effect instructions are those that control the input-output equipment of the GE 412 Process Computer System. The operation of slow speed electro-mechanical devices, such as the paper tape, reader, paper

tape punch, typewriters, and card reader, requires the use of an information buffer. This permits the central processor to operate at normal fast internal speed and yet communicate with these relatively slow peripheral devices. The Conversion, H, M, N, and other special registers are used as buffers for information flowing in and out of the central processor.

It should be noted that the coding examples used to amplify the descriptions of commands relating to peripheral equipment are intended to illustrate the functioning of the commands but may not reflect typical real-time usage because the automatic program interrupt is normally used in the implementation of peripheral equipment in real-time systems.

1. Peripheral Input-Output Instructions

Peripheral input-output instructions control the paper tape reader, paper tape punch, punched card reader, electric typewriter, serial printer, and parallel printer. All information associated with these devices flows through the M, N, or H registers. The electric typewriter may be connected to a paper tape punch in an off-line fashion (not connected to the central processor) for manual preparation of paper tape.

a. Peripherals associated with the M and N Registers.

Information flowing into the computer through the paper tape reader, card reader, or out of the computer through the paper tape punch, serial printer, and electric typewriter passes through the M or N registers. This information is in 7-bit binary-coded-alphanumeric form and the M and N registers are 7 bits in length. The binary-coded-alphanumeric codes are listed in appendix A. A parity bit is generated as characters are punched on paper tape, and checked as characters are read by the paper tape reader. This parity bit is punched in channel 5 on the tape and does not enter the M or N registers. The following diagram illustrates how the standard GE 412 paper tape code is used in conjunction with the 7-bit M and N registers.

Tape Channel	8	7	6	4	3	2	1
M or N register	1	2	3	4	5	6	7

b. Instruction Definitions

SEL	S	2	S=M	2500001
			S=N	2500000

SELECT PERIPHERAL DEVICE ON BUFFER S. The peripheral device specified by the code in buffer S is selected, that is its power is turned on. The selection requires 80 milliseconds and when it has been completed, as indicated by a successful BBR, an additional delay must be programmed for all devices except the serial printer and the IBM Selectric typer to allow the selected device to reach operating speed. The required delays are indicated below. No more than one reader, one typer, and one punch on a given register should have its power on at any time.

Device	Delay
Standard IBM typer	100 milliseconds
Flexowriter	200 milliseconds
Low-speed reader or punch	200 milliseconds
High-speed reader or punch	2 seconds
Card reader	200 milliseconds
Parallel printer	150 milliseconds

OFF	S	2	S=M	2500011
			S=N	2500010

TURN PERIPHERALS ON BUFFER S OFF. All selected devices connected to buffer S are turned off. Completion of the operation is indicated by a successful BBR. The operation requires 80 milliseconds.

				J=1	J=2
BBR	J, S	2	S=M	2514011	2516011
			S=N	2514010	2416010

BRANCH ON BUFFER S READY. Branch to location L + J if buffer S is ready (if its last operation is complete). Take the alternate branch if buffer S is not ready.

				J=1	J=2
BBP	J, S	2	S=M	2514007	2516007
			S=N	2514006	2516006

BRANCH ON BUFFER S PARITY ERROR. Branch to location L + J if the parity error condition is set on buffer S. Take the alternate branch if the condition is not set. This instruction resets the parity error condition but does not reset the visual indicator on the programming and maintenance console.

TYP	S	2	S=M	2500005
			S=N	2500004

TYPE ON BUFFER S. The 7-bit binary-coded-alphanumeric code in buffer S is typed or typed and punched if the punch is slaved to the typewriter.

PCH	S	2	S=M	2500007
			S=N	2500006

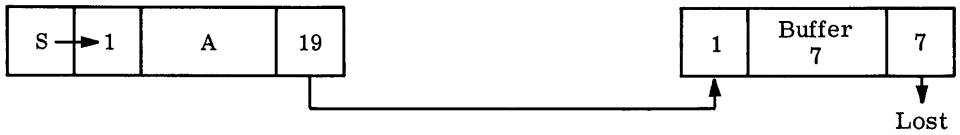
PUNCH ON BUFFER S. The 7-bit code in buffer S is punched. This instruction is not used when the punch is slaved to a typewriter.

RDD	S	2	S=M	2500003
			S=N	2500002

READ INTO BUFFER S. Buffer S is cleared and one 7-bit code is read into buffer S from the paper tape reader or card reader whichever is currently selected.

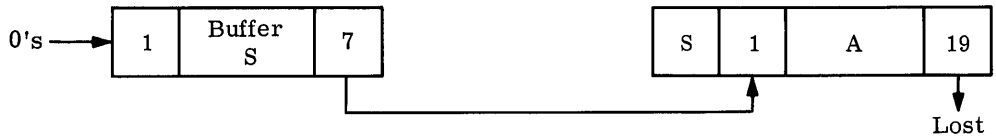
SAB	S, K	2+	S=M	2400400
			S=N	2500200

SHIFT A TO BUFFER S. $C(A)_{1-19}$ and $C(\text{Buffer S})_{1-7}$ are shifted K places to the right together. Bits shifted out of A_{19} shift into buffer S_1. Bits shifted out of buffer S_7 are lost. If the sign of A is plus, zeros fill the vacated positions of A; if the sign of A is minus, ones fill the vacated positions of A. The sign of A is unchanged.



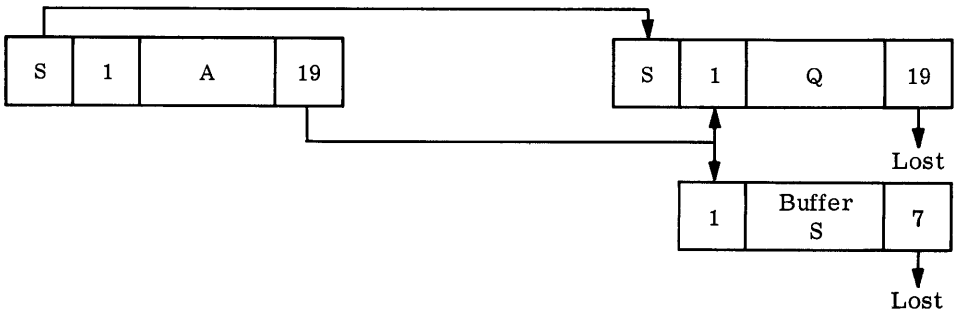
SBA	S, K	2+	S=M	2404000
			S=N	2402000

SHIFT BUFFER S INTO A. $C(\text{Buffer S})_{1-7}$ and $C(A)_{1-19}$ together are shifted K places to the right. Bits shifted out of A_{19} are lost. Bits shifted out of buffer S_7 shift into A_1. The sign of A is unchanged.



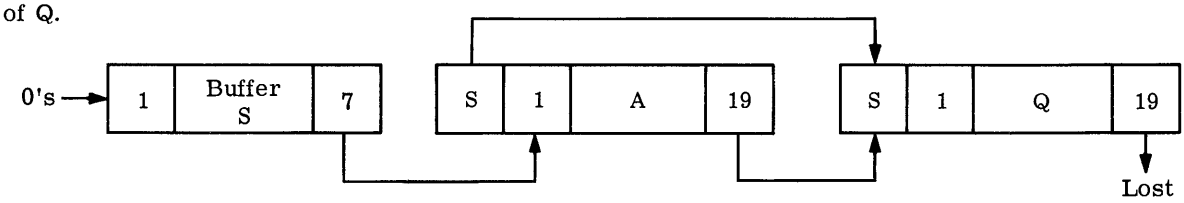
ABQ	S, K	2+	S=M	2400300
			S=N	2400500

SHIFT A INTO BUFFER S AND Q. $C(A)_{1-19}$ are shifted K places to the right into both buffer S and the Q register. Bits shifted out of A_{19} enter both Q_1 and buffer S_1. Bits shifted out of Q_{19} and buffer S_7 are lost. If the sign of A is plus, zeros fill the vacated position of A; if the sign of A is minus, ones fill the vacated position of A. The sign of A replaces the sign of Q. The sign of A is unchanged.



SBD	S, K	2+	S=M	2404100
			S=N	2402100

SHIFT BUFFER S INTO A AND Q DOUBLE. $C(\text{Buffer S})_{1-7}$, $C(A)_{1-19}$, and $C(Q)_{1-19}$ together are shifted K places to the right. Bits shifted out of buffer S_7 shift into A_1, bits out of A_{19} shift into Q_1. Bits shifted out of Q_{19} are lost. Vacated position of buffer S are filled with zeros. The sign of A is unchanged and replaces the sign of Q.



c. Examples of Peripheral Input-Output Instructions

(1) Read in six binary-coded decimal (BCD) characters with the paper tape reader connected to buffer M and selected by code $(050)_8$. The number read in is an integer to be stored in location 04000 at B19 and is an integer number less than $(524,287)_{10}$.

Location of Instruction	Operation Code	Operand Address	X	Actual Form of Instruction in Octal	Remarks
01000	OFF	M		2500011	Turn all devices off
01001	BBR	2, M		2516011	Wait of off to be
01002	BRU	01001		2601001	complete
01003	LDA	02000		0002000	Load A with select code
01004	SAB	M, 7		2400207	Put select code in M
01005	SEL	M		2500001	Select reader
01006	BBR	2, M		2516011	Wait for selection
01007	BRU	01006		2601006	complete

Location of Instruction	Operation Code	Operand Address	X	Actual Form of Instruction in Octal	Remarks
01010	LDA	02001		0002001	Load elapsed time constant
01011	LTC	3		2500021	Start time count 3 and
01012	BTC	2, 3		2516015	wait for 200
01013	BRU	01012		2601012	millisecond delay
01014	LDZ			2504002	
01015	STA	00001		0300001	Set loop counter = 0
01016	MAQ			2504006	Clear Q
01017	RDD	M		2500003	Initiate read.
01020	BBR	2, M		2516011	Wait for character.
01021	BRU	01020		2601020	
01022	SBA	M, 19		2402023	Shift BCD to B19
01023	MPY	02002		1502002	Mult. partial no. by
01024	INX	1	1	1420001	10 and add BCD.
01025	BXL	6	1	0637772	Count no. read and
01026	BRU	01017		2601017	return for next.
01027	XAQ			2504005	When complete store
01030	STA	04000		0304000	result.
01031	next instruction				
02000	(00000000000000101000)			0000050	Select code for reader
02001	(00000000000000001010)			0000012	elapsed time counter constant
02002	(00000000000000001010)			0000012	constant ($10B_{19}$.

(2) Type the decimal equivalent of the binary number stored in location 00100 at B19. The number is less than $(524,287)_{10}$. Use the typewriter on the N register selected by code $(060)_8$. Type the (-) sign if the number minus and a space if positive in front of the number.

Location of Instruction	Operation Code	Operand Address	X	Actual Form of Instruction in Octal	Remarks
01100	ϕFF	N		2500010	Turn off all peripherals on N
01101	BBR	2, N		2516010	Wait for off to be
01102	BRU	01101		2601101	complete
01103	LDA	02100		0002100	Load selector code
01104	SAB	N, 7		2400407	Put it into N
01105	SEL	N		2500000	Select peripheral
01106	BBR	2, N		2516010	Wait for selection
01107	BRU	01106		2601106	to be complete
01110	LDA	02101		0002101	Loat timer constant
01111	LTC	3		2500021	Start delay of 200 msec.
01112	BTC	2, 3		2516015	Wait for delay to
01113	BRU	01112		2601112	be complete
01114	LDA	00100		0000100	Load number to be typed
01115	BPL	1		2514001	Test sign of number
01116	BRU	01123		2601123	
01117	NEG			2504532	Make (-) number positive
01120	MAQ			2504006	Put + number into Q
01121	LDA	02102		0002102	Load minus (-) code
01122	BRU	01124		2601124	
01123	MAQ			2504006	Put + number Q set A=O= Space
01124	LDX	02112	2	0642112	Set loop counter = 0
01125	SAB	N, 7		2400407	Put charceter into N

Location of Instruction	Operation Code	Operand Address	X	Actual Form of Instruction in Octal	Remarks
01126	TYP	N		2500004	Type a character
01127	BXH	-6	2	0537772	Test for
01130	BRU	01140		2601140	End of loop.
01131	DVD	02104	2	1622104	Divide by power of ten*.
01132	INX	1	2	1440001	Increment loop counter.
01133	BZE	1		2514002	If character is zero,
01134	LDA	02103		0002103	load zero code.
01135	BBR	2, N		2516010	Wait for last character
01136	BRU	01135		2601135	to be typed.
01137	BRU	01125		2601125	Return for next character.
01140	next instruction				
02100	00000000000000110000			0000060	Select code for typewriter
02101	00000000000000001010			0000012	Elapsed time counter constant
02102	00000000000000100000			0000040	Code for minus sign
02103	00000000000000010000			0000020	Code for zero
02104	00011000011010100000			0303240	Constants $(100000)_{10}$(B19)
02105	00000010011100010000			0023420	Constant = $(10000)_{10}$(B19)
02106	00000000001111101000			0001750	Constant = $(10000)_{10}$(B19)
02107	00000000000001100100			0000144	Constant - $(100)_{10}$B19
02110	00000000000000001010			0000012	Constant - $(10_{10}$B19
02111	00000000000000000001			0000001	Constant = $(1)_{10}$B19
02112	00000000000000000000			0000000	Constant = 0

*This divide leaves a BCD character in A and the remainder in Q ready to be divided by the next lower power of ten.

d. The Parallel Entry Printer Instructions

The parallel entry line printer is capable of printing at 5 lines per second with 11 numeric or special characters per line. The H register buffers information as it flows from the central processor to the parallel entry printer and is 44 bits in length. The H register holds the 11 binary-coded-decimal (4 bit) characters to be printed on the line printer. The H register is loaded from the A register.

Up to four parallel printers may be connected to the H register and may all be on at the same time, but printing may only be done on one printer at a time. It is possible to have two H registers in a given system and therefore, up to 8 parallel printers.

The binary-coded-decimal format for the standard print position on the printer is shown in the table below. Special symbols, of any variety, may be specified for a certain print position on the printer, but there may only be 12 different characters in each print position.

Standard Print Wheel	Register BCD Code 8	4	2	1
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1

Standard Print Wheel	Register BCD Code 8 4 2 1	
8	1 0 0 0	
9	1 0 0 1	
Blank	1 0 1 0	
-	1 0 1 1	
Blank	1 1 0 0	These codes all print as blanks for all print wheels, standard or non-standard.
Blank	1 1 0 1	
Blank	1 1 1 0	
Blank	1 1 1 1	

The format of the parallel entry printer is as follows

Print Position	11	10	9	8	7	6	5	4	3	2	1
H Register Bits	0-3	4-7	8-11	12-15	16-19	20-23	24-27	28-31	32-35	36-39	40-43
A Register Bits	8-11	12-15	16-19	4-7	8-11	12-15	16-19	4-7	8-11	12-15	16-19
Transfer Command	*Block 3			Block 2				Block 1			

*A_7 determines color (0 = black, 1 = red)

(1) Instruction Definitions

				I=1	I=2
SLH	I, K	2, 3	K=1	2510410	2510417
			K=2	2510510	2510517
			K=3	2510610	2510617
			K=4	2510710	2510717

SELECT PRINTER K AND H REGISTER I. Printer K and H register I is turned on. A delay of 200 milliseconds must be programmed after completion of the SLH instruction (indicated by a successful BRH) to allow the printer to attain operating speed.

OFH	I	2	I=1	2511010
			I=2	2511017

TURN OFF ALL PRINTERS ON H REGISTER I. The power for all printers on H register I is turned off.

				I=1	I=2
LDH	I, K	2	K=1	2511110	2511117
			K=2	2511210	2511217
			K=3	2511310	2511317

LOAD BLOCK K OF H REGISTER I. When K=1, bit positions 0-43 of H register I are cleared and then $C(A)_{4-19}$ replace $C(H)_{28-43}$. When K=2, $C(A)_{4-19}$ replace $C(H)_{12-27}$. When K=3, $C(A)_{8-19}$ replace $C(H)_{0-11}$ and bit 7 of A sets the color selection. If A_7 is 0, printing will be in black. If A_7 is 1, printing will be in red.

				I=1	I=2
PRH	I	2	K=1	2511410	2511417
			K=2	2511510	2511517
			K=3	2511610	2511617
			K=4	2511710	2511717

PRINT ON PRINTER K ON H REGISTER I. The 11 binary-coded-decimal characters in the H register I are printed on printer K.

				J=1	J=2
BRH	J, I	2	I=1	2514024	2516024
			I=2	2514033	2516033

BRANCH ON H REGISTER I READY. Branch to location L + J if the H register I is ready (last operation complete). Take the alternate branch if H register I is not ready.

(2) Example of H Register Instructions

Print one line in black on H register 1, printer 2 containing the identification, sign, and value for the integer quantity stored in location 03000 at B19. Location 03001 contains 5 binary-coded-decimal characters for printing and the format is $I_4\ I_5\ I_1\ I_2\ I_3$. The format of printing should be -

11	10	9	8	7	6	5	4	3	2	1	H Register Position
I_1	I_2	I_3	I_4	I_5	Blank	Sign	V_1	V_2	V_3	V_4	ID, sign value

The following formats are required in the A register to load the H register properly.

		I_1	I_2	I_3		I_4	I_5	Blank	Sign		V_1	V_2	V_3	V_4

Location of Instruction	Operation Code	Operand Address	X	Actual Form of Instruction in Octal	Remarks
05001	SLH	1, 2		2510510	Select printer 2
05002	BRH	2, 1		2516024	Wait for completion
05003	BRU	05002		2605002	of selection
05004	LDA	06000		0006000	Load timer 3 with 200
05005	LTC	3		2500021	Msec delay and
05006	BTC	2, 3		2516015	Wait for time delay
05007	BRU	05006		2605006	complete
05010	DLD	06001		1006001	Load A and Q with zero.
05011	DST	00000		1300000	Set X locations 0, 1, 2,
05012	DST	00002		1300002	and 3 to zero.
05013	LDA	03000		0003000	Load A with value, B19.
05014	BPL	1		2514001	If value is plus, leave
05015	BRU	05020		2605020	zero in X3.
05016	NEG			2504532	If minus, make plus and
05017	INX	1	3	1460001	set X 3 = 1.
05020	MAQ			2504006	Move value to Q, B38
05021	DVD	06002	1	1626002	Divide by power of 10.
05022	SCA	7	2	2400047	Position BCD character.
05023	ORY	00000		2300000	and store in Temp.
05024	LDZ			2504002	Clear A
05025	INX	4	2	1440004	Increment shift count.
05026	INX	1	1	1420001	Increment loop count.
05027	BXL	4	1	0437774	Test for end of loop.
05030	BRU	05021		2605021	Repeat loop.
05031	LDA	00000		0000000	BCD value to A.
05032	LDH	1, 1		2511110	BCD value to H.
05033	LDA	03001		0003001	Load ID into A.
05034	ANA	06006		2206006	Extract I_1, I_2, I_3
05035	LDH	1, 3		2511310	and load into H.
05036	LDA	03001		0003001	Load ID into A.

Location of Instruction	Operation Code	Operand Address	X	Actual Form of Instruction in Octal	Remarks
05037	EXT	06006		2006006	Extract I_4, I_5,
05040	SRA	4		2400004	position for H, and
05041	ADD	06007	3	0166007	add blank and sign.
05042	LDH	1, 2		2511210	Load into H.
05043	PRH	1, 2		2511510	Print on printer 2.
05044	next instruction				
06000	00000000000000001010			0000012	Constant for time counter
06001	00000000000000000000			0000000	Constant zero
06002	00000000001111101000			0001750	Constant $(1000)_{10}$(B19)
06003	00000000000001100100			0000144	Constant $(100)_{10}$(B19)
06004	00000000000000001010			0000012	Constant $(10)_{10}$(B19)
06005	00000000000000000001			0000001	Constant $(1)_{10}$(B19)
06006	00000000111111111111			0007777	Extract constant
06007	00000000000011111111			0000377	Two blanks
06010	00000000000011111011			0000373	One blank and minus sign

2. Scanner-Distributor Instructions

The scanner-distributor is the major communication device between the process and the computer. It mates the digital operation of the computer with the analog operations of process instruments and controllers for on-line data processing and control, as shown in figure 9. It has two modes of operation: analog-to-digital input (scanning) and subcontrol output (distributing). The scanner-distributor has two registers used in its operation: the conversion register for data buffering, and the scanner command register for scanner commands (program control of the scanner-distributor). Up to five scanner-distributors may be employed in a given GE 412 System.

The 12 bit conversion register is physically part of the scanner-distributor and functions as a data buffering register for information flowing into or out of the computer through the scanner-distributor. During scanning (input) operations, the conversion register is an integral part of the analog-to-digital converter and holds the counts (0-4095) proportional to the analog voltage scanned. During subcontrol (output) operations, the conversion register holds the set-up information for electronic or relay drivers for performing such operation as distributing an analog voltage to a process controller, driving a trend recorder, or causing lighted visual displays of information.

a. Analog-to-Digital Mode

The characteristics of an analog sensor such as selection matrix position, polarity, signal attenuation and gain settings for the amplifier, and scanning speed are specified by a scanner command word that is loaded into the scanner command register from the A register under program control. Once the scanner-distributor receives this command, it executes the operation independently of the computer, and thus frees the computer to perform other tasks. Upon the completion of any scanner-distributor operation, an operation complete signal is sent to the computer. This signal may be used to cause program interrupt or its presence may be tested for by the program through a branch command. The format of the scanner command word for analog input is shown in figure 10.

Bit positions 0 and 1 specify the operation code for the scanner-distributor. When equal to 00, they specify high speed analog-to-digital input (48.6 milliseconds per point maximum). Equal to 01, they specify low speed analog-to-digital input (125.4 milliseconds per point maximum). When equal to 10, they specify high speed analog-to-digital input with open thermocouple check. Low speed scanning is more accurate than high speed scanning in the sense that the longer amplifier settling time results in better resolution. (12 bits vs. 10 bits for high speed scanning) Bits 2 through 12 specify the position in the mercury wetted relay matrix of the contact pair to be selected. Bit 13 controls an isolation switch used to isolate high level signals from low level signals and is set to zero for low level, and one for high level signals. Bit 14 is set to one when it is desired to repeat the analog-to-digital conversion of the signal that is currently selected from the matrix.

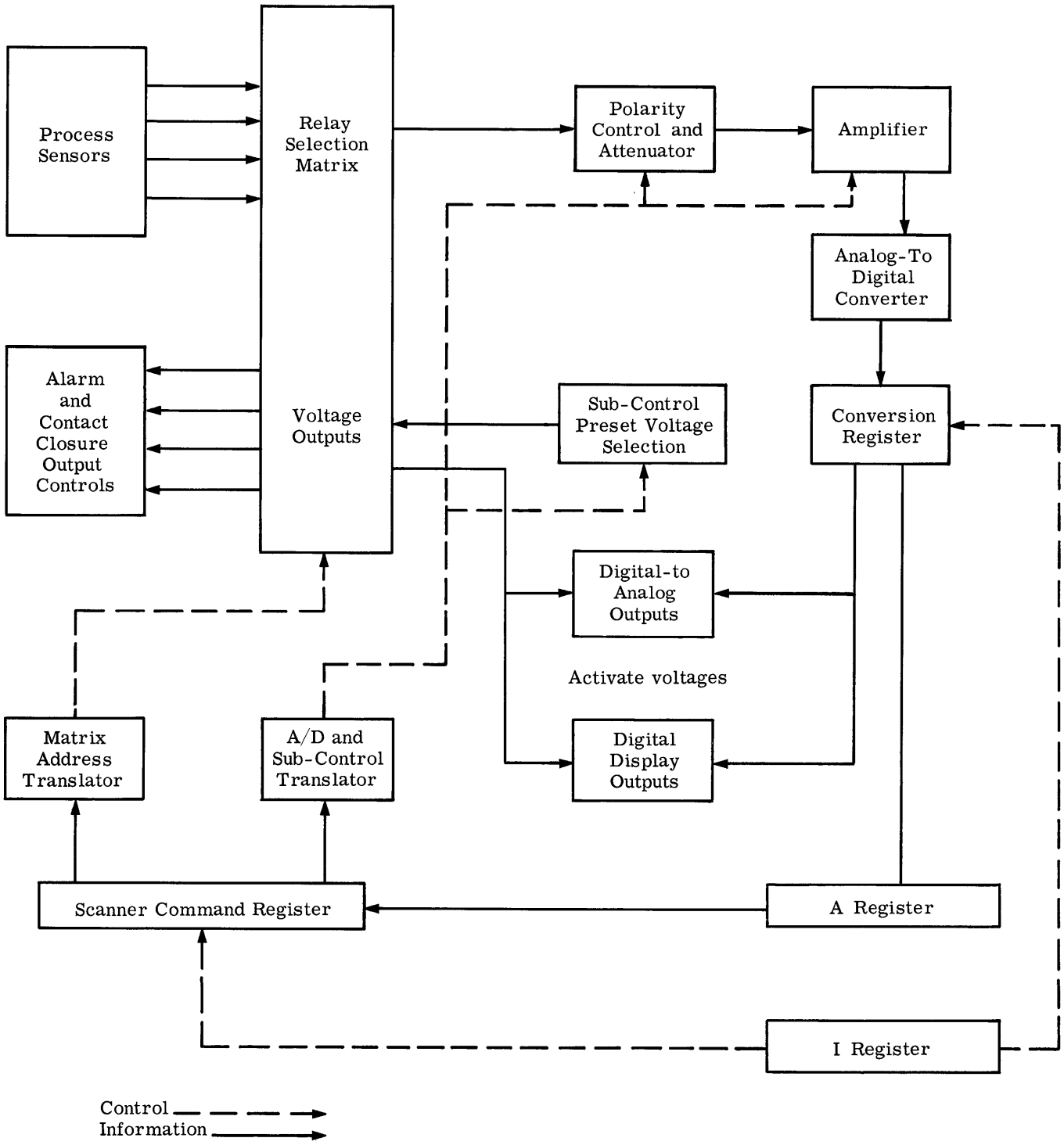


Figure 9. Single Channel Scanner-Distributor

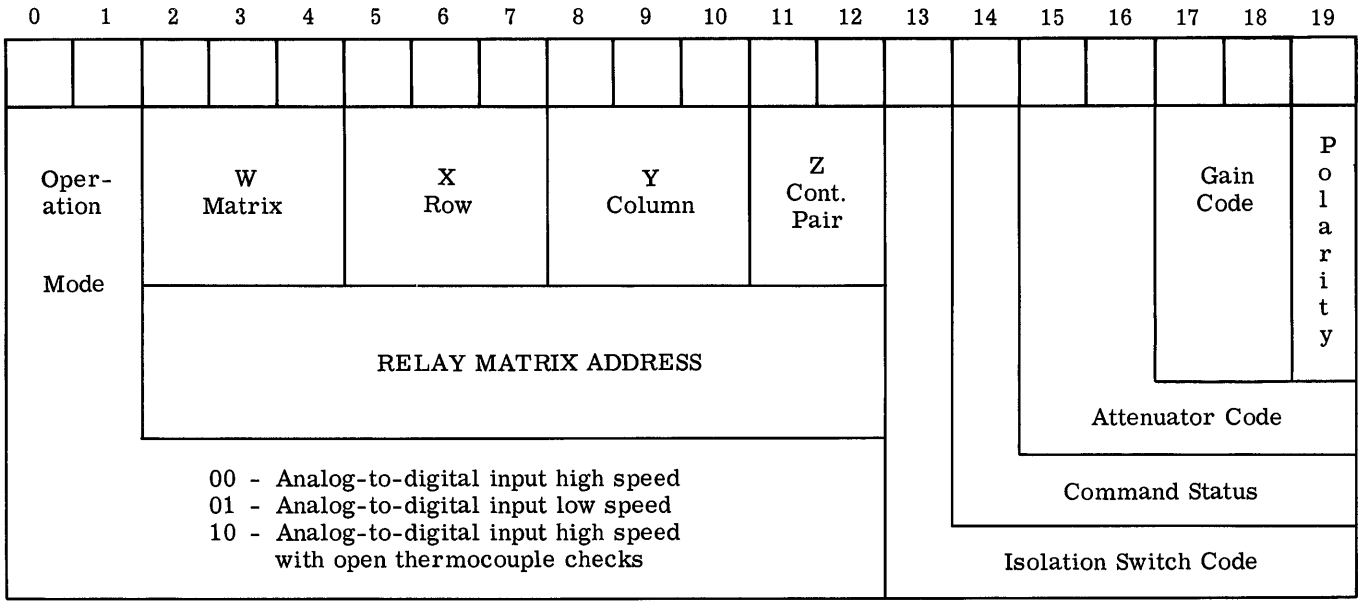


Figure 10. Scanner Command Format for Analog Input

Each repeat of the analog-to-digital conversion requires approximately 640 microseconds. Bits 15 and 16 specify the attenuator setting and bits 17 and 18 specify the gain setting for the amplifier. The resulting ranges are indicated in figure 11. Bit 19 is set to one when negative signals are to be scanned, and zero for positive signals.

Settings Attenuator	Gain	4000 Counts= Output (Volts)	Maximum Output (Volts)	Attenuator Ratio	Gain Factor
11	11	0.010	0.01023	1:1	1000
11	10	0.020	0.02046	1:1	500
11	01	0.040	0.04092	1:1	250
11	00	0.080	0.08184	1:1	125
10	11	0.125	0.127875	12.5:1	1000
10	10	0.250	0.25575	12.5:1	500
10	01	0.500	0.5115	12.5:1	250
10	00	1.00	1.023	12.5:1	125
01	11	2.00	2.046	200:1	1000
01	10	4.00	4.092	200:1	500
01	01	8.00	8.184	200:1	250
01	00	16.00	16.37	200:1	125

Figure 11. Analog Input Full Scale Ranges

b. Subcontrol Mode

The scanner-distributor functions as an output device when it distributes the proper amplitudes of voltage to process and system equipment. The scanner-distributor operating in subcontrol mode is capable of applying one of 9 preset voltages through the relay matrix to specified terminations. The duration of this voltage distribution is also controlled by the scanner-distributor. These voltages may be used to actuate such things as digital-to-analog output conversion, lighted visual displays, and similar operations. They may also be used to open or close a relay contact used to start or stop process equipment, i.e. motors, pumps, etc;

turn on or off annunciators either audible or visual; or step a stepping control device to regulate speed, flow, etc. The format of the scanner command word for subcontrol operations is in figure 12.

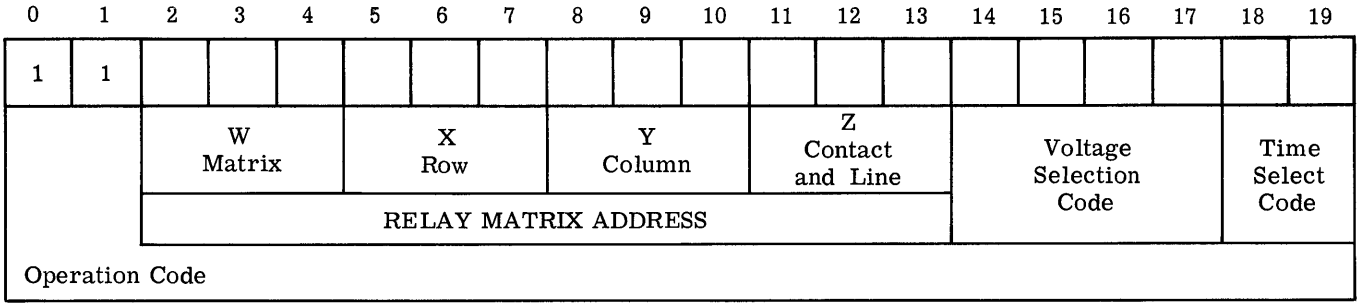


Figure 12. Scanner Command Format for Subcontrol Mode

Bit positions 0 and 1 are always "11" for subcontrol mode. Bits 2 - 13 specify the position of the single contact to be selected in the relay matrix. Bits 14 - 17 select one of nine preset voltages for distribution. Bit 14 equal to "1" is reserved for the distribution of +6 vdc to implement digital-to-analog outputs and digital display outputs. When bit 14 is a "1", bits 15 - 19 are disregarded. With bit 14 set to zero, the configuration in bits 15 - 17 is used to select one of eight other voltages whose amplitudes depend on system requirements. Bits 14 - 17 set to 0000 is reserved for specifying the use of +12 vdc logic voltage that is available in the computer system. Bits 18 - 19 specify one of four time durations for the distribution. Three of these can equal 3.2 to 102.4 milliseconds depending on system requirements. The fourth, when bits 18 - 19 are set to 11, specified that the distribution of the selected voltage is to be continuous and terminated only by initiating a new scanner-distributor operation. In this case of operation complete signal is sent to the computer at the time the voltage is distributed. The timing of subcontrol operations varies with the characteristics of the output. This timing is shown in the following table.

Bit Positions 14	15	16	17	18	19	Maximum msec
1	X	X	X	X	X	57.68
0	X	X	X	X	X	28.88 + D
0	X	X	X	1	1	28.88

D is specified duration of distribution and can equal from 3.2 to 102.4 milliseconds depending on system requirements.

(1) Instruction Definitions

LSC	K	2, 3	K=1	2510103
			K=2	2510203
			K=3	2510403
			K=4	2511003
			K=5	2512003

LOAD SCANNER COMMAND REGISTER K. The scanner command register K is cleared and then loaded with C(A). A is not changed. The loading initiates a new scanner-distributor operation.

				J=1	J=2
BSC	J, K	2	K=1	2514022	2516022
			K=2	2514031	2516031

BRANCH ON SCANNER K OPERATION COMPLETE. Branch to location L + J if the scanner-distributor K has completed its operation. Take the alternate branch if the operation is not complete.

				J=1	J=2
BCO	J, K	2	K=1	2514021	2516021
			K=2	2514032	2516032

BRANCH ON CONVERTER K OVERFLOW. Branch to location L + J if the analog-to-digital converter overflow indicator is on. Take the alternate branch if the overflow indicator is not turned on. The LSC instruction will reset the overflow condition.

RCV	K	2, 3	K=1	2510144
			K=2	2510244
			K=3	2510444
			K=4	2511044
			K=5	2512044

READ CONVERTER K. C(Converter K) $_{1-12}$ replaces C(A) $_{8-19}$. C(A) $_{0-7}$ are replaced by zeros. In single-channel scanners, the C(Converter K) are unchanged. In multiple-channel scanners, the RCV command also switches the converter to the next input channel, initiating the conversion of that channel. (The multiple-input scanner will be discussed later in this manual.)

RDG		2	2500023

READ DIGITAL INPUT. Two 4-bit binary-coded-decimal input characters selected by the scanner-distributor subcontrol are read into A_{1-8} with A_1 holding the most significant bit. This instruction should be preceded by a successful BSC instruction. $C(A)_{S, 9-19}$ are replaced by zeros.

LCV	K	2, 3	K=1	2510101
			K=2	2510201
			K=3	2510401
			K=4	2511001
			K=5	2512001

LOAD CONVERTER REGISTER K FROM A. $C(A)_{8-19}$ are transferred to converter register K. Converter register K is automatically cleared by this instruction prior to the transfer. A is not changed.

(2) Examples of Scanner-Distributor Instructions

(a) Scanning a Thermocouple Sensor

the following program scans an Iron-Constantan thermocouple sensor and converts the reading to degrees Fahrenheit. The conversion equation assumes the temperature to be less than 1000°F and that in this range the relationship between millivolts output and temperature in linear. The equation is based on data from thermocouples with reference junctions at 32°F and a correction must be applied for any variation of the reference from this temperature. The actual current reference junction temperature is assumed to be stored in location $(3000)_8$ at B12. It is further assumed that the linear range of the thermocouple includes the normal range of the reference junction temperature.

The basic equation for the thermocouple, from standard table data, is

$$T = 32.5V + 40.5$$

where T = Temperature in °F
V = Output in millivolts

If the referenne junction temperature differs from that for which the equation is true, a correction must be applied to the measured millivolt output of the thermocouple to adjust the output to that which would have existed if the reference junction had been correct. The equivalent output of the reference junction relative to a standard reference is given by the inverse of the above equation.

$$V_r = \frac{T_r - 40.5}{32.5}$$

This value must be added to the measured thermocouple output. Therefore,

$$T = 32.5\,(V_m + V_r) + 40.5$$

$$\text{or } T = 32.5\left(V_m + \frac{T_r - 40.5}{32.5}\right) + 40.5$$

$$\text{where } T = 32.5\,V_m + T_r$$

If the thermocouple is scanned on the 40 mv range, the count value appearing in the C register will be

$$C_m = 100\,V_m$$

and the final equation is then

$$T = 0.325\,C_m + T_r$$

The thermocouple leads are connected to relay matrix address $(3121)_8$ and the converted temperature is to be stored in location $(05020)_8$ at B19.

Location of Instruction	Operation Code	Operand Address	Octal Instruction	Remarks
02000	LDA	04000	0004000	Get scanner command and
02001	LSC	1	2510103	initiate scan.
02002	BSC	2, 1	2516022	Wait for conversion
02003	BRU	02002	2602002	to be complete.
02004	BCO	1, 1	2514021	If converter overflows
02005	BRU	XXXXX	26XXXXX	go to ERROR Routine
02006	RCV	1	2510144	Read Counts in to A
02007	MAQ		2504006	and move to Q, B19.
02010	MPY	04001	1504001	Multiply by 0.325, B19.
02011	SLD	7	2411007	Shift to B12 and add
02012	ADD	03000	0103000	Ref. Temperature.
02013	SRA	7	2400007	Shift to B19 and
02014	STA	05020	0305020	store result.
02015	next instruction			
04000	00011001010010011010		0312232	Scanner command
04001	00110000000000000000		0600000	$(0.325)_{10}$ at B0.

(b) Scanning a Pressure Sensor

A pressure ranging from 3 to 15 pounds per square inch (psi) is sensed by a transducer that outputs a current ranging from 4 to 20 milliamps, linearly proportional to the pressure sensed. This current is then passed through a 4 ohm resistor which outputs a voltage ranging from .016- .080 volts. This voltage is then sensed by the scanner-distributor on the 80 millivolt full scale range (where 80 millivolts - 4000 counts). The sensor leads are connected to relay matrix address 2131. The following program scans this sensor, converts, the reading to psi and stores this value in location 03017. The following equation is used in the conversion.

Pressure (psi) = Counts x 0.00375.

Location of Instruction	Operation Code	Operand Address	Actual Form of Instruction in Octal	Remarks
02500	LDA	02600	0002600	Load scanner command into
02501	LSC	1	2510003	scanner command register
02502	BSC	2, 1	2516022	Wait for completion
02503	BRU	02502	2602502	of scanner operation
02504	BCO	1, 1	2514021	Test for conversion overflow
02505	BRU	XXXXX	26XXXXX	Go to error routine
02506	RCV	1	2510044	Read counts into A (B19)
02507	MAQ		2504006	Put counts in Q (B19)
02510	MPY	02601	1502601	Multiply by (0.00375)
02511	STA	03017	0303017	Store pressure (B11).
02512	Next instruction			
02600	00010001001010011000		0213230	Scanner command word
02601	01111010111000010100		1727024	(0.00375) Constant (B-8)

(c) Read in the Value Dialed on Two Decimal Manual Entry Switches

The decimal values dialed into manual switches and other similar devices may be read directly into the A register through decimal-to-binary conversion circuits. The read-in is activated by supplying a voltage to the center tap of the manual switch. The 4-bit binary-coded-decimal output of the decimal-to-binary converter is then transferred directly into the A register.

For the example, the center taps of two switches are commonly connected to relay matrix contact address $(6357)_8$ and the output of each switch is directed to a separate decimal-to-binary converter.

The outputs of these two converters are transferred simultaneously to the A register into positions A_{1-4} and A_{5-8}. Store the binary equivalent of these switch positions in location 2000.

Location of Instruction	Operation Code	Operand Address	Actual Form of Instruction in Octal	Remarks
01000	LDA	1100	0001100	Load scanner command into
01001	LSC	1	2510003	scanner command register
01002	BSC	2, 1	2516022	Wait for scanner complete
01003	BRU	01002	2601002	
01004	RDG		2500023	Read in BCD switch readings
01005	STA	02000	0302000	Store switch readings
01006	Next instruction			
01100	11110011101111000011		3635703	Scanner command word

(d) Send a Control Signal to a Process Regulator

Control is normally exercised over the process by supplying set points (analog reference voltages) to process controllers or regulators. These process controllers are self-contained servomechanisms or sub-loop control systems such as fuel flow valve controllers that need only be supplied with a set point periodically. The set points are computed optimum values that are placed in the C register by the program and then, activated by the scanner-distributor in subcontrol, converted to an analog signal that is transmitted to the process controller.

Example: A 10-bit set point for a process controller is stored in location (05010). This set point is to be applied to the digital-to-analog channel which is activated by applying a +6 volt reference voltage to relay matrix contact address $(4333)_8$.

Location of Instruction	Operation Code	Operand Address	Actual Form of Instruction in Octal	Remarks
03000	LDA	05010	0005010	Put digit set point into
03001	LCV	1	2510001	Conversion register
03002	LDA	03100	0003100	Load command into
03003	LSC	1	2510003	Scanner command register
03004	BSC	2, 1	2516022	Wait for completion
03005	BRU	03004	2603004	of output operation
03006	Next instruction			
03100	1110001101101110000 0		3433340	Scanner command word

(3) Multi-Channel Scanner-Distributor

In application requiring high rates of scanning multi-channel scanner-distributors are employed. The following table lists the characteristics of these multi-channel scanner-distributors.

No. of Channels	High Speed Scanning Rate
1	20 Points per second
2	40 Points per second
4	79 Points per second
8	151 Points per second

The multi-channel scanner-distributors have 2, 4, or 8 separate analog conditioning channels consisting of an attenuator, amplifier, and polarity changer. One analog-to-digital converter is employed to convert all channels, being switched from one to the next with a solid state selector. The 2, 4, or 8 analog sensor leads are selected through the relay matrix and connected to the 2, 4, or 8 input conditioning channels in a parallel fashion. All signals are conditioned simultaneously, after which, the analog-to-digital converter is switched from channel to channel converting the output of each channel into counts (0-4095) in the Conversion register. One scanner command word controls the scanning of each group and therefore, all sensors in a group must have the same characteristics (range, polarity, and scan rate).

Upon receiving a scanner command word, the scanner connects the analog-to-digital converter to the first channel and initiates the selection and conditioning of the group of sensors specified. The first RCV instruction causes the contents of the conversion register (digital equivalent of the first channel) to be transferred to the A register and also steps the analog-to-digital converter to the next channel. Each successive RCV reads the digital value for the current channel and steps the converter to the next channel.

Each analog to digital conversion after the first one requires 600 microseconds and a scanner-complete signal is generated after each conversion, including the last one. It is possible to select one of the channels uniquely as follows. If it is desired to read the fifth channel in an eight-channel scanner, four successive RCV commands should be executed after the initial conversion-complete signal is received. Then, after a second conversion-complete signal is received, the fifth RCV command should be executed to obtain the desired reading. It should be remembered, however, that still another conversion-complete signal will be received. This creates no special problem; but if scanning is being accomplished by an interrupt program entered upon receipt of each conversion-complete signal, the program must be aware the additional signal will be generated. The same logic applies to the reading of the last channel also.

(a) Example of multi-channel scanner-distributor programming:

The following program scans eight sensors selected by group relay matrix address $(1220)_8$ on the 40 mv range and stores the counts in eight sequential locations beginning with location 01000. These counts would then be converted to engineering units by another program.

Location of Instruction	Operation Code	Operand Address	X	Actual Form of Instruction in Octal	Remarks
00500	LDA	00600		0000600	Load command into Scanner
00501	LSC	1		2510003	command register
00502	LDX	00601	1	0620601	Let loop counter to zero
00503	BSC	2, 1		2516022	Wait for conversion
00504	BRU	00503		2600503	to be complete
00505	BCO	1, 1		2514021	Test for conversion overflow
00506	BRU	XXXXX		26XXXXX	Go to error routine
00507	RCV	1		2510044	Read counts into A
00510	STA	01000	1	0321000	Store counts
00511	INX	1	1	1420001	Increment loop counter
00512	BXL	-8	1	0437770	Test for end of loop
00513	BRU	00503		2600503	Repeat loop
00514	Next instruction				
00600	00001010010000011010			0122032	Scanner command word
00601	00000000000000000000			0000000	Constant zero

3. Output Distributor

The output distributor is designed to relieve the scanner-distributor of most of its subcontrol functions in systems which require numerous subcontrol functions in addition to heavy loads of analog scanning. Two primary classes of digital output functions are performed by the output distributor. The two functions, described below, are completely independent and a given GE 412 system may include either function alone or both.

a. Multiple Output Function

This function is intended primarily to update the status of annunciators, alarms, decimal displays, analog outputs, regulator references and the like. It is designed for operation into bistable relay devices but is not limited to their use.

The multiple output function is initiated by transferring a control word from the A register into the Multiple Output (MO) Command Register. The MO control logic then decodes bits 0 through 7 as the address of a group of contacts through which the digital information contained in bits 8 through 19 is transmitted to the selected bistable relay devices. At the completion of this action, which requires 3.2 milliseconds, a multiple-output-complete signal is generated which may be tested by a branch command or used to cause automatic program interrupt. An interlock action is included which prevents the initiation of a multiple output action unless the previous action is complete.

Multiple Output Command Format:

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
Row				Column				Data											
Group Matrix Address																			

b. Timed-Contact Function

This function is used whenever a single isolated contact closure (or opening) of a specified time duration is required. The timed-contact function is initiated by transferring a control word from the A register into the Timed-Contact (TC) Command Register. The TC control logic then decodes bits 0 through 7 as the matrix address of the desired contact to be actuated and bits 14 through 19 as the desired time duration. The time duration is specified as a multiple of 12.8 milliseconds, i.e. a time duration code of 10 (octal 12) specifies a duration of 128 milliseconds.

When the complete action (selection, contact closure, delay, dropout) has been completed, which takes 19.2 milliseconds plus the specified duration, a timed-contact-complete signal is generated. The completion signal may be tested by a branch command or used to cause automatic interrupt. Interlock action prevents the initiation of a timed-contact action until a previous action has been completed.

Timed-Contact Command Format:

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
Row				Column				Not Used								Duration Code			
Contact Matrix Address																			

c. Multiple-Output Commands

LDM 2, 3 2510112

LOAD OUTPUT DISTRIBUTOR, MULTIPLE-OUTPUT FUNCTION. The C(A) replace the C(MO). C(A) are not changed. A multiple-output function is initiated.

BDM J 2 J=1 2514034
J=2 2516034

BRANCH ON MULTIPLE OUTPUT OPERATION COMPLETE. Branch to location L + J if the output distributor, multiple-output function has completed its operation. Take the alternate branch if the operation is not complete.

LDT 2, 3 2510012

LOAD OUTPUT DISTRIBUTOR, TIMED-CONTACT FUNCTION. The $C(A)_{0-7,\ 14-19}$ replace the $C(TC)_{0-7,\ 14-19}$ and a timed-contact function is initiated. C(A) are not changed.

BDT J 2 J=1 2514027
J=2 2516027

BRANCH ON TIMED-CONTACT OPERATION COMPLETE. Branch to location L + J if the output distributor, timed-contact function has completed its operation. Take the alternate branch if the operation is not complete.

---

4. Digital Data Accumulator/Digital Fast Scanner

The digital data accumulator/digital fast scanner (DDA/DFS) is a solid state device that provides the GE 412 system with two functional capabilities. Up to five DDA/DFS units may be connected on one GE 412 system.

(1) The DDA accumulates or counts pulses that are normally generated by process sensing equipment, i.e. tachometer, flow meter, killowatt-hour meter, etc. It provides 4, 8, 12, or 16-bit counters capable of counting 15, 255, 4095, and 65535 pulses respectively, which are periodically read into the computer. Each counter is reset to zero by the reading operation.

(2) The DFS senses the status of contact closures related to process sensors, i.e. hot metal detector, over temperature sensor, etc. It scans these contacts in groups of 16 and transmitts the contact status into the computer. The DFS may also be employed to read in the position of decade switches, on-off status of process equipment, and other similar conditions.

Figure 13 shows the DDA/DFS in block diagram form. The selection of a specific counter or contact group, and the output buffering of the information into the computer are common to both DDA and DFS operations. It is not possible, therefore, to operate the DDA and the DFS simultaneously. The 6-bit input selector register is loaded from A_{14-19} by a LAC or LDS instruction, depending on whether a DDA or DFS operation is called for. Loading this register selects the specific counter or group of contacts that is to be read into the computer. The actual read-in is caused by the execution of an RFA instruction. For DDA operations, the contents of the selected counter are transferred into the A register, A_{16-19} for 4-bit counters and A_{12-19} for 8-bit counters. Unused bit positions are reset to zero. For DFS operations the status of the 16 contacts of the selected group are transferred to A_{4-19}. A contact closed is represented by a "1" and a contact open by a "0". A validity bit is also transferred into A_0, with a "1" indicating a valid read-in, and a "0" indicating an invalid read-in. An invalid read-in takes place when the proper voltage has not been applied to the contacts being read-in. An invalid read-in indicates a blown fuse or a connector disconnected.

The execution of each RFA instruction also causes the input selector register to be incremented by one. In this fashion, successive counters or groups of contacts may be read-into the computer in sequence by executing successive RFA instructions. This method requires only one LAC or LDS instruction specifying the starting counter or group. The RFA instruction is used with other specially designed process input equipment and is in a sense a general command used to read-in information from one of many fast access devices.

Special DFS groups are available wherein any change of state of any contact in the group will result in an output signal which may be wired to cause automatic program interrupt. Several such groups may be wired to a single level of interrupt, but if this is done certain precautions must be observed to insure that no change-of-state signals are lost. First, automatic interrupt should be inhibited while the DFS groups are being read. Second, the interrupt program, which reads and examines the groups for the specific contact which changed state to cause the interrupt, should read all of the special groups in succession before interrupt is again permitted.

a. Instruction Definitions

LDS	K	2, 3	K=1	2510102
			K=2	2510202
			K=3	2510402
			K=4	2511002
			K-5	2512002

LOAD DIGITAL SCAN COMMAND REGISTER. Select the digital fast scanner, K, and disconnect the associated digital data accumulator. $C(A)_{14-19}$ replace the contents of the DDA/DFS input selector register, which in turn selects a specific group of contacts for scanning. The DFS will remain selected until disconnected.

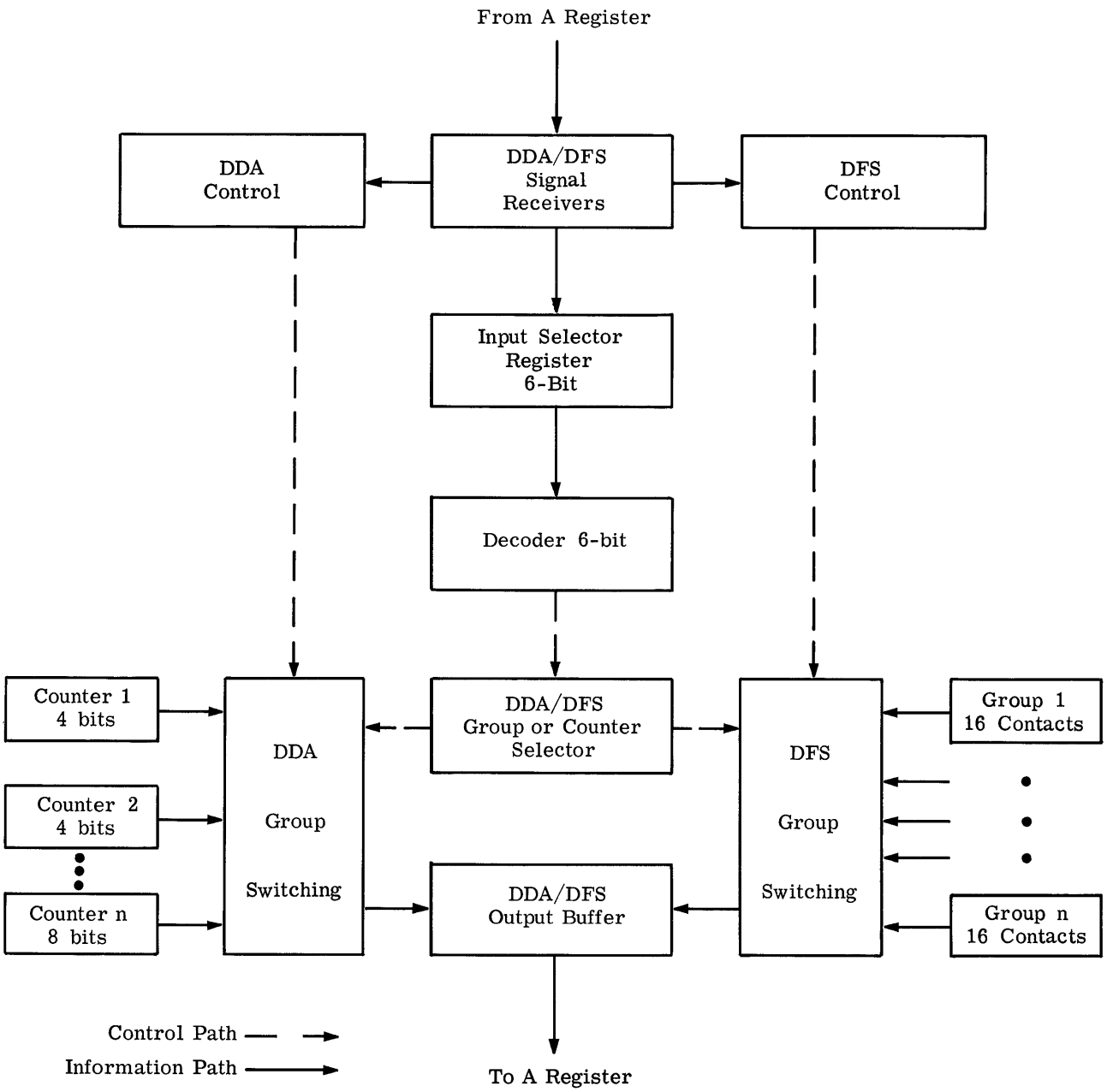


Figure 13. Digital Data Accumulator / Digital Fast Scanner

LAC	K	2, 3	K=1	2510100
			K=2	2510200
			K=3	2510400
			K=4	2511000
			K=5	2512000

LOAD ACCUMULATOR SCAN COMMAND REGISTER. Select digital data accumulator K and disconnect the associated digital fast scanner. $C(A)_{14-19}$ replace the contents of the DDA/DFS input selector register, which in turn selects a specific counter for reading into the computer. The DDA remains selected until disconnected.

RFA	2, 3	2510046

READ FAST ACCESS DEVICE. The contents of the fast access device selected (DDA, DFS, or other special devices) replace the C(A). Unused bits in the A register are set to zero. When used with the DDA/DFS this instruction causes the input selector register to be incremented by one.

OFA	2	2510007

FAST ACCESS DEVICE OFF. Any selected fast access device is disconnected.

b. Examples of DDA/DFS Instructions

(1) Read in counters 1 and 2 in the DDA and add them to the commulative totals stored in locations 01210 and 01022 respectively.

Location of Instruction	Operation Code	Operand Address	Actual Form of Instruction in Octal	Remarks
14000	LDA	15000	0015000	Load counter address into
14001	LAC	1	2510140	Input Select Register
14002	RFA		2510146	Read in counter 1 *
14003	ADD	01210	0101210	Add sum to counter contents
14004	STA	01210	0301210	Store new sum
14005	RFA		2510146	Read in counter 2 *
14006	ADD	01211	0101211	Add sum to counter contents
14007	STA	01211	0301211	Store new sum
14010	Next instruction			
15000	00000000000000000001		0000001	Constant counter address

*Each RFA causes the Input Select Register to be automatically stepped by one.

(2) Determine whether contact number 5, counting left to right, in DFS group $(15)_8$ is open or closed. If it is open transfer to a program starting in location 11500, if closed transfer to a program starting in location 11600.

Location of Instruction	Operation Code	Operand Address	Actual Form of Instruction in Octal	Remarks
06000	LDA	06011	0006011	Load input select
06001	LDS		2510102	Register with DFS group address
06002	RFA		2510146	Read in contact group $(15)_8$

Location of Instruction	Operation Code	Operand Address	Actual Form of Instruction in Octal	Remarks
06003	BPL	1	2514001	If sign - then valid read-in
06004	BRU	XXXXX	26XXXXX	If sign + then invalid read-in
06005	ANA	06012	2206012	Extract contact 5, bit A_8
06006	BZE	1	2514002	
06007	BRU	11500	2611500	Contact open
06010	BRU	11600	2611600	Contact closed
06011	00000000000000001101		0000015	Group address
06012	00000000100000000000		0004000	Extract constant

C. PROGRAMMING AND MAINTENANCE CONSOLE

The programming and maintenance console shown in figure 7, contains the indicators, register displays, and controls necessary for controlling the computer during normal operation and for performing program check-out and maintenance functions.

1. Indicators

There are 12 rectangular indicator lights arranged in a row across the top of the console. One is used to indicate overflow condition in the A register, one is used to indicate the computer is in the permit interrupt mode of operation, and in the 412B one is used to indicate that the computer is accessing an operand in the "upper 8K" of memory. The others are spares which can be used for special system indicators.

2. Alarm Indicators and Controls

There are 12 alarm indicators on the right side of the console, which are lighted-pushbuttons. They are therefore a combination of indicators and reset pushbuttons for certain alarm conditions within the GE 412 System. They may be single or dual indicators, lighted on the top or the bottom; but only one pushbutton is available for each one.

The conditions that are indicated with the indicators are:

a. Core Parity Error
b. Drum Parity Error
c. Core Temperature
d. Stall Alarm
e. Cabinet Over-Temperature
f. Digital Clock Error
g. Paper Tape Parity Error on M Register
h. Paper Tape Parity Error on N Register
i. Primary Power Failure
j. Echo Alarm in Controller Selector

Five dual indicators are spare and used for special system functions.

3. Register Displays

Three rows of indicator lights are located in the center of the console and are used to display the P register, I register, and the A register. One row of 20 indicator lights, called the maintenance display, is located above these three and may be used to display the contents of the Q, Z, W, C_1, C_2, M, N, B, Priority Interrupt, Drum Address, and Core Address registers. The selection of which register to display is made on the Maintenance Display Selector rotary switch located on the bottom of the console. This maintenance display may be used to display the contents of special system registers as defined by the system engineer. These special displays are selected with the Auxiliary Display Selector located on the bottom of the console. This selector is active when the maintenance display selector is in the AUX position. Above the maintenance display

lights is another row of 16 lights, also used primarily for maintenance purposes. These lights indicate the state of internal instruction sequencing and memory accessing priority. (See e and f below.)

4. Controls

a. Key Switch

This key operated switch turns the console on and off. In the "on" position all console switches are enabled. In the "off" position all console switches except the following are disabled:

parity reset switches,
echo alarm reset switch,
the A register toggle switches,
power off switch,
demand pushbutton,
maintenance display selectors.

b. Power Switches and Indicators

Power On and Power Off pushbuttons are provided on the console to turn D. C. voltages on or off. A dual indicator light indicates that the power is "on" or "off" with green and amber lights respectively. When D. C. power is initially turned on, the following units are initialized: Elapsed Time Counters, M Register, N Register, Memory Access Priority, Instruction Sequencing, Drum Command Logic, Automatic Program Interrupt, and System Initialization Signal. The System Initialization Signal is available to initialize such additional equipment as the System Engineer may specify. When power is on and the computer is operating in the manual mode, depressing the Power On pushbutton will initialize the following: Instruction Sequencing, Drum Command Logic, Automatic Program Interrupt, and the System Initialization Signal.

c. Manual/Automatic Toggle Switch

The Manual/Automatic toggle switch is used to select the mode of operation, and is enabled when the Key Switch is in the on position. When this switch is in the "Manual" position all console control switches are enabled. When in the "Automatic" position only the Save P, Save I, Step Switch, and A Register Toggle Switches are enabled.

d. The A Register Controls

(1) The A Register Clear Switch. The A register clear pushbutton clears the A register to zeros when the console is in the manual mode of operation.

(2) The A Register Manual Set Switches. Twenty manual "set A" pushbuttons are provided that, when depressed, set the corresponding position in the A register to one. They are active in the manual mode of operation.

(3) The A Register Toggle Switches. Twenty Toggle switches, corresponding to the 20 positions in the A register, are located in the center of the console. They operate in conjunction with the READ CONSOLE SWITCHES instruction, and represent a zero in the up position and a one in the down position.

e. Instruction/Word Toggle Switch

When the console is on and in the manual mode of operation, the Instruction/Word toggle switch selects the step mode of the computer. In the "Instruction" position it allows a complete instruction to be executed as the Step Switch is depressed. In the "Word" position it allows the completion of only one word time of the execution of an instruction as the Step Switch is depressed, and the upper row of maintenance lights indicates the internal instruction sequencing.

f. Step Switch

When the console is on and in the manual mode of operation, the Step Switch will step the computer either one word or one instruction depending upon the position of the Instruction/Word Toggle Switch. The Step

Switch is also used to initiate automatic operation of the computer after the Manual/Automatic Toggle Switch is placed in the Automatic position.

g. Save P Toggle Switch

The save P toggle switch is used to inhibit the changing of the contents of the P register. It is active when the console is on.

h. Save I Toggle Switch

The save I toggle switch is used to inhibit the changing of the contents of the I register. It is active when the console is on.

i. Transfer A to I Indicator/Switch

Transfer A to I switch causes the contents of the A register to replace the contents of the I register. It is active when the console is on and in the manual mode of operation. The indicator light is on when the switch is active.

j. Exchange A and Q Indicator/Switch

The exchange A and Q switch when depressed causes the contents of the A and Q registers to be exchanged. It is active when the console is on and in the manual mode of operation. The indicator is on when the switch is active.

k. Drum Transfer Indicator/Switch

The drum transfer switch is used in conjunction with other controls on the console to manually initiate a Drum-Core transfer. It is active when the console is on and in the manual mode of operation. The indicator is on when the switch is active.

l. Demand Indicator/Switch

The demand indicator/switch is used to set a flip-flop which operates in conjunction with the BRANCH ON DEMAND instruction. When depressed, the flip-flop is set and the indicator is turned on. When a BRANCH ON DEMAND instruction is executed the indicator is turned off and the flip-flop is reset.

D. CONTROLLER SELECTOR INSTRUCTIONS

The controller selector shares core memory accesses with the magnetic drum and the central processor. Using the controller selector, it is possible to use high speed magnetic tape, high speed line printer, disk storage and data communication systems as input-output media. It is possible through the use of these devices to do large scale data processing and scientific computing on the GE 412 computer.

LCS	D	2	D=0	2501000
			D=1	2501100
			D=2	2501200
			D=3	2501300

LOAD CONTROLLER SELECTOR. Load the control registers of controller selector D from the next two sequential core locations. The constant D specifies which of 4 controller selectors to use. This command initiates the transfer of information between core storage and the device(s) connected to the controller selector.

BCS	J, D, E	2	J=1	2515DOE
			J=2	2517DOE

BRANCH ON CONTROLLER SELECTOR. Branch to location L + J if the condition specified by E is true for controller selector D. Take the alternate branch if condition E is false. E specifies one of 8 conditions applicable to the controller selector D.

BEA	J	2	J=1	2514023
			J=2	2516023

BRANCH ON ECHO ALARM. Branch to location L + J if the echo alarm flip-flop is set. Take the alternate branch if it is not set. This instruction resets the echo alarm flip-flop but does not reset the console echo alarm indicator. The indicator is manually reset by depressing a button on the console. The echo alarm flip-flop is set as the result of an unsuccessful selection operation by the controller selector.

IV. PROGRAMMING TECHNIQUES

A. THE PROCESS ASSEMBLY PROGRAM

1. Introduction

Section III on Programming Fundamentals was devoted to the simple presentation of the basic computer language making use of mnemonic codes to represent instruction operation codes and numerical addresses to represent locations in storage. That section provided all the required information to prepare a program in actual computer language (binary), however this language is cumbersome and difficult to use. The more simplified approach to writing programs would be to use some symbolic language that would be more convenient for the programmer. A symbolic program must later be converted into an actual computer language program before execution. The Process Assembly Program (PAP), a program written for the GE 412, is capable of converting programs written in symbolic form into actual machine language. The symbolic language used to write programs is then called the PAP language, and allows the programmer to refer to computer instructions and computer storage locations in a symbolic form.

The instructions written in PAP language are only symbolic of the actual language instructions and bear a one-to-one relationship to the actual language instructions. The PAP program is first loaded into the GE 412, and then the computer, under control of the PAP program, reads in symbolic PAP language instructions, converts them into actual instructions, and outputs the actual binary language program, called an object program. This object program may then be loaded into the storage of the GE 412 System and executed.

2. PAP Assembly

The format of PAP instructions is similar to the actual instructions shown previously. Instructions are written with location, operation code, operand, tag and comments. The operation code is the only column which must always be present, and it is used to represent one of the acceptable mnemonic operation codes or one of the special pseudo operation codes which will be defined in this section. The symbols used in the location and operand columns are limited only by the programmer's selection and are usually chosen to mnemonically represent data, constants, or routines. This mnemonic reference aids greatly in debugging and correcting programs. A programmer might assign the symbol TYPE to the starting location of a type subroutine, or ALARM to the starting location of an alarm routine. PAP will then assign actual storage addresses to these symbols and remember the assignment so that further reference to that symbolic address may be assigned the same actual address. Address Symbols are usually required only for the starting locations of routines and for locations that are referred to by other instructions in the programs. PAP will assign addresses to other instructions automatically.

A method of assigning storage blocks and starting addresses with actual computer storage locations is provided through the use of pseudo operations. These pseudo operations are written mnemonically the same as computer operation codes, but these codes represent instructions to PAP, rather than instructions for the GE 412. Therefore, the pseudo operation codes never appear in the final actual language program, but rather provide PAP with the necessary information so that it may perform a correct assembly of the symbolic program.

The output of PAP consists of two basic parts, a paper tape output of the actual machine-language instructions in a format compatible with standard loading routines, and a listing of the original input coding together with the octal representation of the machine-language instructions generated. This output listing also includes error codes to call attention to coding errors detected by PAP during the assembly process. A list of the error codes follows.

Code	Meaning
O	Illegal operation code.
A	Possible error in operand field.
T	Possible error in tag field.
L	Illegal character in location field.
M	Symbol in location field multiply defined.
U	Symbol in location field undefined.
S	Symbol table in full.
F	Operation table is full.

3. The PAP Coding Sheet

PAP language instructions are written on the PAP coding sheet, shown in figure 14. The sheet has six separate columns or fields: Location, Operation, Operand, Tag, Comments, and Sequence Number. The PAP program accepts symbolic instructions from paper tape input or punched card input. If paper tape is used, a tab character indicates the end of a field and a carriage return indicates the end of an instruction. The punched card input is fixed in given card columns for each field. The rules and definitions that apply to each field are given below.

a. Location.

The location field is used to identify the symbolic location in storage of the instruction. The entry in this field must be symbolic and is restricted in that plus and minus signs may not be used as part of the symbol. The symbolic representation of a location can be up to 6 characters in length, blanks or spaces will be ignored, i.e., the symbol "A B" is interpreted as "AB". The first character of the symbol must be alphabetic. If an asterisk (*) appears in the first position of the location field, the entire line will be treated as a remark and will not affect the PAP assembly in any way.

b. Operation

The operation field is always three characters in length and is used to enter mnemonic computer operation codes or PAP pseudo operation codes. The list of operation codes is given in Appendix D.

c. Operand

The operand field contains additional symbolic and numeric information necessary for complete definition of the instruction. It may contain a symbolic operand address up to six characters long, a decimal address, constants associated with shift, X location and other instructions, or may be blank for some instructions. The numeric entries are assumed decimal unless otherwise specified, i.e., a "/" preceding an octal constant. Numeric constants in BXH and BXL instruction are automatically negated by PAP and therefore placed in the actual instruction in the 2's complement form. Arithmetic and relative addressing is permitted in the operand field of instructions referring to storage locations as operands. The asterisk (*) is used for relative addressing and represents the address of the current instruction. Symbols appearing in the operand field need not be previously defined, but, somewhere in the program, they must be defined by appearing in the location field of some instruction.

d. Tag

The tag field is a single character field and may be used to enter a numeric character or it may be left blank. The valid entries in this field are 0, 1, 2, or 3 and specify the use of the corresponding automatic address modification location (X location) to be used to modify the instruction. The 0 or blank indicates that no automatic address modification is to take place, or that X location 0 is to be used with LDX, STX, INX, BXL, or BXH or SPB instructions.

e. Comments

The comments field is of variable length and is used at the discretion of the programmer to make remarks associated with each instruction. Comments are never used by PAP in the assembly, but the information is very helpful when debugging or changing a program. When programs are prepared for entry to the computer in paper tape form, this field is punched on the tape. The field has a maximum length of 24 characters when punched on cards.

f. Sequence

The identification field is used to sequence number input cards and this field is not used on paper tape input. A maximum of 5 characters may be entered into this field.

4. Pseudo Operation Codes

PAP uses a group of terms called pseudo instructions in addition to the mnemonic codes for the instructions in the normal repertoire of the GE 412. A pseudo instruction is a symbolic representation of

GENERAL ELECTRIC

PROCESS ASSEMBLY PROGRAM CODING FORM

PROGRAM		DATE
PROGRAMMER	COMPUTER	Page of

LOCATION	OPERATION	OPERAND	TAG	COMMENTS	SEQUENCE
1	8	12	29	31	56

Process Computer Section, Industry Control Department — Phoenix, Arizona

Figure 14. PAP Coding Sheet

information required by PAP for the proper assembly of a program. Pseudo instructions have the same general form as a computer instruction, however, they are never executed by the computer as part of the actual program. These pseudo instructions provide information to PAP but do not actually become part of the final program. A list of pseudo instructions available for use with PAP is given below with a complete description of usage. Some examples are given to illustrate correct usage of the pseudo instructions and to illustrate the assembly program's reaction to erroneous usage.

a. SLC SET LOCATION COUNTER

The assembly program assigns consecutive memory locations to the instructions being assembled. The SLC pseudo instruction enables the programmer to control the location of his program in memory by setting the initial value of the location counter used by the assembly program and resetting it whenever necessary. Unless otherwise specified, the location counter will automatically be set to zero at the beginning of the assembly. The operand field is used to specify the manner in which the location counter is to be set. The several options are described below.

(1) If the operand field contains a value, the location counter will be set to that value. The value may be a decimal number, an octal number (indicated by a 1 preceding the number), or a previously defined symbol. A composite operand consisting of a summation of permissible operands may also be used.

(2) If the operand consists of the single letter, E, the location counter will be set to the next even value. If this results in a "skipped" location, a NOP instruction will be inserted at that skipped location.

(3) If the operand consists of the single letter, O, the location counter will be set to the next odd value. If this results in a "skipped" location, a NOP instruction will be inserted at that skipped location.

(4) If the first character of the operand is a plus sign, PAP advances the location counter by the amount specified, thus skipping over a block of locations to reserve them for data storage or similar purposes. Any of the operands defined in option (1) above may follow the plus sign to specify the size of the block to be reserved. If the location field of the SLC pseudo instruction contains a symbol, that symbol will be assigned to the first location of the reserved block.

b. DCW CONSTRUCT A DRUM CONTROL WORD

The standard loader routine is capable of loading information directly into magnetic drum storage (using an area of core memory as a buffer). If this is desired, it is necessary that the address word punched in the output paper tape begin with a "7" and that the beginning location be specified as a track and sector address for the drum. Any program loaded into drum memory will eventually have to be transferred into core memory for operation, however, and it must be assembled to operate in core. The DCW pseudo instruction provides the assembly program with the information necessary for generating the address word required for specifying to the loader the correct drum address and also to generate the coding for the area of core memory in which the program will ultimately operate.

To fulfill these requirements, the DCW pseudo instruction must be supplied with three operands separated by commas. The first operand gives the drum track number, the second gives the drum sector number, and the third operand specifies the core starting address for which the program should be assembled.

These operands may be decimal, octal, or symbolic. If they are symbolic, the symbols must have been previously defined.

c. FOR LOADER FORMAT SELECTION

A program written to operate in a given area of core memory normally will not operate properly in any other area of memory. For instance, a branch instruction intended to cause a transfer to alternate coding under certain conditions will not have the desired effect if the alternate coding is not in the expected location. However, the loader routine is capable of loading a given program into a section of memory other than that for which it was originally assembled, modifying the operand addresses as required to permit the program to operate in its new location, providing the load tape is in the proper format.

The assembly program will produce an output paper tape in absolute (nonrelocatable) format unless instructed otherwise. If a relocatable program or block of instructions is desired, the FOR pseudo instruction must be used.

If the operand field of the FOR pseudo instruction contains the single letter "R" all following instructions will be punched into the output tape in relocatable format until the assembler encounters a FOR pseudo instruction with the single letter "A" in its operand field.

When operating in the relocatable mode, the assembler punches the BCD code for the letter "R" at the end of each instruction containing an operand address which should be modified when the program is relocated.

Symbols defined by EQL may not be relocated and will not be assembled in relocatable format even if preceded by a FOR R pseudo instruction. Also, operand addresses specified in absolute on the coding sheet or by symbols defined while the assembler is operating in the absolute mode will not be relocated.

d. EQL ASSIGN A VALUE TO A SYMBOL

The EQL pseudo instruction requires a symbol in the location field and a value in the operand field. The value in the operand field may be decimal, octal (indicated by /), or symbolic. If it is symbolic, the symbol must have been previously defined. Composite operands comprised of a summation of permissible values are also permissible.

Two symbolic operands are reserved for special usage. The single letter "M" in the operand field associates the symbol in the location field with the M register. Likewise, the single letter "N" associates the location symbol to the N register.

e. DEC CONSTRUCT A SINGLE-PRECISION DECIMAL CONSTANT

The DEC pseudo instruction allows the programmer to enter decimal constants directly as part of the program to be assembled by PAP. The assembly program makes the necessary conversion to binary and positions the result at the specified scale factor. Integer, fractional, and mixed numbers are permissible with the letter "E" used to denote exponent of a power-of-ten multiplier and the letter "B" used to denote scaling of the resulting binary number relative to the machine binary point. If E is not given, it is assumed equal to zero, and if B is not given it is assumed equal to 19.

f. DDC CONSTRUCT A DOUBLE-PRECISION DECIMAL CONSTANT

The DDC pseudo instruction is identical to the DEC pseudo instruction except that a double-length (2-word) constant is constructed and assigned to the next two available locations. If B is not given, it will be assumed equal to 38.

g. OCT CONSTRUCT AN OCTAL CONSTANT

The OCT pseudo instruction allows the programmer to enter octal constants as part of the program to be assembled by PAP. A right-justified, integer octal value will be generated and assigned to the next available location. Up to seven digits may be specified and they may be preceded by a minus sign.

h. ALF CONSTRUCT AN ALPHANUMERIC CONSTANT

The ALF pseudo instruction allows the programmer to store, as constants for his program, 6-bit BCD alphanumeric characters, three to a word, right-justified.

The location field may contain a symbol through which the program can refer to the constant, and the operand field may contain any three alphanumeric characters recognizable to PAP. (A list of such characters may be found in Section IV. A. 5.)

i. END END OF ASSEMBLY

The END pseudo instruction directs the PAP program to punch a transfer code word into the output paper tape and to complete the assembly. If requested by the PAP operator, the assembly program will then add to the printed listing a table of EQL pseudo instructions showing the actual octal addresses assigned to the symbols defined by the program just assembled. This EQL table is also punched into paper tape in a format such that it may later be used to preassign locations to symbols before assembling another program or reassembling a corrected version of the same program.

j. DEF DEFINE A NEW OPERATION CODE
DES DESIGNATE A NEW OPERATION CODE

These two pseudo instructions are available to simplify the task of adding new mnemonic operation codes to the list of those recognizable to PAP. DES is used by PAP03 and DEF is used by PAP04.

While their use is simple, it requires some knowledge of the internal logic of the PAP programs. For this reason, it is suggested that a PAP manual be consulted for their descriptions.

NOTE: It should be noted that the next-to-last column of the examples, which are shown as actual PAP output listings, shows the actual location in octal, and the last column is the octal representation of the information actually punched in the paper tape output of the assembly program. Each group of seven octal characters is a word of program to be stored in memory by the loader routine excepting those which begin with a "4" or a "6". Words beginning with a "4" contain an address at which the loader is to begin storing the following words. The "4" signifies to the loader that a new address is being specified. Words beginning with a "6" similarly instruct the loader to skip a specified number of locations. The number of locations to be skipped is specified in octal.

5. Character Set Recognized by PAP

The following are the only characters recognized by PAP.

Alphabetic:	A through Z
Numeric:	0 through 9
Special:	+ (plus sign)
	- (minus sign)
	. (period, decimal point)
	, (comma)
	/ (slash)
	* (asterisk)
	(blank, space)

```
*          EXAMPLES OF PAP PSEUDO OPERATION CODES.
*
*
*      SLC PSEUDO CODE
*
       SLC 512                  DECIMAL OPERAND.
START  LDA SAM
       STA JOE
       BRU *
       SLC /2000                OCTAL OPERAND.
       LDA SAM
       STA JOE
       BRU START
SAM    SLC +10                  RESERVE TEN LOCATIONS.
JOE    SLC +/12                 RESERVE TEN LOCATIONS.
       SLC SAM+24               SYMBOLIC OPERAND.
BEGIN  LDA SAM+2
       STA JOE+4
       BRU START
       SLC E                    SET COUNTER EVEN.
JONES  SLC +2                   RESERVE TWO LOCATIONS.
       DLD JONES
       BRU *
       SLC O                    SET COUNTER ODD.
BOB    SLC +1                   RESERVE ONE LOCATION.
       LDA BOB
       STA SAM
       BRU BEGIN
       SLC +2
       SLC                      ERROR, NO OPERAND.
       LDA BETTY                SEE FOR PSEUDO CODE
       STA JOE
       BRU *
*
*      DCW PSEUDO CODE
*
       DCW 100,32,3000          DECIMAL OPERANDS.
       LDA SAM
       DCW /100,/32,/3000       OCTAL OPERANDS.
       LDA 1000
TRACK  EQL 100                     SEE EQL DEFINITION
SECTOR EQL 32                      IN NEXT SECTION.
       DCW TRACK,SECTOR,JOE     SYMBOLIC OPERANDS.
       LDA /1000
       DCW TRACK,32,/3000       MIXED OPERANDS.
       LDA SAM
*
*      FOR PSEUDO CODE
*
       LDX JANE            1
       INX 1               1
       STX JOAN            1
       FOR R                    REQUEST RELOCATABLE FORM
BETTY  LDA SAM             1
       BRU *+10                 RELOCATABLE
CHET   SUB JOE
       STA /4231
       NEG
BILL   STO BETTY                RELOCATABLE
       LDA 1000
       STO BILL                 RELOCATABLE
       ANA CHET                 RELOCATABLE
       STA BOB
       LDA JONES
       FOR A
```

```
0
0
0
0
0
00001          4001000
00002   01000  0002003
00003   01001  0302015
00004   01002  2601002
00005          4002000
00006   02000  0002003
00007   02001  0302015
00010   02002  2601000
00011          6000012
00012          6000012
00013          4002033
00014   02033  0002005
00015   02034  0302021
00016   02035  2601000
00017
00020          6000002
00021   02040  1002036
00022   02041  2602041
00023   02042  2504000
00024          6000001
00025   02044  0002043
00026   02045  0302003
00027   02046  2602033
00030          6000002
00031 A        4000000
00032   00000  0003004 R
00033   00001  0302015
00034   00002  2600002
0
0
0
00035          7144040
00036   05670  0002003
00037          7100032
00040   03000  0001750
00041
00042
00043          7144040
00044   02015  0001000
00045          7144040
00046   03000  0002003
0
0
0
00047   03001  0620144
00050   03002  1420001
00051   03003  1720117
00052
00053 M 03004  0022003
00054   03005  2603017 R
00055   03006  0202015
00056   03007  0304231
00057   03010  2504532
00060   03011  2703004 R
00061   03012  0001750
00062   03013  2703011 R
00063   03014  2203006 R
00064   03015  0302043
00065   03016  0002036
00066
```

```
        STA MARY                                        00067   03017  0300144
        BRU *-8                                         00070   03020  2603010
*                                                       0
*       EQL PSEUDO CODE                                 0
*                                                       0
MARY    EQL 100                 DECIMAL OPERAND.        00071
SUE     EQL /100                OCTAL OPERAND.          00072
JANE    EQL MARY                SYMBOLIC OPERAND.       00073
IN      EQL N                   N REGISTER DEFINITION.  00074
OUT     EQL M                   M REGISTER DEFINITION.  00075
        LDA MARY                                        00076   03021  0000144
        STA SUE                                         00077   03022  0300100
        SAB OUT,7                                       00100   03023  2400207
        SBA IN,19                                       00101   03024  2404023
        LDA JANE                                        00102   03025  0000144
JIM     EQL JAMES               ERROR, JAMES UNDEFINED. 00103
        LDA JIM                                         00104   03026  0000000
JAMES   EQL                     ERROR, NO OPERAND.      00105 A
        LDA JAMES                                       00106   03027  0000000
JOAN    EQL /1178               ERROR, OPERAND NOT OCTAL 00107
        LDA JOAN                                        00110   03030  0000117
*                                                       0
*       DEC PSEUDO CODE                                 0
*                                                       0
        DEC 932                                         00111   03031  0001644
        DEC +932                                        00112   03032  0001644
        DEC 932.0                                       00113   03033  0001644
        DEC 932.75                                      00114   03034  0001644
        DEC 932.75B16                                   00115   03035  0016446
        DEC 9.3275E2B16                                 00116   03036  0016446
        DEC 9.3275 E2 B16       NOTE THAT SPACES MAY BE 00117   03037  0016446
        DEC -9.3275 E2 B16      USED FOR READABILITY.   00120   03040  3761332
        DEC .05B0                                       00121   03041  0063146
        DEC 0.05B3                                      00122   03042  0006314
        DEC 0.05B-3                                     00123   03043  0631463
        DEC 5 E-2 B-3                                   00124   03044  0631463
        DEC 50E-3B-3                                    00125   03045  0631463
        DEC .0000E3B-3                                  00126   03046  0631463
        DEC .05                 B19 YIELDS ZERO         00127   03047  0000000
        DEC 524288              ERROR, TOO LARGE        00130 A 03050  0000000
*                                                       0
*       DDC PSEUDO CODE                                 0
*                                                       0
        DDC 3.1415926B19                                00131   03051  0000003
                                                        00132   03052  0220773
        DDC 31415926E-7B19                              00133   03053  0000003
                                                        00134   03054  0220773
        DDC -10000000B30                                00135   03055  3766355
                                                        00136   03056  2300000
        DDC 3.4567 E-11 B-29                            00137   03057  0023001
                                                        00140   03060  1374000
*                                                       0
*       OCT PSEUDO CODE                                 0
*                                                       0
        OCT 1234567                                     00141   03061  1234567
        OCT 123                                         00142   03062  0000123
        OCT -123                ERROR, - NOT PERMITTED  00143 A 03063  0000000
        OCT 1198                ERROR, NOT OCTAL        00144 A 03064  0000011
        OCT 7654321             ERROR, TOO LARGE        00145 A 03065  3654321
        OCT /123                ERROR, / NOT REQUIRED   00146   03066  0000000
*                                                       0
*       ALF PSEUDO CODE                                 0
*                                                       0
        ALF ABC                                         00147   03067  0616263
        ALF  DE                                         00150   03070  0006465
```

```
        ALF   F                                   00151    03071  0000066
        ALF ABCD                ERROR, TOO MANY   00152 A 03072  0616263
        ALF    E                ERROR, TOO MANY   00153 A 03073  0000000
*                                                 0
*       END PSEUDO CODE                           0
*                                                 0
        END START               END OF ASSEMBLY   00154          6101000
```

```
* 6.                SAMPLE PAP   PROGRAM TO ILLUSTRATE   0
*                        PRIME   FEATURES OF PAP          0
*                                                         0
        SLC /2000                                        00001          4002000
START   OFF N                   TURN OFF N-REGISTER      00002   02000  2500010
        BBR 2,N                   PERIPHERALS.           00003   02001  2516010
        BRU *-1                                          00004   02002  2602001
        LDA SELCD               TURN ON TYPER.           00005   02003  0002112
        SAB N,7                                          00006   02004  2400407
        SEL N                                            00007   02005  2500000
        BBR 2,N                 WAIT FOR SELECTION.      00010   02006  2516010
        BRU *-1                                          00011   02007  2602006
        LDA WAIT                INITIATE 200 MSEC. DELAY 00012   02010  0002113
        LTC 3                     FOR OPERATING SPEED.   00013   02011  2500021
        BTC 2,3                                          00014   02012  2516015
        BRU *-1                                          00015   02013  2602012
        LDX ZERO              1 ZERO WORD COUNT.         00016   02014  0622114
WORD    LDX ZERO              2 ZERO CHARACTER COUNT.    00017   02015  0642114
        LDA HEAD              1 GET A WORD AND POSITION  00020   02016  0022131
        SRD 18                    IT IN Q.               00021   02017  2400122
CHAR    BBR 2,N                 WAIT FOR LAST TYPER      00022   02020  2516010
        BRU *-1                   ACTION TO FINISH.      00023   02021  2602020
        SLD 6                   SHIFT NEXT CHARACTER TO  00024   02022  2411006
        SAB N,7                   A AND THEN TO N AND    00025   02023  2400407
        TYP N                     INITIATE ACTION.       00026   02024  2500004
        INX 1                 2 INCREASE CHARACTER COUNT 00027   02025  1440001
        BXL 3                 2   BY 1 AND TEST. IF NOT  00030   02026  0457775
        BRU CHAR                  YET 3, RETURN FOR NEXT 00031   02027  2602020
        INX 1                 1 INCREASE WORD COUNT BY 1 00032   02030  1420001
        BXL 51                1   AND TEST. IF NOT DONE  00033   02031  0437715
        BRU WORD                  RETURN FOR NEXT WORD.  00034   02032  2602015
        LDZ                     CLEAR A AND              00035   02033  2504002
        STA X                     SET X TO ZERO.         00036   02034  0302126
CALC    LDA B                   GET B AND MOVE IT TO Q   00037   02035  0002125
        MAQ                       FOR MULTIPLICATION.    00040   02036  2504006
        LDA A                   GET A AND CALCULATE Y    00041   02037  0002124
        MPY X                     AS  A + BX.            00042   02040  1502126
        DST Y                   STORE Y.                 00043 A 02041  1302127
        LDX ZERO              2 SET X/Y KEY TO X.        00044   02042  0642114
TYPEY   LDX ZERO              3 SET DIGIT INDEX TO ZERO. 00045   02043  0662114
        LDA VALUE             2 GET X OR Y AND MOVE IT   00046   02044  0042126
        MAQ                       TO Q FOR DIVISION.     00047   02045  2504006
TYPE    DVD TENX              3 GENERATE NEXT DIGIT IN A 00050   02046  1662115
        BZE 1                                            00051   02047  2514002
        BRU *+3                 IF IT IS NOT ZERO, STORE 00052   02050  2602053
        STA SUPCD                 IT AS SUPPRESSION KEY. 00053   02051  0302122
        BRU *+4                 IF IT IS ZERO, EXAMINE   00054   02052  2602056
        LDA SUPCD                 SUPPRESSION KEY.       00055   02053  0002122
        BZE 2                   IF ZERO IS TO BE TYPED,  00056   02054  2516002
        LDA ZEROC                 GET ZERO CODE.         00057   02055  0002123
        BBR 2,N                 WAIT FOR LAST TYPER      00060   02056  2516010
        BRU *-1                   ACTION TO FINISH.      00061   02057  2602056
        SAB N,7                 TYPE ZERO OR SPACE OR    00062   02060  2400407
        TYP N                     THE NON-ZERO DIGIT.    00063   02061  2500004
        INX 1                 3 INCREASE DIGIT INDEX.    00064   02062  1460001
```

```
        BXL 4                  3 IF LESS THAN FOUR, GO     00065   02063   0477774
        BRU TYPE                   BACK TO TYPE NEXT.      00066   02064   2602046
        BXL 5                  3 INSURE THAT SUPPRESSION   00067   02065   0477773
        LDO                        CODE IS NON-ZERO FOR    00070   02066   2504022
        STA SUPCD                  5TH AND ZERO FOR 1ST.   00071   02067   0302122
        BXL 5                  3 IF 5TH DIGIT NOT TYPED,   00072   02070   0477773
        BRU TYPE                   GO TO TYPE IT.          00073   02071   2602046
        INX 1                  2 INCREASE X/Y KEY.         00074   02072   1440001
        BXL 2                  2 IF NOW SET TO Y, GO TO    00075   02073   0457776
        BRU TYPEY                  TYPE Y VALUE.           00076   02074   2602043
        LDA CARRT                TYPE A CARRIAGE RETURN    00077   02075   0002130
        BBR 2,N                    AT END OF EACH LINE     00100   02076   2516010
        BRU *-1                    OF TABLE.               00101   02077   2602076
        SAB N,7                                            00102   02100   2400407
        TYP N                                              00103   02101   2500004
        LDX X                  1 GET LAST X VALUE.         00104   02102   0622126
        BXH 20                 1 IF IT IS 20, GO TO TURN   00105   02103   0537754
        BRU *+4                    OFF TYPER. IF NOT,      00106   02104   2602110
        INX 1                  1 INCREASE X BY 1,          00107   02105   1420001
        STX X                  1   STORE IT, AND GO TO    00110   02106   1722126
        BRU CALC                   CALCULATE NEXT Y.       00111   02107   2602035
        OFF N                    IF FINISHED, TURN OFF     00112   02110   2500010
        BRU *                      TYPER AND STOP.         00113   02111   2602111
N       EQL N                    DEFINE N FOR N-REGISTER   00114
SELCD   OCT 60                   TYPER SELECTION CODE      00115   02112   0000060
WAIT    OCT 14                   200 MSEC. CONSTANT.       00116   02113   0000014
ZERO    OCT 0                    DEFINE ZERO               00117   02114   0000000
TENX    DEC 1E4B19               TABLE OF POWERS OF TEN    00120   02115   0023420
        DEC 1E3                    FOR GENERATING BCD      00121   02116   0001750
        DEC 100                    NUMBERS.                00122   02117   0000144
        DEC 10                                             00123   02120   0000012
        DEC 1                                              00124   02121   0000001
SUPCD   OCT 0                    ZERO SUPPRESSION KEY      00125   02122   0000000
ZEROC   OCT 20                   ZERO CODE                 00126   02123   0000020
A       DEC 200                  INTERCEPT CONSTANT        00127   02124   0000310
B       DEC -10                  SLOPE CONSTANT            00130   02125   3777766
        SLC E                    TO ENABLE DST Y IN ODD    00131
X       SLC +1                   STORAGE FOR X             00132           6000001
Y       SLC +1                   STORAGE FOR Y             00133           6000001
CARRT   OCT 100                  CARRIAGE RETURN CODE      00134   02130   0000100
VALUE   EQL X                                              00135
HEAD    OCT 3003235              TABLE OF BCD CHARACTERS   00136   02131   3003235
        ALF THE                    FOR TYPING OUT THE      00137   02132   0237065
        ALF  FO                    HEADING INFORMATION,    00140   02133   0006646
        ALF LLO                    STARTING WITH A         00141   02134   0434346
        ALF WIN                    CARRIAGE RETURN.        00142   02135   0267145
        ALF G T                                            00143   02136   0670023
        ALF ABL                                            00144   02137   0616243
        ALF E I                                            00145   02140   0650071
        ALF S G                                            00146   02141   0220067
        ALF ENE                                            00147   02142   0654565
        ALF RAT                                            00150   02143   0516123
        ALF ED                                             00151   02144   0656400
        ALF FRO                                            00152   02145   0665146
        ALF M T                                            00153   02146   0440023
        ALF HE                                             00154   02147   0706500
        ALF EQU                                            00155   02150   0655024
        ALF ATI                                            00156   02151   0612371
        ALF ON                                             00157   02152   0464500
        OCT 3003235                                        00160   02153   3003235
        OCT 3000000                                        00161   02154   3000000
        ALF                                                00162   02155   0000000
        ALF                                                00163   02156   0000000
        ALF                                                00164   02157   0000000
        ALF Y E                                            00165   02160   0300065
```

```
ALF QUA                                   00166   02161   0502461
ALF LS                                    00167   02162   0432200
ALF 200                                   00170   02163   0022020
ALF  -                                    00171   02164   0004000
ALF 10X                                   00172   02165   0012027
OCT 3003235                               00173   02166   3003235
OCT 3006646                               00174   02167   3006646
ALF R V                                   00175   02170   0510025
ALF ALU                                   00176   02171   0614324
ALF ES                                    00177   02172   0652200
ALF OF                                    00200   02173   0466600
ALF X I                                   00201   02174   0270071
ALF N S                                   00202   02175   0450022
ALF TEP                                   00203   02176   0236547
ALF S O                                   00204   02177   0220046
ALF F 1                                   00205   02200   0660001
ALF  FR                                   00206   02201   0006651
ALF OM                                    00207   02202   0464400
ALF O T                                   00210   02203   0200023
ALF O 2                                   00211   02204   0460002
ALF O.                                    00212   02205   0207300
OCT 3003235                               00213   02206   3003235
OCT 3000000                               00214   02207   3000000
ALF   X                                   00215   02210   0000027
ALF                                       00216   02211   0000000
ALF Y                                     00217   02212   0300000
OCT 3003235                               00220   02213   3003235
END START              END-TRANSFER CARD  00221           6102000
```

B. SUBROUTINE PROGRAMMING

1. Introduction

Subroutines are one of the most powerful and convenient tools available to the programmer. When constructing the logical sequencing required in a program, it is often discovered the same general function has occurred several times during the entire program. If it were possible to transfer program control from the main sequence to a secondary sequence, and later return to the same point in the main sequence, the recurring function would only require one set of instructions to accomplish any number of performances of the specific function. The series of Common instructions which accomplish the desired function is designated a "sub-routine".

To cite an example, consider the solution of:

$$X = a\sqrt{b} + c\sqrt{d}$$

The function of determining the square-root of a value occurs twice within the same equation. Through the use of a subroutine to determine square-root, one set of instructions is required to accomplish the function each time square-root occurs in the entire main sequence. The functions which can be accomplished by the subroutine technique need only satisfy one condition. That is-that the instructions required to accomplish the

function be contained in a definable block. These functions would include, but are not restricted to, basic arithmetic sequences, complex mathematical evaluations, information and data transfers, or input-output routines. By employing the subroutine principle, a programmer may develop a "building-block" program, with each block performing a separate phase of the main program. The main program sequence is then only required to provide the necessary direction, intermediate preparation, and linkages (often designated calling sequences) between various subroutine sections of the program.

In order to accomplish the demands of the main program, the programmer must include within each subroutine certain basic requirements. It is of primary importance that the logical power of the subroutine be universal enough to handle any conceivable situation presented by the main program. The subroutine must further provide for a single common entrance, but be able to interpret and accomplish a variable exit for returning to the main program. The subroutine must provide the result(s) to be used by the main program in the form and location specified by the main program. If the subroutine must use registers or other components which may contain information necessary for subsequent operations of the main program, these components must be restored to their original form prior to reentry into the main program.

For correct and useful application of a subroutine, the main program must likewise satisfy certain minimum requirements. The main program must call for the correct entrance location to activate the subroutine. The main program must specify the desired reentry point or subroutine exit location. Finally, the main program must provide all information required by the subroutine to accomplish the function. This information, often referred to as the argument, is generally supplied in a specific form and location. The extent and detail of the information required by the subroutine will vary with each subroutine, but the requirements for supplying this information lie with the main program.

In summary, the use of subroutines and "building-block" subroutine techniques can provide the programmer a considerable saving of memory space and programming time and effort. The very slight expense in additional complexity and space of subroutine linkages or calling sequence are generally outweighed by the savings in memory space and programming effort. However, the specific use of subroutines within any given program will depend on the programmer's experience and ingenuity to a much greater extent than the inherent requirements of the program itself. The use of subroutines almost always make the execution time longer than normal programming.

2. Use of Subroutines

The use of subroutines can best be explained by using an example. Consider sensing the analog signal from a differential pressure sensor with the GE 412, and using the output of the sensor to compute a flow rate. The sensor output, 0-200 inches of water, corresponds to 0-500,000 pounds per hour water flow. The relationship between inches of water sensed and pounds per hour flow is shown in the following equation:

$$W = K_1 \cdot K_2 \cdot (\Delta P)^{1/2}$$

where:

W = Flow (lbs/hr)

ΔP = Differential pressure (inches of H_2O)

K_1 = Orifice Constant (determined by calibration) = 49,890

K_2 = Correction coefficient = $\dfrac{P_{actual}}{P_{calibration}} \times \dfrac{T_{calibration}}{T_{actual}}$

P = Pressure

T = Temperature

The output of the differential pressure sensor (0-200 inches of water) is converted via a transducer to 4-20 milliamps. This signal is then passed through a 100 ohm resistor and converted to a 4/10-2 volt signal. This voltage is then sensed on the 0-2 volt range of the scanner-distributor and converted to 800-4000 counts. The equation relating counts to inches of water is then:

$$\Delta P = (.0625) \text{ Counts} - 50.$$

The following program illustrates reading this sensor and using the reading to compute flow. It is assumed that K_2 has previously been computed from current temperature and pressure readings and stored in location 00400 with (B3).

The main program must satisfy the conditions of setting up the proper linkage to the subroutine The subroutine in this case is used to compute the square-root function. This linkage is specified as:

Entry Conditions:

1. 12-bit argument in the A register, B12

2. Exit address in location 0003

Exit Conditions:

1. Square-root of the argument is in A, B6

Subroutine Program

A polynominal of the form AN^2+BN+C is used by the subroutine to approximate the square-root of numbers that have even scale factors. The method used is as follows:

Consider:

$$n = 2^B N$$

where:

n = any number with an even scale factor
B = scale factor of n
$1/4 \leq N < 1$

The approximation is accomplished using N; and the scale factor for the square-root is $\frac{B}{2}$.

If:

$1/4 \leq N < 1/2$, A = -.563063
B = 1.24912
C = .223801

If:

$1/2 \leq N < 1$ A = -.187688
B = .866579
C = .321985

This approximation has a maximum error of ± .1% for 12-bit numbers.

LOCATION	OPERATION	OPERAND	TAG	COMMENTS
	LØC	500		DEFINE STARTING ADDRESS
MAIN	LDA	CØN1		LØAD CØMMAND INTØ
	LSC	1		SCANNER COMMAND REGISTER
	BSC	2, 1		WAIT FOR SCANNER
	BRU	*-1		COMPLETE
	RCV	1		READ IN COUNTS
	MAQ			PUT COUNTS INTO Q B19
	MPY	CØN2		(.25xCOUNTS) B18
	SLD	6		(.25xCOUNTS) B12
	SUB	CØN3		(.25xCOUNTS) -50 B12
	SPB	SQRØØT	1	GO TO SQUARE ROOT SUBROUTINE
	MAQ			SQRØØT OF DELTAP B6 IN Q
	MPY	K2		K1 x DELTAP B9
	MAQ			

LOCATION	OPERATION	OPERAND	TAG	COMMENTS
	MPY	K1		K2xK1xDELTAP B28
	SRD	10		B38
	XAQ			
	STA	FLØW		STORE FLOW IN STORAGE
	BRU	NEXT		GØ TØ NEXT PRØGRAM
CØN1	ØCT	0345316		SCANNER CØMMAND WØRD
CØN2	DEC	.0625B-1		CØNSTANT
CØN3	DEC	50B12		CØNSTANT
K1	DEC	49890		ØRIFICE CØNSTANT B19
K2	EQØ	400		CØRRECTIØN CØNSTANT B3
FLØW	DEC	0		

3. Square Root Routine

LOCATION	OPERATION	OPERAND	TAG	COMMENTS
*				SQUARE RØØT SUBRØUTINE
	LØC	1000		ASSIGN STARTING ADDRESS
SQRØØT	STX	XLØC2	2	SAVE CØNTENTS ØF X LØCATION 2
	NØR	18		NØRMALIZE ARGUMENT
	MAQ			PUT IT IN Q
	LDA	0		TEST NØ. ØF LEADING ZERØ
	BEV	2		
	BRU	SQRTA		IF ODD THEN N LESS THAN 1/2
	LDX	ZERØ	2	IF EVEN THEN N GREATER THAN 1/2
SQRTB	XAQ			PUT NØRMALIZED ARGUMENT IN A
	SRA	0	2	MAKE ARG. BE . 1XX ØR .01XX
	STA	TEMP		STØRE ARGUMENT TEMPØRARILY
	MAQ			PUT ARGUMENT IN Q (B0)
	MPY	CØN	2	(AXN) (B4)
	ADD	CØN+2	2	((AXN)+B) (B4)
	MAQ			
	MPY	TEMP		((AXN)+B)X N (B4)
	SLD	4		(B0)
	ADD	CØN+4	2	((AN+B)N)+C (B0)
	STA	TEMP		STØRE TEMPØRARILY
	LDA	CON+6		(18) DECIMAL (B19)
	SUB	0		NUMBER ØF LEADING ZERØS IN A
	SRA	1		DIVIDE BY 2 PAIRS ØF LEADING O'S
	STA	2		PUT NØ. ØF PAIRS ØF O'S IN XL2
	LDA	TEMP		SQRØØT ØF N (B0)
	SRA	0	2	SQRØØT ØF N (B=1/2 ØF ØRIG)
	LDX	XLØC2	2	RESTØRE CØNTENTS ØF X LØCATIØN 2
	BRU	1	1	EXIT BACK TØ MAIN PRØGRAM
SQRTA	LDX	ØNE	2	SET KEY TØ 1
	BRU	SQRTB		
CØN	DEC	-.187688 B4		A CØNSTANTS
	DEC	-.563063 B4		A
	DEC	.866579 B4		B
	DEC	1.24912 B4		B
	DEC	.321985 B0		C
	DEC	.223801 B0		C
	DEC	18		(18) B19
ØNE	DEC	1		(1) B19
ZERØ	DEC	0		(0)
XLØC2	DEC	0		

C. REAL-TIME PROGRAMMING

Process computing applications require the computer to operate on a real-time basis; sensing variables, correlating these variables with standard conditions, and applying control as the process dynamically operates. There can be very little delay between the sensing and the application of control. The functions of scanning, monitoring, logging, performance calculations, and controlling must be accomplished by the program in the available computing time without falling behind the operation of the process. Automatic program interrupt, elapsed time counters, and the digital clock are the basis of real-time programming in the GE 412 process computing system. This computer hardware and a program, called the Executive Control Program (ECP), coordinate the use of available computing time by all of the functional programs that do the work in the system. A typical system flow chart is shown in figure 15.

1. Automatic Program Interrupt

The automatic program interrupt system consists of twelve levels of interruption, each corresponding to a position in the automatic program interrupt register. Any condition that can be related to an open or closed contact (bistable) can be used to cause program interruption. The overflow condition from an elapsed time counter, the complete signals from input-output equipment, demand push-buttons, limit switches, limit sensors, and other process equipment may be used as conditions that cause automatic program interruption. The exact employment of the twelve levels of interruption will vary from system to system, but the programming approach to coordinating the many asynchronous happenings in a process computer system is fairly uniform.

2. Elapsed Time Counters and the Digital Clock

The four elapsed time counters in the GE 412 system are capable of accumulating increments of time and signalling the program upon completion of the count. This communication is done by setting an indicator that the program can branch on, or by using the overflow signal from the particular time counter to cause automatic program interrupt directly. The elapsed time counters are normally employed by the program for controlling scan frequencies, logging frequencies, and many other time dependent functions. The elapsed time counters are also used as an integral part of the executive control program to control the effective use of the available computing time in the GE 412 System. Periodically the ECP reads the digital clock to reference time to the hour, minute and second of true time.

3. The Executive Control Program (ECP)

The executive control program is the executor of "real-time" in the process control program. It governs the sharing of available computing time by all of the functional programs within the process control system. Since the importance of the functional programs vary, a priority is assigned by the ECP along with a frequence of execution. The ECP works directly with the elapsed time counters, digital clock, and the automatic program interrupt system to effective control the frequency of execution and priority of all functional programs.

4. Functional Programs

Each process control program consists of a variable number of functional routines that accomplish the scanning, alarming, monitoring, performance evaluation, operator demand, and controlling functions. Each of these programs accomplishes a finite part of the overall process control program. The number and complexity of the functional routines varies greatly from one application to another. The ECP is responsible for the proper entry into these routines at the right time, and if a functional program happens to be interrupted by a higher priority routine, the ECP must also re-enter the interrupted routine at the exact position of interruption. Functional routines are therefore written as a straight line program and the automatic interrupt will interrupt it, remembering where to re-enter, so that higher priority routine may take precedence.

5. Simplified Process Computing System Program

To illustrate the concepts of real-time programming for the GE 412 Computer System a simplified process computing system program is described herein. It is representative of approximately 5% of a true system program, however, it displays all of the real-time characteristics of a true system program. The system described scans, monitors, alarms, logs, and does an on-demand display for 100 analog inputs. The general system flow chart is shown in figure 15. As shown by this diagram elapsed time counters, scanner

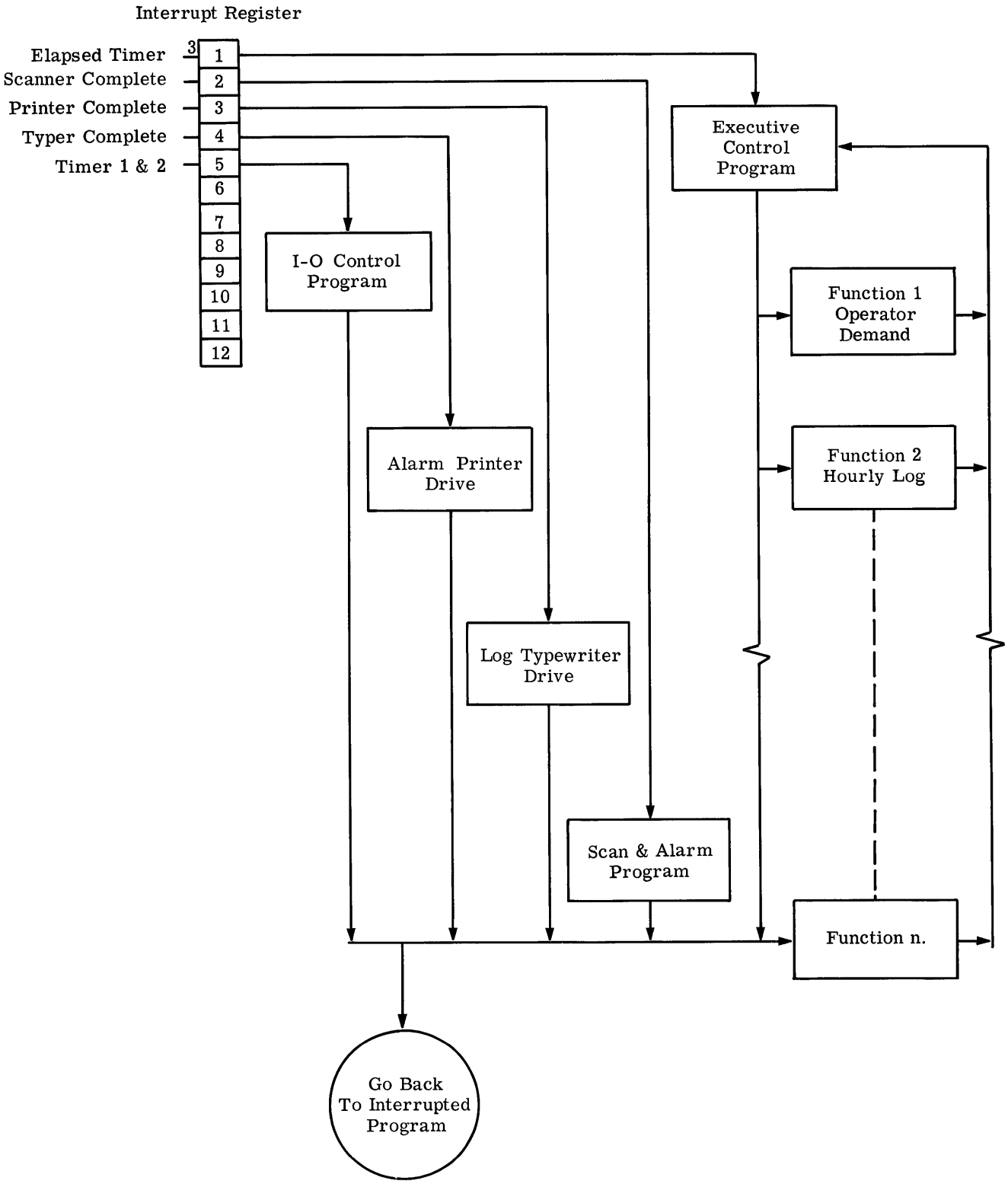


Figure 15. Basic System Flow Chart

complete signals, alarm printer ready, and log typewriter ready signals are used to cause interrupt. The coordination of input and output of information is done using programs, called drivers, which simply keep the peripheral device busy, if there is something for it to do. The time counter interrupts are used to coordinate time in the system. As described previously the ECP controls of use of computer time by the functional routines. Only two functional routines are specified in the system, but normally a true system program would contain many functional routines.

a. Executive Control Program (ECP)

The basis of the operation of the ECP is an interruption from a time counter that enters the ECP once every 500 milliseconds (could be less if necessary). Upon entry the ECP saves the contents of all registers applicable to the interrupted program in an area in storage set aside for that particular program, so that it may properly re-enter the program when time becomes available. The ECP then updates a relative count of time in storage. This relative time is then compared with the time at which the functional programs are to be executed in order of priority, starting with priority one and descending through all priorities. If no functional routine is required the ECP simply waits for time to pass. Finding a functional routine that is required, the ECP loads the registers for that routine and re-enters it either at the beginning or, if this routine was interrupted, at the position of interruption. A program number associated with the operating program is kept in storage so that the ECP always knows what program is running. The program number of the ECP is "O", functional routine 1 is numbered "1" and so on. This program number is used by the ECP to store the register contents in the proper place in storage. Figure 16 shows a flow chart for the ECP.

b. Scan Program

The scan program drives the scanner-distributor at full speed scanning analog inputs. Inputs are read-in, stored in a table, checked against high and low limits and alarmed on the alarm printer in read when the input is out of limits the first time. If the input is within limits, a check is made to see if it was out of limits last time. If it was, the point is again printed on the alarm printer in black. When the 100 analog input points have been scanned the scan program starts from the beginning and scans them again. If at any time an output is required to be performed by the scanner-distributor, the point being scanned is interrupted and the output is initiated. When the output is completed, the program that required the output must then re-initiate the scan of the point that was interrupted.

(1) The Engineering Conversion Subroutine

The engineering conversion subroutine converts the raw counts signal from the scanner into engineering units using the equation indicated by the index table. Conversion is accomplished using a second order polynomial and 32 sets of coefficients. (Simplified from true system program.)

(2) Alarm Assembly Subroutine

The alarm assembly subroutine reads the digital clock, converts the alarm value into binary-coded-decimal, and places all of this information along with the alarm indication (high, low, etc.) in a queue table for the alarm printer drive program. If the printer is inactive the alarm assembly routine turns the alarm printer on and initiates a 200 millisecond delay to allow the printer to attain operating speed.

(3) The Alarm Printer Drive Program

The alarm printer drive program in an interrupt program that simply gets the next item to be printed out of an alarm queue table and initiates the printing of it. If the queue table is empty, it then turns the power off on the alarm printer. The time is printed only once every minute while alarms are being printed.

(4) Input-Output Control

The input-output control routine allows time counter to be used in delaying the use of peripherals until they are up to operating speed. As the delay expires to input-output control routine start the action required by transferring to the appropriate driver.

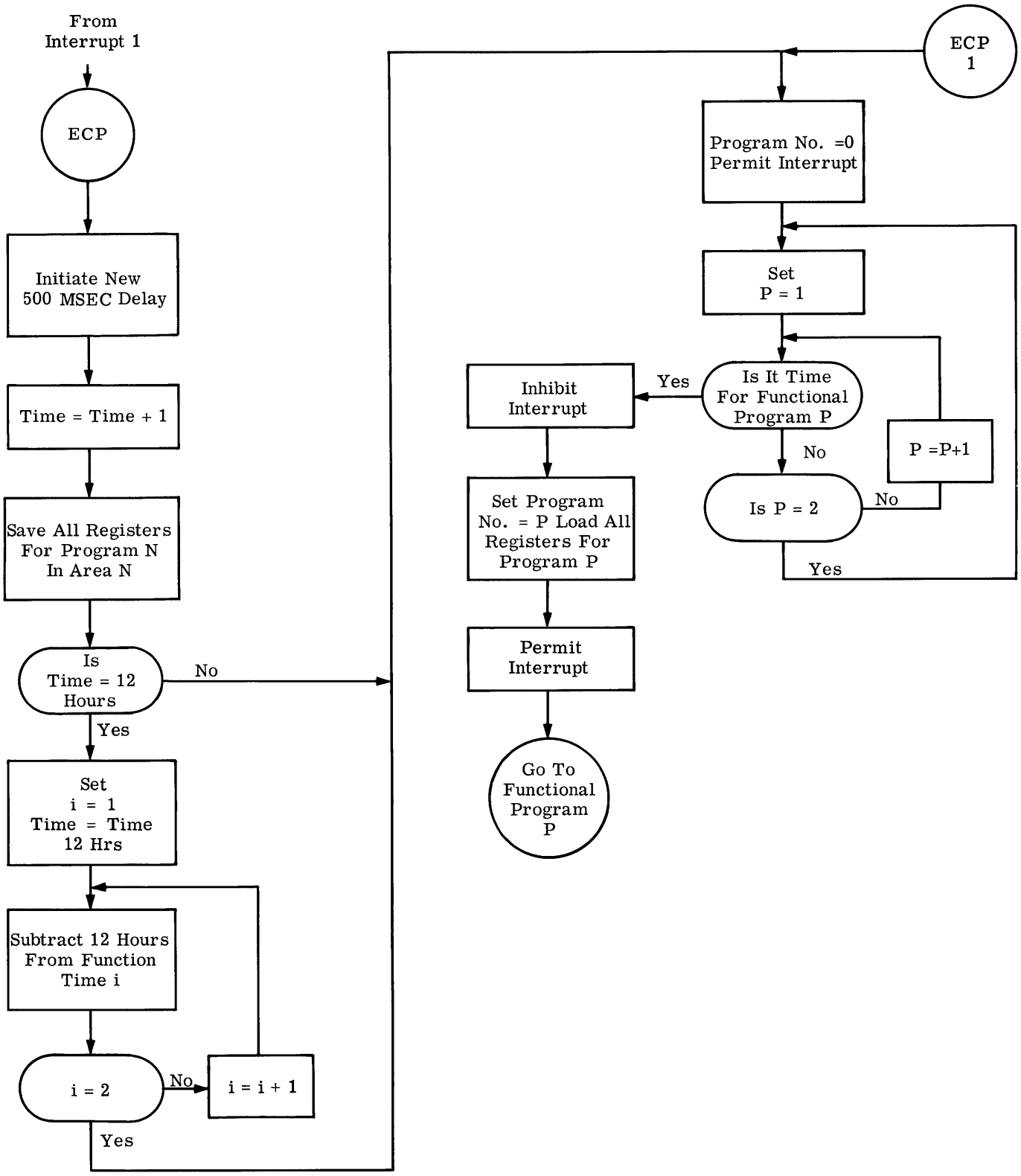


Figure 16. Executive Control Program

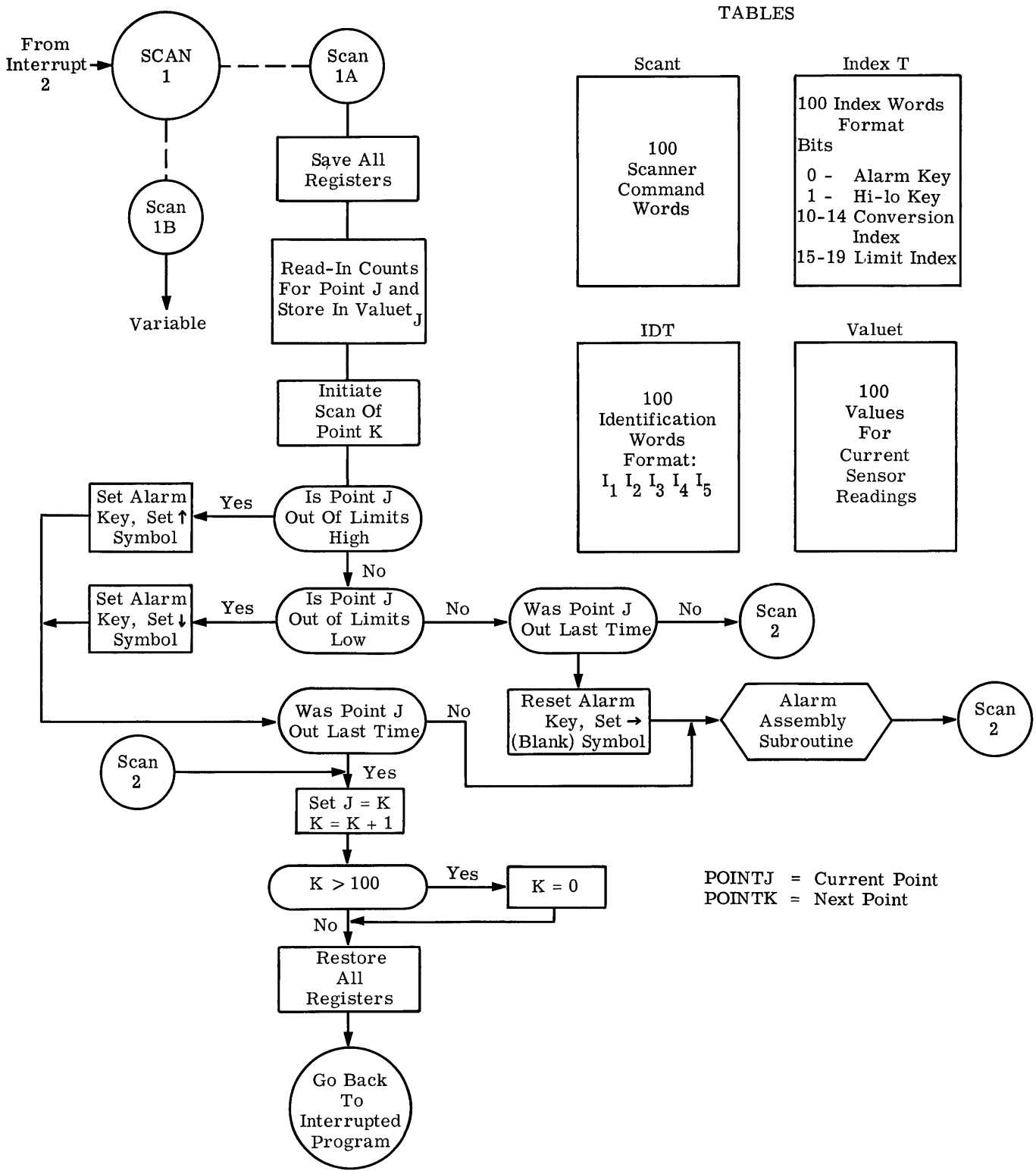


Figure 17. Scan Program

ALARM QUEUE TABLE

AQUE1	AQUE2	AQUE3	AGUE4	
Format: Identification	Format: Identification	Format: Value	Format: Time	1
Color $I_1I_2I_3$	I_4I_5 — ↑	$V_1V_2V_3V_4$	$T_1T_2T_3T_4$	20

Four Words Per Item

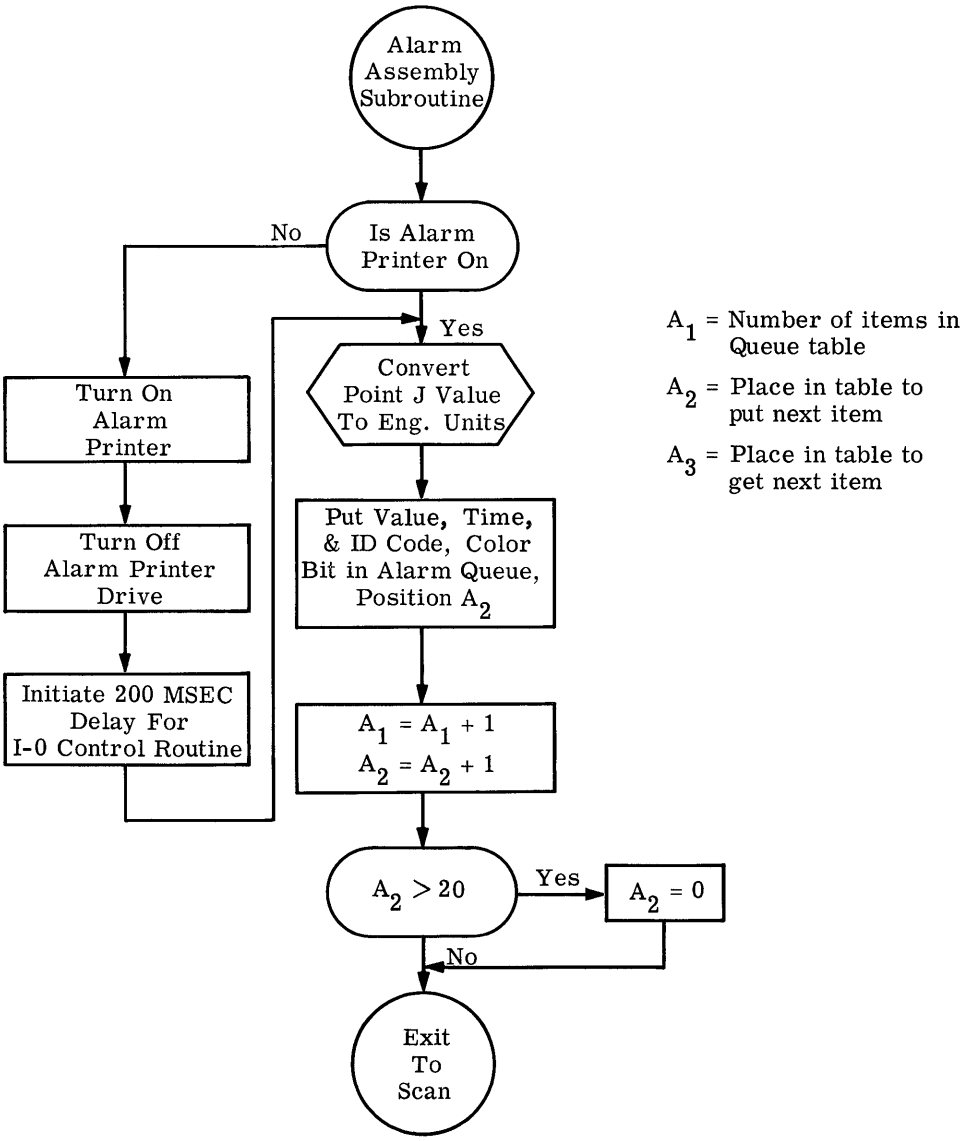


Figure 18. Alarm Assembly Routine

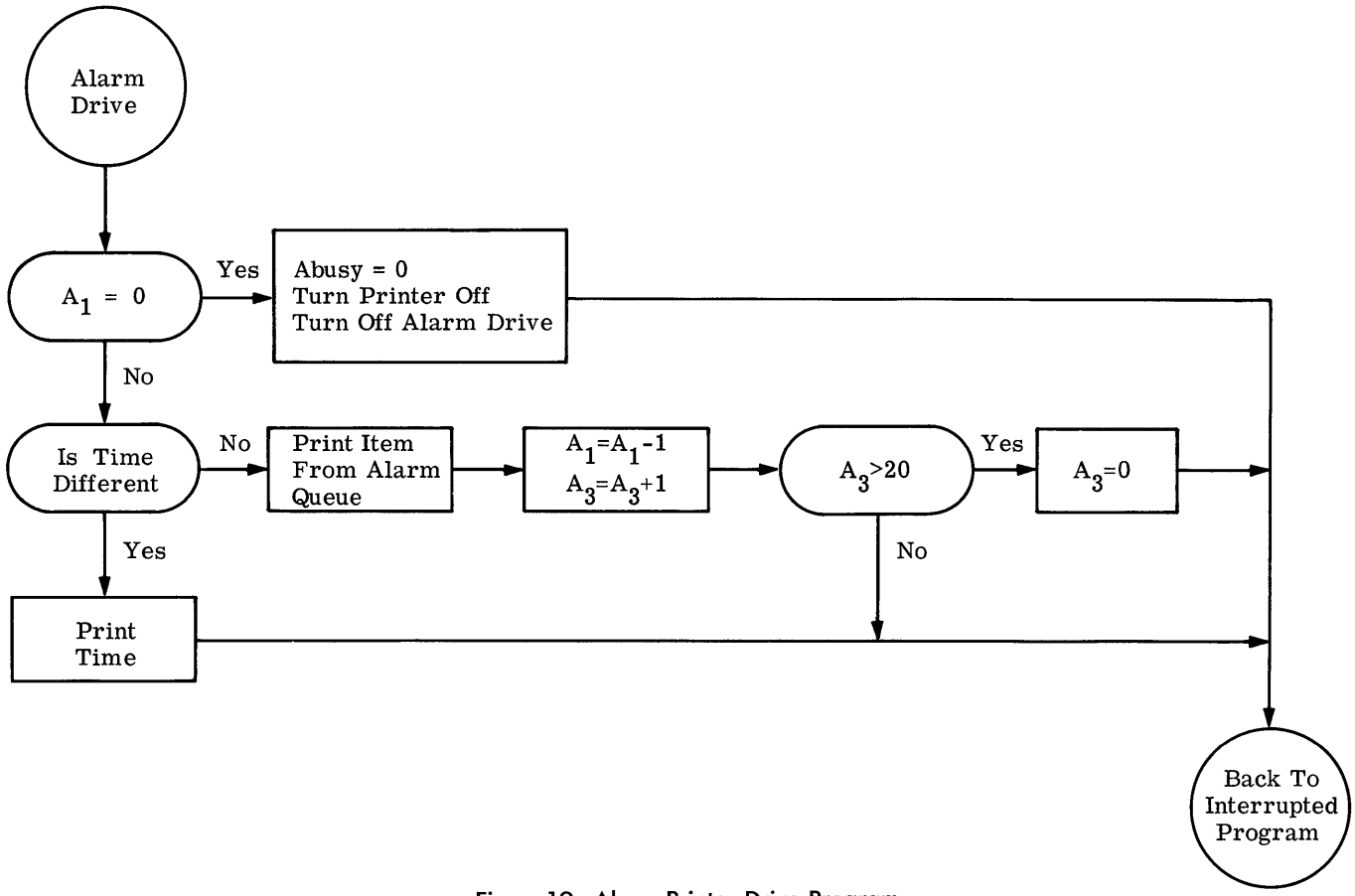


Figure 19. Alarm Printer Drive Program

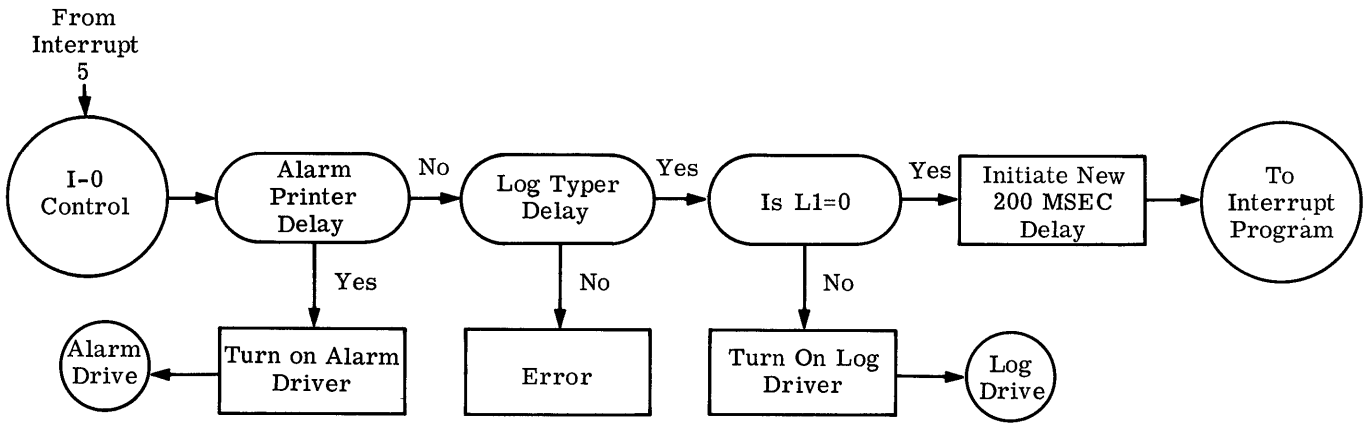


Figure 20. Input-Output Control

c. The Demand Program

The demand program interrogates a demand push-button once every second and if a demand is indicated, initiates the action called for. Upon finding a request, the demand program reads the decade switches on the operators console to get the indentification number of the analog input point. The current reading for that point is then displayed on the digital display of the operators console. The demand indicator is then turned off, and the analog input that was interrupted is re-initiated.

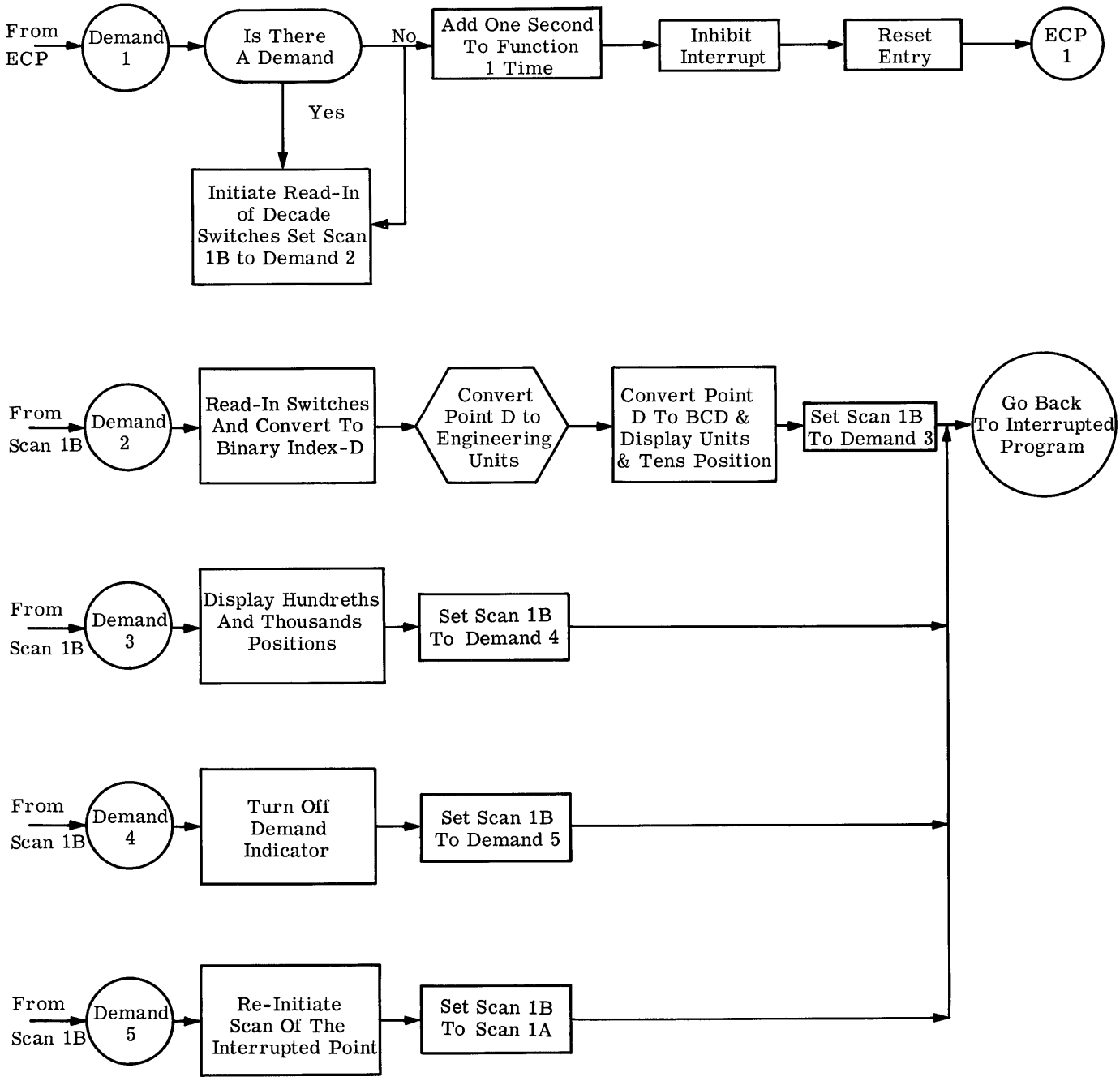


Figure 21. Demand Program

d. Log Program

Once every hour the ECP enters the Log Program. The 100 current analog input readings are converted into engineering units and placed in a log value table. The values are taken out of this table one at a time, converted to binary-coded-decimal with leading zero suppression, and stored character at a time in a log typewriter queue table. When the table becomes full the log program waits for the log drive program to take some of the characters out before putting any more in. The log drive program is an interrupt program that sends one character at a time to the typewriter from the log character queue table. When the drive program has typed all of the characters it then types a carriage return, and turns the typewriter off.

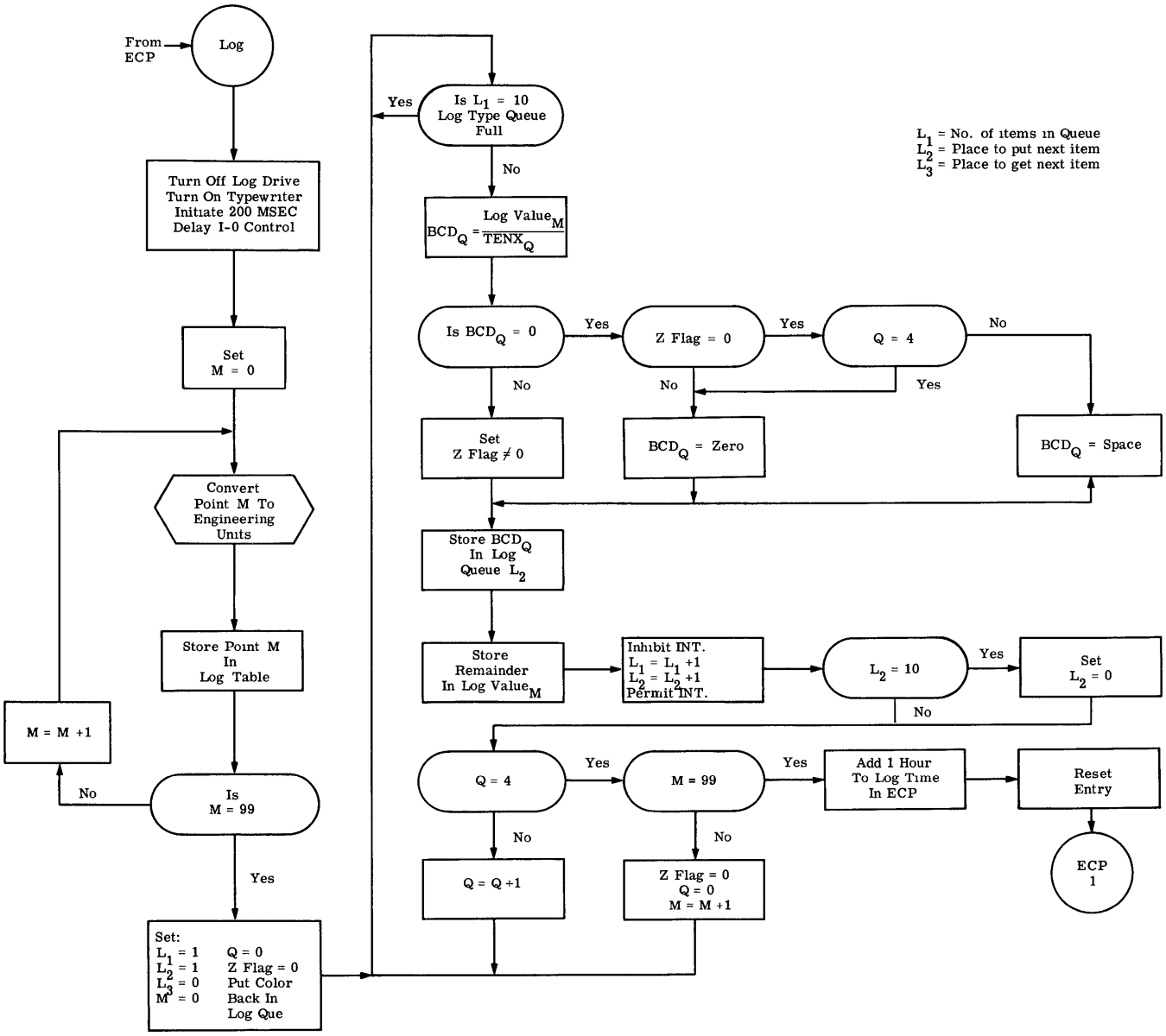


Figure 22. Log Program

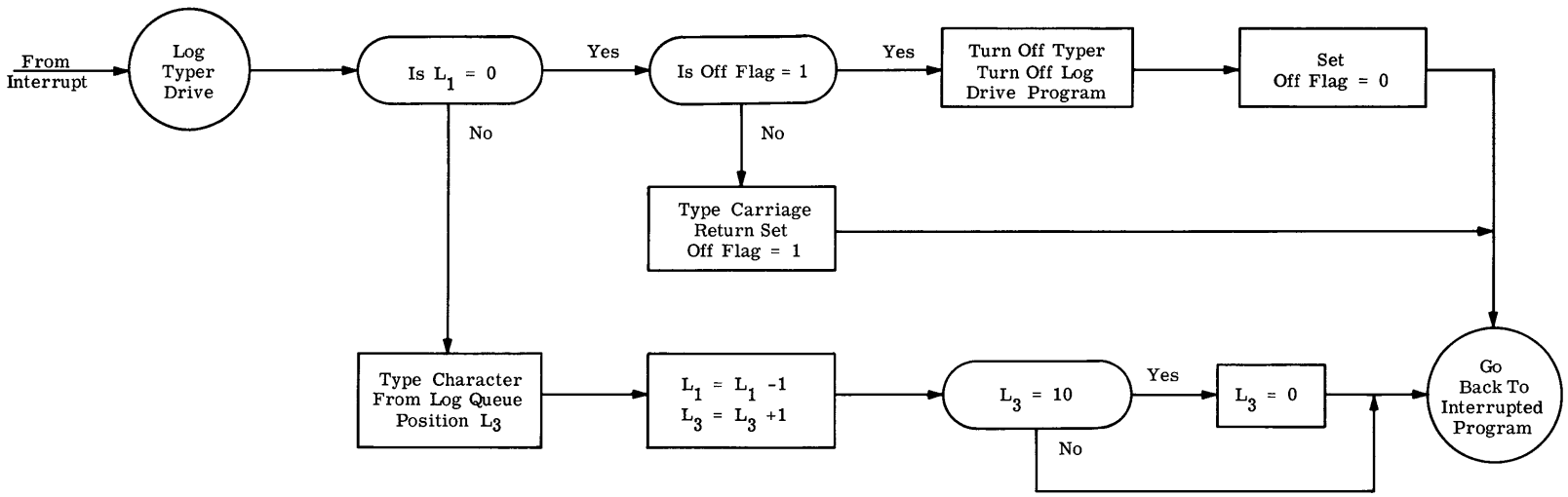


Figure 23. Log Typer Drive Program

```
        SLC /1000                                   00001          4001000
ECP     DST SAVEAQ                                  00002   01000  1301076
        LDA ETCON3          ENTRY FROM INTERRUPT    00003   01001  0001137
        LTC 3               INITIATE 500 MSEC DELAY 00004   01002  2500021
        LDA TIME            ADD ONE TO              00005   01003  0001135
        ADO                 RELATIVE COUNT OF       00006   01004  2504032
        STA TIME            SYSTEM TIME             00007   01005  0301135
        DLD 0                                       00010   01006  1000000
        DST SAVE01                                  00011   01007  1301100
        LDA PROGNO          MULTIPLY PROGRAM NUMBER 00012   01010  0001136
        SLA 3               BY 8 TO INDEX REGISTER  00013   01011  2410003
        STA 1               STORAGE AREA            00014   01012  0300001
        DLD SAVEAQ          SAVE ALL                00015   01013  1001076
        DST REG           1 REGISTER FOR            00016   01014  1321102
        DLD SAVE01          PROGRAM N IN            00017   01015  1001100
        DST REG+2         1 REGISTER                00020   01016  1321104
        DLD 2               AREA N                  00021   01017  1000002
        DST REG+4         1                         00022   01020  1321106
        LDA 6                                       00023   01021  0000006
        STA REG+6         1                         00024   01022  0321110
        LDA TIME                                    00025   01023  0001135
        SUB CON864                                  00026   01024  0201141
        BZE 2                                       00027   01025  2516002
        BRU ECPX                                    00030   01026  2601037
        STA TIME                                    00031   01027  0301135
        LDX ONE           1 IT IS 12 HOURS          00032   01030  0621140
ECPW    LDA FTIME         1                         00033   01031  0021132
        SUB CON864                                  00034   01032  0201141
        STA FTIME         1                         00035   01033  0321132
        INX 1             1                         00036   01034  1420001
        BXL 3             1                         00037   01035  0437775
        BRU ECPW                                    00040   01036  2601031
ECPX    IAI                                         00041   01037  2500013
        LDZ                 SET PROGRAM NUMBER      00042   01040  2504002
        STA PROGNO          TO 0 FOR ECP            00043   01041  0301136
        LDX ONE           1 SET P=1                 00044   01042  0621140
        PAI                                         00045   01043  2500012
ECPB    LDA TIME            IS IT TIME FOR          00046   01044  0001135
        SUB FTIME         1 FUNCTIONAL ROUTINE P    00047   01045  0221132
        BPL 1                                       00050   01046  2514001
        BRU ECPA            IF SO GO TO ECPA        00051   01047  2601054
        INX 1             1 P=P+1                   00052   01050  1420001
        BXL 3             1 IS P=2                  00053   01051  0437775
        BRU ECPB            IF NOT REPEAT LOOP      00054   01052  2601044
        BRU ECPB-2          IF SO START LOOP AGAIN  00055   01053  2601042
ECPA    IAI                 INHIBIT INTERRUPT       00056   01054  2500013
        LDA 1                                       00057   01055  0000001
        STA PROGNO          SET PROGRAM NUMBER=P    00060   01056  0301136
        SLA 3                                       00061   01057  2410003
        STA 1                                       00062   01060  0300001
        DLD REG+4         1 LOAD ALL                00063   01061  1021106
        DST 2               REGISTERS               00064   01062  1300002
        DLD REG           1 FOR                     00065   01063  1021102
        DST SAVEAQ          PROGRAM                 00066   01064  1301076
        LDA REG+6         1 P                       00067   01065  0021110
        STO ECPC                                    00070   01066  2701074
        DLD REG+2         1                         00071   01067  1021104
        DST 0                                       00072   01070  1300000
        DLD SAVEAQ                                  00073   01071  1001076
        SAI 3                                       00074   01072  2500016
        PAI                                         00075   01073  2500012
ECPC    BRU *               GO TO PROGRAM P         00076   01074  2601074
```

```
        SLC E                                          00077   01075  2504000
SAVEAQ  SLC +2                TEMP STORAGE FOR A AND Q 00100          6000002
SAVE01  SLC +2                TEMP STOR FOR X1 AND X2  00101          6000002
REG     SLC +24               REGISTER STORAGE BLOCK   00102          6000030
FTIME   SLC +3                                         00103          6000003
TIME    DEC 0                                          00104   01135  0000000
PROGNO  DEC 0                                          00105   01136  0000000
ETCON3  OCT 10036             CONSTANT FOR 500 MSEC    00106   01137  0010036
ONE     DEC 1                                          00107   01140  0000001
CON864  DEC 86400             COUNTS = 12 HOURS        00110   01141  0250600
*                                                      0
*                                                      0
*THE        SCAN PROGRAM                               0
*  ENTER FROM INTERRUPT 2                              0
SCAN    STX SAVEX1          1 SAVE                     00111   01142  1721223
        STX SAVEX2          2 ALL                      00112   01143  1741224
        STX SAVEX3          3 REGISTERS                00113   01144  1761225
        DST SAVEAQ                                     00114   01145  1301076
        LDX POINTJ          1                          00115   01146  0621230
        LDX POINTK          2                          00116   01147  0641231
        RCV 1                 READ-IN AND STORE        00117   01150  2510144
        STA VALUET          1 READING FROM POINTJ      00120   01151  0321235
        LDA SCANT           2 INITIATE SCAN OF         00121   01152  0041247
        LSC 1                 POINTK                   00122   01153  2510103
        LDA INDEXT          1 EXTRACT LIMIT INDEX      00123   01154  0021261
        ANA MASKA             FROM INDEX WORD          00124   01155  2201337
        STA 3                 AND STORE IN XL3         00125   01156  0300003
        LDA HIGHT           3 IS POINTJ                00126   01157  0061311
        SUB VALUET          1 OUT OF LIMITS            00127   01160  0221235
        BPL 2                 HIGH                     00130   01161  2516001
        BRU OUTHI                                      00131   01162  2601204
        LDA VALUET          1 IS POINTJ                00132   01163  0021235
        SUB LOWT            3 OUT OF LIMITS            00133   01164  0261305
        BPL 2                 LOW                      00134   01165  2516001
        BRU OUTLO                                      00135   01166  2601215
        LDA INDEXT          1 WAS POINTJ OUT OF        00136   01167  0021261
        BPL 2                 LIMITS LAST TIME         00137   01170  2516001
        BRU OLAST                                      00140   01171  2601220
SCANA   STX POINTJ          2 SET J=K                  00141   01172  1741230
        INX 1               2 K=K+1                    00142   01173  1440001
        BXH 10              2 IS K GREATER THAN 10     00143   01174  0557766
        LDX ZERO            2 IF SO SET K=0            00144   01175  0641226
        STX POINTK          2                          00145   01176  1741231
        LDX SAVEX1          1 RESTORE ALL              00146   01177  0621223
        LDX SAVEX2          2 REGISTERS                00147   01200  0641224
        LDX SAVEX3          3 AND GO BACK              00150   01201  0661225
        DLD SAVEAQ            TO THE                   00151   01202  1001076
        BRU 5                 INTERRUPTED PROGRAM      00152   01203  2600005
OUTHI   LDA INDEXT          1                          00153   01204  0021261
        ANA MASKB                                      00154   01205  2201227
        ADD CON1B1            SET ALARM INDICATOR TO   00155   01206  0101233
        STA INDEXT          1 HIGH                     00156   01207  0321261
        BPL 2                 WAS IT OUT LAST TIME     00157   01210  2516001
        BRU SCANA             IF SO BYPASS             00160   01211  2601172
        LDA CON2B0                                     00161   01212  0001234
        ORY INDEXT          1                          00162   01213  2321261
        BRU ALARM             IF NOT, GO TO ALM ROUTIN 00163   01214  2601345
OUTLO   LDA INDEXT          1 SET ALARM INDICATOR      00164   01215  0021261
        ANA MASKB             TO LOW                   00165   01216  2201227
        BRU OUTHI+3                                    00166   01217  2601207
OLAST   ANA MASKC             IF NOT SET OUTLAST INDEX 00167   01220  2201232
        STA INDEXT          1 AND ALARM                00170   01221  0321261
```

```
        BRU ALARM                                   00171   01222   2601345
SAVEX1  DEC 0                                       00172   01223   0000000
SAVEX2  DEC 0                                       00173   01224   0000000
SAVEX3  DEC 0                                       00174   01225   0000000
ZERO    DEC 0                                       00175   01226   0000000
MASKB   OCT 2777777                                 00176   01227   2777777
POINTJ  DEC 0                                       00177   01230   0000000
POINTK  DEC 1                                       00200   01231   0000001
MASKC   OCT 0777777                                 00201   01232   0777777
CON1B1  OCT 1000000                                 00202   01233   1000000
CON2B0  OCT 2000000                                 00203   01234   2000000
VALUET  SLC +10                 VALUE TABLE         00204           6000012
SCANT   OCT 0111030             SCANNER COMMAND TABLE 00205 01247   0111030
        OCT 0141030                                 00206   01250   0141030
        OCT 0151030                                 00207   01251   0151030
        OCT 0161030                                 00210   01252   0161030
        OCT 0112030                                 00211   01253   0112030
        OCT 0122030                                 00212   01254   0122030
        OCT 0132030                                 00213   01255   0132030
        OCT 0142030                                 00214   01256   0142030
        OCT 0121030                                 00215   01257   0121030
        OCT 0152030                                 00216   01260   0152030
INDEXT  OCT 0                   INDEX TABLE         00217   01261   0000000
        OCT 1                                       00220   01262   0000001
        OCT 2                                       00221   01263   0000002
        OCT 3                                       00222   01264   0000003
        OCT 3                                       00223   01265   0000003
        OCT 2                                       00224   01266   0000002
        OCT 1                                       00225   01267   0000001
        OCT 0                                       00226   01270   0000000
        OCT 3                                       00227   01271   0000003
        OCT 1                                       00230   01272   0000001
IDT     OCT 1                   IDENTIFICATION TABLE 00231  01273   0000001
        OCT 2                                       00232   01274   0000002
        OCT 3                                       00233   01275   0000003
        OCT 4                                       00234   01276   0000004
        OCT 5                                       00235   01277   0000005
        OCT 6                                       00236   01300   0000006
        OCT 7                                       00237   01301   0000007
        OCT 10                                      00240   01302   0000010
        OCT 11                                      00241   01303   0000011
        OCT 20                                      00242   01304   0000020
LOWT    DEC 500                 LOW LIMIT TABLE     00243   01305   0000764
        DEC 1000                                    00244   01306   0001750
        DEC 1500                                    00245   01307   0002734
        DEC 2000                                    00246   01310   0003720
HIGHT   DEC 1000                HIGH LIMIT TABLE    00247   01311   0001750
        DEC 2000                                    00250   01312   0003720
        DEC 3000                                    00251   01313   0005670
        DEC 3500                                    00252   01314   0006654
*                                                   0
*                                                   0
*       ENGINEERING             CONVERSION SUBROUTINE 0
*                               ARGUMENT IN A       0
*                               INDEX IN XL1        0
*                               RETURN IN XL3       0
CONVER  IAI                     INHIBIT INTERRUPT   00253   01315   2500013
        SLA 7                                       00254   01316   2410007
        STA COUNTS              B12                 00255   01317   0301340
        MAQ                     PUT COUNTS IN Q B12 00256   01320   2504006
        LDA INDEXT,1                                00257   01321   0021261
        SRA 5                                       00260   01322   2400005
```

```
        ANA MASKA                 0000037                    00261   01323   2201337
        STX CSAVE2            2                              00262   01324   1741341
        STA 2                                                00263   01325   0300002
        LDZ                                                  00264   01326   2504002
        MPY COEFA             2 (AX) B12                     00265   01327   1541342
        ADD COEFB             2 (AX)+B  B12                  00266   01330   0141343
        MAQ                                                  00267   01331   2504006
        MPY COUNTS                ((AX)+B)X  B24             00270   01332   1501340
        SLD 5                     ((AX)+B)X  B19             00271   01333   2411005
        ADD COEFC             2 (((AX)+B)X)+C=ENG UNITS      00272   01334   0141344
        LDX CSAVE2            2                              00273   01335   0641341
        BRU 1                 3 RETURN TO MAIN PROGRAM       00274   01336   2660001
MASKA   OCT 37                                               00275   01337   0000037
COUNTS  DEC 0                                                00276   01340   0000000
CSAVE2  DEC 0                                                00277   01341   0000000
COEFA   DEC 0                     A COEFFICIENTS B0          00300   01342   0000000
COEFB   DEC 1B12                  B COEFFICIENTS B12         00301   01343   0000200
COEFC   DEC 0                     C COEFFICIENTS B19         00302   01344   0000000
*                                                            0
*                                                            0
*       ALARM ASSEMBLY ROUTINE                               0
                                                             00303
ALARM   LDA ABUSY                 IS ALARM PRINTER           00304   01345   0001450
        BZE 2                     BUSY                       00305   01346   2516002
        BRU ALARMA                IF SO BYPASS               00306   01347   2601357
        OCT 2510410               TURN ON PRINTER SLH 1,1    00307   01350   2510410
        LDA CONBRU                TURN OFF PRINTER DRIVE     00310   01351   0001451
        STA 10                                               00311   01352   0300012
        LDA ETIME2                200 MSEC                   00312   01353   0001452
        LTC 1                     INITIATE DELAY             00313   01354   2500017
        LDQ                       SET ABUSY=1                00314   01355   2504022
        STA ABUSY                                            00315   01356   0301450
ALARMA  LDA VALUET            1 LOAD COUNTS                  00316   01357   0021235
        SPB CONVER            3 CONVERT COUNTS TO ENG UN     00317   01360   0761315
        MAQ                       PUT VALUE IN Q B19         00320   01361   2504006
        DVD CON1E3                                           00321   01362   1601453
        SLA 12                                               00322   01363   2410014
        STX ASAVE2            2 SAVE XL2                     00323   01364   1741456
        LDX A2                2                              00324   01365   0641460
        STA AQUE3             2 CONVERT                      00325   01366   0341540
        LDZ                       VALUE                      00326   01367   2504002
        DVD CON1E2                 TO                        00327   01370   1601454
        SLA 8                     BCD AND                    00330   01371   2410010
        ORY AQUE3             2    STORE                     00331   01372   2341540
        LDZ                         IN                       00332   01373   2504002
        DVD CON1E1                 ALARM                     00333   01374   1601455
        SLA 4                     QUEUE                      00334   01375   2410004
        ORY AQUE3             2                              00335   01376   2341540
        XAQ                                                  00336   01377   2504005
        ORY AQUE3             2                              00337   01400   2341540
        LDA IDT               1 GET ID                       00340   01401   0021273
        SRD 8                     POSITION I1,I2,I3          00341   01402   2400110
        ANA MASKD                 0007777                    00342   01403   2201461
        STA AQUE1             2 STORE  IN ALARM TABLE        00343   01404   0341470
        LDA INDEXT            1 TEST INDEX TO                00344   01405   0021261
        BPL 1                     SET COLOR OF               00345   01406   2514001
        BRU *+3                   PRINTING                   00346   01407   2601412
        LDA CON1B7                PUT IN RED                 00347   01410   0001464
        ORY AQUE1             2                              00350   01411   2341470
        SLD 16                    POSITION  I4,I5            00351   01412   2411020
        ANA MASKE                                            00352   01413   2201462
        STA AQUE2             2 STORE IN  ALARM TABLE        00353   01414   0341514
```

```
        LDA INDEXT          1 TEST FOR                  00354   01415   0021261
        BPL 1                 ALARM INDICATOR           00355   01416   2514001
        BRU ALARMB                                      00356   01417   2601427
        SRA 18                                          00357   01420   2400022
        BEV 1                                           00360   01421   2514000
        BRU *+3                                         00361   01422   2601425
        LDA CONAU             ARROW UP                  00362   01423   0001465
        BRU *+4                                         00363   01424   2601430
        LDA CONAD             ARROW DOWN                00364   01425   0001466
        BRU *+2                                         00365   01426   2601430
ALARMB  LDA CONB              BLANK  BLANK              00366   01427   0001467
        ORY AQUE2           2                           00367   01430   2341514
        RCL                   READ CLOCK                00370   01431   2510051
        SRD 14                AND                       00371   01432   2400116
        SLA 1                 POSITION                  00372   01433   2410001
        SLD 7                 FOR                       00373   01434   2411007
        ANA MASKF                     0037777           00374   01435   2201463
        STA AQUE4           2 PRINTING                  00375   01436   0341564
        LDA A1                                          00376   01437   0001457
        ADO                                             00377   01440   2504032
        STA A1                A1=A1+1                   00400   01441   0301457
        INX 1               2 A2=A2+1                   00401   01442   1440001
        BXH 20              2 IS A2 GREATER THAN 19     00402   01443   0557754
        LDX ZERO            2 A2=0                      00403   01444   0641226
        STX A2              2                           00404   01445   1741460
        LDX ASAVE2          2                           00405   01446   0641456
        BRU SCANA             RETURN TO SCAN            00406   01447   2601172
ABUSY   DEC 0                                           00407   01450   0000000
CONBRU  BRU 4                                           00410   01451   2600004
ETIME2  OCT 10076             200 MSEC DELAY            00411   01452   0010076
CON1E3  DEC 1000              CONSTANTS                 00412   01453   0001750
CON1E2  DEC 100                                         00413   01454   0000144
CON1E1  DEC 10                                          00414   01455   0000012
ASAVE2  DEC 0                                           00415   01456   0000000
A1      DEC 0                                           00416   01457   0000000
A2      DEC 0                                           00417   01460   0000000
MASKD   OCT 0007777                                     00420   01461   0007777
MASKE   OCT 0177400                                     00421   01462   0177400
MASKF   OCT 0037777                                     00422   01463   0037777
CON1B7  OCT 0010000                                     00423   01464   0010000
CONAU   OCT 0000057           ARROW UP + BLANK          00424   01465   0000057
CONAD   OCT 0000077           ARROW DOWN + BLANK        00425   01466   0000077
CONB    OCT 0000377           BLANK + BLANK             00426   01467   0000377
AQUE1   SLC +20               ALARM                     00427           6000024
AQUE2   SLC +20                 PRINTER                 00430           6000024
AQUE3   SLC +20                   DRIVE                 00431           6000024
AQUE4   SLC +20                     TABLE               00432           6000024
*                                                       0
*                                                       0
*       ALARM DRIVER PROGRAM                            0
ADRIVE  LDA A1                FROM INTERRUPT3           00433   01610   0001457
        BZE 1                 IS A1=0                   00434   01611   2514002
        BRU ADRIVA            IF SO                     00435   01612   2601646
        STX ASAVE1          1 SAVE XLOCATION1           00436   01613   1721655
        LDX A3              1                           00437   01614   0621654
        LDA ATIME             IS TIME SAME AS           00440   01615   0001656
        SUB AQUE4           1 LAST PRINTED              00441   01616   0221564
        BZE 2                                           00442   01617   2516002
        BRU ADRIVB                                      00443   01620   2601641
        LDA AQUE3           1 LOAD H REGISTER           00444   01621   0021540
        OCT 2511110             AND (OH1,1              00445   01622   2511110
        LDA AQUE2           1 PRINT ONE                 00446   01623   0021514
```

```
        OCT 2511210            LINE LDH 1,2              00447   01624   2511210
        LDA AQUE1            1                           00450   01625   0021470
        OCT 2511310            LDH1,3                    00451   01626   2511310
        OCT 2511410            PRH 1,1                   00452   01627   2511410
        LDA A1                                           00453   01630   0001457
        SBO                    A1=A1-1                   00454   01631   2504112
        STA A1                                           00455   01632   0301457
        INX 1                1 A3=A3+1                   00456   01633   1420001
        BXH 20               1 IS A3 GREATER THAN 19     00457   01634   0537754
        LDX ZERO             1 SET A3=0                  00460   01635   0621226
        STX A3               1                           00461   01636   1721654
ADRIVC  LDX ASAVE1           1                           00462   01637   0621655
        BRU 4                                            00463   01640   2600004
ADRIVB  LDA AQUE4            1 PRINT                     00464   01641   0021564
        STA ATIME                                        00465   01642   0301656
        OCT 2511110            LDH1,1                    00466   01643   2511110
        OCT 2511410            PRH 1,1                   00467   01644   2511410
        BRU ADRIVC                                       00470   01645   2601637
ADRIVA  LDA CONBRU             TURN OFF                  00471   01646   0001451
        STA 10                 ALARM DRIVE               00472   01647   0300012
        OCT 2511010            TURN PRINTER OFF          00473   01650   2511010
        LDZ                                              00474   01651   2504002
        STA ABUSY              SET ABUSY=0               00475   01652   0301450
        BRU 4                                            00476   01653   2600004
A3      DEC 0                                            00477   01654   0000000
ASAVE1  DEC 0                                            00500   01655   0000000
ATIME   DEC 0                                            00501   01656   0000000
*                                                        0
*                                                        0
*       I/O CONTROL PROGRAM                              0
IOCONT  BTC 1,1                FROM INTERRUPT            00502   01657   2514013
        BRU APRINT             ALARM PRINTER DELAY       00503   01660   2601664
        BTC 1,2                                          00504   01661   2514014
        BRU LOGTYP             LOG TYPER DELAY           00505   01662   2601667
        BRU 4                  ERROR                     00506   01663   2600004
APRINT  LDA *+2                TURN ON ALARM             00507   01664   0001666
        STA 10                 DRIVER                    00510   01665   0300012
        BRU ADRIVE             GO TO ALARM DRIVER        00511   01666   2601610
LOGTYP  LDA L1                                           00512   01667   0002107
        BZE 1                  IS L1=0                   00513   01670   2514002
        BRU NEWDEL             IF SO 200MSEC MORE DELAY  00514   01671   2601675
        LDA *+2                TURN ON                   00515   01672   0001674
        STA 11                 LOG DRIVE ROUTINE         00516   01673   0300013
        BRU LOGDR                                        00517   01674   2601700
NEWDEL  LDA ETIME2                                       00520   01675   0001452
        LTC 2                                            00521   01676   2500020
        BRU 4                  RETURN TO INTERRUPTED PR  00522   01677   2600004
*                                                        0
*                                                        0
*LOG    DRIVER ROUTINE         ENTRY FOR INTERRUPT 4     0
LOGDR   LDA L1                                           00523   01700   0002107
        BZE 1                                            00524   01701   2514002
        BRU LOGDRA                                       00525   01702   2601720
        SBO                                              00526   01703   2504112
        STA L1                 L1=L1-1                   00527   01704   0302107
        STX SAVEX3           3 SAVE XL3                  00530   01705   1761225
        LDX L3               3                           00531   01706   0662111
        LDA LOGQUE           3                           00532   01707   0062062
        SAB N,7                TYPE CHARACTER            00533   01710   2400407
        TYP N                  FROM POSITION L3          00534   01711   2500004
        INX 1                3 L3=L3+1                   00535   01712   1460001
        BXH 10               3 IS L3=10                  00536   01713   0577766
```

```
        LDX ZERO          3 L3=0                      00537   01714   0661226
        STX L3            3                           00540   01715   1762111
        LDX SAVEX3        3 RESTORE XL3               00541   01716   0661225
        BRU 4               RETURN TO INTERRUPTED     00542   01717   2600004
LOGDRA  LDA OFLAG           IS OFF FLAG=0             00543   01720   0001737
        BZE 2                                         00544   01721   2516002
        BRU LOGDRB                                    00545   01722   2601731
        LDA CRCODE          TYPE                      00546   01723   0001740
        SAB N,7             CARRAIGE                  00547   01724   2400407
        TYP N               RETURN                    00550   01725   2500004
        LDO                                           00551   01726   2504022
        STA OFLAG           SET OFF FLAG=1            00552   01727   0301737
        BRU 4                                         00553   01730   2600004
LOGDRB  OFF N               TURN TYPER OFF            00554   01731   2500010
        LDA CONBRU                                    00555   01732   0001451
        STA 11              TURN OFF LOG DRIVE        00556   01733   0300013
        LDZ                                           00557   01734   2504002
        STA OFLAG           SET OFF FLAG=0            00560   01735   0301737
        BRU 4                                         00561   01736   2600004
OFLAG   DEC 0                                         00562   01737   0000000
CRCODE  OCT 100                                       00563   01740   0000100
*                                                     0
*                                                     0
*HOUR   LOG PROGRAM ENTRY  FROM ECP FUCTION NUMBER 40  0
LOG     LDA TYPSEL                                    00564   01741   0002046
        SAB N,7             TURN ON TYPEWRITER        00565   01742   2400407
        SEL N                                         00566   01743   2500000
        LDA CONBRU          TURN OFF LOG              00567   01744   0001451
        STA 11              DRIVER PROGRAM            00570   01745   0300013
        LDA ETIME2          SET 200 MSEC DELAY        00571   01746   0001452
        LTC 2               FOR IO CONTROL ROUTINE    00572   01747   2500020
        LDX ZERO          1 SET M=ZERO                00573   01750   0621226
LOGA    LDA VALUET        1                           00574   01751   0021235
        SPB CONVER        3 CONVERT TO ENG UNITS      00575   01752   0761315
                                                      00576
        PAI                                           00577   01753   2500012
        STA LOGT          1                           00600   01754   0322047
        INX 1             1 M=M+1                     00601   01755   1420001
        BXL 10            1 IS M GREATER THAN 9       00602   01756   0437766
        BRU LOGA            REPEAT 100 TIMES          00603   01757   2601751
        LDZ                                           00604   01760   2504002
        STA ZEROF           ZERO FLAG=0               00605   01761   0302101
        STA L3              L3=0                      00606   01762   0302111
        LDX ONE           2 L2=1                      00607   01763   0641140
        STX L1            2 L1=1                      00610   01764   1742107
        LDX ZERO          1 M=0                       00611   01765   0621226
        LDX ZERO          3 Q=0                       00612   01766   0661226
        LDA TYPBLK          PUT BLACK CODE IN         00613   01767   0002061
        STA LOGQUE          QUEUE TABLE               00614   01770   0302062
LOGB    LDA L1                                        00615   01771   0002107
        SUB CON1E1                                    00616   01772   0201455
        BPL 1               IS L1=10                  00617   01773   2514001
        BRU LOGB            IF SO THEN WAIT           00620   01774   2601771
        LDA LOGT          1 GET VALUE OR REMAINDER    00621   01775   0022047
        MAQ                                           00622   01776   2504006
        DVD TENX          3                           00623   01777   1662074
        BZE 1                                         00624   02000   2514002
        BRU LOGC                                      00625   02001   2602022
        STA ZEROF           SET ZERO FLAG NOT=0       00626   02002   0302101
LOGD    STA LOGQUE        2 STORE BCD IN LOGQUE L2    00627   02003   0342062
        XAQ                                           00630   02004   2504005
        STA LOGT          1 STORE REMAINDER           00631   02005   0322047
```

```
        IAI                                          00632   02006   2500013
        LDA L1                                       00633   02007   0002107
        ADO                     L1=L1+1              00634   02010   2504032
        STA L1                                       00635   02011   0302107
        PAI                                          00636   02012   2500012
        INX 1                 2 L2=L2+1              00637   02013   1440001
        BXH 10                2 IF L2 GREATER THAN 9 00640   02014   0557766
        LDX ZERO              2 L2=0                 00641   02015   0641226
        BXH 4                 3 IS Q=4               00642   02016   0577774
        BRU LOGE                                     00643   02017   2602032
        INX 1                 3 Q=Q+1                00644   02020   1460001
        BRU LOGB                                     00645   02021   2601771
LOGC    LDA ZEROF               LEADING ZERO         00646   02022   0002101
        BZE 1                   SUPPRESSION  ZEROF=0 00647   02023   2514002
        BRU *+3                                      00650   02024   2602027
        LDA ZEROC               LOAD TYPE CODE FOR ZERO 00651 02025  0002102
        BRU LOGD                                     00652   02026   2602003
        BXH 4                 3 Q=4                  00653   02027   0577774
        BRU *-3                                      00654   02030   2602025
        BRU LOGD                SPACE                00655   02031   2602003
LOGE    BXH 9                 1 M=9                  00656   02032   0537767
        BRU LOGF                                     00657   02033   2602040
        LDX ZERO              3 Q=0                  00660   02034   0661226
        STX ZEROF             3 ZEROF=0              00661   02035   1762101
        INX 1                 1 M=M+1                00662   02036   1420001
        BRU LOGB                                     00663   02037   2601771
LOGF    LDA FTIME+2                                  00664   02040   0001134
        ADD CON72               ADD 1HOUR TO         00665   02041   0102103
        STA FTIME+2             LOG TIME IN ECP      00666   02042   0301134
        LDA CONLOG              RESET                00667   02043   0002104
        STA REG+22              ENTRY                00670   02044   0301130
        BRU ECPX                GO TO ECP            00671   02045   2601037
TYPSEL  OCT 60                                       00672   02046   0000060
LOGT    SLC +10                 LOG TABLE            00673           6000012
TYPBLK  OCT 35                                       00674   02061   0000035
LOGQUE  SLC +10                 LOG CHARACTER QUEUE. 00675           6000012
N       EQL N                                        00676
TENX    DEC 10000                                    00677   02074   0023420
        DEC 1000                                     00700   02075   0001750
        DEC 100                                      00701   02076   0000144
        DEC 10                                       00702   02077   0000012
        DEC 1                                        00703   02100   0000001
ZEROF   DEC 0                                        00704   02101   0000000
ZEROC   OCT 20                                       00705   02102   0000020
CON72   DEC 7200                                     00706   02103   0016040
CONLOG  BRU LOG                                      00707   02104   2601741
M       DEC 0                                        00710   02105   0000000
Q       DEC 0                                        00711   02106   0000000
L1      DEC 0                                        00712   02107   0000000
L2      DEC 0                                        00713   02110   0000000
L3      DEC 0                                        00714   02111   0000000
*                                                    0
*                                                    0
*         DEMAND PROGRAM FUNCTION NUMBER 1           0
DEMAND  BRD 1                                        00715   02112   2514017
        BRU DEMA                                     00716   02113   2602124
DEMNDA  LDA FTIME+1                                  00717   02114   0001133
        ADD TWO                 ADD 1SEC TO DEMAND TIME 00720 02115  0102233
        STA FTIME+1                                  00721   02116   0301133
        IAI                                          00722   02117   2500013
        LDA CONDEM                                   00723   02120   0002123
        STA REG+14              RESET ENTRY FROM ECP 00724   02121   0301120
```

```
        BRU ECPX                                    00725   02122  2601037
CONDEM  BRU DEMAND          CONSTANT                00726   02123  2602112
DEMA    LDA CONSC1          INITIATE READ IN        00727   02124  0002227
        LSC 1               OF  DECADE SWITCHES     00730   02125  2510103
        LDA CONDM2          SET SCAN1B TO           00731   02126  0002131
        STA SCAN            DEMAND2                 00732   02127  0301142
        BRU DEMNDA          GO BACK TO ECP          00733   02130  2602114
CONDM2  BRU DEMND2          CONSTANT                00734   02131  2602132
DEMND2  DST SAVEAQ          FROM SCANNER INTERRUPT  00735   02132  1301076
        STX SAVEX1        1                         00736   02133  1721223
        STX SAVEX3        3                         00737   02134  1761225
        RDG                                         00740   02135  2500023


        SRD 15              CONVERT                 00741   02136  2400117
        XAQ                  INDEX                  00742   02137  2504005
        SRA 15                TO                    00743   02140  2400017
        MPY CON1E1           BINARY                 00744   02141  1501455
        DST 1                                       00745 A 02142  1300001
        LDA VALUET        1                         00746   02143  0021235
        SPB CONVER        3 VALUE IN ENG UNITS      00747   02144  0761315
        MAQ                                         00750   02145  2504006
        DVD CON1E1                                  00751   02146  1601455
        XAQ                                         00752   02147  2504005
        SLA 8                                       00753   02150  2410010
        STA TEMPD           UNITS                   00754   02151  0302234
        LDZ                                         00755   02152  2504002
        DVD CON1E1                                  00756   02153  1601455
        XAQ                                         00757   02154  2504005
        SLA 4                                       00760   02155  2410004
        ADD TEMPD            TENS                   00761   02156  0102234
        LCV 1               PUT UNITS AND TENS IN C 00762   02157  2510101
        LDA CONSC2          INITIATE SUB CONTROL    00763   02160  0002230
        LSC 1               TO DISPLAY UNITS + TENS 00764   02161  2510103
        LDZ                                         00765   02162  2504002
        DVD CON1E1                                  00766   02163  1601455
        SLA 4                                       00767   02164  2410004
        STA TEMPD                                   00770   02165  0302234
        XAQ                                         00771   02166  2504005
        SLA 8                                       00772   02167  2410010
        ORY TEMPD           STORE HUND + THOUS TEMP 00773   02170  2302234
        LDA CONDM3          SET SCAN 1B TO          00774   02171  0002177
        STA SCAN            DEMAND3                 00775   02172  0301142
        LDX SAVEX1        1                         00776   02173  0621223
        LDX SAVEX3        3                         00777   02174  0661225
        DLD SAVEAQ                                  01000   02175  1001076
        BRU 5               GO BACK TO INTERRUPTED  01001   02176  2600005
CONDM3  BRU DEMND3                                  01002   02177  2602200
DEMND3  LDA TEMPD                                   01003   02200  0002234
        LCV 1                                       01004   02201  2510101
        LDA CONSC3          INITIATE DISPLAY OF     01005   02202  0002231
        LSC 1               HUNDRETHS + THOUSANDS   01006   02203  2510103
        LDA CONDM4          SET SCAN 1B             01007   02204  0002207
        STA SCAN            TO DEMAND 4             01010   02205  0301142
        BRU 4                                       01011   02206  2600004
CONDM4  BRU DEMND4                                  01012   02207  2602210
DEMND4  LDA CONSC4          TURN OFF DEMAND         01013   02210  0002232
        LSC 1                 LIGHT                 01014   02211  2510103
        LDA CONDM5          SET SCAN 1B             01015   02212  0002215
        STA SCAN            TO DEMAND5              01016   02213  0301142
        BRU 4                                       01017   02214  2600004
CONDM5  BRU DEMND5                                  01020   02215  2602216
```

```
DEMND5 STX SAVEX1          1                    01021  02216  1721223
       LDX POINTJ          1 RE-INITIATE SCAN OF 01022  02217  0621230
       LDA SCANT           1 INTERUPTED POINT.  01023  02220  0021247
       LSC 1                                    01024  02221  2510103
       LDA CONDM6            SET SCAN 1B BACK TO 01025  02222  0002226
       STA SCAN              SCAN 1A            01026  02223  0301142
       LDX SAVEX1          1                    01027  02224  0621223
       BRU 4                                    01030  02225  2600004
CONDM6 STX SAVEX1          1 CONSTANT           01031  02226  1721223
CONSC1 OCT 3030203           SCANNER            01032  02227  3030203
CONSC2 OCT 3010240           COMMANDS FOR       01033  02230  3010240
CONSC3 OCT 3010340           SUB-CONTROL        01034  02231  3010340
CONSC4 OCT 3040004                              01035  02232  3040004
TWO    DEC 2                                    01036  02233  0000002
TEMPD  DEC 0                                    01037  02234  0000000
       SLC /3000                                01040         4003000
INITLZ LDZ                                      01041  03000  2504002
       STA 1                                    01042  03001  0300001
       STA POINTJ                               01043  03002  0301230
       STA TIME                                 01044  03003  0301135
       STA ABUSY                                01045  03004  0301450
       STA A1                                   01046  03005  0301457
       STA A2                                   01047  03006  0301460
       STA A3                                   01050  03007  0301654
       STA OFLAG                                01051  03010  0301737
       LDA TABLE           1                    01052  03011  0023040
       STA 4               1                    01053  03012  0320004
       INX 1               1                    01054  03013  1420001
       BXL 16              1                    01055  03014  0437760
       BRU *-4                                  01056  03015  2603011
       LDO                                      01057  03016  2504022
       STA POINTK                               01060  03017  0301231
       ADO                                      01061  03020  2504032
       STA FTIME+1                              01062  03021  0301133
       LDA CONDEM                               01063  03022  0002123
       STA REG+14                               01064  03023  0301120
       LDA CONLOG                               01065  03024  0002104
       STA REG+22                               01066  03025  0301130
       LDA CON72                                01067  03026  0002103
       STA FTIME+2                              01070  03027  0301134
       LDA ETCON3                               01071  03030  0001137
       LTC 3                                    01072  03031  2500021
       LDZ                                      01073  03032  2504002
       LTC 1                                    01074  03033  2500017
       LTC 2                                    01075  03034  2500020
       LDA SCANT                                01076  03035  0001247
       LSC 1                                    01077  03036  2510103
       BRU ECPX                                 01100  03037  2601037
TABLE  LDA 7                                    01101  03040  0000007
       PAI                                      01102  03041  2500012
       BRU *                                    01103  03042  2603042
       LDA 0                                    01104  03043  0000000
       BRU ECP                                  01105  03044  2601000
       BRU SCAN                                 01106  03045  2601142
       BRU ADRIVE                               01107  03046  2601610
       BRU LOGDR                                01110  03047  2601700
       BRU IOCONT                               01111  03050  2601657
       BRU 4                                    01112  03051  2600004
       BRU 4                                    01113  03052  2600004
       BRU 4                                    01114  03053  2600004
       BRU 4                                    01115  03054  2600004
       BRU 4                                    01116  03055  2600004
       BRU 4                                    01117  03056  2600004
       BRU 4                                    01120  03057  2600004
       END INITLZ                               01121         6103000
```

APPENDIX A. BINARY CODED DIGITS

Character	N1	N2	N3	Parity	N4	N5	N6	N7	Hollerith Code	Octal Code	Normally Available on Numeric Typewriter
	N REGISTER BIT										
0 (zero)	0	0	1		0	0	0	0	0	020	X
1	0	0	0		0	0	0	1	1	001	X
2	0	0	0		0	0	1	0	2	002	X
3	0	0	0	X	0	0	1	1	3	003	X
4	0	0	0		0	1	0	0	4	004	X
5	0	0	0	X	0	1	0	1	5	005	X
6	0	0	0	X	0	1	1	0	6	006	X
7	0	0	0		0	1	1	1	7	007	X
8	0	0	0		1	0	0	0	8	010	X
9	0	0	0	X	1	0	0	1	9	011	X
A	0	1	1		0	0	0	1	1 & 12	061	
B	0	1	1		0	0	1	0	2 & 12	062	
C	0	1	1	X	0	0	1	1	3 & 12	063	
D	0	1	1		0	1	0	0	4 & 12	064	
E	0	1	1	X	0	1	0	1	5 & 12	065	
F	0	1	1	X	0	1	1	0	6 & 12	066	
G	0	1	1		0	1	1	1	7 & 12	067	
H	0	1	1		1	0	0	0	8 & 12	070	
I	0	1	1	X	1	0	0	1	9 & 12	071	
J	0	1	0	X	0	0	0	1	1 & 11	041	
K	0	1	0	X	0	0	1	0	2 & 11	042	
L	0	1	0		0	0	1	1	3 & 11	043	
M	0	1	0	X	0	1	0	0	4 & 11	044	
N	0	1	0		0	1	0	1	5 & 11	045	
O	0	1	0		0	1	1	0	6 & 11	046	
P	0	1	0	X	0	1	1	1	7 & 11	047	
Q	0	1	0	X	1	0	0	0	8 & 11	050	
R	0	1	0		1	0	0	1	9 & 11	051	
S	0	0	1	X	0	0	1	0	0 & 2	022	
T	0	0	1		0	0	1	1	0 & 3	023	
U	0	0	1	X	0	1	0	0	0 & 4	024	
V	0	0	1		0	1	0	1	0 & 5	025	
W	0	0	1		0	1	1	0	0 & 6	026	
X	0	0	1	X	0	1	1	1	0 & 7	027	
Y	0	0	1	X	1	0	0	0	0 & 8	030	
Z	0	0	1		1	0	0	1	0 & 9	031	

Character	N REGISTER BIT N1	N2	N3	Parity	N4	N5	N6	N7	Hollerith Code	Octal Code	Normally Available on Numeric Typewriter
Space (blank key or space bar)	0	0	0	X	0	0	0	0	Blank	000	X
- (hyphen)	0	1	0		0	0	0	0	11	040	X
/ (slash)	0	0	1	X	0	0	0	1	0 & 1	021	
$ (dollar)	0	1	0	X	1	0	1	1	3, 8 & 11	053	
, (comma)	0	0	1	X	1	0	1	1	0, 3 & 8	033	
. (period)	0	1	1		1	0	1	1	3, 8 & 12	073	X
Tabulate	0	0	1	X	1	1	1	0	0, 6 & 8	036	X
Carriage Return	1	0	0		0	0	0	0	7 & 9	100	X
Print Red	0	0	1		1	0	1	0	0, 2 & 8	032	X
Print Black	0	0	1	X	1	1	0	1	0, 5 & 8	035	X

APPENDIX B. FLOW CHARTING AND 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.

When the system flow chart is completed, other flow charts in much greater detail are drawn from the individual blocks. These more detailed flow charts are the charts that the programmer usually uses when he "codes" the program. In these flow charts, reference is sometimes made to actual computer instructions such as ADD, OSA, RSA, STA. Flow charts of this type are of considerable help when "coding" and debugging 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. For this reason, it was deemed advisable to standardize the symbols used in flow charts throughout the computer industry in order to simplify communications between programmers as well as manufacturers. The Standardization Committee of the Association for Computing Machinery has recommended certain flow-charting symbols for specific uses which are included in this appendix.

Other 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 alphabetics 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.

Numerics	Alphabetics
2	Z̸ ("zee")
4, 9	
0	Ō or Ø ("oh")
1	I ("eye")
5	S or S̸ ("ess")
7 (seven)	

Suggested Flow Chart Symbols
for Simplified Flow Charting

Symbol	Usage
	Function or Operation Description
	Logic "Flow" (Follow the Arrows)
	Decision, Test, Comparison (2-or-3-way split)
	Subroutine
	Entrance, Exit, Stop
A A	Fixed Connector (Same symbol)
B_n B_1 B_2 B_3	Variable Connector (switch function)

APPENDIX C. 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	0149	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	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	3000 to 3377
Decimal	1356 to 1791

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	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	3400 to 3777
Decimal	1792 to 2047

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	2042	2043	2044	2045	2046	2047

Octal-Decimal Integer ConversionTable

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	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	5000 to 5377
Decimal	2560 to 2815

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	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	3537	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	5400 to 5777
Decimal	2816 to 3071

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	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	2354	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	7000 to 7377
Decimal	3584 to 3839

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	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	7400 to 7777
Decimal	3840 to 4095

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	4090	4091	4092	4093	4094	4095

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

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

APPENDIX D. OPERATION CODES IN ORDER BY MNEMONICS

Symbols	Octal	Description	Execution Time
ABQ S,K	2400300 N,0	SHIFT A INTO BUFFER S AND INTO Q, K PLACES	2+
ABQ S,K	2400500 M,0	SHIFT A INTO BUFFER S AND INTO Q, K PLACES	2+
ADD Y	0100000 0	ADD Y TO A	2
ADO	2504032	ADD ONE TO A	2
ANA Y	2200000 0	AND Y TO A	2
BBP J,S	2514006 1,N	BRANCH ON BUFFER S PARITY ERROR	2
BBP J,S	2514007 1,M	BRANCH ON BUFFER S PARITY ERROR	2
BBP J,S	2516006 2,N	BRANCH ON BUFFER S PARITY ERROR	2
BBP J,S	2516007 2,M	BRANCH ON BUFFER S PARITY ERROR	2
BBR J,S	2514010 1,N	BRANCH ON BUFFER S READY	2
BBR J,S	2514011 1,M	BRANCH ON BUFFER S READY	2
BBR J,S	2516010 2,N	BRANCH ON BUFFER S READY	2
BBR J,S	2516011 2,M	BRANCH ON BUFFER S READY	2
BCL J	2514012 1	BRABCH ON CLOCK VALID	2
BCL J	2516012 2	BRABCH ON CLOCK VALID	2
BCO J,K	2514021 1,1	BRANCH ON CONVERTER K OVERFLOW	2
BCO J,K	2514032 1,2	BRANCH ON CONVERTER K OVERFLOW	2
BCO J,K	2516021 2,1	BRANCH ON CONVERTER K OVERFLOW	2
BCO J,K	2516032 2,2	BRANCH ON CONVERTER K OVERFLOW	2
BCS J,D,E	2515000 1,0,0	BRANCH ON CONTROLLER SELECTOR, MINIMUM OPERANDS	2
BCS J,D,E	2517307 2,3,7	BRANCH ON CONTROLLER SELECTOR, MAXIMUM OPERANDS	2
BDC J	2514020 1	BRANCH ON DRUM OPERATION COMPLETE	2
BDC J	2516020 2	BRANCH ON DRUM OPERATION COMPLETE	2
BDM J	2514034 1	BRANCH ON MULTIPLE OUTPUT COMPLETE	2
BDM J	2516034 2	BRANCH ON MULTIPLE OUTPUT COMPLETE	2
BDT J	2514027 1	BRANCH ON TIMED CONTACT COMPLETE	2
BDT J	2516027 2	BRANCH ON TIMED CONTACT COMPLETE	2
BEA J	2514023 1	BRANCH ON FCHO ALARM	2
BEA J	2516023 2	BRANCH ON FCHO ALARM	2
BEV J	2514000 1	BRANCH ON EVEN	2
BEV J	2516000 2	BRANCH ON EVEN	2
BOV J	2514003 1	BRANCH ON OVERFLOW	2
BOV J	2516003 2	BRANCH ON OVERFLOW	2
BPC J	2514004 1	BRANCH ON PARITY ERROR, CORE	2
BPC J	2516004 2	BRANCH ON PARITY ERROR, CORE	2
BPD J	2514005 1	BRANCH ON PARITY ERROR, DRUM	2
BPD J	2516005 2	BRANCH ON PARITY ERROR, DRUM	2
BPL J	2514001 1	BRANCH ON PLUS	2
BPL J	2516001 2	BRANCH ON PLUS	2
BRD J	2514017 1	BRANCH ON DEMAND	2
BRD J	2516017 2	BRANCH ON DEMAND	2
BRH J,I	2514024 1,1	BRANCH ON H-REGISTER I READY	2
BRH J,I	2514033 1,2	BRANCH ON H-REGISTER I READY	2
BRH J,I	2516024 2,1	BRANCH ON H-REGISTER I READY	2
BRH J,I	2516033 2,2	BRANCH ON H-REGISTER I READY	2
BRU Y	2600000 0	BRANCH UNCONDITIONALLY	1
BSC J,K	2514022 1,1	BRANCH ON SCANNER OPERATION COMPLETE	2
BSC J,K	2514031 1,2	BRANCH ON SCANNER OPERATION COMPLETE	2
BSC J,K	2516022 2,1	BRANCH ON SCANNER OPERATION COMPLETE	2
BSC J,K	2516031 2,2	BRANCH ON SCANNER OPERATION COMPLETE	2
BTC J,K	2514013 1,1	BRANCH ON TIME COUNTER OVERFLOW	2
BTC J,K	2514014 1,2	BRANCH ON TIME COUNTER OVERFLOW	2
BTC J,K	2514015 1,3	BRANCH ON TIME COUNTER OVERFLOW	2
BTC J,K	2514016 1,4	BRANCH ON TIME COUNTER OVERFLOW	2
BTC J,K	2516013 2,1	BRANCH ON TIME COUNTER OVERFLOW	2
BTC J,K	2516014 2,2	BRANCH ON TIME COUNTER OVERFLOW	2

BTC J,K	2516015	2,3	BRANCH ON TIME COUNTER OVERFLOW	2
BTC J,K	2516016	2,4	BRANCH ON TIME COUNTER OVERFLOW	2
BXH K,X	0500000	0,0	BRANCH ON X EQUAL TO OR GREATER THAN K	3
BXL K,X	0400000	0,0	BRANCH ON X LESS THAN K	3
BZE J	2514002	1	BRANCH ON ZERO	2
BZE J	2516002	2	BRANCH ON ZERO	2
CHS	2504040		CHANGE SIGN OF A	2
CPL	2504512		COMPLEMENT A	2
DAD Y	1100000	0	DOUBLE-LENGTH ADD Y	3
DLD Y	1000000	0	DOUBLE-LENGTH LOAD FROM Y	3
DNO K	2411040	0	DOUBLE-LENGTH NORMALIZE, K PLACES MAXIMUM	2+
DST Y	1300000	0	DOUBLE-LENGTH STORE AT Y	3
DSU Y	1200000	0	DOUBLE-LENGTH SUBTRACT Y	3
DVD Y	1600000	0	DIVIDE BY Y	25,28
ERA Y	2100000	0	EXCLUSIVE OR INTO A FROM Y	2
EXT Y	2000000	0	EXTRACT FROM A INTO Y	2
IAI	2500013		INHIBIT AUTOMATIC INTERUPT	2
INX K,X	1400000	0,0	INCREMENT X BY K	3
JMP Y	3700000	0	JUMP UNCONDITIONALLY TO Y	1
LAC K	2510100	1	LOAD ACCUMULATOR SCAN COMMAND REGISTER	2,3
LAC K	2510200	2	LOAD ACCUMULATOR SCAN COMMAND REGISTER	2,3
LAC K	2510400	3	LOAD ACCUMULATOR SCAN COMMAND REGISTER	2,3
LAC K	2511000	4	LOAD ACCUMULATOR SCAN COMMAND REGISTER	2,3
LAC K	2513000	5	LOAD ACCUMULATOR SCAN COMMAND REGISTER	2,3
LAQ	2504001		LOAD A FROM Q	2
LCS D	2501000	0	LOAD CONTROLLER SELECTOR D COMMAND REGISTER	2
LCS D	2501101	1	LOAD CONTROLLER SELECTOR D COMMAND REGISTER	2
LCS D	2501200	2	LOAD CONTROLLER SELECTOR D COMMAND REGISTER	2
LCS D	2501300	3	LOAD CONTROLLER SELECTOR D COMMAND REGISTER	2
LCV K	2510101	1	LOAD CONVERTER REGISTER K FROM A	2,3
LCV K	2510201	2	LOAD CONVERTER REGISTER K FROM A	2,3
LCV K	2510401	3	LOAD CONVERTER REGISTER K FROM A	2,3
LCV K	2511001	4	LOAD CONVERTER REGISTER K FROM A	2,3
LCV K	2512001	5	LOAD CONVERTER REGISTER K FROM A	2,3
LDA Y	0000000	0	LOAD A FROM Y	2
LDC	2500026		LOAD DRUM COMMAND REGISTER FROM A	2
LDH I,N	2511110	1,1	LOAD H-REGISTER I, BLOCK N, FROM A	2,3
LDH I,N	2511117	2,1	LOAD H-REGISTER I, BLOCK N, FROM A	2,3
LDH I,N	2511210	1,2	LOAD H-REGISTER I, BLOCK N, FROM A	2,3
LDH I,N	2511217	2,2	LOAD H-REGISTER I, BLOCK N, FROM A	2,3
LDH I,N	2511310	1,3	LOAD H-REGISTER I, BLOCK N, FROM A	2,3
LDH I,N	2511317	2,3	LOAD H-REGISTER I, BLOCK N, FROM A	2,3
LDM	2510112		LOAD OUTPUT DISTRIBUTOR MULTIPLE-OUTPUT FUNCTION	2
LDO	2504022		LOAD ONE INTO A	2
LDS K	2510102		LOAD DIGITAL SCAN COMMAND REGISTER FROM A	2,3
LDS K	2510202		LOAD DIGITAL SCAN COMMAND REGISTER FROM A	2,3
LDS K	2510402		LOAD DIGITAL SCAN COMMAND REGISTER FROM A	2,3
LDS K	2511002		LOAD DIGITAL SCAN COMMAND REGISTER FROM A	2,3
LDS K	2512002		LOAD DIGITAL SCAN COMMAND REGISTER FROM A	2,3
LDT	2510012		LOAD OUTPUT DISTRIBUTOR TIMED-CONTACT FUNCTION	2
LDX Y,X	0600000	0,0	LOAD X-LOCATION FROM Y	3
LDZ	2504002		LOAD ZERO INTO A	2
LLA K	2410100	0	LOGICAL LEFT SHIFT A, K PLACES	2+
LLC K	2410300	0	LOGICAL LEFT SHIFT A CIRCULAR, K PLACES	2+
LLD K	2411100	0	LOGICAL LEFT SHIFT DOUBLE, K PLACES	2+
LMO	2504102		LOAD MINUS ONE INTO A	2
LQA	2504004		LOAD Q FROM A	2
LSC K	2510103	1	LOAD SCANNER COMMAND REGISTER FROM A	2,3
LSC K	2510203	2	LOAD SCANNER COMMAND REGISTER FROM A	2,3
LSC K	2510403	3	LOAD SCANNER COMMAND REGISTER FROM A	2,3
LSC K	2511003	4	LOAD SCANNER COMMAND REGISTER FROM A	2,3
LSC K	2512003	5	LOAD SCANNER COMMAND REGISTER FROM A	2,3
LTC K	2500017	1	LOAD TIME COUNTER K	2
LTC K	2500020	2	LOAD TIME COUNTER K	2
LTC K	2500021	3	LOAD TIME COUNTER K	2
LTC K	2500022	4	LOAD TIME COUNTER K	2
MAQ	2504006		MOVE A TO Q	2
MPY Y	1500000	0	MULTIPLY BY Y	13-18
NEG	2504532		NEGATE A	2

NOP	2504000	NO OPERATION	2
NOR K	2410040 0	NORMALIZE, K PLACES MAXIMUM	2+
OFA	2510007	TURN OFF FAST ACCESS DEVICES	2
OFF S	2500010 N	TURN OFF PERIPHERALS ON BUFFER S	2
OFF S	2500011 M	TURN OFF PERIPHERALS ON BUFFER S	2
OFH I	2511010 1	TURN OFF PRINTERS ON H-REGISTER I	2
OFH I	2511017 2	TURN OFF PRINTERS ON H-REGISTER I	2
OQA K	2401040 0	OR Q AND A INTO A, K PLACES	2+
ORY Y	2300000 0	OR A INTO Y	2
PAI	2500012	PERMIT AUTOMATIC INTERUPT	2
PCH S	2500006 N	PUNCH ON BUFFER S	2
PCH S	2500007 M	PUNCH ON BUFFER S	2
PRH I,K	2511410 1,1	PRINT ON H-REGISTER I, PRINTER K	2,3
PRH I,K	2511417 2,1	PRINT ON H-REGISTER I, PRINTER K	2,3
PRH I,K	2511510 1,2	PRINT ON H-REGISTER I, PRINTER K	2,3
PRH I,K	2511517 2,2	PRINT ON H-REGISTER I, PRINTER K	2,3
PRH I,K	2511610 1,3	PRINT ON H-REGISTER I, PRINTER K	2,3
PRH I,K	2511617 2,3	PRINT ON H-REGISTER I, PRINTER K	2,3
PRH I,K	2511710 1,4	PRINT ON H-REGISTER I, PRINTER K	2,3
PRH I,K	2511717 2,4	PRINT ON H-REGISTER I, PRINTER K	2,3
RCL	2510051	READ DIGITAL CLOCK INTO A	2
RCS	2500024	READ CONSOLE SWITCHES INTO A	2
RCV K	2510144 1	READ CONVERTER K INTO A	2,3
RCV K	2510244 2	READ CONVERTER K INTO A	2,3
RCV K	2510444 3	READ CONVERTER K INTO A	2,3
RCV K	2511044 4	READ CONVERTER K INTO A	2,3
RCV K	2512044 5	READ CONVERTER K INTO A	2,3
RDD S	2500002 N	READ ON BUFFER S	2
RDD S	2500003 M	READ ON BUFFER S	2
RDG	2500023	READ DIGITAL INPUT	2
RFA	2510046	READ FAST ACCESS DEVICE	2,3
SAB S,K	2400200 M,0	SHIFT A INTO BUFFER S, K PLACES	2+
SAB S,K	2400400 N,0	SHIFT A INTO BUFFER S, K PLACES	2+
SAI K	2500014 1	SELECT AUTOMATIC INTERUPT GROUP K	2
SAI K	2500015 2	SELECT AUTOMATIC INTERUPT GROUP K	2
SAI K	2500016 3	SELECT AUTOMATIC INTERUPT GROUP K	2
SBA S,K	2402000 M,0	SHIFT BUFFER S INTO A, K PLACES	2+
SBA S,K	2404000 N,0	SHIFT BUFFER S INTO A, K PLACES	2+
SBD S,K	2402100 M,0	SHIFT BUFFER S INTO AQ DOUBLE, K PLACES	2+
SBD S,K	2404100 N,0	SHIFT BUFFER S INTO AQ DOUBLE, K PLACES	2+
SBO	2504112	SUBTRACT ONE FROM A	2
SCA K	2400040 0	SHIFT RIGHT CIRCULAR A, K PLACES	2+
SCD K	2401100 0	SHIFT RIGHT CIRCULAR DOUBLE, K PLACES	2+
SEL S	2500000 N	SELECT DEVICE ON BUFFER S	2
SEL S	2500001 M	SELECT DEVICE ON BUFFER S	2
SLA K	2410000 0	SHIFT LEFT A, K PLACES	2+
SLD K	2411000 0	SHIFT LEFT DOUBLE, K PLACES	2+
SLH I,K	2510410 1,1	SELECT H-REGISTER I, PRINTER K	2,3
SLH I,K	2510417 2,1	SELECT H-REGISTER I, PRINTER K	2,3
SLH I,K	2510510 1,2	SELECT H-REGISTER I, PRINTER K	2,3
SLH I,K	2510517 2,2	SELECT H-REGISTER I, PRINTER K	2,3
SLH I,K	2510610 1,3	SELECT H-REGISTER I, PRINTER K	2,3
SLH I,K	2510617 2,3	SELECT H-REGISTER I, PRINTER K	2,3
SLH I,K	2510710 1,4	SELECT H-REGISTER I, PRINTER K	2,3
SLH I,K	2510717 2,4	SELECT H-REGISTER I, PRINTER K	2,3
SPB Y,X	0700000 0,0	STORE P AT X AND BRANCH UNCONDITIONALLY TO Y	2
SPJ Y	3740000 0	STORE P AT X2 AND JUMP UNCONDITIONALLY TO Y	2
SQA K	2401000 0	SHIFT Q RIGHT INTO A, K PLACES	2+
SRA K	2400000 0	SHIFT RIGHT A, K PLACES	2+
SRD K	2400100 0	SHIFT RIGHT DOUBLE, K PLACES	2+
SSA	2500025	SET STALL ALARM	2
STA Y	0300000 0	STORE A AT Y	2
STO Y	2700000 0	STORE OPERAND ADDRESS AT Y	2
STX Y,X	1700000 0,0	STORE X-LOCATION AT Y	3
SUB Y	0200000 0	SUBTRACT Y FROM A	2
TYP S	2500004 N	TYPE ON BUFFER S	2
TYP S	2500005 M	TYPE ON BUFFER S	2
XAQ	2504005	EXCHANGE A AND Q	2
XEC Y	3400000 0	EXECUTE THE INSTRUCTION AT Y	1+

APPENDIX E. OPERATION CODES IN ORDER BY OCTAL

Symbols	Octal	Description	Execution Time
LDA Y	0000000 0	LOAD A FROM Y	2
ADD Y	0100000 0	ADD Y TO A	2
SUB Y	0200000 0	SUBTRACT Y FROM A	2
STA Y	0300000 0	STORE A AT Y	2
BXL K,X	0400000 0,0	BRANCH ON X LESS THAN K	3
BXH K,X	0500000 0,0	BRANCH ON X EQUAL TO OR GREATER THAN K	3
LDX Y,X	0600000 0,0	LOAD X-LOCATION FROM Y	3
SPB Y,X	0700000 0,0	STORE P AT X AND BRANCH UNCONDITIONALLY TO Y	2
DLD Y	1000000 0	DOUBLE-LENGTH LOAD FROM Y	3
DAD Y	1100000 0	DOUBLE-LENGTH ADD Y	3
DSU Y	1200000 0	DOUBLE-LENGTH SUBTRACT Y	3
DST Y	1300000 0	DOUBLE-LENGTH STORE AT Y	3
INX K,X	1400000 0,0	INCREMENT X BY K	3
MPY Y	1500000 0	MULTIPLY BY Y	13-18
DVD Y	1600000 0	DIVIDE BY Y	25,28
STX Y,X	1700000 0,0	STORE X-LOCATION AT Y	3
EXT Y	2000000 0	EXTRACT FROM A INTO Y	2
ERA Y	2100000 0	EXCLUSIVE OR INTO A FROM Y	2
ANA Y	2200000 0	AND Y TO A	2
ORY Y	2300000 0	OR A INTO Y	2
SRA K	2400000 0	SHIFT RIGHT A, K PLACES	2+
SCA K	2400040 0	SHIFT RIGHT CIRCULAR A, K PLACES	2+
SRD K	2400100 0	SHIFT RIGHT DOUBLE, K PLACES	2+
SAB S,K	2400200 M,0	SHIFT A INTO BUFFER S, K PLACES	2+
ABQ S,K	2400300 N,0	SHIFT A INTO BUFFER S AND INTO Q, K PLACES	2+
SAB S,K	2400400 N,0	SHIFT A INTO BUFFER S, K PLACES	2+
ABQ S,K	2400500 M,0	SHIFT A INTO BUFFER S AND INTO Q, K PLACES	2+
SQA K	2401000 0	SHIFT Q RIGHT INTO A, K PLACES	2+
OQA K	2401040 0	OR Q AND A INTO A, K PLACES	2+
SCD K	2401100 0	SHIFT RIGHT CIRCULAR DOUBLE, K PLACES	2+
SBA S,K	2402000 M,0	SHIFT BUFFER S INTO A, K PLACES	2+
SBD S,K	2402100 M,0	SHIFT BUFFER S INTO AQ DOUBLE, K PLACES	2+
SBA S,K	2404000 N,0	SHIFT BUFFER S INTO A, K PLACES	2+
SBD S,K	2404100 N,0	SHIFT BUFFER S INTO AQ DOUBLE, K PLACES	2+
SLA K	2410000 0	SHIFT LEFT A, K PLACES	2+
NOR K	2410040 0	NORMALIZE, K PLACES MAXIMUM	2+
LLA K	2410100 0	LOGICAL LEFT SHIFT A, K PLACES	2+
LLC K	2410300 0	LOGICAL LEFT SHIFT A CIRCULAR, K PLACES	2+
SLD K	2411000 0	SHIFT LEFT DOUBLE, K PLACES	2+
DNO K	2411040 0	DOUBLE-LENGTH NORMALIZE, K PLACES MAXIMUM	2+
LLD K	2411100 0	LOGICAL LEFT SHIFT DOUBLE, K PLACES	2+
SEL S	2500000 N	SELECT DEVICE ON BUFFER S	2
SEL S	2500001 M	SELECT DEVICE ON BUFFER S	2
RDD S	2500002 N	READ ON BUFFER S	2
RDD S	2500003 M	READ ON BUFFER S	2
TYP S	2500004 N	TYPE ON BUFFER S	2
TYP S	2500005 M	TYPE ON BUFFER S	2
PCH S	2500006 N	PUNCH ON BUFFER S	2
PCH S	2500007 M	PUNCH ON BUFFER S	2
OFF S	2500010 N	TURN OFF PERIPHERALS ON BUFFER S	2
OFF S	2500011 M	TURN OFF PERIPHERALS ON BUFFER S	2
PAI	2500012	PERMIT AUTOMATIC INTERUPT	2
IAI	2500013	INHIBIT AUTOMATIC INTERUPT	2
SAI K	2500014 1	SELECT AUTOMATIC INTERUPT GROUP K	2
SAI K	2500015 2	SELECT AUTOMATIC INTERUPT GROUP K	2
SAI K	2500016 3	SELECT AUTOMATIC INTERUPT GROUP K	2

LTC K	2500017 1	LOAD TIME COUNTER K	2
LTC K	2500020 2	LOAD TIME COUNTER K	2
LTC K	2500021 3	LOAD TIME COUNTER K	2
LTC K	2500022 4	LOAD TIME COUNTER K	2
RDG	2500023	READ DIGITAL INPUT	2
RCS	2500024	READ CONSOLE SWITCHES INTO A	2
SSA	2500025	SET STALL ALARM	2
LDC	2500026	LOAD DRUM COMMAND REGISTER FROM A	2
LCS D	2501000 0	LOAD CONTROLLER SELECTOR D COMMAND REGISTER	2
LCS D	2501101 1	LOAD CONTROLLER SELECTOR D COMMAND REGISTER	2
LCS D	2501200 2	LOAD CONTROLLER SELECTOR D COMMAND REGISTER	2
LCS D	2501300 3	LOAD CONTROLLER SELECTOR D COMMAND REGISTER	2
NOP	2504000	NO OPERATION	2
LAQ	2504001	LOAD A FROM Q	2
LDZ	2504002	LOAD ZERO INTO A	2
LQA	2504004	LOAD Q FROM A	2
XAQ	2504005	EXCHANGE A AND Q	2
MAQ	2504006	MOVE A TO Q	2
LDO	2504022	LOAD ONE INTO A	2
ADO	2504032	ADD ONE TO A	2
CHS	2504040	CHANGE SIGN OF A	2
LMO	2504102	LOAD MINUS ONE INTO A	2
SBO	2504112	SUBTRACT ONE FROM A	2
CPL	2504512	COMPLEMENT A	2
NEG	2504532	NEGATE A	2
OFA	2510007	TURN OFF FAST ACCESS DEVICES	2
LDT	2510012	LOAD OUTPUT DISTRIBUTOR TIMED-CONTACT FUNCTION	2
RFA	2510046	READ FAST ACCESS DEVICE	2,3
RCL	2510051	READ DIGITAL CLOCK INTO A	2
LAC K	2510100 1	LOAD ACCUMULATOR SCAN COMMAND REGISTER	2,3
LCV K	2510101 1	LOAD CONVERTER REGISTER K FROM A	2,3
LDS K	2510102	LOAD DIGITAL SCAN COMMAND REGISTER FROM A	2,3
LSC K	2510103 1	LOAD SCANNER COMMAND REGISTER FROM A	2,3
LDM	2510112	LOAD OUTPUT DISTRIBUTOR MULTIPLE-OUTPUT FUNCTION	2
RCV K	2510144 1	READ CONVERTER K INTO A	2,3
LAC K	2510200 2	LOAD ACCUMULATOR SCAN COMMAND REGISTER	2,3
LCV K	2510201 2	LOAD CONVERTER REGISTER K FROM A	2,3
LDS K	2510202	LOAD DIGITAL SCAN COMMAND REGISTER FROM A	2,3
LSC K	2510203 2	LOAD SCANNER COMMAND REGISTER FROM A	2,3
RCV K	2510244 2	READ CONVERTER K INTO A	2,3
LAC K	2510400 3	LOAD ACCUMULATOR SCAN COMMAND REGISTER	2,3
LCV K	2510401 3	LOAD CONVERTER REGISTER K FROM A	2,3
LDS K	2510402	LOAD DIGITAL SCAN COMMAND REGISTER FROM A	2,3
LSC K	2510403 3	LOAD SCANNER COMMAND REGISTER FROM A	2,3
SLH I,K	2510410 1,1	SELECT H-REGISTER I, PRINTER K	2,3
SLH I,K	2510417 2,1	SELECT H-REGISTER I, PRINTER K	2,3
RCV K	2510444 3	READ CONVERTER K INTO A	2,3
SLH I,K	2510510 1,2	SELECT H-REGISTER I, PRINTER K	2,3
SLH I,K	2510517 2,2	SELECT H-REGISTER I, PRINTER K	2,3
SLH I,K	2510610 1,3	SELECT H-REGISTER I, PRINTER K	2,3
SLH I,K	2510617 2,3	SELECT H-REGISTER I, PRINTER K	2,3
SLH I,K	2510710 1,4	SELECT H-REGISTER I, PRINTER K	2,3
SLH I,K	2510717 2,4	SELECT H-REGISTER I, PRINTER K	2,3
LAC K	2511000 4	LOAD ACCUMULATOR SCAN COMMAND REGISTER	2,3
LCV K	2511001 4	LOAD CONVERTER REGISTER K FROM A	2,3
LDS K	2511002	LOAD DIGITAL SCAN COMMAND REGISTER FROM A	2,3
LSC K	2511003 4	LOAD SCANNER COMMAND REGISTER FROM A	2,3
OFH I	2511010 1	TURN OFF PRINTERS ON H-REGISTER I	2
OFH I	2511017 2	TURN OFF PRINTERS ON H-REGISTER I	2
RCV K	2511044 4	READ CONVERTER K INTO A	2,3
LDH I,N	2511110 1,1	LOAD H-REGISTER I, BLOCK N, FROM A	2,3
LDH I,N	2511117 2,1	LOAD H-REGISTER I, BLOCK N, FROM A	2,3
LDH I,N	2511210 1,2	LOAD H-REGISTER I, BLOCK N, FROM A	2,3
LDH I,N	2511217 2,2	LOAD H-REGISTER I, BLOCK N, FROM A	2,3
LDH I,N	2511310 1,3	LOAD H-REGISTER I, BLOCK N, FROM A	2,3
LDH I,N	2511317 2,3	LOAD H-REGISTER I, BLOCK N, FROM A	2,3
PRH I,K	2511410 1,1	PRINT ON H-REGISTER I, PRINTER K	2,3
PRH I,K	2511417 2,1	PRINT ON H-REGISTER I, PRINTER K	2,3
PRH I,K	2511510 1,2	PRINT ON H-REGISTER I, PRINTER K	2,3

PRH I,K	2511517	2,2	PRINT ON H-REGISTER I, PRINTER K	2,3
PRH I,K	2511610	1,3	PRINT ON H-REGISTER I, PRINTER K	2,3
PRH I,K	2511617	2,3	PRINT ON H-REGISTER I, PRINTER K	2,3
PRH I,K	2511710	1,4	PRINT ON H-REGISTER I, PRINTER K	2,3
PRH I,K	2511717	2,4	PRINT ON H-REGISTER I, PRINTER K	2,3
LCV K	2512001	5	LOAD CONVERTER REGISTER K FROM A	2,3
LDS K	2512002		LOAD DIGITAL SCAN COMMAND REGISTER FROM A	2,3
LSC K	2512003	5	LOAD SCANNER COMMAND REGISTER FROM A	2,3
RCV K	2512044	5	READ CONVERTER K INTO A	2,3
LAC K	2513000	5	LOAD ACCUMULATOR SCAN COMMAND REGISTER	2,3
BEV J	2514000	1	BRANCH ON EVEN	2
BPL J	2514001	1	BRANCH ON PLUS	2
BZE J	2514002	1	BRANCH ON ZERO	2
BOV J	2514003	1	BRANCH ON OVERFLOW	2
BPC J	2514004	1	BRANCH ON PARITY ERROR, CORE	2
BPD J	2514005	1	BRANCH ON PARITY ERROR, DRUM	2
BBP J,S	2514006	1,N	BRANCH ON BUFFER S PARITY ERROR	2
BBP J,S	2514007	1,M	BRANCH ON BUFFER S PARITY ERROR	2
BBR J,S	2514010	1,N	BRANCH ON BUFFER S READY	2
BBR J,S	2514011	1,M	BRANCH ON BUFFER S READY	2
BCL J	2514012	1	BRABCH ON CLOCK VALID	2
BTC J,K	2514013	1,1	BRANCH ON TIME COUNTER OVERFLOW	2
BTC J,K	2514014	1,2	BRANCH ON TIME COUNTER OVERFLOW	2
BTC J,K	2514015	1,3	BRANCH ON TIME COUNTER OVERFLOW	2
BTC J,K	2514016	1,4	BRANCH ON TIME COUNTER OVERFLOW	2
BRD J	2514017	1	BRANCH ON DEMAND	2
BDC J	2514020	1	BRANCH ON DRUM OPERATION COMPLETE	2
BCO J,K	2514021	1,1	BRANCH ON CONVERTER K OVERFLOW	2
BSC J,K	2514022	1,1	BRANCH ON SCANNER OPERATION COMPLETE	2
BEA J	2514023	1	BRANCH ON ECHO ALARM	2
BRH J,I	2514024	1,1	BRANCH ON H-REGISTER I READY	2
BDT J	2514027	1	BRANCH ON TIMED CONTACT COMPLETE	2
BSC J,K	2514031	1,2	BRANCH ON SCANNER OPERATION COMPLETE	2
BCO J,K	2514032	1,2	BRANCH ON CONVERTER K OVERFLOW	2
BRH J,I	2514033	1,2	BRANCH ON H-REGISTER I READY	2
BDM J	2514034	1	BRANCH ON MULTIPLE OUTPUT COMPLETE	2
BCS J,D,E	2515000	1,0,0	BRANCH ON CONTROLLER SELECTOR, MINIMUM OPERANDS	2
BEV J	2516000	2	BRANCH ON EVEN	2
BPL J	2516001	2	BRANCH ON PLUS	2
BZE J	2516002	2	BRANCH ON ZERO	2
BOV J	2516003	2	BRANCH ON OVERFLOW	2
BPC J	2516004	2	BRANCH ON PARITY ERROR, CORE	2
BPD J	2516005	2	BRANCH ON PARITY ERROR, DRUM	2
BBP J,S	2516006	2,N	BRANCH ON BUFFER S PARITY ERROR	2
BBP J,S	2516007	2,M	BRANCH ON BUFFER S PARITY ERROR	2
BBR J,S	2516010	2,N	BRANCH ON BUFFER S READY	2
BBR J,S	2516011	2,M	BRANCH ON BUFFER S READY	2
BCL J	2516012	2	BRABCH ON CLOCK VALID	2
BTC J,K	2516013	2,1	BRANCH ON TIME COUNTER OVERFLOW	2
BTC J,K	2516014	2,2	BRANCH ON TIME COUNTER OVERFLOW	2
BTC J,K	2516015	2,3	BRANCH ON TIME COUNTER OVERFLOW	2
BTC J,K	2516016	2,4	BRANCH ON TIME COUNTER OVERFLOW	2
BRD J	2516017	2	BRANCH ON DEMAND	2
BDC J	2516020	2	BRANCH ON DRUM OPERATION COMPLETE	2
BCO J,K	2516021	2,1	BRANCH ON CONVERTER K OVERFLOW	2
BSC J,K	2516022	2,1	BRANCH ON SCANNER OPERATION COMPLETE	2
BEA J	2516023	2	BRANCH ON ECHO ALARM	2
BRH J,I	2516024	2,1	BRANCH ON H-REGISTER I READY	2
BDT J	2516027	2	BRANCH ON TIMED CONTACT COMPLETE	2
BSC J,K	2516031	2,2	BRANCH ON SCANNER OPERATION COMPLETE	2
BCO J,K	2516032	2,2	BRANCH ON CONVERTER K OVERFLOW	2
BRH J,I	2516033	2,2	BRANCH ON H-REGISTER I READY	2
BDM J	2516034	2	BRANCH ON MULTIPLE OUTPUT COMPLETE	2
BCS J,D,E	2517307	2,3,7	BRANCH ON CONTROLLER SELECTOR, MAXIMUM OPERANDS	2
BRU Y	2600000	0	BRANCH UNCONDITIONALLY	1
STO Y	2700000	0	STORE OPERAND ADDRESS AT Y	2
XEC Y	3400000	0	EXECUTE THE INSTRUCTION AT Y	1+
JMP Y	3700000	0	JUMP UNCONDITIONALLY TO Y	1
SPJ Y	3740000	0	STORE P AT X2 AND JUMP UNCONDITIONALLY TO Y	2

APPENDIX F. INSTRUCTION FORMATS

Notes

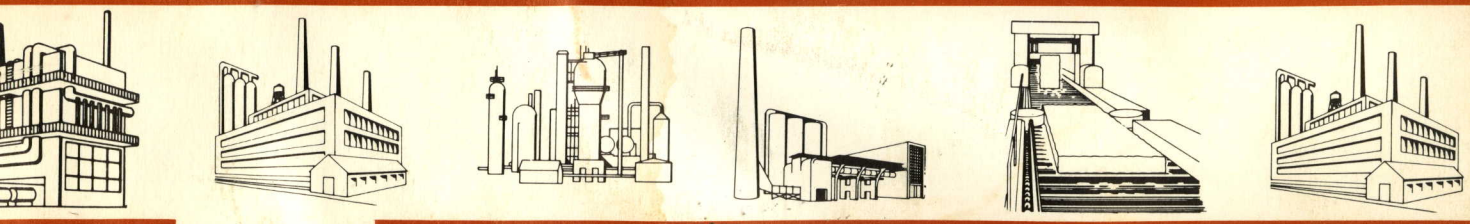
1. For BXH and BXL, K must appear in 2's complement form, e.g. BXH with K=5 and X=2 is 0557773.
2. Octal representation of bits 14-19 for SAI is K+13.
3. Octal representation of bits S, N, K, I depend upon specific command.
4. D is limited to a single bit in positions 9-13 for selection of one of up to 5 scanners or DDA/DFS devices.
5. Model 412B systems only.
6. In model 412B systems, K will be increased by 4096 if bit 7 is a 1 in INX commands.
7. In model 412B systems, Y is limited to values less than 8191 for all instructions except SPB. If the SPB instruction is in a location greater than 8191, Y will be increased by 8192 when the instruction is executed.

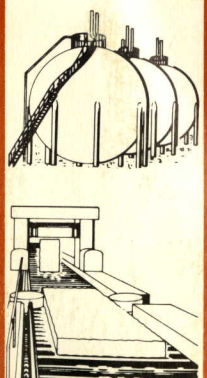
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
Memory Access	Operation Code					X		^{7}Y												
X-Location	Operation Code					X		^{1,6}K												
Right Shift	1	0	1	0	0	X	0	N→A	M→A	Q→A	A→N	A→M	A→Q	A→A	K					
Left Shift	1	0	1	0	0	X	1	0	0	Q→A	0	A→A	Logical	Normalize	K					
Special Device	1	0	1	0	1	X	0	0	0	0	Operation Sub-Code									
															^{2}K					
															^{3}S					
Controller-Selector	1	0	1	0	1	X	0	0	0	1	0	D		Operation Sub-Code						
Data Transfer	1	0	1	0	1	X	0	1	Operation Sub-Code											
External Effect	1	0	1	0	1	X	1	0	^{4}D											
									Operation Sub-Code											
											^{3}N, K				^{3}I					
Conditional Branch	1	0	1	0	1	X	1	1	J	0	Operation Sub-Code									
															^{3}K, S, I					
Controller-Selector Branch	1	0	1	0	1	X	1	1	J	1	0	D		0	0	0	E			
^{5}16 K Branch	1	1	1	1	1	P→X_2	Y													

FOR QUALITY...
PRODUCTIVITY
...PROFIT

PROCESS COMPUTERS
FOR AUTOMATION

PROCESS COMPUTER SECTION PHOENIX, ARIZONA





Progress Is Our Most Important Product

GENERAL ELECTRIC

PCP56A-2M-111

Litho in U.S.A.