

*Reference Manual*

**CONTROL DATA<sup>®</sup>  
1604-C COMPUTER**

**CONTROL DATA**  
CORPORATION

## 1604 INSTRUCTIONS

			Page				Page
00	ZRO	(not used)		40	SST	Selective Set	2-34
01	ARS	A Right Shift	2-14	41	SCL	Selective Clear	2-35
02	QRS	Q Right Shift	2-14	42	SCM	Selective Complement	2-34
03	LRS	AQ Right Shift	2-14	43	SSU	Selective Substitute	2-36
04	ENQ	Enter Q	2-26	44	LDL	Load Logical	2-36
05	ALS	A Left Shift	2-14	45	ADL	Add Logical	2-36
06	QLS	Q Left Shift	2-15	46	SBL	Subtract Logical	2-36
07	LLS	AQ Left Shift	2-15	47	STL	Store Logical	2-36
10	ENA	Enter A	2-26	50	ENI	Enter Index	2-27
11	INA	Increase A	2-26	51	INI	Increase Index	2-27
12	LDA	Load A	2-11	52	LIU	Load Index, U	2-13
13	LAC	Load A, Complement	2-11	53	LIL	Load Index, L	2-13
14	ADD	Add	2-18	54	ISK	Index Skip	2-17
15	SUB	Subtract	2-18	55	IJP	Index Jump	2-17
16	LDQ	Load Q	2-11	56	SIU	Store Index, U	2-13
17	LQC	Load Q, Complement	2-11	57	SIL	Store Index, L	2-13
20	STA	Store A	2-12	60	SAU	Substitute Address, U	2-16
21	STQ	Store Q	2-12	61	SAL	Substitute Address, L	2-16
22	AJP	A Jump	2-28, 31	62	INT	Input Transfer	2-41
23	QJP	Q Jump	2-29, 32	63	OUT	Output Transfer	2-41
24	MUI	Multiply Integer	2-19	64	EQS	Equality Search	2-37
25	DVI	Divide Integer	2-20	65	THS	Threshold Search	2-37
26	MUF	Multiply Fractional	2-21	66	MEQ	Masked Equality	2-38
27	DVF	Divide Fractional	2-21	67	MTH	Masked Threshold	2-38
30	FAD	Floating Add	2-21	70	RAD	Replace Add	2-39
31	FSB	Floating Subtract	2-22	71	RSB	Replace Subtract	2-39
32	FMU	Floating Multiply	2-23	72	RAO	Replace Add One	2-39
33	FDV	Floating Divide	2-24	73	RSO	Replace Subtract One	2-40
34	SCA	Scale A	2-25	74	EXF	External Function	3-3
35	SCQ	Scale AQ	2-25	75	SLJ	Selective Jump	2-30, 32
36	SSK	Storage Skip	2-33	76	SLS	Selective Stop	2-30, 32
37	SSH	Storage Shift	2-33	77	SEV	(not used)	

*Reference Manual*

**CONTROL DATA<sup>®</sup>  
1604-C COMPUTER**

**CONTROL DATA**  
CORPORATION



# C O N T E N T S

## Chapter 1 - Description

1604 Characteristics	1-1
Logical Description	1-2
Storage Section	1-2
Input - Output	1-4
Arithmetic Section	1-4
Control Section	1-6

## Chapter 2 - Description of Instructions

Word Format	2-1
Execution Address	2-2
Address Modification	2-2
Execution of a Pair of Instructions	2-4
Instructions	2-5
Instruction Execution Time	2-6
Order of Instructions	2-7
Data Transmission	2-11
Shifting	2-14
Address Modification	2-16
Arithmetic	2-18
Fixed	2-18
Floating	2-21
Scale	2-25
No Address	2-26
Jumps and Stops	2-28
Normal	2-28
Return	2-30
Storage Test	2-33
Logical	2-34

## CONTENTS (Cont'd)

Storage Search	2-37
Replace	2-39
Transfer	2-40

### Chapter 3 - Input-Output

Methods of Data Exchange	3-1
Transfer Channel	3-1
Buffer Channels	3-1
Initiation and Control of Data Exchange	3-2
Transfer	3-2
Buffer	3-2
Console Input-Output Equipment	3-9
Typewriter	3-10
Paper Tape Reader	3-10
Paper Tape Punch	3-12
Internal EXF Select Instructions	3-13
Internal EXF Sense Instructions	3-14
Console EXF Select Codes	3-14
Input	3-14
Typewriter	3-14
Paper Tape Reader	3-15
Output	3-15
Typewriter	3-15
Paper Tape Punch	3-16
Console EXF Sense Codes	3-16
Typewriter	3-16
Paper Tape Reader	3-17
Paper Tape Punch	3-17
Programmed Examples for Console	3-18

### Chapter 4 - Operations

Description of Indicators and Control Switches	4-1
Reader and Punch Controls	4-6
Auto Load Control	4-7

## CONTENTS (Cont'd)

Operation	4-8
Load Program Entering	4-8
Starting Operation With Pre-Stored Load Program	4-8
Reader	4-9
Punch	4-10
Typewriter	4-11
Tape Transports	4-11
606 Tape Unit	4-12
1607 Tape Unit	4-16
File Protection Ring	4-19
Emergency Procedure	4-20

## Glossary

### Appendix Section

I	Number Systems	1
II	Faults	19
III	Table of Powers of 2	21
IV	Octal - Decimal Integer Conversion Table	22
V	Octal - Decimal Fraction Conversion Table	26
VI	EXF and Character Codes	29
VII	Magnetic Tape BCD Codes	39
VIII	Flexowriter Codes	40
IX	Punched Card Codes	41
X	Input/Output Typewriter Codes	42
XI	1612 Printer Codes	43

## FIGURES

### Chapter 1 - Description

1-1	Typical 1604 System	1-5
-----	---------------------	-----

### Chapter 3 - Input Output

3-1	Recognition of I/O Requests	3-6
3-2	Seven Level Punched Paper Tape	3-11

### Chapter 4 - Operation

4-1	Center Panel of Console	4-1
4-2	Console Display	4-2
4-3	Manual Controls	4-5
4-4	Reader and Punch Controls	4-6
4-5	Paper Tape Reader	4-9
4-6	Paper Tape Punch	4-10
4-7	Operator Control Panel	4-12
4-8	606 Tape Load and Unload Mechanics	4-14
4-9	1607 Tape Unit	4-18
4-10	File Protection Ring	4-20

## TABLES

### Chapter 1 - Description

1-1	Registers of the Computer	1-3
1-2	Arithmetic Properties of Registers	1-3

### Chapter 3 - Input-Output

3-1	Typical Interrupt Routine	3-8
-----	---------------------------	-----

### Chapter 4 - Operation

4-1	Conditions Indicated by Console Background Lights	4-3
4-2	Main Computer Controls	4-4
4-3	Reader and Punch Controls	4-6
4-4	606 Controls and Indicators	4-12
4-5	1607 Controls and Indicators	4-17
4-6	Emergency Procedure	4-20



## PREFACE

This manual describes the characteristics, instructions, and manual controls of the CONTROL DATA\* 1604 computer. Throughout the manual "1604" designation is used meaning "1604-C" which has become the official designation. Instruction Descriptions in this manual are also applicable to the CONTROL DATA\* 1604-B computer.

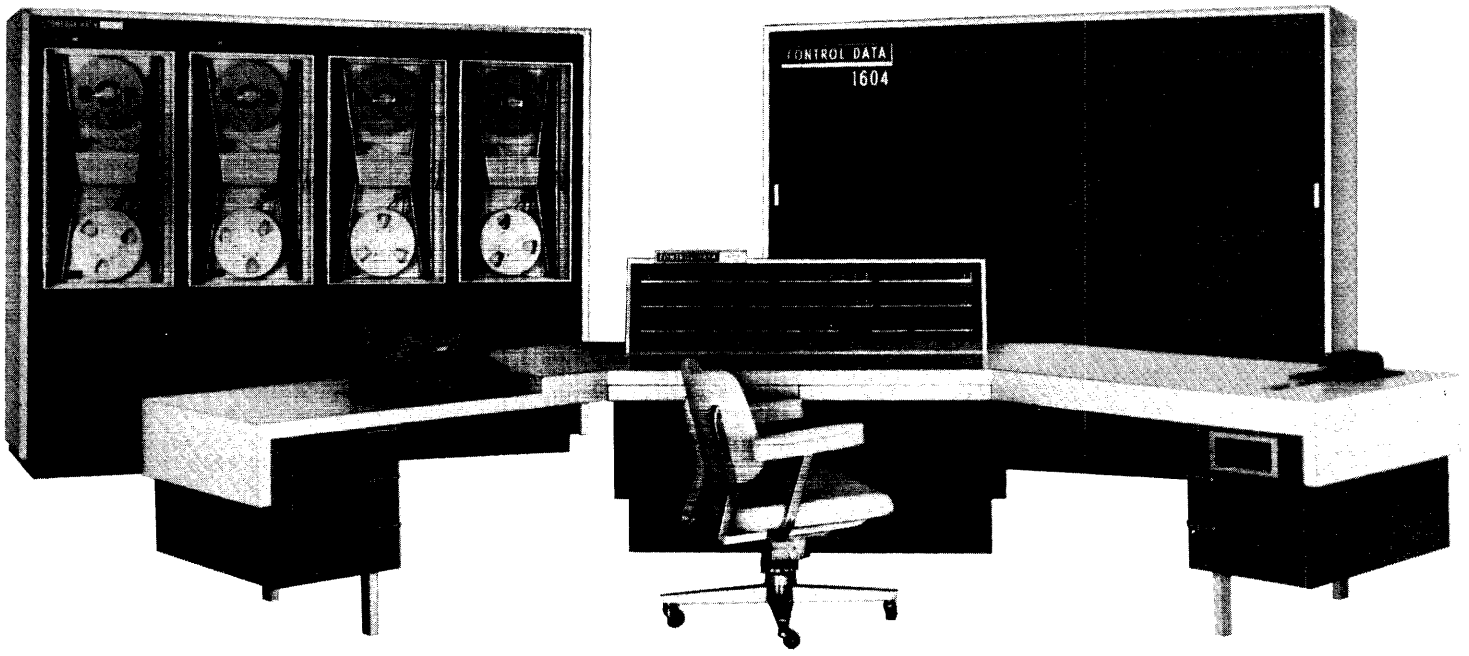
---

\*Registered trademark of Control Data Corporation





# Description



The CONTROL DATA\* 1604 is a stored-program, general-purpose digital computer with a large storage capacity, exceedingly fast computation and transfer speeds, and special provisions for input-output communication. The 1604 is designed to handle large-volume data processing and to solve large scale scientific problems. The compact equipment, constructed from solid-state components throughout, is suitable for use in a semi-permanent office environment.

## 1604 CHARACTERISTICS

Stored-program general-purpose digital computer	Program interrupt
Parallel mode of operation	Console, includes:
48-bit word, 2 instructions per word	Photo-electric paper tape reader
Single address logic	Paper tape punch
Operation code                   6 bits	Electric typewriter
Designator                        3 bits	Register contents displayed in octal
Base Execution Address   15 bits	
Six 15-bit index registers	Flexible instructions
Indirect addressing	Fixed-point arithmetic
Magnetic core storage	(integer and fractional)
32,768 48-bit words	Floating-point arithmetic
Two independent 16,384 word banks alternately phased	Logical and masking operations
4.8 usec effective cycle time (representative program)	Indexing
6.4 usec total cycle time	Storage searching
Input-output	Binary arithmetic
Transmission of 48-bit words	Parallel addition in 1.2 usec without access
Three separate buffer input channels	Modulus $2^{48}-1$ (one's complement)
Three separate buffer output channels	Real-time clock
High-speed transfer channel	Completely solid-state
4.8 usec per word	Diode logic
	Transistor amplifiers

---

\* Registered trademark of Control Data Corporation

## LOGICAL DESCRIPTION

The computer performs calculations and processes data in a parallel binary mode through the step-by-step execution of individual instructions which are stored internally along with the data.

Functionally, the computer may be divided into four major sections. INPUT-OUTPUT provides communication between the computer and the external equipment; ARITHMETIC performs the arithmetic and logical operations required for executing instructions; STORAGE provides internal storage for data and instructions; and CONTROL coordinates and sequences all operations for executing an instruction by obtaining the instruction from storage and translating it into commands for the other sections.

The registers in the computer are identified by letters (table 1-1). The arithmetic properties of the registers are detailed in table 1-2. The operational registers usually hold the end result of an operation. Their contents are displayed on the console and may be manually changed by the operator. The transient registers used in formulating the result are secondary registers. They are not displayed and cannot be manually changed.

### STORAGE SECTION

The magnetic core storage section of the 1604 Computer provides high-speed, random access storage for 32,768 words. It consists of two independent storage units each with a capacity of 16,384 words. These units operate together during the execution of a stored program and thus are considered as one 32,768 word storage system.

A word is 48 bits in length and is used in two ways: as two 24-bit instructions or as a 48-bit operand (data word). The location of each word in storage is identified by an assigned number or address. When a word is taken (read) from or entered (written) into storage, a reference is made to the storage address which holds the word. All odd storage addresses are located in one storage unit; all even addresses in the other.

The cycle time, or time for a complete storage reference, is 6.4 microseconds. Since the storage cycles of the two sections overlap one another in the execution of a program, the average effective cycle time for random addresses is about 4.8 microseconds.

TABLE 1-1. REGISTERS OF THE COMPUTER

Register	Function	Register	Function
A*	Accumulator	U <sup>1*</sup>	Program Control
Q*	Auxiliary Arithmetic	U <sup>2</sup>	Auxiliary Program Control
B <sup>1</sup> through B <sup>6*</sup>	Index registers (six)	R	Address Buffer
P*	Program Address	X	Exchange

\* Operational Registers

TABLE 1-2. ARITHMETIC PROPERTIES OF REGISTERS

Register	No. of Stages	Modulus	Complement Notation*	Arithmetic	Result
A	48	$2^{48}-1$	one's	subtractive	signed**
Q	48	$2^{48}-1$	one's		signed
U <sup>2</sup>	15	$2^{15}-1$	one's	subtractive	signed
P	15	$2^{15}$	two's	additive	unsigned
R	15	$2^{15}$	two's	subtractive	unsigned

\* Refer to Appendix

\*\*The result of an arithmetic operation in A satisfies  $A \leq 2^{47}-1$  since A always is treated as a signed quantity. When the result in A is zero, it is always represented by 000...00 except when 111...11 is added to 111...11. In this case, the result is 111...11 (negative zero).

## INPUT-OUTPUT

The input-output section of the computer handles the flow of information to and from the computer. Prior to executing a program, the data and instructions which comprise the program (input) are loaded into computer storage. After computation is completed, the results (output) are transmitted from storage to an external equipment. All information is transmitted by way of the X register in the form of 48-bit words.

The computer communicates with external equipment through six independent buffer channels which provide for the normal exchange of data (figure 1-1).

Input: Channel 1	Output: Channel 2
Channel 3	Channel 4
Channel 5	Channel 6

The input and output buffer channels are paired, channels 1 and 2, channels 3 and 4, and channels 5 and 6. Every external equipment is connected to one of these pairs. It is possible to connect as many as eight different equipments to any given pair of channels. All six buffer channels may be concurrently transmitting information. However, only one external equipment can use any one buffer channel at any given instant.

In the 1604 computer, input-output operations are independent of the main computer program. When data is transmitted, the main computer program initiates an automatic cycle which buffers data to and from computer storage. The main computer program then continues while the actual buffering of data is carried out independently and automatically.

This process of asynchronous input-output operations will be termed a buffer. Buffer transmissions employ independent access to computer storage because computation continues while the external equipment is loading or unloading information from computer storage at a rate dictated by the external equipment.

## ARITHMETIC SECTION

The arithmetic section of the 1604 computer consists of two operational registers, A and Q, and one secondary register, X.

The A register (accumulator) is the principal arithmetic register. Some of the more important functions of A are:



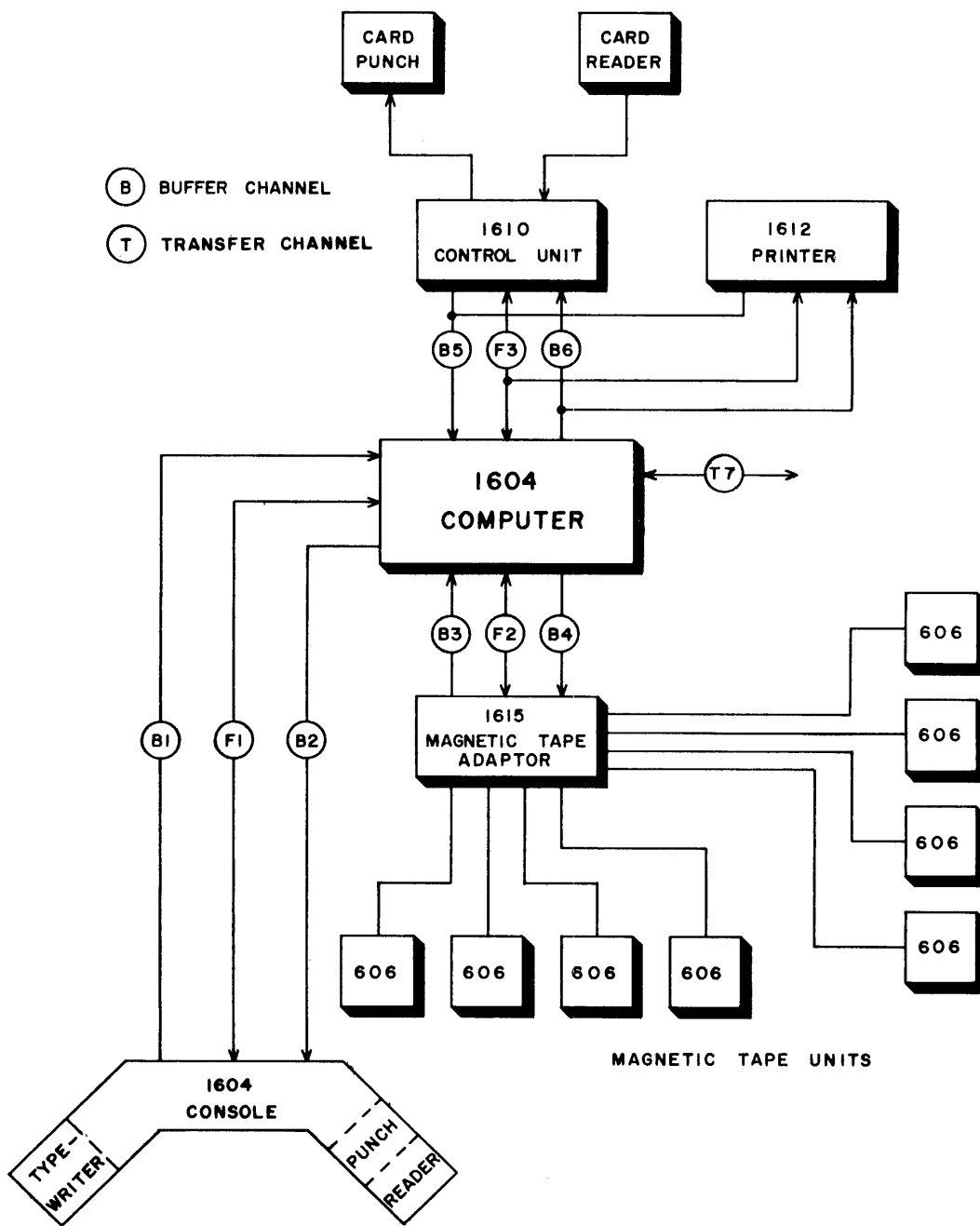


Figure 1-1. Typical 1604 System

- 1) Arithmetic operation - A initially holds one of the operands in addition, subtraction, multiplication and division. The result is usually held in A.
- 2) Shifting - A may be shifted to the right or left separately or in conjunction with Q. Right shifting is open-ended; the lowest bits are discarded and sign extended. Left shifting is circular; the highest order bit appears in the lowest order stage after each shift; all other bits move one place to the left.
- 3) Control for conditional instructions - A holds the word which conditions jump and search instructions.

The Q register is an auxiliary arithmetic register and is generally used in conjunction with the A register. The principal functions of Q are:

- 1) Providing temporary storage of contents of A while A is used for another arithmetic operation.
- 2) Forming a double-length register, AQ or QA.
- 3) Shifting to the right or left, separately or in conjunction with A.
- 4) Participating with the A register in multiplication, division and logical product operations (masking).

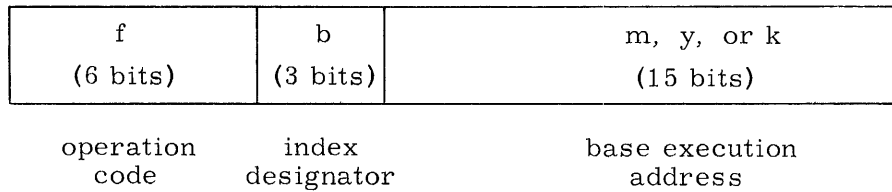
The exchange, or X register, is the communication center of the computer. All internal transmissions between the arithmetic section or the input-output section and the rest of the computer are made through X.

## **CONTROL SECTION**

The control section directs the operations required to execute instructions and to exchange data with external equipment. It also establishes the timing relationships needed to perform the operations in the proper sequence.

The control section acquires a program step from storage, interprets it and sends the necessary commands to other sections. A program step is a pair of 24-bit instructions which together occupy one storage location as a 48-bit word. The higher-order 24 bits are the upper instruction; the remaining 24 bits, the lower instruction.

### Instruction Format



Each of the 62 instructions has a unique 6-bit operation code which specifies the operation to be performed.

The index designator generally specifies one of the six index registers whose content is to be added to the execution address. This process is called address modification. However, the index designator may also specify indirect addressing or condition jump and stop instructions.

The execution address may be used in one of three ways: as an address, m, of an operand; as an operand, y; or as a shift count, k.

The eight operational registers in the control section are P,  $U^1$  and  $B^1$  through  $B^6$ .

The program address counter (P) is a two's complement additive register. It provides program continuity by generating in sequence the storage addresses which contain the individual program steps. Usually at the completion of each program step the count in P is advanced by one to specify the address of the next program step.

The program control register ( $U^1$ ) holds a program step while the two instructions contained in it are executed. The upper instruction is usually executed first followed by the lower instruction. After executing an instruction, a half exit, full exit, or jump exit is performed. A half exit always allows the lower instruction of a program step to be executed. A full exit advances the count in P by one and executes the upper instruction of the new program step specified by the contents of P. A jump exit allows a new sequence of instructions to be executed; the storage location of the new instruction is specified by the execution address of the jump instruction. The execution address, in this case, is entered into P and specifies the starting location of a new sequence of program steps.

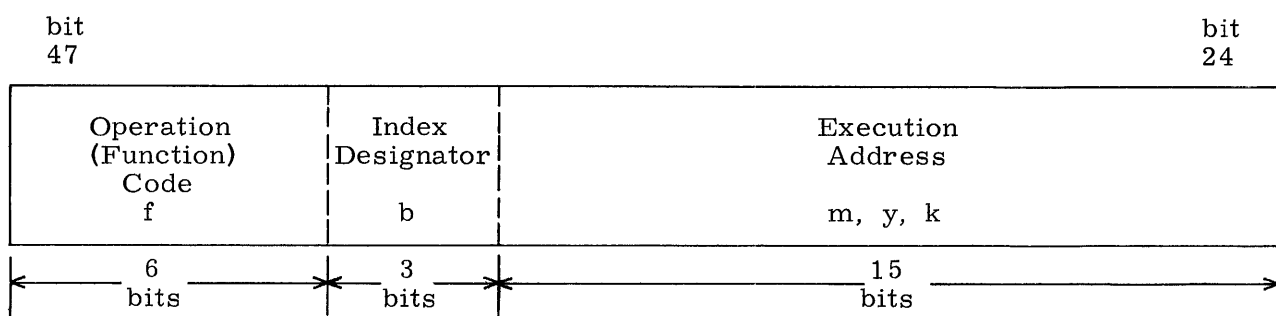
The auxiliary program control register ( $U^2$ ) is a 15-bit subtractive accumulator used primarily in the modification of the base execution address. The contents of the specified index register are transmitted to the address buffer register (R), which has provisions for counting, complementing and storing. The contents of R are then added to the contents of  $U^2$  which holds the execution address.



# Description of Instructions

## WORD FORMAT

A computer word consists of 48 bits and may be interpreted as one 48-bit data word or two 24-bit instructions. Each instruction is composed of three parts or codes: operation code, index designator, and execution address. The higher-order 24 bits of the word are called the upper instruction and the lower-order 24 bits are called the lower instruction.



Code	Range	
Operation f	01 - 76 <sub>8</sub>	Specifies the operation to be performed. A 00 or 77 code is interpreted as a fault which stops computer operation.
Execution Address m, y, k	00000 through 77777 <sub>8</sub>	Used in one of three ways: 1) as a shift count k 2) as an operand address, m 3) as an operand, y
Index Designator b	0 1-6  7	No address modification Relative address modification Specifies the index designator whose contents are to be added to the execution address. (Refer to jump and stop instructions for exceptions.) Indirect addressing

## Execution Address

The base execution address may be used as: (1) a shift count,  $k$ ; (2) an operand,  $y$ ; (3) an address of an operand,  $m$ , in storage. The execution address may also be modified or unmodified depending on the index designator. If unmodified, the address is represented by the lower-case symbol  $k$ ,  $y$ , or  $m$ ; if the address is modified the symbols are capitalized. The following examples point out the relationship between the unmodified and modified execution address.

The modified shift count  $K$  is represented by:

$$1) \quad K = k + (B^b) \text{ where:} \quad \begin{array}{l} K \quad = \text{modified shift count} \\ k \quad = \text{unmodified shift count (execution address)} \\ (B^b) = \text{contents of index register } b. \end{array}$$

If the index designator = 0, then  $K = k$ .

The modified operand  $Y$  is represented by:

$$2) \quad Y = y + (B^b) \text{ where:} \quad \begin{array}{l} Y \quad = \text{modified operand} \\ y \quad = \text{unmodified operand (execution address)} \\ (B^b) = \text{contents of index register } b. \end{array}$$

If the index designator = 0, then  $Y = y$ .

The modified operand address  $M$  is represented by:

$$3) \quad M = m + (B^b) \text{ where:} \quad \begin{array}{l} M \quad = \text{modified address of operand} \\ m \quad = \text{unmodified address of operand ( execution address)} \\ (B^b) = \text{contents of index register } b. \end{array}$$

If the index designator = 0, then  $M = m$ . Note that (3) is the only case in which the execution address is interpreted as an address of an operand.

## Address Modification

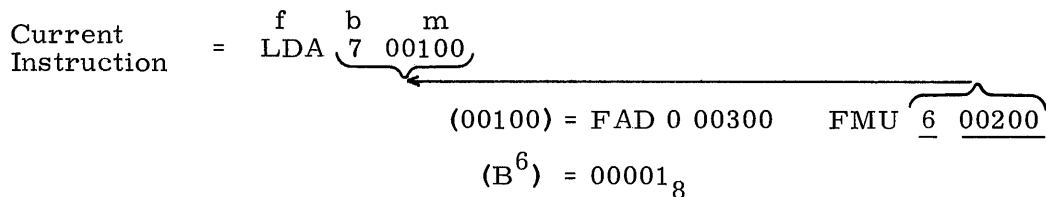
The three possible modes of address modification are identified by the index designators as follows:

- 1)  $b = 0$  No Address Modification. In this mode the execution address is interpreted without modification; nothing is added to or subtracted from it. (Direct addressing.)

- 2)  $b = 1-6$  Relative Address Modification. In this mode the execution address is modified and is equal to the initial execution address plus the contents of the designated index register. One's complement arithmetic is used in determining the modified execution address.
- 3)  $b = 7$  Indirect Addressing. In this mode a storage reference is made to the location designated by the execution address. The 48-bit word is read from storage and the lower-order 18 bits of the word are interpreted as the  $b$  designator (3 bits) and execution address (15 bits) of the present instruction. The new index designator may refer to any one of the three modes.

Examples:

- 1) No Address Modification  $\begin{matrix} f & b & m \\ \text{LDA } 0 & \text{address} \end{matrix}$   
 This instruction is interpreted as load accumulator from the storage location designated by the sum of the execution address and the contents of the specified index register,  $B^b$ . Since  $b = 0$ , no index register is designated and  $m$  specifies the storage location whose contents are loaded into A.
- 2) Relative Address Modification  $\begin{matrix} f & b & m \\ \text{LDA } 6 & \text{address} \end{matrix}$   $(B^6) = 00001_8$   
 In this example, the accumulator is loaded from the storage location designated by the execution address plus the contents of index register 6. Therefore, the contents of the storage location named by the execution address plus  $00001_8$  is loaded into the accumulator.  $M = m + (B^b)$ .
- 3) Indirect Addressing



When the  $b$  designator of the current instruction is 7, the mode is indirect addressing. The lower 18 bits of the contents of the storage location designated by the execution address, 00100, are read from storage into the  $U^1$  register where they are interpreted as the index designator and execution address of the current instruction.



The index designator is inspected again and because it is not 0 or 7 the relative address mode exists. (Note that the new index designator could reference any one of the three modes of address modification.) The execution address, 00200, plus the contents of  $B^6$ ,  $00001_8$  specify the storage location whose contents will be loaded into the accumulator.  $M = 00200_8 + (00001_8) = 00201_8$

### Execution of a Pair of Instructions

Example:

	f	b	m		f	b	m
(00300) = LDA	0	00310		ADD	1	00210	
(00301) = STA	0	00400		SLS	0	00301	
		$(B^1) = 00101_8$					

The P register holds address 00300 (an even lowest bit indicates the address of the program step is in the even storage unit). The storage reference is initiated; the 48-bit word is read from address 00300 and entered into  $U^1$ . Computer operation is now dependent upon the interpretation of the 24-bit instruction in the upper half of  $U^1$ .

The operation code, LDA, and the index designator, 0, are translated. The function of the upper instruction, LDA, is to load the A register with the contents of the designated storage location. Because the index designator is 0, the execution address is not modified. The translation of the operation code initiates the sequence of the commands which execute the instruction and the operand in address 00310 is loaded into A.

The lower instruction in  $U^1$  is now translated. The ADD instruction causes the quantity in storage location M to be added to the contents of the A register. Since the index designator is not 0 or 7, the contents of the index register are added to the execution address to form M.  $M = m + (B^b) = 00210_8 + 00101_8 = 00311_8$ . The contents of storage address 00311 are added to the contents of the A register completing the instruction. The contents of the P register are increased by one and the next pair of instructions at address 00301 is read from storage and executed.

# INSTRUCTIONS

The 62 computer instructions are described on the following pages, (EXF instructions are discussed in detail in chapter three). The title line contains the numeric code, the mnemonic code and format, name, and average execution time of the instruction. Abbreviations and symbols are defined as follows:

A	Accumulator
$A_n$	The binary digit in position n of the A register
→	Transmit to
b	Index designator
$B^b$	Designated index register
Exit (Full)	Proceed to upper instruction of next program step
Half exit	Proceed to lower instruction of same program step
j	The condition designator for jump and stop instructions
k	Unmodified shift count
K	Modified shift count. $K = k + (B^b)$
LA	Lower address - execution address portion of lower instruction of a program step.
m	Unmodified operand address
M	Modified operand address. $M = m + (B^b)$
( )	Contents of a register or storage location
( )'	One's complement contents of a register or storage location
( )f	Final contents of a register or storage location
( )i	Initial contents of a register or storage location
Q	Auxiliary arithmetic register
UA	Upper address
X	Exchange register
y	Unmodified operand
Y	Modified operand. $Y = y + (B^b)$

## **Instruction Execution Time**

The time needed to execute an instruction varies from application to application because of the following factors.

If the instruction occupies the upper position in an instruction word, the time needed to read the word from storage must be considered.

If consecutive storage references are made to the same storage unit (even-even or odd-odd) the read access time from storage will be maximized.

If indirect addressing is specified, at least one additional reference will be needed to complete the instruction. (The new index designator may itself specify indirect addressing.)

If an input-output request exists, the request will, in most cases, be processed before the next instruction is executed. (Refer to chapter three.)

If a storage reference is made at the end of the preceding instruction, execution of the next instruction may be delayed.

The instruction execution times listed on the following pages were compiled by averaging the times for a long list of the same instructions. The list was arranged for typical values of the factors.

## ORDER OF INSTRUCTIONS

Numeric Code	Mnemonic Code	Name	Timing*
<b>DATA TRANSMISSION</b>			
12	LDA	LOAD A	} 7.2
13	LAC	LOAD A COMPLEMENT	
16	LDQ	LOAD Q	
17	LQC	LOAD Q COMPLEMENT	
20	STA	STORE A	
21	STQ	STORE Q	
52	LIU	LOAD INDEX (UPPER)	
53	LIL	LOAD INDEX (LOWER)	
56	SIU	STORE INDEX (UPPER)	
57	SIL	STORE INDEX (LOWER)	
<b>SHIFTING</b>			
01	ARS	A RIGHT SHIFT	} 2.8 + .4s**
02	QRS	Q RIGHT SHIFT	
03	LRS	AQ RIGHT SHIFT	
05	ALS	A LEFT SHIFT	
06	QLS	Q LEFT SHIFT	
07	LLS	AQ LEFT SHIFT	
<b>ADDRESS MODIFICATION</b>			
60	SAU	SUBSTITUTE ADDRESS (UPPER)	7.2
61	SAL	SUBSTITUTE ADDRESS (LOWER)	7.2
54	ISK	INDEX SKIP	5.6
55	IJP	INDEX JUMP	4.4

\* Timing is average execution time in usec

\*\* s = Number of places shifted

## ORDER OF INSTRUCTIONS

### ARITHMETIC (Fixed)

14	ADD	ADD	7.2
15	SUB	SUBTRACT	7.2
24	MUI	MULTIPLY INTEGER	$25.2 + .8n^*$
25	DVI	DIVIDE INTEGER	65.2
26	MUF	MULTIPLY FRACTIONAL	$25.2 + .8n^*$
27	DVF	DIVIDE FRACTIONAL	65.2

### ARITHMETIC (Floating)

30	FAD	FLOATING ADD	18.8
31	FSB	FLOATING SUBTRACT	18.8
32	FMU	FLOATING MULTIPLY	36.0
33	FDV	FLOATING DIVIDE	56.0
34	SCA	SCALE A	$2.8 + .4s^{**}$
35	SCQ	SCALE AQ	$2.8 + .4s^{**}$

### NO ADDRESS

04	ENQ	ENTER Q	} 3.0
10	ENA	ENTER A	
11	INA	INCREASE A	
50	ENI	ENTER INDEX	
51	INI	INCREASE INDEX	

### JUMPS AND STOPS (Normal)

22	AJP	A JUMP	} 7.2
23	QJP	Q JUMP	
75	SLJ	SELECTIVE JUMP	
76	SLS	SELECTIVE STOP	

\* n = Number of ones in multiplier

\*\* s = Number of positions shifted

## ORDER OF INSTRUCTIONS

### JUMPS AND STOPS (Return)

22	AJP	A JUMP	}	
23	QJP	Q JUMP		
75	SLJ	SELECTIVE JUMP		
76	SLS	SELECTIVE STOP		

7.2

### STORAGE TEST

36	SSK	STORAGE SKIP		8.8
37	SSH	STORAGE SHIFT		12.8

### LOGICAL

40	SST	SELECTIVE SET	}	
42	SCM	SELECTIVE COMPLEMENT		
41	SCL	SELECTIVE CLEAR		
43	SSU	SELECTIVE SUBSTITUTE		
44	LDL	LOAD LOGICAL	}	
45	ADL	ADD LOGICAL		
46	SBL	SUBTRACT LOGICAL		
47	STL	STORE LOGICAL		

7.2

7.4

7.2

### STORAGE SEARCH

64	EQS	EQUALITY SEARCH	}	
65	THS	THRESHOLD SEARCH		
66	MEQ	MASKED EQUALITY		
67	MTH	MASKED THRESHOLD		

4.0 + 3.6r\*

---

\* r = Number of repeated executions

## ORDER OF INSTRUCTIONS

### REPLACE

70	RAD	REPLACE ADD
71	RSB	REPLACE SUBTRACT
72	RAO	REPLACE ADD ONE
73	RSO	REPLACE SUBTRACT ONE

} 13.2

### TRANSFER

62	INT	INPUT TRANSFER
63	OUT	OUTPUT TRANSFER

} 4.0 + 4.8r\*

---

\* r = Number of repeated executions

# DATA TRANSMISSION

- 1) Relative address modification does not take place during LIU, LIL, SIU or SIL instructions. Indirect addressing will occur if b=7. A b=0 condition effects a pass.
- 2) All modes of address modification apply to the remaining data transmission instructions.
- 3) During the execution of data transmission instructions, one storage reference is made. If indirect addressing is designated, at least two storage references are made.

**LDA b m** 12          Load A

7.2 usec

Replaces the contents of A with a 48-bit operand contained in storage location M. The initial contents of A are changed during execution; the contents of M remain unchanged.

**LAC b m** 13          Load A Complement

7.2 usec

Replaces the contents of A with the complement of a 48-bit operand contained in storage location M. The initial contents of A are changed during execution; the contents of M remain unchanged.

**LDQ b m** 16          Load Q

7.2 usec

Replaces the contents of Q with a 48-bit operand contained in storage location M. The initial contents of Q are changed during execution; the contents of address M remain unchanged.

**LQC b m** 17          Load Q Complement

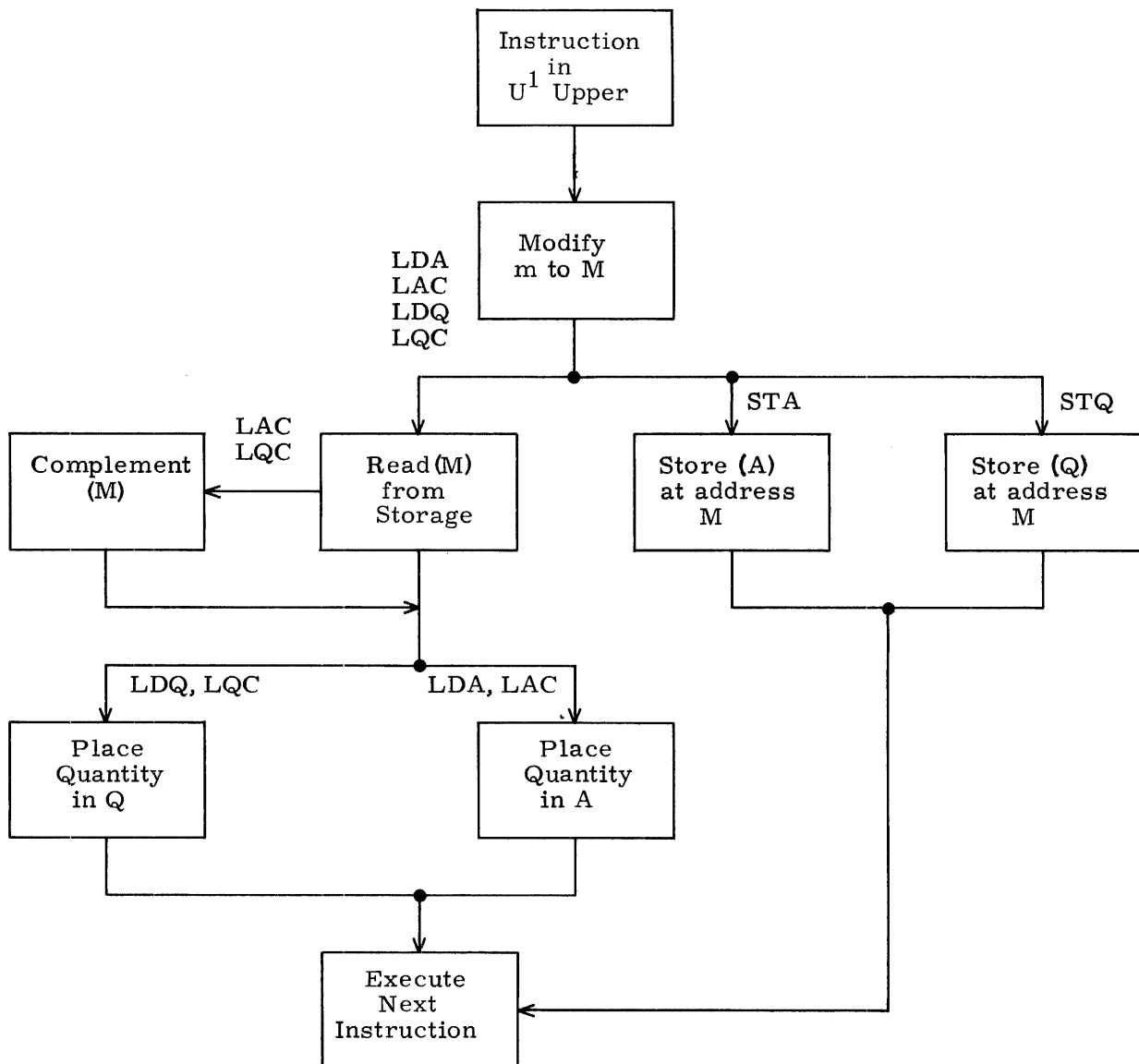
7.2 usec

Replaces the contents of Q with the complement of a 48-bit operand contained in storage location M. The initial contents of Q are changed during execution; the contents of address M remain unchanged.



**STA b m** 20 Store A 7.2 usec  
 Replaces the contents of the designated storage location, M, with the contents of A. The initial contents of A remain unchanged.

**STQ b m** 21 Store Q 7.2 usec  
 Replaces the contents of the designated storage location, M, with the contents of Q. The initial contents of Q remain unchanged.



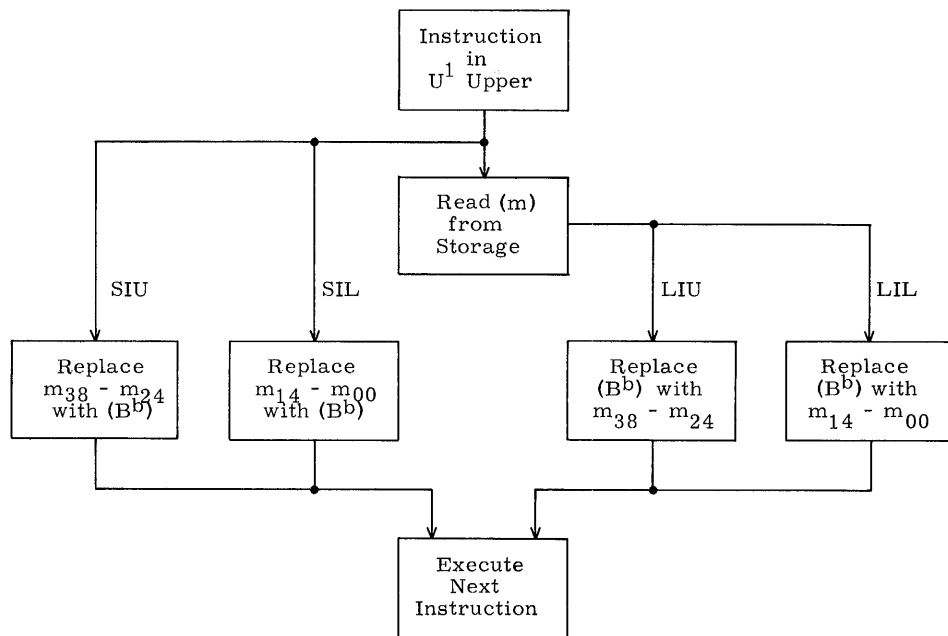
**LDA, LAC, LDQ, LQC, STA, and STQ**

**LIU b m** 52 Load Index Upper 7.2 usec  
 Replaces the contents of the designated index register with the upper address portion of storage location m. If  $b = 0$  this instruction becomes a pass (do nothing) instruction. Initial contents of m remain unchanged.

**LIL b m** 53 Load Index Lower 7.2 usec  
 Replaces the contents of the designated index register with the lower address portion of storage location m. If  $b = 0$  this instruction becomes a pass (do nothing) instruction. Initial contents of m remain unchanged.

**SIU b m** 56 Store Index Upper 7.2 usec  
 Replaces the upper address portion of storage location m with the contents of the designated index register. The remaining bits of the word in storage remain unchanged. If  $b = 0$ ,  $(m_{ua})$  is cleared. Initial contents of  $B^b$  remain unchanged.

**SIL b m** 57 Store Index Lower 7.2 usec  
 Replaces the lower address portion of storage location m with the contents of the designated index register. The remaining bits of the word in storage remain unchanged. If  $b = 0$ ,  $(m_{la})$  is cleared. Initial contents of  $B^b$  remain unchanged.



**LIU, LIL, SIU, and SIL**

# SHIFTING

- 1) All modes of address modification apply to these instructions.
- 2) If the modified shift count, K, is greater than  $127_{10}$ , a fault indicator is set. Regardless of the magnitude of count, however, the required number of shifts is executed. (K is reduced by one count for each shift executed and when K = 0, shifting stops.)
- 3) Shifting must be completed before an input-output or interrupt request can be processed. (See chapter three.)

**ARS b k** 01          A Right Shift     2.8 + .4s\* usec

Shifts contents of A to the right K places. The sign is extended and the lower bits are discarded. The largest practical shift count is  $47_{10}$  since the register is now an extension of the sign bit.

**QRS b k** 02          Q Right Shift     2.8 + .4s usec

Shifts contents of Q to the right K places. The sign is extended and the lower bits are discarded. The largest practical shift count is  $47_{10}$  since the register is now an extension of the sign bit.

**LRS b k** 03          Long Right Shift     2.8 + .4s usec

Shifts contents of AQ to the right K places as one 96-bit register. The A register is considered as the leftmost 48 bits and the Q register as the rightmost 48 bits. The sign of A is extended. The lower order bits of A replace the higher order bits of Q and the lower order bits of Q are discarded. The largest practical shift count is  $95_{10}$  since AQ is now an extension of the sign of A.

**ALS b k** 05          A Left Shift     2.8 + .4s usec

Shifts contents of A to the left K places, left circular. The higher order bits of A replace the lower order bits. The largest practical shift count  $48_{10}$  returns the register to its original state.

---

\* s = Number of positions shifted

**QLS b k** 06 Q Left Shift

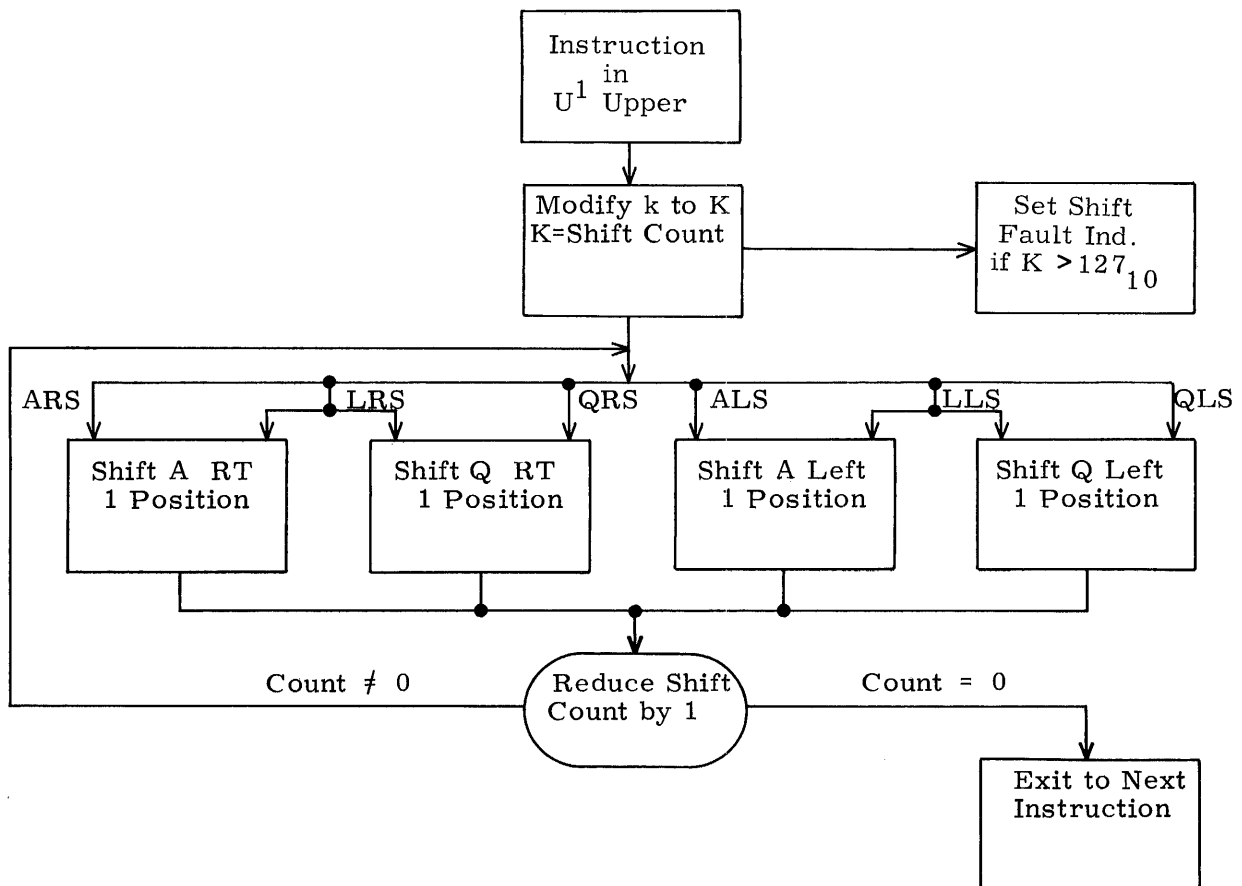
2.8 + .4s usec

Shifts contents of Q to the left K places, left circular. The higher order bits of Q replace the lower order bits. The largest practical shift count  $48_{10}$  returns the register to its original state.

**LLS b k** 07 Long Left Shift

2.8 + .4s usec

Shifts contents of AQ to the left K places, left circular, as one 96-bit register. The higher order bits of A replace the lower order bits of Q and the higher order bits of Q replace the lower order bits of A. The largest practical shift count  $96_{10}$  returns AQ to its original state.



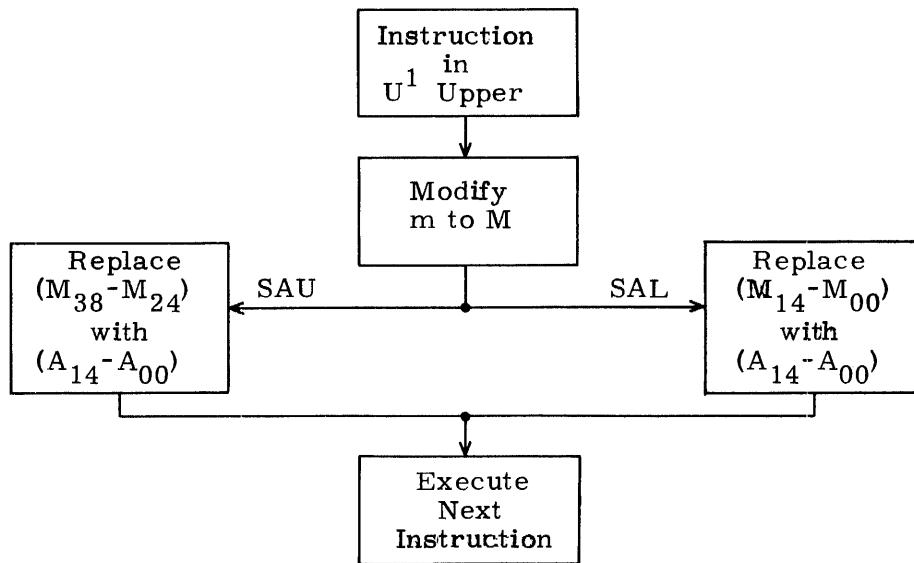
### Shift Instructions

# ADDRESS MODIFICATION

- 1) All modes of address modification apply to SAU and SAL instructions.
- 2) Relative address modification does not take place during ISK or IJP instructions. Indirect addressing occurs if  $b=7$ , and a pass occurs if  $b=0$ .
- 3) During execution of ISK and IJP instructions, no storage reference is made unless indirect addressing is specified which requires at least one reference. For SAU and SAL instructions, one reference is always made. If indirect addressing is designated, at least one additional reference will be needed to complete the instruction.

**SAU  $b m$  60**      Substitute Address Upper      7.2 usec  
 Replaces the upper address portion of M with the lower order 15 bits of A.  
 Remaining bits of M are not modified and the initial contents of A are unchanged.

**SAL  $b m$  61**      Substitute Address Lower      7.2 usec  
 Replaces the lower address portion of M with the lower order 15 bits of A.  
 Remaining bits of M are not modified and the initial contents of A are unchanged.



**SAU and SAL**

**ISK b y** 54

Index Skip

5.6 usec

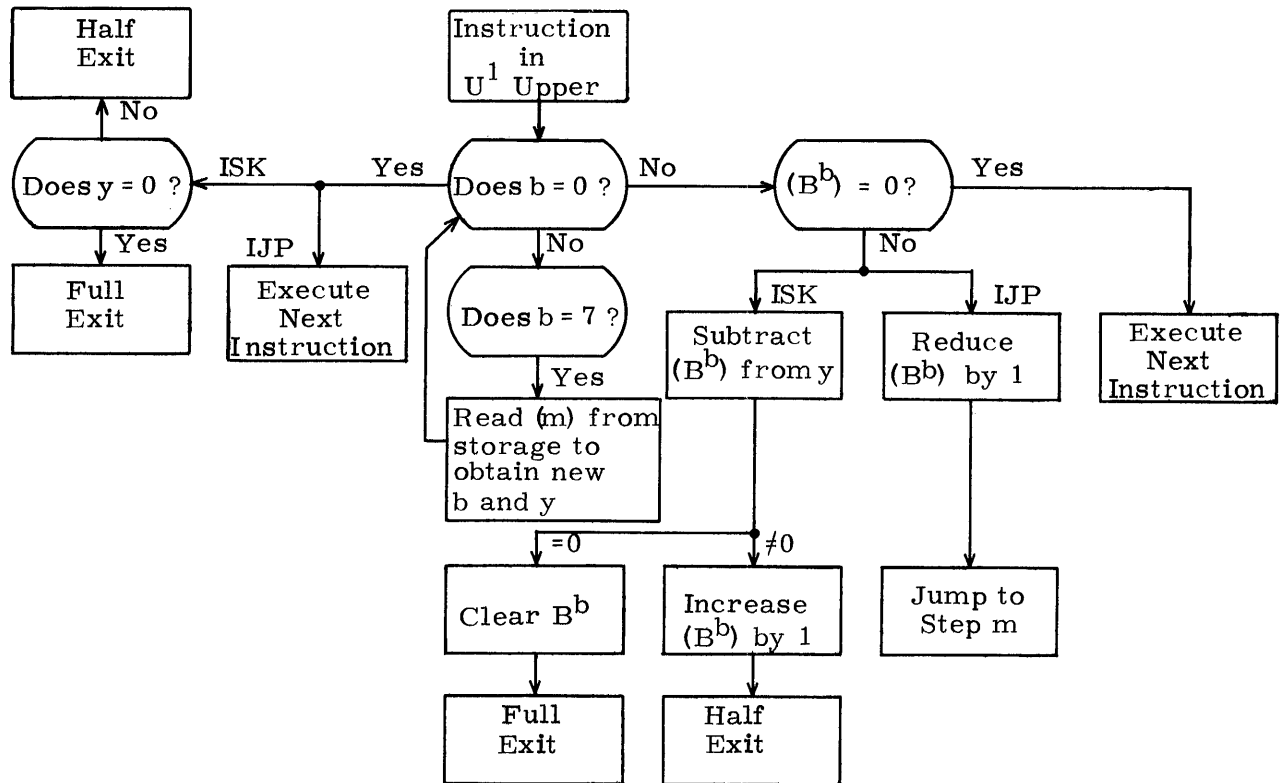
Compares  $(B^b)$  with  $y$ . If the two quantities are equal,  $B^b$  is cleared and a full exit is performed. If the quantities are unequal,  $(B^b)$  is increased one count in the R register and a half exit is performed. Because the R register is a two's complement subtractive counter, it is possible to count through negative zero and positive zero. (See appendix.) If  $b = 0$  and  $y \neq 0$ , a half exit is taken. If  $b = 0$  and  $y = 0$ , a full exit is taken. ISK is usually restricted to the upper instruction. If used as a lower instruction it will half exit upon itself until the full exit condition is satisfied. If  $b = 0$  and  $y \neq 0$ , the condition will never be satisfied.

**IJP b m** 55

Index Jump

4.4 usec

Examines  $(B^b)$ . If this quantity is not zero, the quantity is reduced one count and a jump is executed to program step  $m$ . The counting operation is performed in the R register but negative zero is not generated because IJP terminates at positive zero. (See appendix.) The index jump can be used in the upper or lower instruction without reservation; it executes a normal jump upon satisfaction of the jump condition.



**ISK and IJP**

# ARITHMETIC

- 1) All modes of address modification apply to these instructions.
- 2) One storage reference is made for each instruction unless indirect addressing is designated. In this case, at least two references are made.
- 3) If the capacity of the A register  $\pm (2^{47}-1)$  is exceeded during the execution of the instructions an arithmetic overflow fault is produced. When executing the DVI or DVF instructions, if the result exceeds the capacity of the Q register  $\pm (2^{47}-1)$  a divide fault is produced. (Refer to appendix.)

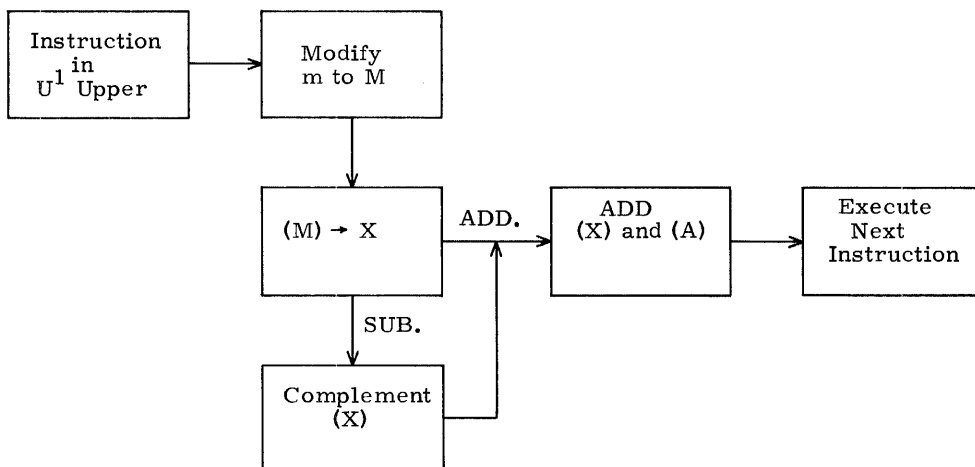
## FIXED

**ADD b m** 14                      Add    7.2 usec

Adds a 48-bit operand obtained from storage location M to contents of A. A negative zero may be produced by this instruction if (A) and (M) are initially negative zero. The contents of storage address M remain unchanged.

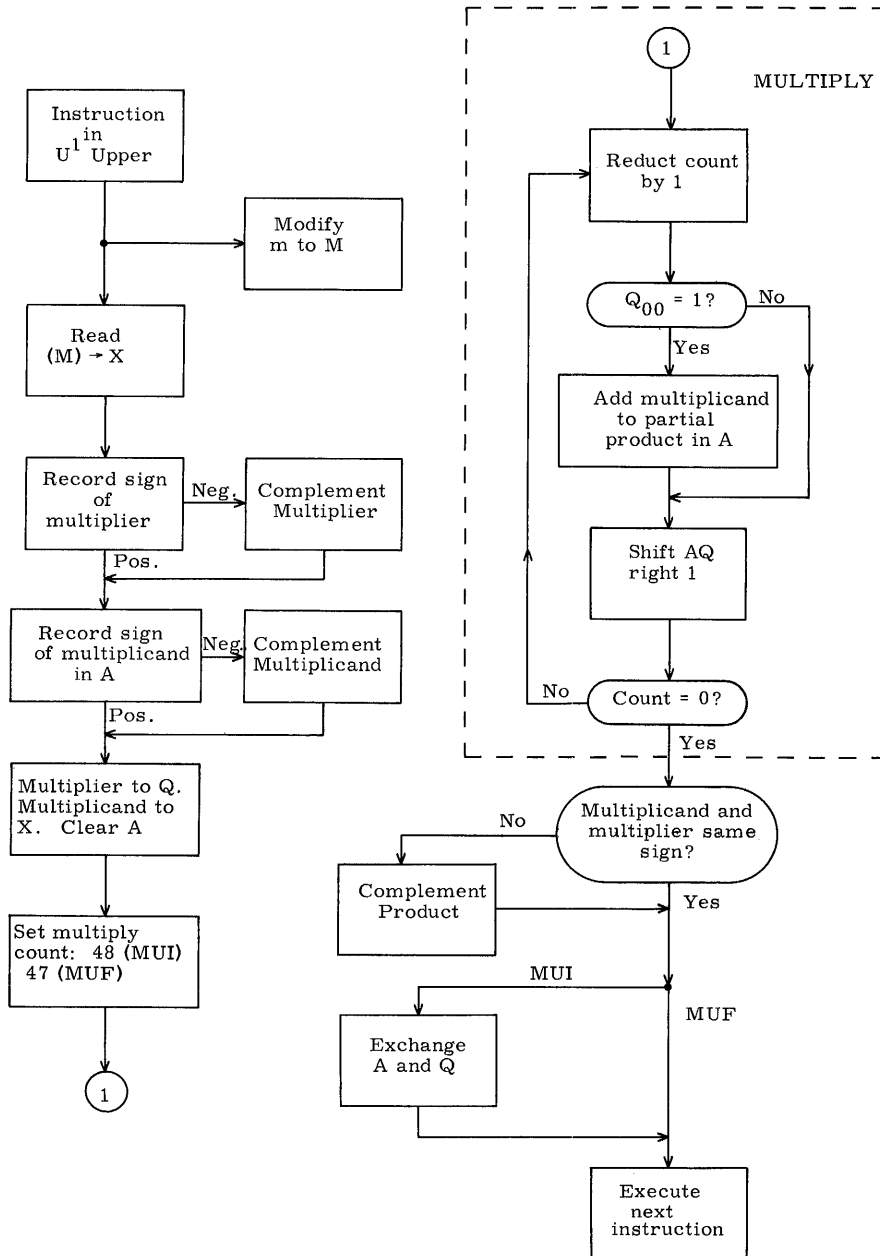
**SUB b m** 15                      Subtract    7.2 usec

Obtains a 48-bit operand from storage location M and subtracts it from the initial contents of A. A negative zero will be produced if the initial contents of A are negative zero and that of storage location M are positive zero. The contents of address M remain unchanged.



## ADD and SUB

Forms a 96-bit product from two 48-bit operands. The multiplier must be loaded into A prior to execution of the instruction. The execution address specifies the storage location of the multiplicand. The product is contained in QA as a 96-bit quantity. The operands are considered as integers and therefore the binary point is assumed to be at the lower order (right hand) end of the A register.

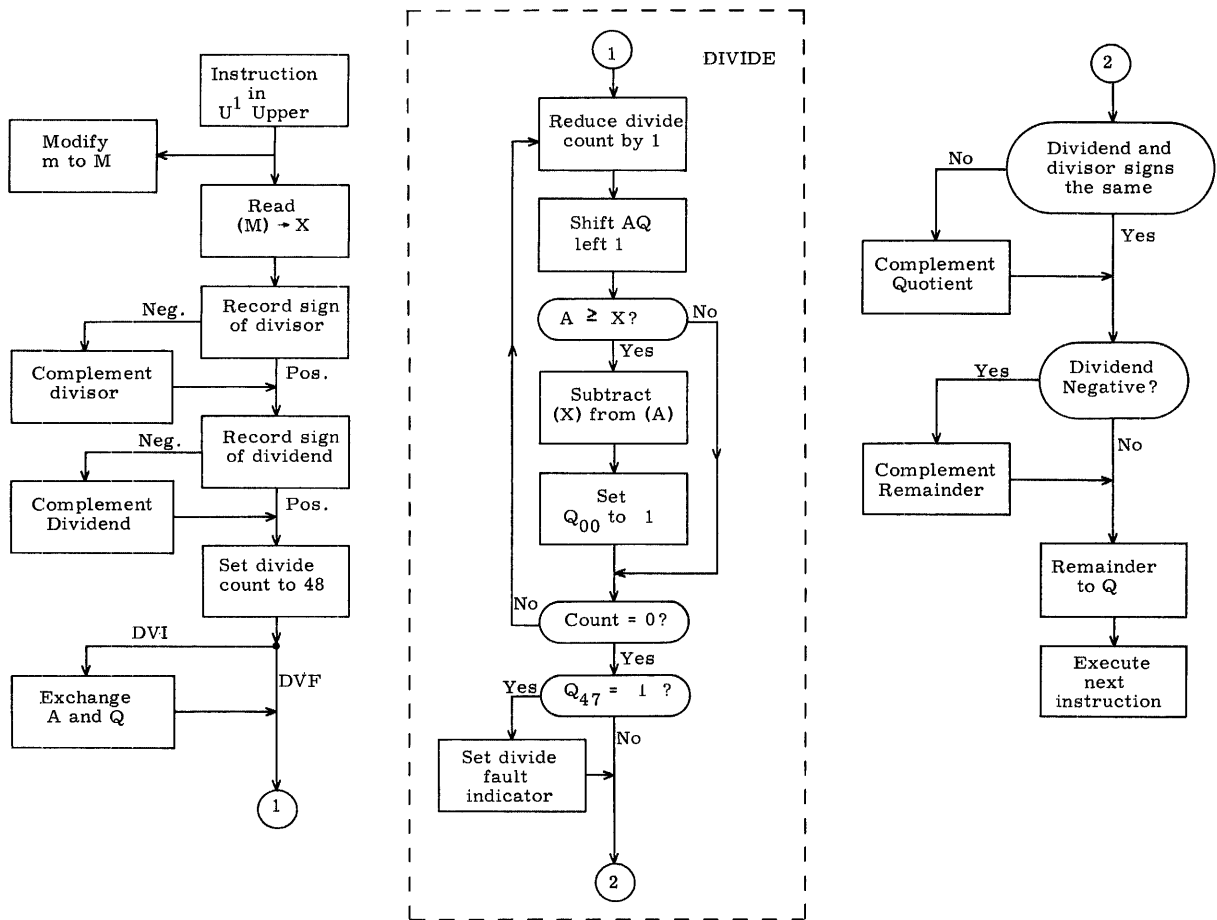


**MUI and MUF**

\* n = Number of ones in multiplier



Divides a 96-bit integer dividend by a 48-bit integer divisor. The 96-bit dividend must be formed in the QA register prior to executing the instruction. If a 48-bit dividend is loaded into A, the sign of Q must be set. That is, the sign of the dividend in A must be extended throughout Q. The 48-bit divisor is read from the storage location specified by the execution address. The quotient is formed in A and the remainder is left in Q at the end of the operation. Dividend and remainder have the same sign.



**DVI and DVF**

**MUF b m** 26            Multiply Fractional                            25.2 + .8n\* usec

Forms a 96-bit product from two 48-bit operands. The operands are treated as fractions with the binary point immediately to the right of the sign bit. The multiplier must be loaded into A prior to executing the instruction. The multiplicand is read into X from the storage location specified by M. The 96-bit product is contained in AQ.

**DVF b m** 27            Divide Fractional                                    65.2 usec

Divides a 96-bit quantity by a 48-bit divisor. All operands are treated as fractions with the binary point immediately to the right of the sign bit. The 96-bit dividend must be loaded into AQ prior to executing this instruction. If a 48-bit dividend is loaded into Q, the sign of Q must be extended throughout A. At the end of this operation the quotient is left in A and the remainder in Q. Remainder and dividend have the same sign.

1) Refer to appendix for a discussion of floating point format.

2) All modes of address modification apply.

## FLOATING

3) One storage reference is made unless indirect addressing is designated. In this case, at least two references are made.

4) Floating point range faults (overflow-underflow) occur if the exponent exceeds  $2^{10}-1$  in absolute value. Refer to appendix.

**FAD b m** 30            Floating Add    18.8 usec

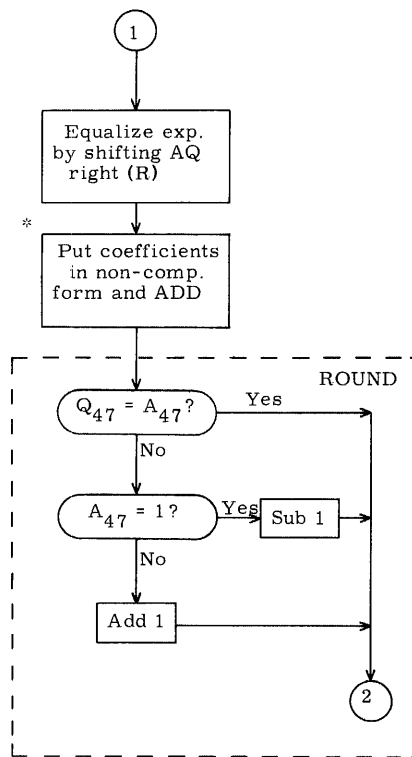
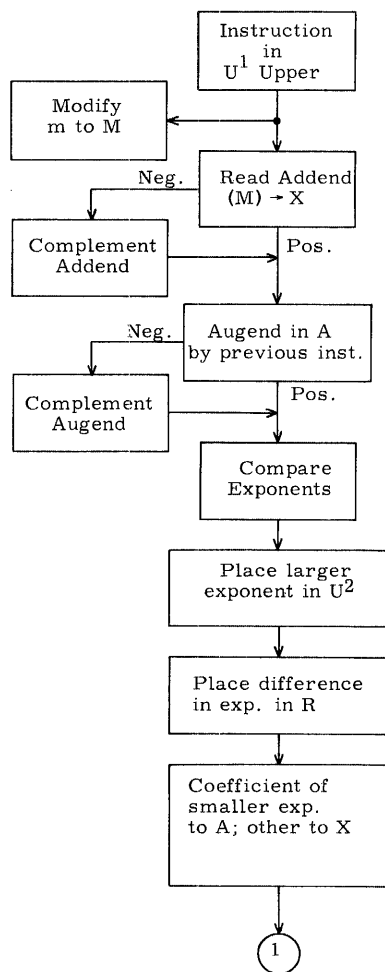
Forms the sum of two operands packed in floating point format. A floating point operand is read from storage location M and added to the floating point word in A. The result is normalized, rounded, and retained in A at the end of the operation. Q contains only the residue of the rounding operation at the end of the sequence.

---

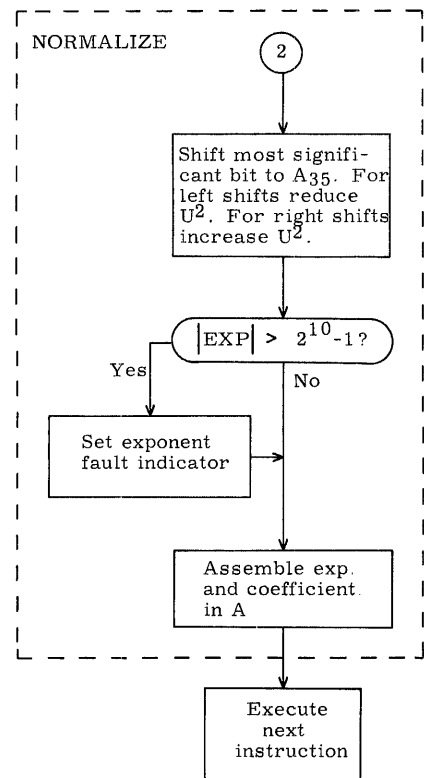
\* n = Number of ones in multiplier

Forms the difference of two 48-bit operands in floating point format. The subtrahend is acquired from storage address M and is subtracted from the minuend in A. The result is rounded and normalized if necessary and retained in A. The residue from the rounding operation is left in Q at the end of the sequence.

The basic steps executed in a FSB are the same as those for FAD except the coefficients are subtracted rather than added.

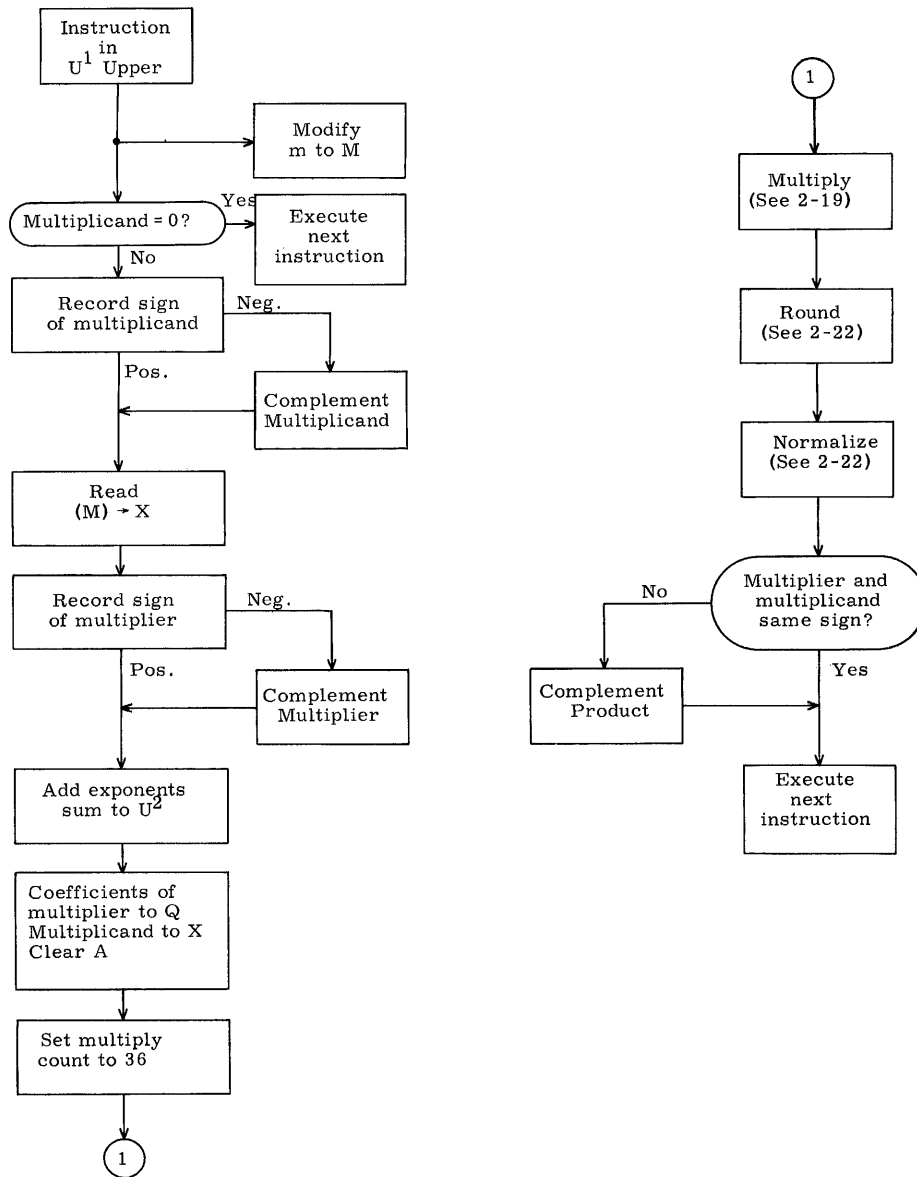


\* FSB the coefficients are subtracted

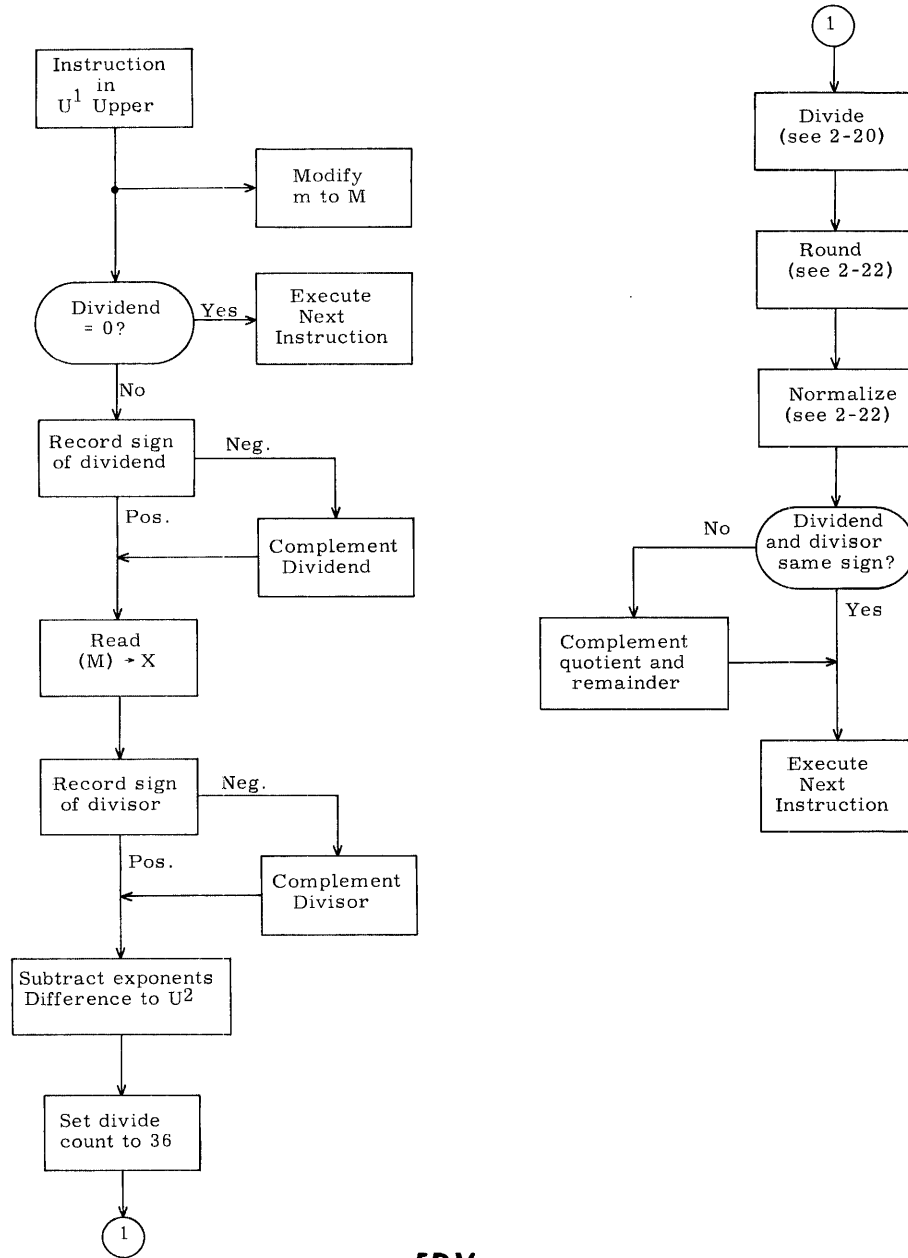


**FAD and FSB**

Forms the product of an operand in floating point format with the previous contents of A also in floating point format. The operand is read from storage location M. The product is rounded and normalized if necessary and retained in A. The residue from the rounding operation is left in Q at the end of the sequence.



Forms the quotient of two 48-bit operands in floating point format. The dividend must be loaded into A prior to executing this instruction. The divisor is read from the storage location specified by M. The quotient is rounded and normalized if necessary and retained in A at the end of the operation. The residue from the rounding operation is left in Q at the end of the operation.



FDV

- 1) Address modification does not apply. Rather, the index register is used to preserve the scale factor.
- 2) If  $b = 0$ , scaling is executed but the scale factor is lost.
- 3) If  $b = 7$ , indirect addressing is used and at least one storage reference is made.
- 4) If  $(A)_i$  is already scaled or equal to positive or negative zero,  $k \rightarrow B^b$  and scaling is not executed.
- 5) If the execution address is initially equal to 0,  $B^b$  is cleared and no scaling takes place.
- 6) The shift fault indicator is not affected by this instruction.

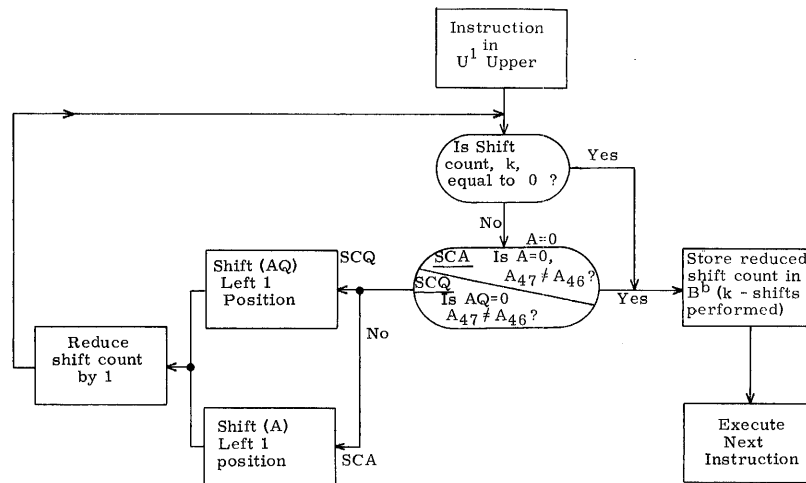
**SCALE**

**SCA b k** 34                      Scale A    2.8 + .4s\* usec

Shifts A left circularly until the most significant digit is to the right of the sign bit or until  $k = 0$ . Shift count  $k$  is reduced by one for each shift and terminates when  $k = 0$  or the most significant digit is to the right of the sign bit. Upon termination the count (scale factor) is entered in the designated index register.

**SCQ b k** 35                      Scale AQ    2.8 + .4s usec

Shifts AQ left circularly until the most significant digit is to the right of the sign bit. Shift count  $k$  is reduced by one for each shift. Operation terminates when  $k = 0$  or the most significant digit is to the right of the sign bit. Upon termination the count (scale factor) is entered in the designated index register.



**SCA and SCQ**

\* s = Number of positions shifted

# NO ADDRESS

- 1) All modes of address modification apply to ENQ, ENA, and INA instructions.
- 2) Address modification does not take place during ENI and INI instructions. The only mode recognized is indirect addressing.
- 3) No storage reference is made during these instructions unless indirect addressing is designated. In this case, at least one storage reference is made.

**ENQ b y** 04                      Enter Q    3.0 usec

The 15-bit operand, Y, is entered into Q and its highest order bit is extended in the remaining 33 bits. The largest positive 15-bit operand that can be entered into Q is  $37777_8$  ( $2^{14}-1$ ) and its "0" bit will be duplicated in each of the remaining 33 bits of Q. Negative zero will be formed in Q if:

- 1)  $(B^b) = 77777_8$  and  $y = 77777_8$  or
- 2)  $b = 0$  and  $y = 77777_8$ .

**ENA b y** 10                      Enter A    3.0 usec

The 15-bit operand, Y, is entered into the A register and its highest order bit is extended in the remaining 33 bits. The largest positive 15-bit operand that can be entered into A is  $37777_8$  ( $2^{14}-1$ ) and the "0" bit will be duplicated in each of the remaining 33 bits. Negative zero will be formed in A if:

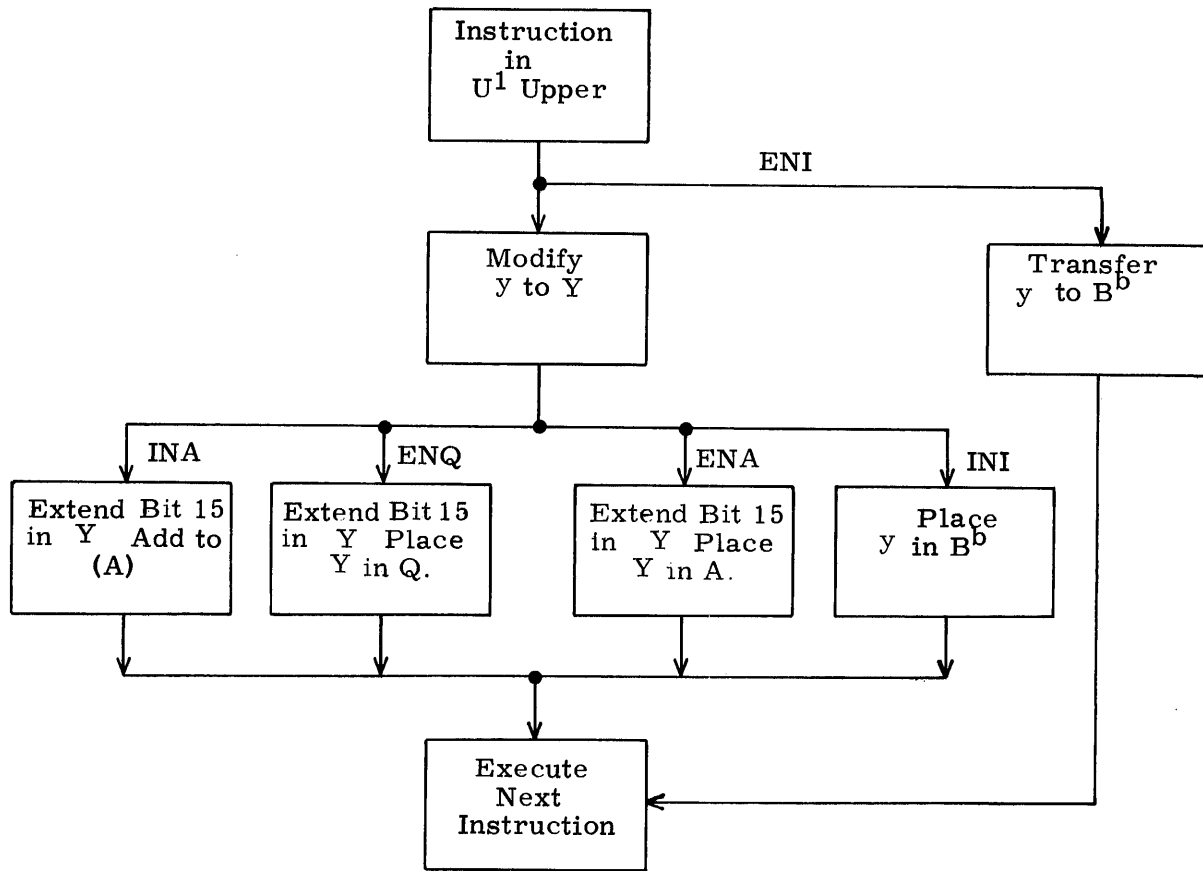
- 1)  $(B^b) = 77777_8$  and  $y = 77777_8$  or
- 2)  $b = 0$  and  $y = 77777_8$ .

**INA b y** 11                      Increase A    3.0 usec

Adds Y to A. The 15-bit operand Y is placed in X and its highest order bit is extended in the remaining 33 bits. The operand in X is added to (A).

**ENI b y** 50 Enter Index 3.0 usec  
 Replaces ( $B^b$ ) with the operand y. If b = 0, this instruction becomes a pass (do nothing) instruction.

**INI b y** 51 Increase Index 3.0 usec  
 Increases ( $B^b$ ) by the operand y. If the b designator is zero, this instruction becomes a pass (do nothing) instruction.



**No Address**



# JUMPS AND STOPS

1) Address modification does not apply to these instructions.

## **NORMAL**

2) One storage reference is made.

## **Jump**

A jump instruction causes a current program sequence to terminate and initiates a new sequence at a different location in storage. The Program Address register, P, provides the continuity between program steps and always contains the storage location of the current program step.

When a jump instruction occurs, P is cleared and a new address is entered. In all jump instructions, the execution address, m, specifies the beginning address of the new program sequence. The word at address m is read from storage, placed in  $U^1$  and the upper instruction (first instruction of the new sequence) is executed.

Some of the jump instructions are conditional upon a register containing a specific value or upon the position of an operator's jump or stop key on the console. If the criterion is satisfied, the jump is made to location m. If it is not satisfied, the program proceeds in its regular sequence to the next instruction.

A jump instruction may appear in either position in a program step. If the jump instruction appears in the first (upper) part of the program step and the jump is taken, the second (lower) part of the program step is not executed. If the instruction appears in the lower part, the upper part is executed in the normal manner.

**AJP j m** 22 A Jump 7.2 usec

Jumps to m if the conditions of the A register specified by the jump designator, j, exist. If not, the next instruction is executed.

j = 0 Jump if (A) = 0

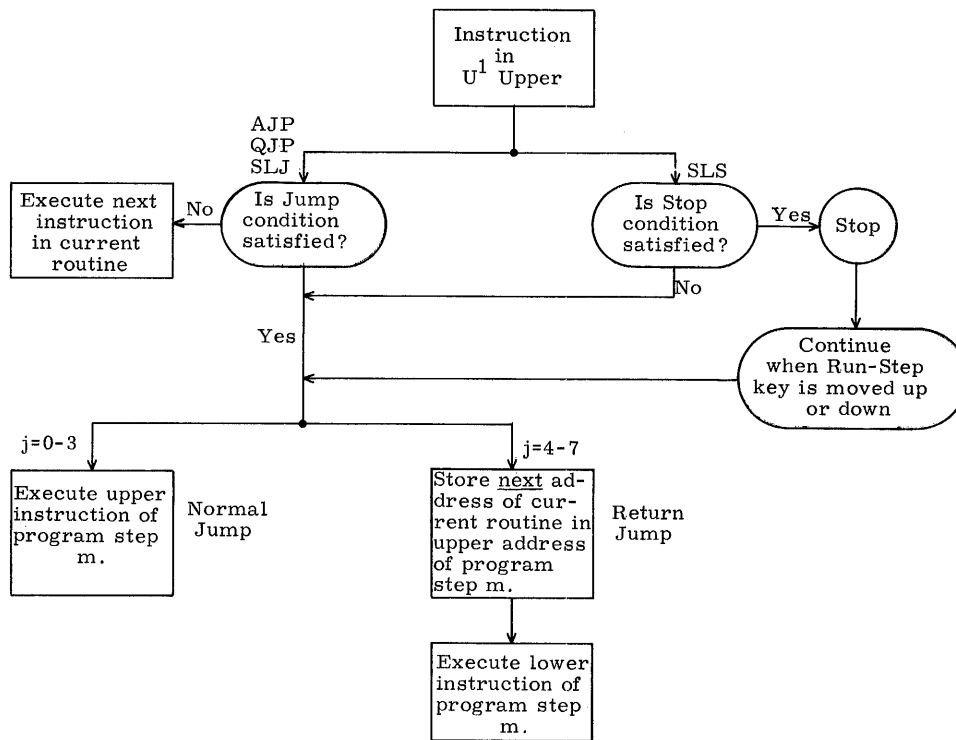
j = 1 Jump if (A)  $\neq$  0

j = 2 Jump if (A) = +

j = 3 Jump if (A) = -

When (A) is negative zero the interpretation is:

- j = 0 The jump is executed because, in this case, negative zero is recognized as positive zero.
- j = 1 The jump is not executed.
- j = 2 The jump is not executed because the sign bit is a "1".
- j = 3 The jump is executed because the sign bit is a "1".



### AJP, QJP, SLJ, and SLS

**QJP j m** 23

Q Jump

7.2 usec

Jumps to m if the condition of the Q register specified by the jump designator, j, exists. If not, the next instruction is executed.

- j = 0 Jump if (Q) = 0
- j = 1 Jump if (Q) ≠ 0
- j = 2 Jump if (Q) = +
- j = 3 Jump if (Q) = -

When (Q) is negative zero the AJP interpretation applies.

**SLJ j m**

75                      Selective Jump                                      7.2 usec

Jumps to m if the condition of the jump keys specified by j exists. If not, the next instruction is executed.

- j = 0 Jump unconditionally
- j = 1 Jump if jump key 1 is set
- j = 2 Jump if jump key 2 is set
- j = 3 Jump if jump key 3 is set

**SLS j m**      76                      Selective Stop                                      7.2 usec

Stops at present step in the sequence if the condition of the stop key specified by j exists. If the stop condition exists, the stop is executed, and the jump is executed unconditionally when the Run-Step key is moved to the run or step position. If the stop condition is not satisfied, the jump is executed unconditionally.

- j = 0 Stop unconditionally
- j = 1 Stop if stop key 1 is set
- j = 2 Stop if stop key 2 is set
- j = 3 Stop if stop key 3 is set

- 1) Address modification does not apply to these instructions.

**RETURN JUMP**

- 2) One storage reference is made.

A return jump begins a new program sequence at the lower instruction portion of the program step to which the jump is made. At the same time, the address portion of the upper instruction of that program step is replaced with the address of the next program step in the main program. This instruction allows a return to the main program after completing the subprogram sequence.

**AJP j m** 22

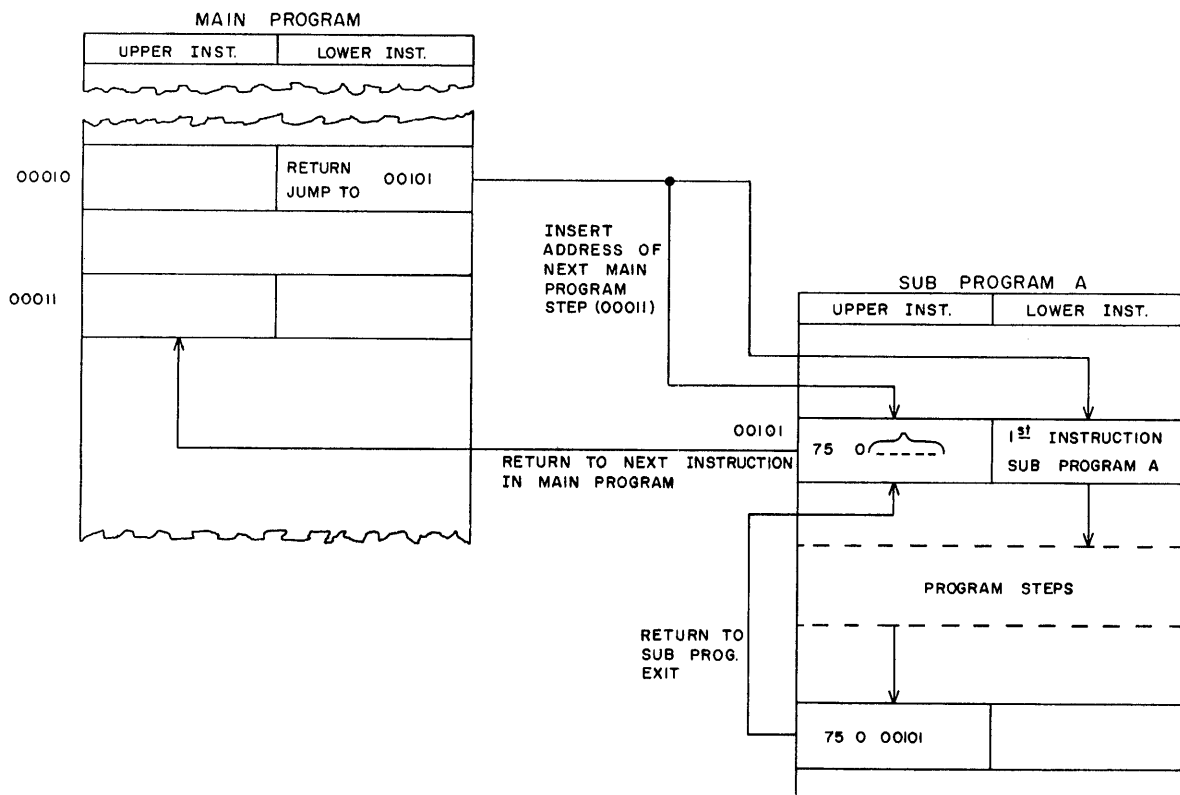
A Jump

7.2 usec

Executes a return jump to storage location m if the condition of the A register specified by j exists. If not, the next instruction is executed.

- j = 4 Return jump if (A) = 0
- j = 5 Return jump if (A) ≠ 0
- j = 6 Return jump if (A) = +
- j = 7 Return jump if (A) = -

Note: If (A) = negative zero, refer to the AJP instruction.



### Return Jump

**QJP j m** 23 Q Jump 7.2 usec

Executes a return jump to storage location m if the condition of the Q register specified by j exists. If not, the next instruction is executed.

j = 4 Return jump if (Q) = 0

j = 5 Return jump if (Q) ≠ 0

j = 6 Return jump if (Q) = +

j = 7 Return jump if (Q) = -

Note: If (Q) = negative zero, refer to the AJP instruction.

**SLJ j m** 75 Selective Jump 7.2 usec

Executes a return jump to storage location m on condition j where condition j represents the setting of the jump keys. If the condition is not satisfied, the next instruction is executed.

j = 4 Return jump unconditionally

j = 5 Return jump if jump key 1 is set

j = 6 Return jump if jump key 2 is set

j = 7 Return jump if jump key 3 is set

Note: The set position of a jump key is in the up position.

**SLS j m** 76 Selective Stop 7.2 usec

Stops on condition j and executes a return jump to storage location m if the Run-Step key is moved in the run or step position. If the stop condition is satisfied, the stop is executed and the return jump is executed when the Run-Step key is moved in either position. If the stop condition is not satisfied, the stop is not executed and the return jump is executed unconditionally.

j = 4 Stop unconditionally

j = 5 Stop if stop key 1 is set

j = 6 Stop if stop key 2 is set

j = 7 Stop if stop key 3 is set

# STORAGE TEST

- 1) All modes of address modification apply to these instructions.
- 2) At least one storage reference is made unless indirect addressing is designated in which case at least two storage references are made.

**SSK b m** 36

Storage Skip

8.8 usec

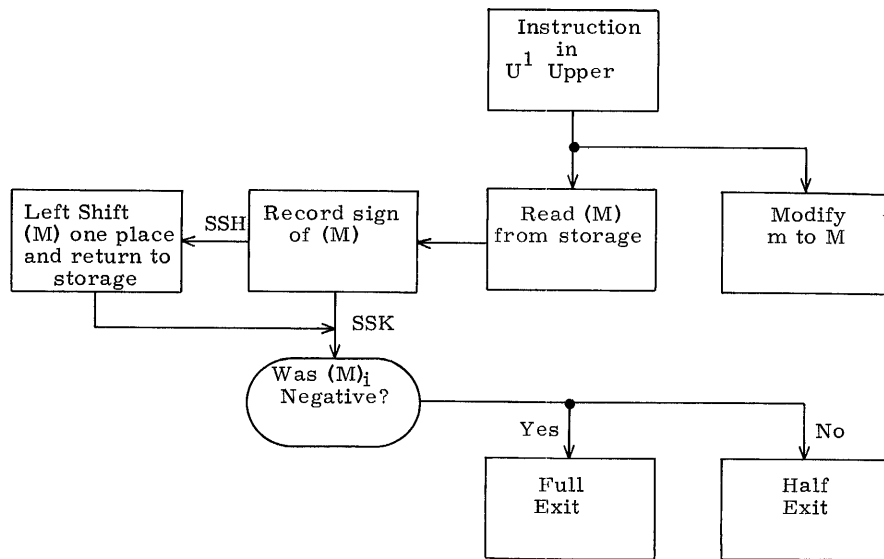
Senses the sign bit of the operand in storage location M. If the sign is negative, a full exit is taken. If the sign is positive, a half exit is taken. The contents of the operational registers are left unmodified. SSK is usually restricted to an upper instruction. If used as a lower instruction and the sign of (M) is negative, a full exit will be executed. If the sign is positive, it will half exit upon itself and never execute a full exit.

**SSH b m** 37

Storage Shift

12.8 usec

Senses the sign bit of the quantity in storage location M. If the sign bit is negative a full exit is taken, and if the quantity is positive a half exit is taken. In either case the quantity is shifted left circularly one bit before the exit. This instruction is usually restricted to the upper position. If used as a lower instruction and the sign of (M) is positive, the instruction will half exit upon itself until a negative sign bit is found. The contents of the operational registers are left unmodified.



**SSH and SSK**

# LOGICAL

- 1) All modes of address modification apply to these instructions.
- 2) The LDL, ADL, SBL and STL instructions achieve their result by forming a logical product. A logical product is a bit by bit multiplication of two binary numbers (logical AND condition):

$$\begin{array}{ll} 0 \times 0 = 0 & 1 \times 0 = 0 \\ 0 \times 1 = 0 & 1 \times 1 = 1 \end{array}$$

- 3) A logical product is used, in many cases, to select specific portions of an operand for entry into another operation. For example, if only a specific portion of an operand in storage is to be added to (A), as the operand passes through X it is subjected to a mask comprised of a predetermined pattern of "0's" and "1's". Forming the logical product of (X) and the mask causes X to retain the original contents only in those stages which have corresponding "1's" in the mask. When only the selected bits remain in X, the instruction proceeds to conclusion.

**SST b m** 40 Selective Set 7.2 usec

Sets the individual bits of A to "1" where there are corresponding "1's" in the word at storage location M. "0" bits in the word at storage location M do not modify the corresponding bits in A. In a bit by bit comparison of (A) and (M) there are four possible combinations of bits.

1) $(A)_i = 1$	2) $(A)_i = 1$	3) $(A)_i = 0$	4) $(A)_i = 0$
$(M)_i = 1$	$(M)_i = 0$	$(M)_i = 1$	$(M)_i = 0$
$(A)_f = 1$	$(A)_f = 1$	$(A)_f = 1$	$(A)_f = 0$
$(M)_f = 1$	$(M)_f = 0$	$(M)_f = 1$	$(M)_f = 0$

**SCM b m** 42 Selective Complement 7.2 usec

Individual bits of A are complemented where there are corresponding "1's" in the word at storage location M. If the corresponding bits at M are "0's", the associated bits of A remain unchanged.

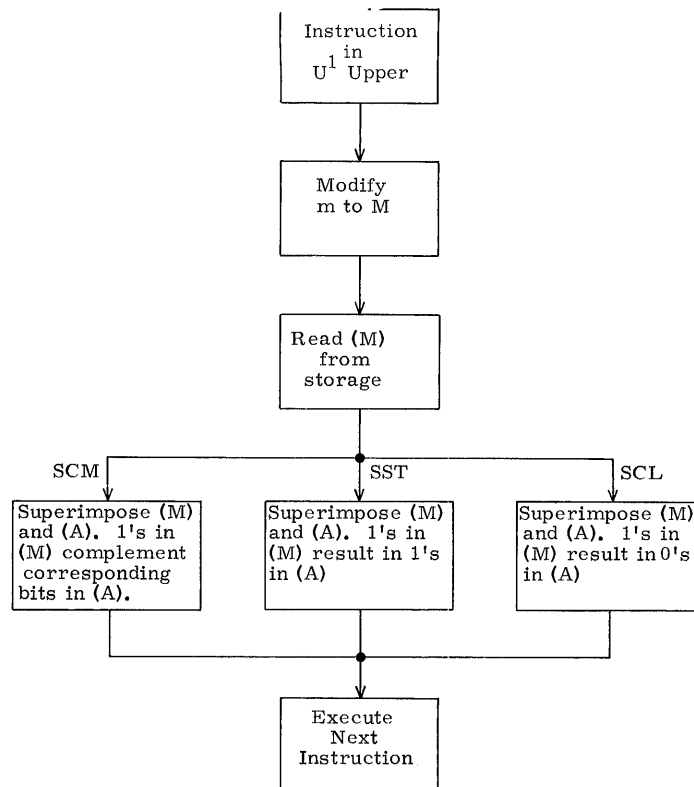
1) $(A)_i = 1$	2) $(A)_i = 1$	3) $(A)_i = 0$	4) $(A)_i = 0$
$(M)_i = 1$	$(M)_i = 0$	$(M)_i = 1$	$(M)_i = 0$
$(A)_f = 0$	$(A)_f = 1$	$(A)_f = 1$	$(A)_f = 0$
$(M)_f = 1$	$(M)_f = 0$	$(M)_f = 1$	$(M)_f = 0$

**SCL b m** 41 Selective Clear 7.2 usec

Clears individual bits of A where there are corresponding "1's" in the word at storage location M. If the corresponding bits at M are "0's", the associated bits of A remain unchanged.

In a bit by bit comparison of (A) and (M) there are four possible combinations of bits.

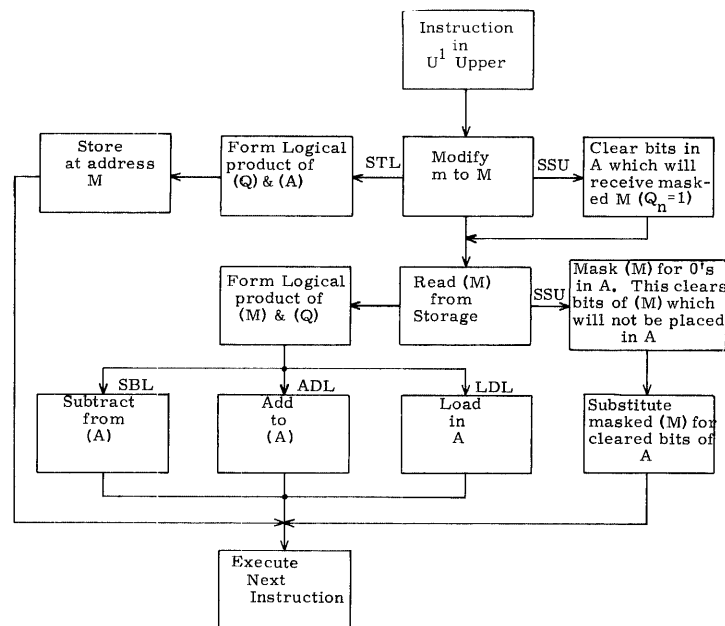
1) $(A)_i = 1$	2) $(A)_i = 1$	3) $(A)_i = 0$	4) $(A)_i = 0$
$(M)_i = 1$	$(M)_i = 0$	$(M)_i = 1$	$(M)_i = 0$
$(A)_f = 0$	$(A)_f = 1$	$(A)_f = 0$	$(A)_f = 0$
$(M)_f = 1$	$(M)_f = 0$	$(M)_f = 1$	$(M)_f = 0$



**SCM, SST, and SCL**



- SSU b m** 43 Selective Substitute 7.4 usec  
 Substitutes selected portions of an operand at storage address M into the A register where there are corresponding "1's" in the Q register (mask). The portions of A not masked by "1's" in Q are left unmodified.
- LDL b m** 44 Load Logical 7.4 usec  
 Loads A with the logical product of Q and the contents of the designated storage location, M. The operand can be in either Q or M.
- ADL b m** 45 Add Logical 7.4 usec  
 Adds to A the logical product of Q and the quantity in location M; the mask may be in Q or storage. Once the logical product is formed addition follows normal rules (appendix).
- SBL b m** 46 Subtract Logical 7.4 usec  
 Subtracts from A the logical product of the Q register and the quantity in storage location M. The mask may be in Q or storage. When the logical product is formed, the subtraction proceeds in the normal manner. (See appendix.)
- STL b m** 47 Store Logical 7.2 usec  
 Replaces the bits in storage location M with the logical product of Q and A registers. Neither (A) nor (Q) are modified. The mask may be located in A or Q.



**ADL, LDL, SBL, SSU, and STL**

# STORAGE SEARCH

- 1) If  $b = 0$  in the following instructions only the word at storage location  $m$  will be searched.
- 2) If  $b = 7$ , indirect addressing is used to obtain the execution address and  $b$  designator.
- 3) If  $(B^b) = 0$ , no search is made.

**EQS  $b m$**  64      Equality Search       $4.0 + 3.6r^* \text{ usec}$

Searches a list of operands to find one that is equal to  $A$ . The number of items to be searched is specified by  $B^b$ . These items are in sequential addresses beginning at the location specified by  $m$ . The search begins with the last address,  $m + B^b - 1$ .  $B^b$  is reduced one count for each word that is searched until an operand is found that equals  $A$  or until  $B^b$  equals zero. If the search is terminated by finding an operand that equals  $A$ , a full exit is made. The address of the operand satisfying this condition is given by the sum of  $m$  and the final contents of  $B^b$ . If no operand is found that equals  $A$ , a half exit is taken. Positive zero and minus zero are recognized as the same quantity. When EQS is used as a lower instruction, the next instruction will always be executed when the search terminates.

**THS  $b m$**  65      Threshold Search       $4.0 + 3.6r \text{ usec}$

Searches a list of operands to find one that is greater than  $A$ . The number of items to be searched is specified by  $B^b$ . These items are located in sequential addresses beginning at the location specified by  $m$ . The search begins with the last address,  $m + B^b - 1$ . The content of the index register is reduced by one for each operand examined. The search continues until an operand is reached that is greater than  $A$  or until  $B^b$  is reduced to zero. If the search is terminated by finding an operand greater than the value in  $A$ , a full exit is performed. The address of the operand satisfying the condition is given by the sum of  $m$  and the final contents of  $B^b$ . If no operand in the list is greater than the value in  $A$ , a half exit is performed. If THS is used as a lower instruction, the next instruction will be executed when search terminates. In the comparison made here positive zero is considered as greater than minus zero.

---

\*  $r$  = Number of words searched

**MEQ b m** 66

Masked Equality Search

4.0 + 3.6r\* usec

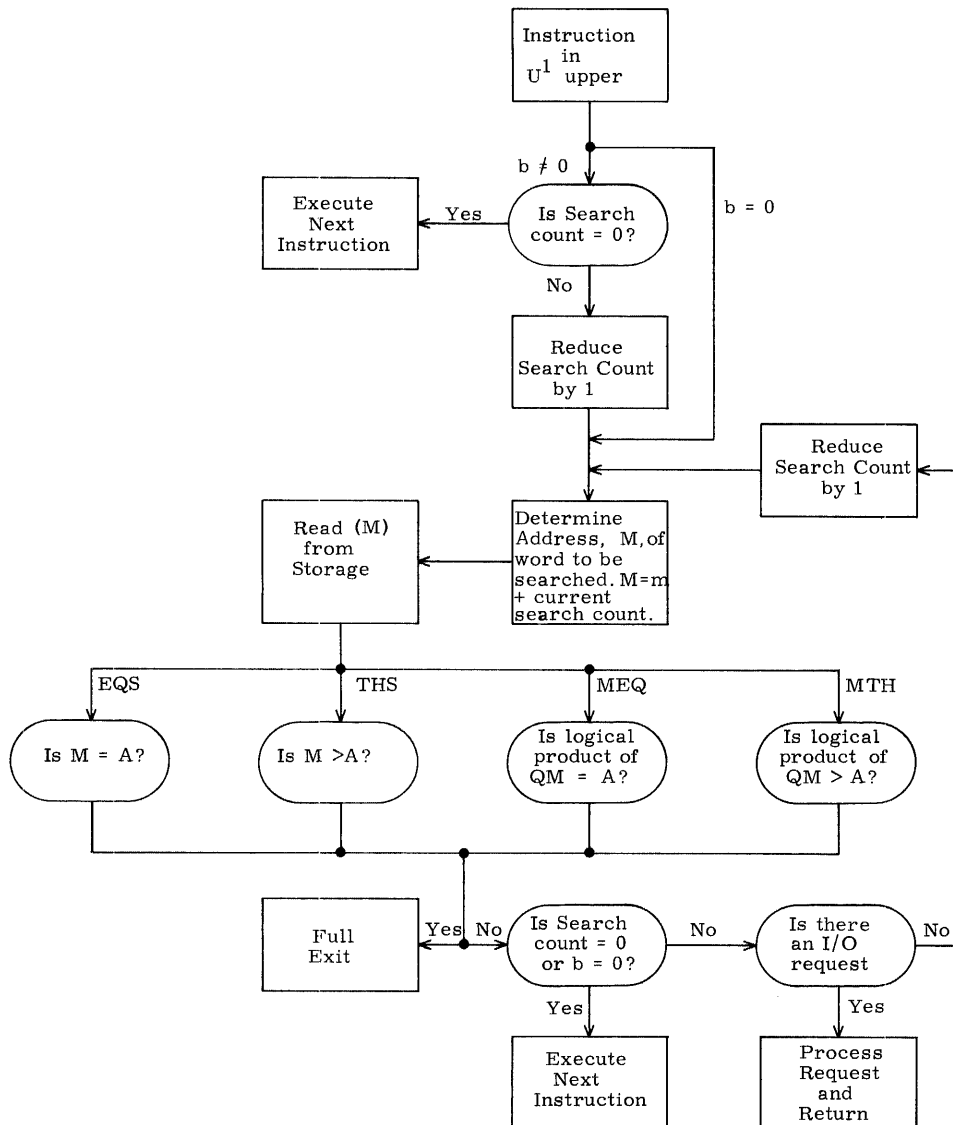
Searches a list of operands to find one such that the logical product of (Q) and (M) is equal to (A). This instruction, except for the mask, operates in the same manner as an equality search.

**MTH b m** 67

Masked Threshold Search

4.0 + 3.6r usec

Searches a list of operands to find one such that the logical product of (Q) and (M) is greater than (A). Except for the mask, this instruction operates in the same manner as the threshold search.



### Search

\* r = Number of words searched

# REPLACE

- 1) All modes of address modification apply to these instructions.
- 2) During the execution of the replace instructions, two storage references are made. If indirect addressing is designated, at least three references are made.
- 3) If the capacity of the A register  $\pm (2^{47}-1)$  is exceeded during the execution of the following instructions, an arithmetic overflow fault is produced. (Refer to appendix.)

**RAD b m** 70                      Replace Add    13.2 usec

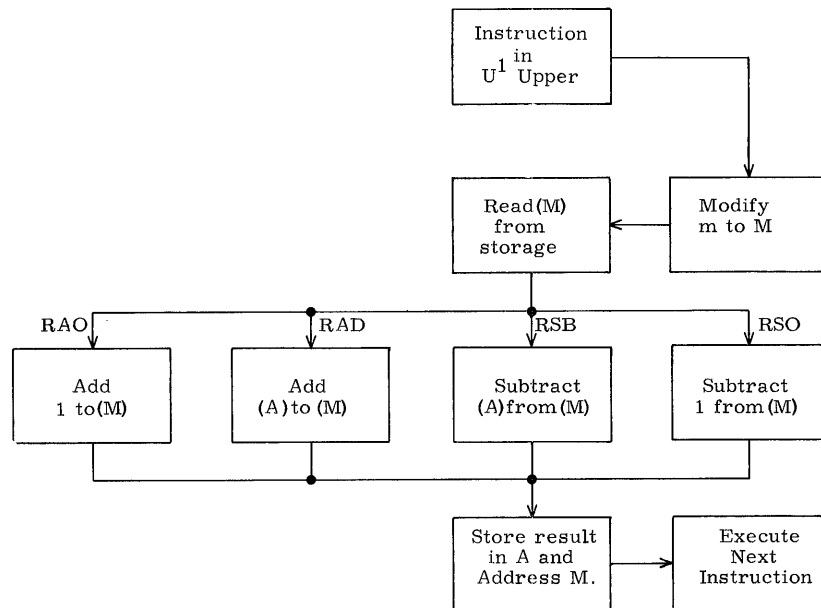
Obtains a 48-bit operand from storage location M and adds it to the initial contents of A. The sum is left in A and is also transmitted to location M.

**RSB b m** 71                      Replace Subtract    13.2 usec

Subtracts (A) from (M) and places the result in both the A register and location M.

**RAO b m** 72                      Replace Add One    13.2 usec

Replaces the operand in storage location M with its original value plus one. The result is also placed in A.

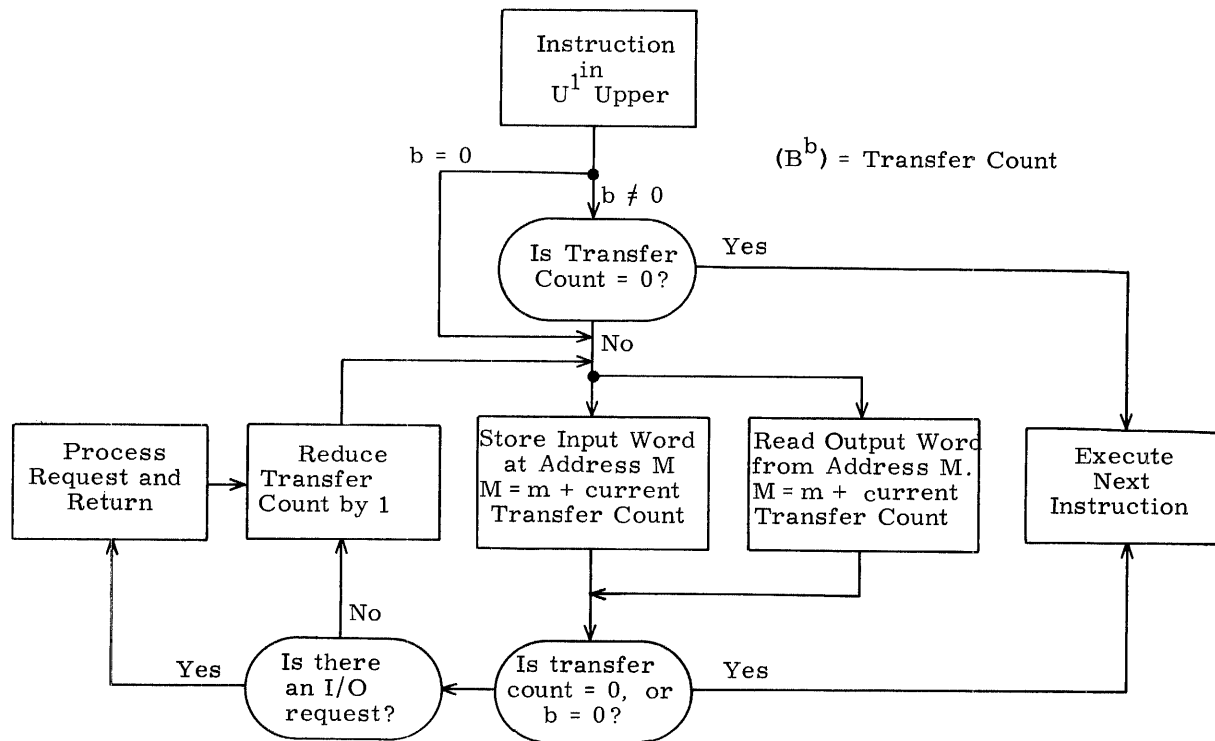


## Replace

Replaces the operand in storage location M with its original contents minus one. The difference is also left in A; the original contents of A and M are destroyed.

# TRANSFER

- 1) Address modification does not take place in the following instructions. The only mode recognized is indirect addressing.
- 2) The index registers contain the number of words to be transferred into or out of the computer via channel 7.
- 3) When a transfer is in progress all other computer operations stop except the processing of input-output requests. (See chapter three.)
- 4) If  $b = 0$ , one word is transferred to or from address  $m$ .



**Transfer**

**INT b m** 62 Input Transfer  $4.0 + 4.8r^*$  usec

Transfers a block of data from an external equipment into storage. The number of words to be transferred is specified by  $B^b$ . These words are stored in sequential addresses beginning at the location specified by the execution address,  $m$ . The transfer begins by storing the first input word in the last address in the sequence,  $m + B^b - 1$ . As each word is transferred,  $B^b$  is reduced by one until it is equal to zero.

**OUT b m** 63 Output Transfer  $4.0 + 4.8r$  usec

Transfers a block of data from computer storage to an external equipment. The number of words to be transferred is specified by  $B^b$ . The words to be transferred are located in sequential addresses beginning at the location specified by the execution address,  $m$ . The transfer begins by obtaining the first output word from the last address,  $m + B^b - 1$ . As each word is transferred  $B^b$  is reduced by one until it is equal to zero.

---

\*  $r$  = Number of words transferred



3

# Input-Output



## METHODS OF DATA EXCHANGE

The computer communicates with external equipment via a single transfer channel or six buffer channels. The transfer channel which provides for very high speed communication is program initiated and controlled. The buffer channels provide for the normal exchange of data and, although program initiated, operate independently of the program.

### HIGH SPEED TRANSFER CHANNEL

The high speed transfer channel (channel 7) handles both input and output communications between computer and high speed equipments. Information is transferred between the computer and external equipment in blocks at a word by word rate. The transfer rate is usually dependent on the speed of the external equipment as the computer can perform transfers at a maximum (approximate) rate of one word every 5 microseconds.

As many as 8 different equipments (optimum conditions) may be connected to the transfer channel. However, only one equipment can use the channel at any given instant and the current transfer operation must continue to completion before a different equipment can use the channel.

### BUFFER CHANNELS

The six independent buffer channels are grouped in three pairs:

Input:	Channel 1	Output:	Channel 2
	Channel 3		Channel 4
	Channel 5		Channel 6

Every external equipment is connected to one of these pairs. All six buffer channels may be concurrently transmitting information. However, only one external equipment can use any one buffer channel at a given instant. The rate of data flow on the buffer channels is, as with the transfer channel, dependent on the operating speed of the external equipment.

# INITIATION AND CONTROL OF DATA EXCHANGE

## TRANSFER

A transfer operation is initiated and controlled by the computer program. An INT or OUT instruction transfers the number of words designated by the contents of an index register. The starting storage location of the transfer is specified by the execution address of the instruction. (Refer to chapter 2 for a discussion of the INT or OUT instructions.)

Once initiated, the transfer operation continues to completion. While the transferring of words is in progress all computer operations stop except the processing of buffer input-output requests. (Refer to pages 3-5 and 3-6.)

## BUFFER

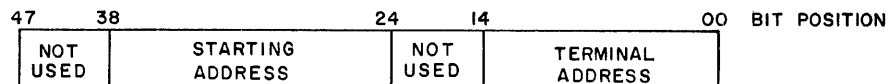
A buffer operation is program initiated but, in contrast to transfer operations, proceeds under controls that are independent of the main program.

### Buffer Control Word

Information is buffered in blocks at a word by word rate. The initial and terminal storage addresses of the block comprise the buffer control word. Each of the six buffer channels has an assigned storage address which holds the buffer control word.

<u>Address of Control Word</u>	<u>Channel</u>
00001	1
00002	2
00003	3
00004	4
00005	5
00006	6

Buffer Control Word



The terminal address is one greater than the last address to be used in the buffer. Prior to initiating a buffer operation, the terminating address must be entered into the lower address of the control word. The starting address is automatically entered into the word when the buffer is initiated by an EXF instruction.

### External Function (EXF) Instructions

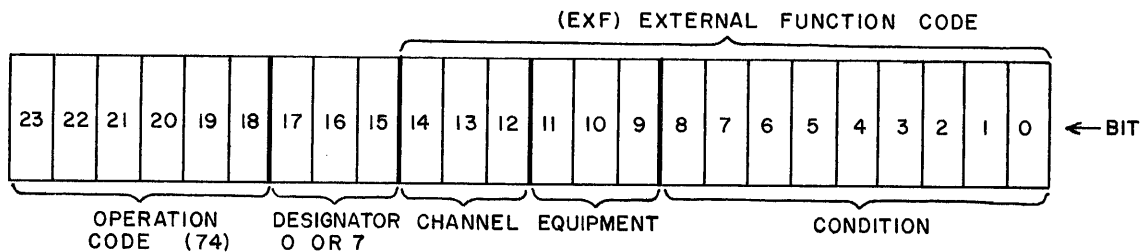
The EXF instructions initiate a buffer, sense for specified conditions, and select operations and equipment. EXF codes are listed in appendix 6.

There are three kinds of external instructions:

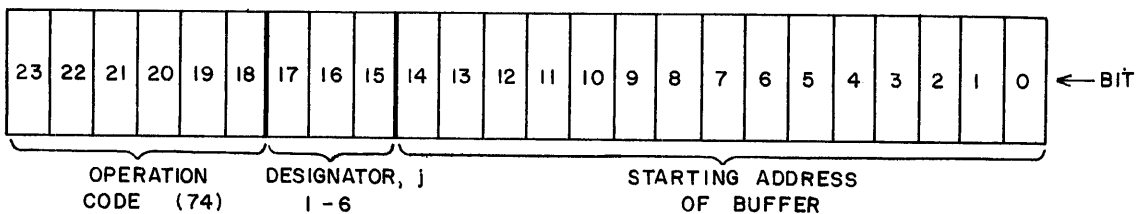
Select	74 0 XXXXX
Sense	74 7 XXXXX
Activate	74 j XXXXX

The composition of an external function instruction is shown below.

#### Select and Sense



#### Activate



The 74 0 (EXF Select) instructions select the external equipment which is to communicate with the computer and/or its mode of operation. The select instructions do not activate the buffer but, rather, establish initial operating conditions within the designated equipment so that information will be properly processed when the buffer is activated.

The EXF 7 instructions sense the condition of an external equipment or the internal conditions (faults) of the computer and will execute a full exit or half exit depending on the presence or absence of the condition.

The location of a 74 7 y instruction within an instruction word determines whether a skip or a wait will be performed.

When used in the upper instruction position (Example 1) a 74 7 y is a skip instruction.

Example 1 :	(00010)	74 7 00010	75 0 40000
	(00011)	53 1 00005	16 1 00032

In this example the translation of the upper instruction of a program step 00010 is Exit on Channel 1 active. If channel 1 is active the next instruction to be executed would be the upper instruction of step 00011, i. e., 53 1 00005. If channel 1 were inactive, the lower instruction of step 00010 would be executed.

When used in the lower position (Example 2) a 74 7 y is a wait instruction.

Example 2 :	(00100)	74 2 00600	74 7 00021
	(00101)	54 2 00005	75 0 00072

In this case, the translation of the lower instruction of step 00100 is Exit on Channel 2 inactive. If channel 2 is inactive a full exit is performed to the next pair of instructions, program step 00101. If, however, the channel is active, the instruction half exits and repeats itself until the channel becomes inactive. The sensing of conditions in no way alters the condition.

The EXF j instructions activate buffer channel j where j equals 1-6. The execution address of the instruction, y, must designate the starting address of the region in storage. These instructions are the only instructions which can initiate a buffer.

The following steps should be completed prior to initiating a buffer operation.

- 1) Sense for: (a) equipment ready and (b) channel inactive.
- 2) Select the external equipment and its mode of operation.
- 3) Substitute the terminal address into the buffer control word.

An equipment is ready if there is no motion, that is, a transmission is not taking place.

A buffer channel is active while data is being buffered. A buffer channel is inactive if the previous input-output operation has been completed.

The buffer must be satisfied (starting address = terminating address) to inactivate the channel. This can be accomplished by activating the channel (74 j instruction) at the terminating address. This makes the channel inactive but no additional information is transmitted.

### **Buffer Scanner**

Data exchange on each of the separate buffer channels is initiated by the program but proceeds under a control that is independent of the main program. This control is the buffer scanner.

The buffer scanner prevents one external equipment from dominating all others by sequentially monitoring the six buffer channels, the interrupt line and the real time clock to sense when any of these require action. When one of the positions demands action, the scanner stops. One word is processed and the scanner is released to resume its monitoring. If none of the positions demand action, a full scan is made in 3.2 usec. If, however, all eight channels demand processing at the same time, the time lapse for processing successive words on the same channel will be 200 usec.

Figure 3-1 shows the relationship between executing instructions and recognizing action requests (input-output requests) from the buffer channels, interrupt line and real time clock. During the time an input-output request is being processed, the scanner is released and resumes monitoring. Four more channels are scanned before the processing of the request is completed. If the scanner halts (indicating one of the four channels demands action) this request will be processed before the main program is allowed to continue.

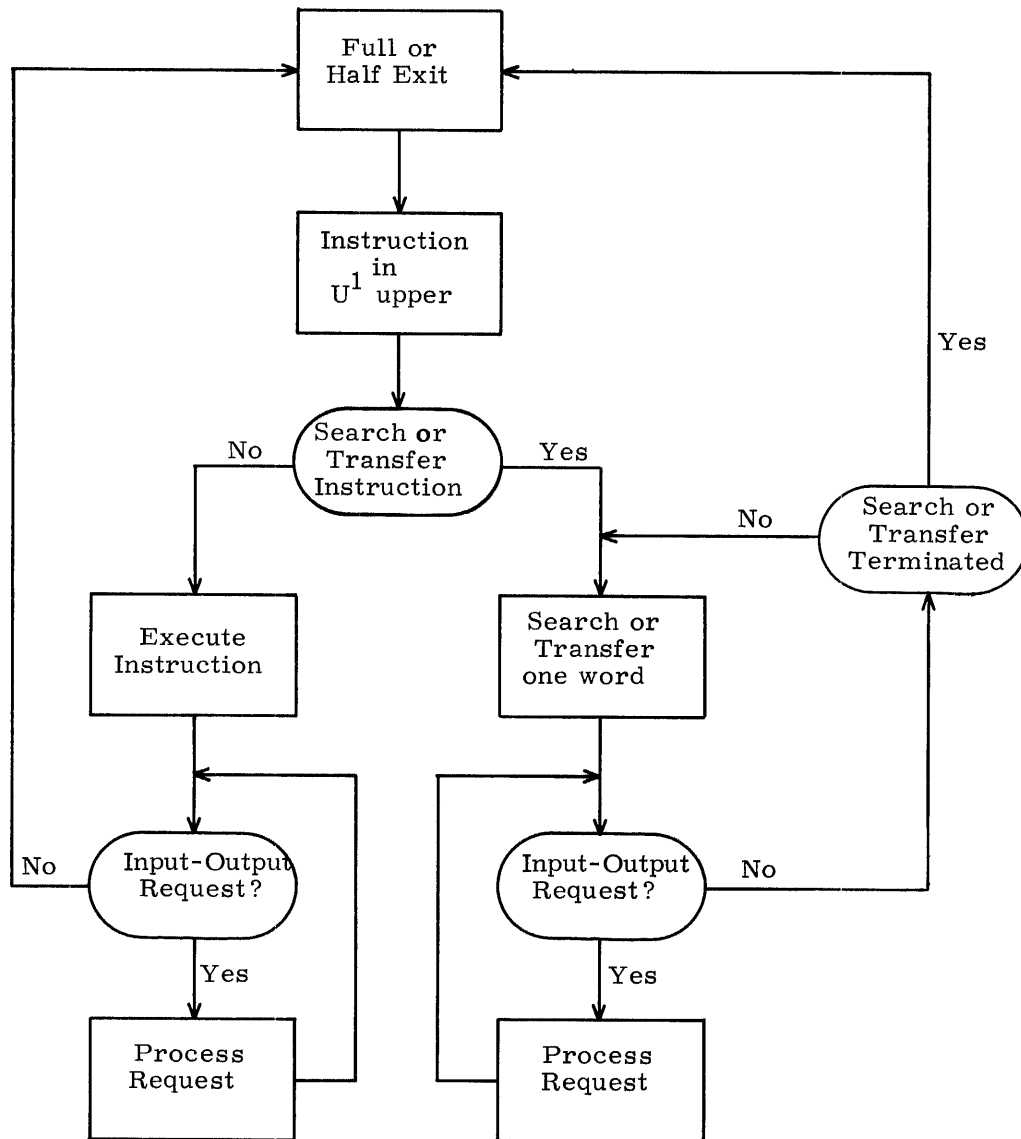


Figure 3-1. Recognition of I/O Requests

## Interrupt

In each piece of external equipment as well as in parts of the internal computer control, certain conditions may arise which make it necessary that the main program be notified of their presence. The signal which notifies the computer of these conditions is called an interrupt and is program controlled. If an interrupt is desired when a particular condition arises, an external function select code (74. 0) must select an interrupt on that condition. Unless such selection is made, no interrupt is produced when the condition arises. (See appendix 6 for the codes.)

When an interrupt occurs, the main program is halted and a previously programmed routine of instructions (interrupt routine) is performed which must determine the cause of the interruption and take appropriate action. After completing these operations, the interrupt routine must return to the main program. The main program resumes at the exact point from which the entrance to the interrupt routine was made.

### Interrupt Routine

The interrupt is processed by performing a return jump to address 00007. This is a special address allocated for use as the entrance point to the interrupt routine and for the return from this routine to the main program.

Typically, address 00007 (interrupt control word) contains two unconditional jump instructions. For example:

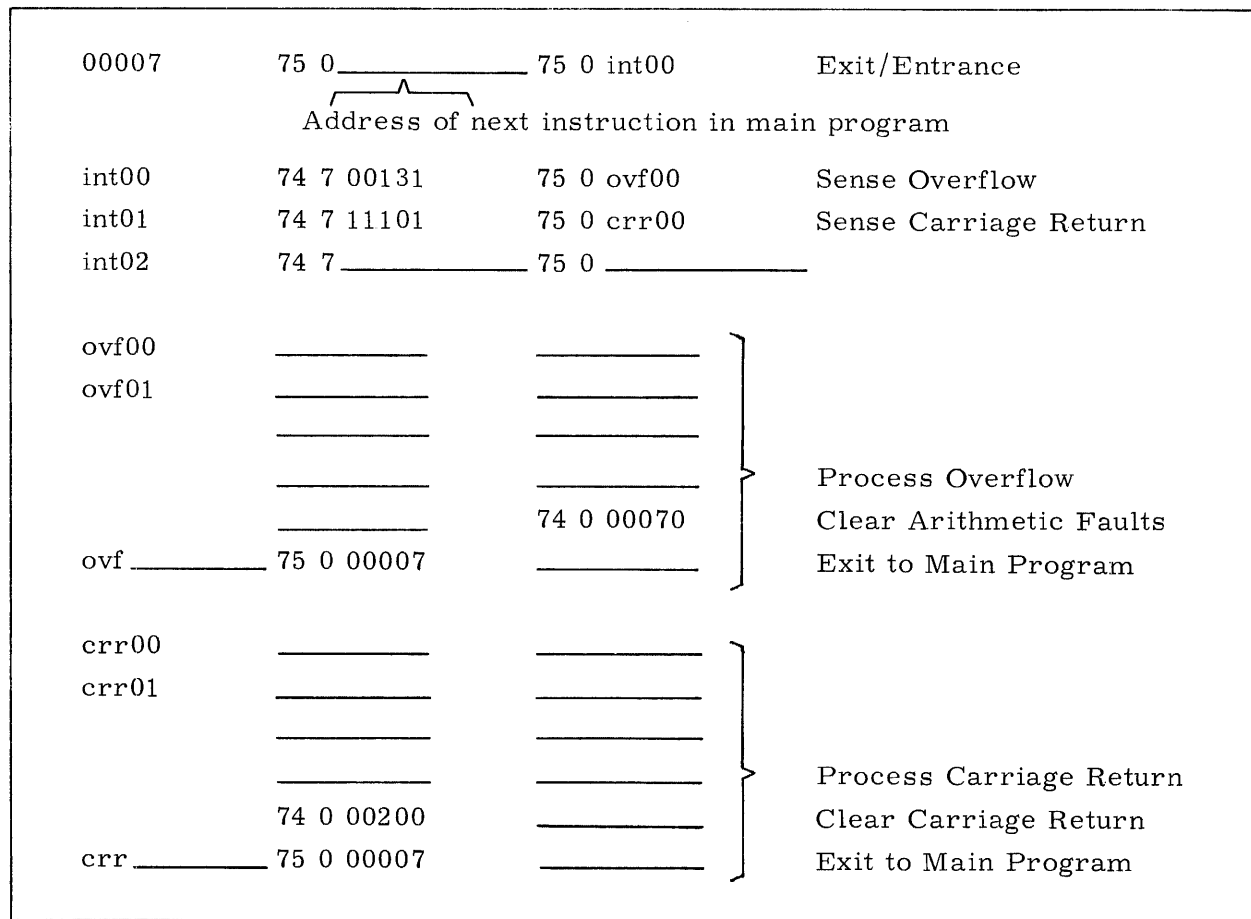
(00007)	75 0 XXXXX	75 0 YYYYY
---------	------------	------------

The upper instruction provides for the return to the main program upon completion of the interrupt routine. To accomplish this the upper address portion (XXXXX) is loaded with the contents of the P register when the interrupt routine is entered. The lower instruction jumps to the interrupt routine which begins with an instruction whose address is indicated in the lower address part of 00007 (YYYYY).

In general, the interrupt routine (table 3-1) checks for all possible interrupt conditions by means of sense (74. 7) instructions. After determining which selected condition caused the interrupt a jump is made to that portion of the routine which processes the interrupt.

After having been interrupted the computer cannot again be interrupted without returning to the main program via location 00007.

TABLE 3-1. TYPICAL INTERRUPT ROUTINE



When an interrupt occurs, unless the indication of the condition or the interrupt selection is removed, the program will again be immediately interrupted upon return to the main program.

In the process overflow fault above, all arithmetic faults are cleared. In the process carriage return portion of this interrupt routine the interrupt signal is removed by a 74 0 00200 (clear carriage return character). The interrupt selection may be removed by a 74 0 11100 (no interrupt on carriage return).



### **Real Time Clock**

Address 00000 in core storage may be selected to provide a continuously operating record of elapsed time. The 48-bit quantity stored there is advanced by one every 1/60 of a second (accuracy is maintained by the 60-cycle power source). The content of address 00000, which may be sampled at any time by the program, gives elapsed time from the start of the real-time clock operation. The clock may be started by 74 0 01000 and is stopped by 74 0 02000. Starting the clock does not pre-set address 00000 in any way, but begins the periodic incrementing of its previous contents.

By selecting interrupt on arithmetic fault, and presetting the contents of address 00000, the real-time clock may be used to provide an interrupt of the main computer program. When interrupt occurs, check location 00000 for overflow.

## **CONSOLE INPUT-OUTPUT EQUIPMENT**

Three input-output devices mounted on the console are standard equipment with the 1604 computer. A Teletype high speed punch and a high speed tape reader provide for the processing of perforated paper tape. An IBM typewriter, modified by Soroban Engineering, Inc., provides for direct keyboard entry of data and for printed copy output. The console input-output units communicate with the central computer via buffer channel 1 (input) and 2 (output). Other input-output units may share these channels but console input-output units use only these channels.

Data may be transmitted between the console equipments and the computer in either the character or the assembly mode. In the character mode a 7-bit character is buffered one character at a time. The 7-bit character occupies the lowest bit positions of a 48-bit word; the upper 41 bits are "0's".

In the assembly mode the 48-bit word, consisting of eight 6-bit characters, is buffered. During an input buffer in the assembly mode eight successive characters are assembled into a 48-bit word and sent to the computer. The first character occupies the upper 6 bits of the word; the last character occupies the lower order 6 bits.

For an output buffer in the assembly mode a 48-bit word from the computer is disassembled into eight characters, the upper 6 bits first.

## **TYPEWRITER**

The typewriter may be used as a keyboard input device or as an output device for producing printed copy; during output it types approximately 10 characters per second.

All of the typewriter characters and functions are represented by unique combinations of 6 bits. (Codes are in appendix 6.) During a keyboard input operation, striking a character key causes the coder to produce the code which is sent to the computer. Space is the only coded typewriter control function which is sent to the computer. For typewriter output, a 6-bit character code sent to the decoder causes the typewriter to print the selected character or perform the designated control function.

If the keyboard is selected by code 1114X, the interrupt signal occurs for each carriage return (CR).

If an illegal code (unlisted) is sent to the typewriter from the computer, operation hangs up. Striking the CR, backspace or shift keys will allow operation to be resumed.

A zero code (all "0" bits) which constitutes a do nothing code is used to fill out a 48-bit word in the assembly mode.

## **PAPER TAPE READER**

The paper tape reader enters information stored on punched paper tape into the computer. The reader, which is always connected to channel 1, operates at a maximum rate of 350 characters (frames) per second; the time interval between successive characters from the reader is 3.3 ms.

Manual controls for the reader are on the punch and reader control panel of the console. When the Reader Mode switch is set to Assembly, the tape is positioned at the first frame of the first word (load point); when it is set to Character, the tape moves ahead one frame.

Information is stored on paper tape in seven levels. A frame, which is across the width of the tape, can store 7 bits (figure 3-2). The sprocket or feed holes between levels 3 and 4 generate signals to time and control the reading of the tape.

In the assembly mode, level 7 is used as a control rather than an information level. The first of the eight characters in a word is indicated by a hole in the control level.

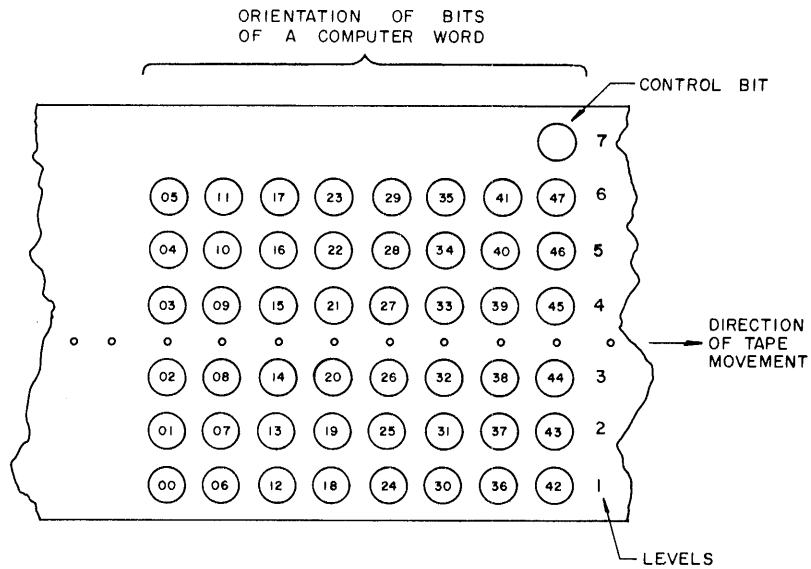


Figure 3-2. Seven Level Punched Paper Tape

Reader tape motion stops on any one of three conditions:

- 1) When buffer operation terminates (assembly or character mode).
- 2) When the load point in assembly mode is reached.
- 3) Absence of a 7th level every 8th character in the assembly mode.

The reader End of Tape indicator is set on any of three conditions:

- 1) On a computer Master Clear.
- 2) Absence of a 7th level every 8th character in the assembly mode.
- 3) By a 74 0 11210 instruction. This instruction is used to indicate the end of information in the character mode.

After reading all information on the tape in the assembly mode, tape motion stops and the End of Tape indicator is set because the 7th level control bit is missing. In the character mode, however, motion stops when the buffer operation is satisfied but the End of Tape indicator remains cleared. A 74 0 11210 instruction may be programmed to set the End of Tape indicator after the buffer terminates. The state of the End of Tape indicator, regardless of the mode of operation, may be used to determine if all information on the paper tape has been read.

### **PAPER TAPE PUNCH**

The punch which prepares paper tape output is always connected to buffer channel 2. Nominal operating rate is 60 characters per second. In character mode, the lower 7 bits of each word sent are punched; the upper bits are ignored. In assembly mode, eight 6-bit characters are punched per word. The upper 6 bits are punched first, with the 7th level supplied.

On the punch, the feedout lever provides for punching out leader. A micro-switch is mounted near the roll of paper tape that supplies the punch. When the supply is low, the switch opens and provides an out of tape indication which may be sensed.

## INTERNAL EXF SELECT INSTRUCTIONS

74 0 000C0	Interrupt on Channel C Inactive Selects interrupt when channel C becomes inactive. C = 1 - 6 An interrupt signal is generated whenever the channel becomes inactive. More than one interrupt can be selected. The interrupt remains selected until cleared.
000C1	Remove Selection Above Interrupt on channel C inactive selection removed
00100	Interrupt on Arithmetic Faults Selects interrupt on occurrence of any arithmetic fault; remains selected until cleared
00101	Remove Selection Above Interrupt on arithmetic faults selection removed
00070	Clear All Arithmetic Faults Removes all arithmetic fault indications and turns off arithmetic fault background lights on console
01000	Start Real-Time Clock Begins process of incrementing previous contents of address 00000 by one each 16.6 ms; address 00000 is not cleared by starting clock
02000	Stop Real-Time Clock Halts process of incrementing address 00000. The contents of 00000 remain unchanged.
C0000	Clear All Channel C Selections Clears all previous selections made on the designated channel C except interrupt on channel C inactive
00200	Clear the Carriage-Return-Typed Indicator

## INTERNAL EXF SENSE INSTRUCTIONS

74 7 000C0	Exit on Channel C Active Full exit if channel C is active; if not, half exit
000C1	Exit on Channel C Inactive Full exit if channel C is inactive; if active, half exit
00110	Exit on Divide Fault; half exit if no divide fault
00111	Exit on No Divide Fault; half exit if divide fault
00120	Exit on Shift Fault; half exit if no shift fault
00121	Exit on No Shift Fault; half exit if shift fault
00130	Exit on Overflow Fault; half exit if no overflow fault
00131	Exit on No Overflow Fault; half exit if overflow fault
00140	Exit on Exponent Fault; half exit if no exponent fault
00141	Exit on No Exponent Fault; half exit if exponent fault

## CONSOLE EXF SELECT CODES

(Always Channel 1 and 2, Equipment #1)

### INPUT TYPEWRITER

74 0 11100	Select the Typewriter for Input and No Interrupt on Carriage Return Selects keyboard (character mode only) Interrupt selection cleared, Carriage Return indicator cleared
11140	Select the Typewriter for Input and Interrupt on Carriage Return Selects keyboard (character mode only) Interrupt selection set, Carriage Return indicator cleared. The next carriage return, which is not output, will set the Carriage Return FF and cause an interrupt. The interrupt selection can be cleared by the external master clear or the 74 0 11100 select only.

## PAPER TAPE READER

74 0 11210	Set End of Tape Indicator Sets the End of Tape indicator * Clears interrupt on end of tape
11200	Select the Paper Tape Reader and No interrupt on End of Tape Selects the reader Interrupt on end of tape cleared
11220	Select the Paper Tape Reader and Interrupt on End of Tape Selects the reader Interrupt on end of tape set. If the End of Tape indicator is set, the interrupt will be immediate.

## OUTPUT TYPEWRITER

74 0 21100	Select the Typewriter for Output in the Assembly Mode Selects keyboard to print **
21110	Select the Typewriter for Output in the Character Mode Selects keyboard to print **

---

\*This select is usually used in character mode operation only. The End of Tape indicates the logical end of tape, and can be cleared externally only by moving the switch (on the reader control) to the character or assembly position. Master Clear selects the paper tape reader and sets the End of Tape indicator. When the End of Tape indicator has been set the reader is "not ready".

\*\*Will not change the Carriage Return FF nor the interrupt selection on channel 1. The code "00" will be ignored, all other illegal codes will cause the typewriter to hang-up. It is released by performing a function, usually spacing.

## PAPER TAPE PUNCH

74 0 21200	Select the Paper Tape Punch, Assembly Mode Selects the punch, sets mode to assembly Turns the punch motor on
21210	Select the Paper Tape Punch, Character Mode Selects the punch, sets mode to character Turns the punch motor on
21240	Turn the Punch Motor Off

## CONSOLE EXF SENSE CODES

### TYPEWRITER

(Sensed on Input Channel Only)

74 7 11100	Full Exit if Carriage Return Performed on Input If a carriage return (which was not the result of an output) has been performed since the last input select, a full exit is executed; if not, a half exit.
11101	Full Exit if No Carriage Return Typed on Input If the Carriage Return indicator is not set, a full exit is executed; if set, a half exit.
11140	Full Exit Lower Case If the typewriter keyboard is in the lower case a full exit is performed.
11141	Full Exit Upper Case If the typewriter keyboard is in the upper case a full exit is performed.



## PAPER TAPE READER

- 74 7 11200 Full Exit on End of Tape Indicator Set  
If the End of Tape indicator is set a full exit is performed; if not, a half exit.
- 11201 Full Exit on No End of Tape Indicator Set  
If the End of Tape indicator is not set a full exit is performed; if set, a half exit.
- 11210 Full Exit on Assembly Mode  
If the paper tape reader is in the assembly mode a full exit is performed; if not, a half exit.
- 11211 Full Exit on Character Mode  
If the paper tape reader is in the character mode a full exit is performed; if not, a half exit.

## PAPER TAPE PUNCH

- 74 7 21200 Full Exit on Out of Tape  
If the paper tape punch is out of tape, a full exit is performed; if not, a half exit.
- 21201 Full Exit on Not Out of Tape  
If the paper tape punch is not out of tape, a full exit is performed; if out of tape, a half exit.

## PROGRAMMED EXAMPLES FOR CONSOLE

Example 1: Type  $50_{10}$  characters which are stored in FILE (one character per word in the lower 6 bits).

<u>LOCN</u>	<u>OPN</u>	<u>B</u>	<u>OPERAND OR M-TERM</u>	<u>REMARKS</u>
FILE	BSS		50	
START	ENA	0	FILE + 50	Terminating Address
-	EXF	7	00021B	Wait for Channel 2 to be Inactive
	STA	0	2	Control Word for Channel 2
	EXF	0	21110B	Select Typewriter Character Mode
	EXF	2	FILE	Activate Channel 2 at First Word Address

The program would continue from this point while the output goes on.

Example 2: Call for  $10_{10}$  characters to be typed as input to the computer. (Typewriter input is in the character mode only, therefore  $10_{10}$  words must be read in.) If an error is made before the tenth character is typed, press the carriage return key and call for ten new characters. Continue the main program until all ten characters are typed without error.

TYPE	BSS		10	
CONTROL	ENA	0	TYPE + 10	Ten Words, the First at Type
-	EXF	7	00011B	Wait for Channel 1 to be Inactive
	EXF	0	11140B	Select Typewriter for Input and Interrupt on CR
	SAL	0	1	Terminating Address to Control Word on CR
	EXF	1	TYPE	Activate Channel 1, the First Character to Type
	EXF	0	00010	Select Interrupt on Channel 1 Inactive

The program would continue from this point. When 10 characters had been typed or a carriage return performed, an interrupt would be executed (return jump to 00007).

00007	SLJ	0	XXXXX	Address to Return in Main Program Supplied
	NOP			
00010	EXF	7	11100B	Full Exit if Carriage Return
	SLJ		13B	Must Have Been Channel 1 Inactive
00011	EXF	0	00200B	Clear Carriage Return Indicator
	EXF	1	TYPE	Call for 10 New Characters
00012	SLJ	0	7	Return to Main Program via 00007
00013	XXX	X	XXXXX	The program to interpret the input would be here. The return to the main program must eventually be via location 00007.

Example 3: Punch out on paper tape the contents of location DUMP up to but not including DUMP + 1000<sub>g</sub>. Punch all of each word.

<u>LOCN</u>	<u>OPN</u>	<u>B</u>	<u>OPERAND OR M-TERM</u>	<u>REMARKS</u>
DUMP	BSS		1000B	
OUT	NOP			
-	EXF	7	21B	Wait for Channel 2 to be Inactive
	EXF	0	21200B	Select Paper Tape Punch, Assembly Mode
	ENI	1	DUMP + 1000B	Terminating Address
	SIL	1	2	Store in Location 00002
	EXF	2	DUMP	Activate Channel 2 at FWA

The program would continue from this point while the output goes on.

Example 4: Read a paper tape and type the characters until a carriage return code is read (45<sub>g</sub>). Read them into location CHAR and following locations.

CHAR	BSS	(Maximum for tape used)		
-	ENA	0	CHAR	Initialize the Activate Instructions
	SAU	0	ACT 1	
	SAU	0	ACT 2	
	SAU	0	ACT 2 + 1	
-	EXF	7	21B	Wait for Channel 2 to be Inactive
	EXF	0	21110B	Select the Typewriter for Output
-	EXF	7	11B	Wait for Channel 1 to be Inactive
	EXF	0	11200B	Select the Paper Tape Reader
	ENA		45B	Enter the Code for Carriage Return
	ENI	6	CHAR + 1	Set Up First Terminating Address
	NOP			
NEXT	SIL	6	1	Store in Control Word 00001
	SIL	6	2	Store in Control Word 00002
ACT 1	EXF	1	CHAR	Call for One Character
	EXF	7	11B	Wait
ACT 2	EXF	2	CHAR	Type the Character
	NOP			
	EQS	0	CHAR	Was it a Carriage Return?
	SLJ		CONT	No (next page)
	SLJ		DECODE	Yes

<u>LOCN</u>	<u>OPN</u>	<u>B</u>	<u>OPERAND OR M-TERM</u>	<u>REMARKS</u>
CONT	SIU	6	ACT 1	New Input Address
	SIU	6	ACT 2	New Output Address
	SIU	6	ACT 2 + 1	
	INI	6	1	Increase Address
	SLJ	0	NEXT	Continue Reading
DECODE	At this point the line of input could be decoded. The carriage return code is in the location specified by $(B^b) - 1$ .			



# Operation

## DESCRIPTION OF INDICATORS AND CONTROL SWITCHES

All main computer controls and indicators are on the console. Functional significance of console background lights is listed in table 4-1; computer controls are described in table 4-2.

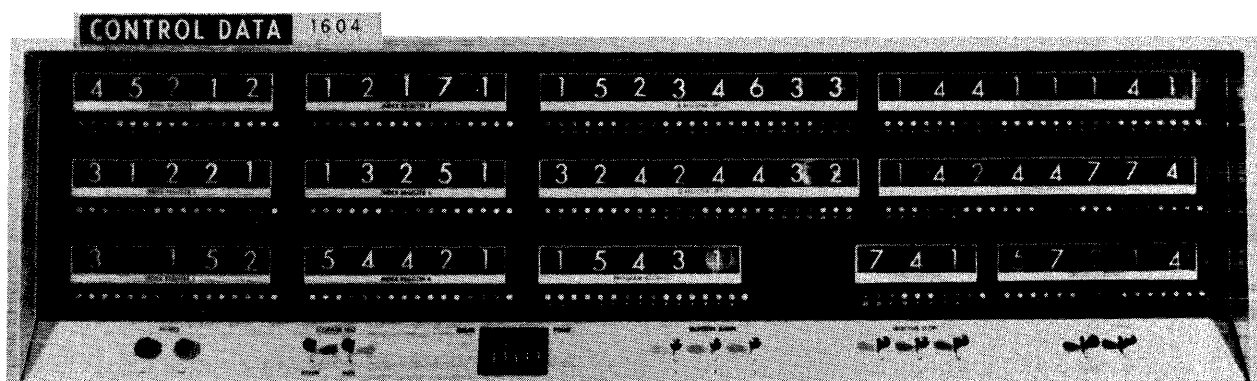


Figure 4-1. Center Panel of Console

The indicators are lamp modules, each of which displays a single octal digit. The lamps, in response to signals from the computer, display the contents of the operational registers in octal form only when the computer is stopped; the display is blank when the computer is running. Each indicator has three push buttons which are numbered in the powers of two, from right to left, starting with zero. Pressing a push button forces that particular stage of the register to the SET state. Each group of three buttons represents an octal digit. Different shades of blue are used in adjacent octal groups; within an octal group the three buttons are of the same shade.

At the right end of each register is a CLEAR push button (white). This button will clear the individual FFs within that register. SET and CLEAR push buttons should be used only when the computer is stopped; otherwise errors may result.

Conditions which stop the computer are listed below. When these conditions exist register contents may be altered by setting or clearing.

- 1) Illegal function codes 00 and 77
- 2) Selective Stops (instructions 76)
- 3) Breakpoint Stop
- 4) Pressing START-STEP switch
- 5) Pressing CLEAR switch (internal master clear)

At some of the modules there are colored background lights which indicate certain internal conditions (Figure 4-2, Table 4-1). A light is identified by the register in which it is located and its position in the register. For example, AL-4 is fourth from the left in A register left. In general, red lights signify faults and blue lights signify special operating conditions. The background lights may be illuminated when the computer is running as well as when it is stopped.

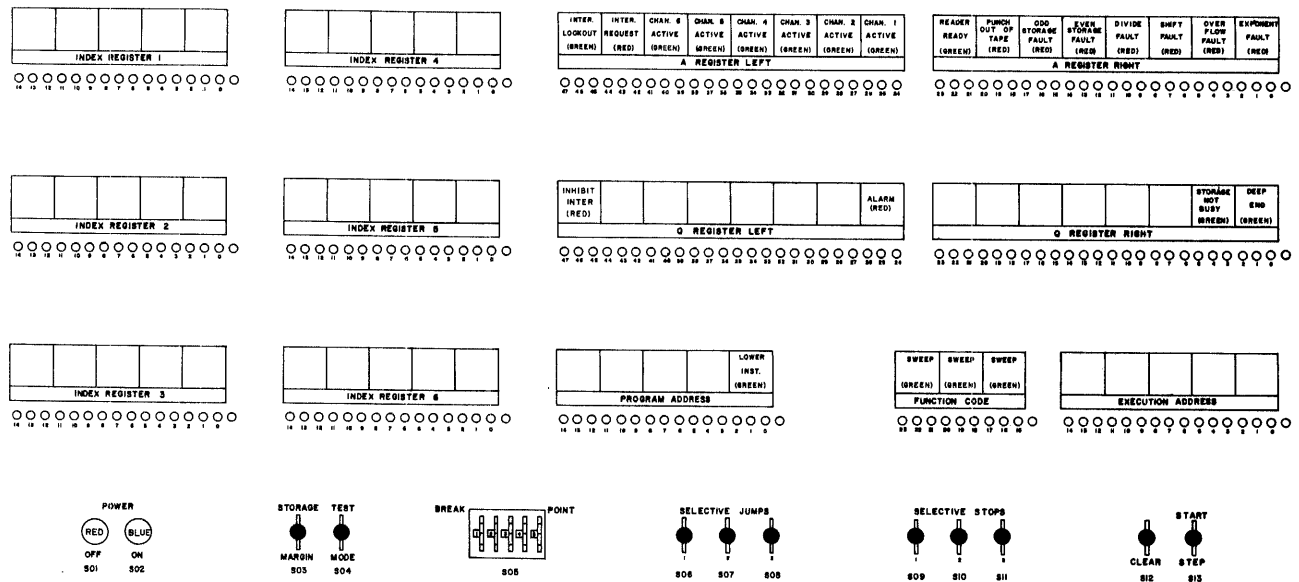


Figure 4-2. Console Display

TABLE 4-1. CONDITIONS INDICATED BY CONSOLE BACKGROUND LIGHTS

Light	Condition
AL-1 (blue)	<u>Interrupt Lockout</u> - Computer is in interrupt routine.
AL-2 (red)	<u>Interrupt Request</u> - Interrupt request signal is being received from interrupt circuit.
AL-3 (blue)	<u>Channel 6 Active</u> - Channel 6 is in use for output buffer.
AL-4 (blue)	<u>Channel 5 Active</u> - Channel 5 is in use for input buffer.
AL-5 (blue)	<u>Channel 4 Active</u> - Channel 4 is in use for output buffer.
AL-6 (blue)	<u>Channel 3 Active</u> - Channel 3 is in use for input buffer.
AL-7 (blue)	<u>Channel 2 Active</u> - Channel 2 is in use for output buffer.
AL-8 (blue)	<u>Channel 1 Active</u> - Channel 1 is in use for input buffer.
AR-1 (blue)	<u>Reader Ready</u> - (1) Paper tape is at load point, ready for an input buffer; or (2) input buffer paper tape is in progress.
AR-2 (red)	<u>Punch Out of Tape</u> - Punch tape reel is nearly empty.
AR-3 (red)	<u>Odd Storage Fault</u> - Fault in sequence chain of odd storage unit; storage unit is inoperative until master cleared.
AR-4 (red)	<u>Even Storage Fault</u> - Fault in sequence chain of even storage unit; storage unit is inoperative until master cleared.
AR-5 (red)	<u>Divide Fault</u> - Improper divide instruction executed.
AR-6 (red)	<u>Shift Fault</u> - Shift count greater than 127 (decimal).
AR-7 (red)	<u>Overflow Fault</u> - Required sum or difference exceeds capacity of A register.
AR-8 (red)	<u>Exponent Fault</u> - In a floating-point instruction, exponent of result is $2^{10}$ or greater.
QL-8 (red)	<u>Alarm</u> - Indicates abnormal condition.
QR-7 (green)	<u>Storage Not Busy</u> - Indicates storage is not in use.
QR-8 (blue)	<u>Deep End</u> - Computer fails to complete operation in step mode.
PA-5 (blue)	<u>Lower Instruction</u> - Lower instruction is indicated.
FUNCTION CODE (blue) (3 lights)	<u>Sweep</u> - Computer is in sweep mode (Mode switch is down).



TABLE 4-2. MAIN COMPUTER CONTROLS

Control		Function
POWER push button	ON - green	Applies d-c power to computer by energizing contactor in primary power lines of motor-generator.
	OFF - red	Removes d-c and a-c power from computer by de-energizing contactor in primary power lines of motor-generator.
STORAGE TEST Lever switch locks in up, down and neutral positions.	MARGIN	Varies the bias applied to storage sense amplifiers. Used for maintenance purposes only; should be in neutral position at all other times.
	MODE	UP: an instruction is executed repeatedly in either the Step or Start mode.  DOWN: contents of consecutive storage locations may be manually examined by depressing Step. Consecutive half-words are displayed in function code and execution address registers but are not executed.
BREAKPOINT Five 8-position switches can be set to octal address 00000 through 77777.		Provides for selection of any storage address as a breakpoint address. Computer stops when program address and breakpoint address are equal, just prior to performing the upper instruction at the breakpoint address.
SELECTIVE JUMPS 1, 2, 3 Three lever switches lock in upper positions, momentary in down positions.		Provide manual conditions for instruction 75, normal jumps, b = 1, 2 or 3, return jumps, b = 5, 6 or 7 (Same function in all positions)
SELECTIVE STOPS 1, 2, 3 Three lever switches lock in upper position, momentary in down positions.		Provide manual conditions for stopping the computer on instruction 76, b = 1, 2, 3, 5, 6 or 7 (Same function in all positions)
CLEAR Lever switch, momentary in up and down positions.		DOWN master clears the computer, clears all operational registers and most control FFs.  UP master clears external equipment, causing most of the registers and control FFs of the external equipment to be cleared and the paper tape reader to be selected.

TABLE 4-2. (CONT'D.)

Control	Function
<p>START-STEP</p> <p>Lever switch, momentary in up and down positions.</p>	<p>START (up) selects high-speed mode in which a program of instructions and auxiliary operations proceeds until completed or stopped.</p> <p>STEP (down) selects Step mode. Each time switch is pressed a single instruction is executed and computer stops (all buffer requests are completed before operation stops). Step selection overrides any previous selection of Start.</p>
<p>VOLUME CONTROL</p> <p>Black knob under console desk</p>	<p>Controls volume of signal from console loudspeaker.</p>
<p>*SET Push Buttons</p> <p>Numbered in the powers of 2, beginning with zero. Each group of 3 is an octal digit.</p>	<p>Allow for manual entry of a quantity into a given register. Forces that particular stage of the register to the Set state.</p>
<p>*CLEAR Push Button</p>	<p>Clears the individual FFs within that register.</p>

\* Should be used only when the computer is stopped

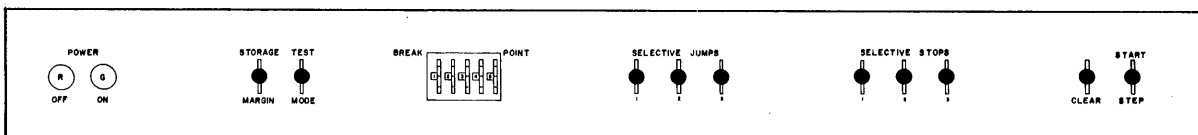


Figure 4-3. Manual Controls

## READER AND PUNCH CONTROLS

TABLE 4-3. READER AND PUNCH CONTROLS

Switch	Function
PUNCH MOTOR	Turns punch motor on or off. (Motor may also be turned on under program control.)
SELECT-TAPE FEED	Select enables use of the punch.  Tape Feed causes leader to be punched.
READER MOTOR	Turns reader motor on or off. (Motor cannot be turned on by any other means.)
CHARACTER-ASSEMBLY	In Character mode each character is sent to computer separately.  In Assembly mode eight consecutive characters are assembled into a word to be sent to computer.

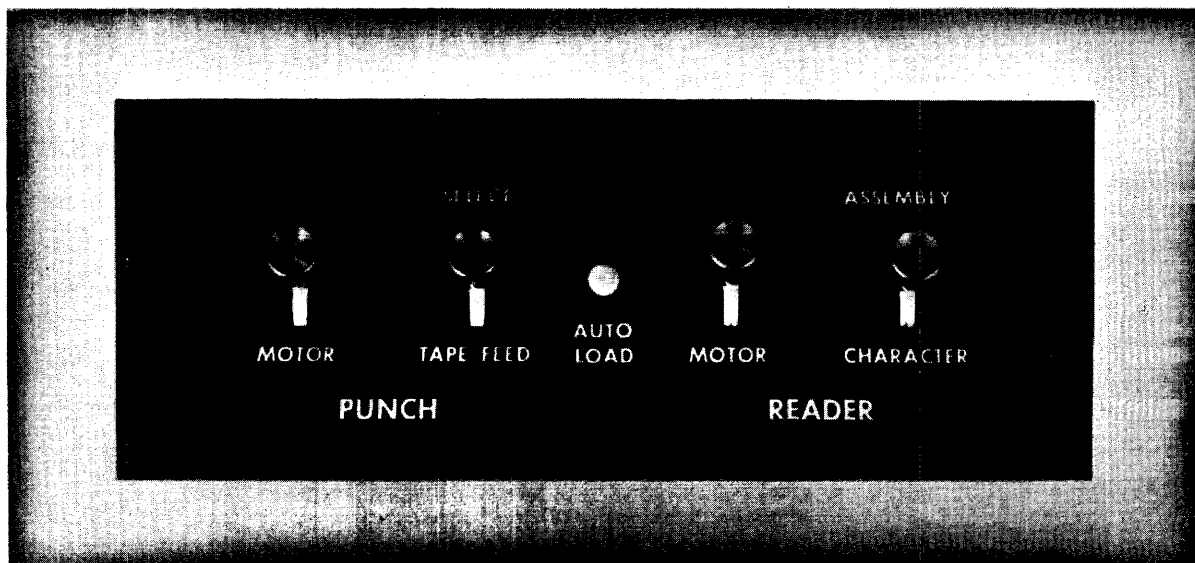


Figure 4-4. Reader and Punch Controls: Auto Load Control

## AUTO LOAD CONTROL

The Auto Load button initiates a bootstrap routine to read into memory (via channel 3) the first record from magnetic tape #1, on 1615 or 1607 equipment #2.

Pressing the Auto Load button selects the tape and loads the bootstrap routine into memory locations 00000 and 00001, and puts an address of 32000 (arbitrary and > 00004) in the lower address of 00003.

The program appears as:

(00000)	74 0 32005	Rewind the tape
	74 7 32000	Wait for ready
(00001)	74 3 00002	Activate, FWA = 00002
	74 7 32000	Wait for ready
(00002)	XX X XXXXX	Will be the first
	XX X XXXXX	word read from tape
(00003)	74 3 00002	
	74 7 32000	

If Breakpoint is not set to 00000 or 00001 the routine will be executed. The first word read from tape will be read into location 00002 and be executed as soon as the tape is ready again.

Location 00003 will become XXX00004XXXYYYYY from the second word on tape, inserting a new terminating address.

## OPERATION

The 1604 is a stored-program computer. To load a program in the computer a load program (basic service library) is needed. The load program is entered manually. A paper tape reader, a paper tape punch, an electric typewriter, and a set of magnetic tapes are some of the important external devices used for communicating with the 1604. The programmer, before operating any of these devices, should make himself familiar with instructions for these devices and they should be followed in the order recommended.

### LOAD PROGRAM ENTERING

A load program to be entered in storage is usually on bi-octal paper tape.

The following procedure enters the load program:

- 1) Turn on power.
- 2) Master clear, both internal and external.
- 3) Press Start/Step switch once.
- 4) Clear function code and set to 200.
- 5) Clear execution address and set to 00001.
- 6) Set terminal address of buffer in lowest five octal digits of A register right.
- 7) Press Start/Step switch once.
- 8) Load tape into reader.
- 9) Turn on reader motor (wait 10 seconds).
- 10) Raise reader Mode switch to ASSEMBLY position.
- 11) Clear function code and set to 741.
- 12) Clear execution address and set to initial address of buffer.
- 13) Press Start/Step switch once. Wait until tape loads (console lights come on).
- 14) Press Clear switch.
- 15) Perform steps 2 through 8 of operation with pre-stored program.

### STARTING OPERATION WITH PRE-STORED LOAD PROGRAM

When a general loading program which provides for loading other programs is held in storage, the starting procedure is as follows:

- 1) Turn on power (Power On, figure 4-3).
- 2) Make required manual selections:

Selective Jumps  
Selective Stops  
Breakpoint

- 3) Set in operation the external device or devices selected to communicate with the computer. (Follow the instructions for the devices given in this chapter.)
- 4) Master clear, both internal and external (press Clear, then raise it).
- 5) Set Program Address register to address of first instruction of program.
- 6) Begin computer operation (raise Start switch).
- 7) To shut down the equipment after the operation has stopped, follow the instructions as given for each external device.
- 8) Press Power Off button, which disconnects power from all equipments.

## READER

The reader is a CONTROL DATA 350 Paper Tape Reader (figure 4-5). It can read either a 5, a 7, or an 8-level tape. For a bi-octal tape with the 7th level control holes, assembly mode is selected; for a flex or other code, character mode is selected.

- 1) Check if tape basket is at the proper place. Do not allow the tape to fall on the floor.
- 2) Turn tape release lever clockwise to raise tape guide plate.
- 3) Select the desired tape level by means of the tape level switch.

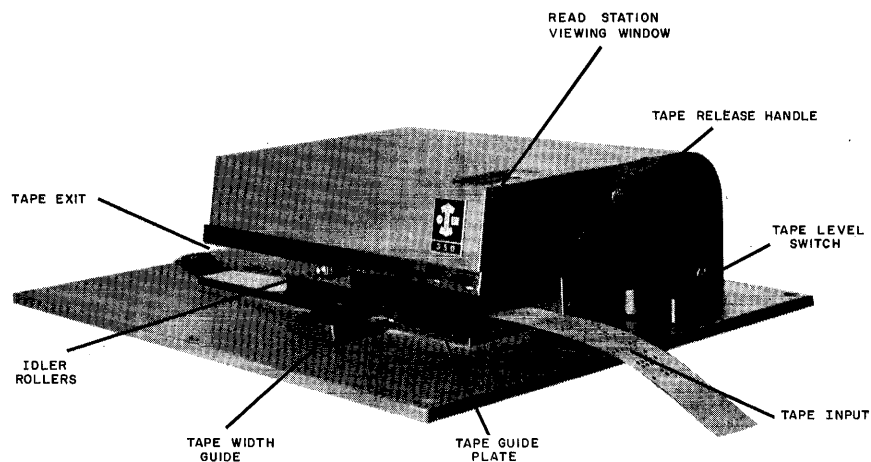


Figure 4-5. Paper Tape Reader

- 4) Holding the tape guide down, slide it so that the marker rests above the proper etched mark on the tape deck surface. The outer position is for 8-level tape, the center for 7-level, and the inner for 5-level.
- 5) Insert tape as shown in figure 4-5. Make sure that the tape is properly aligned.
- 6) Turn Tape Release lever counterclockwise to lower the tape guide.
- 7) Select the desired mode of operation by the Mode switch (figure 4-4) on the control panel.
- 8) Turn on Reader switch on control panel (figure 4-4).
- 9) After the reader has read the tape, remove paper tape from reader and basket; rewind tapes.
- 10) Turn off reader motor.

## PUNCH

The paper tape punch (figure 4-6) is mounted on a hinged rack at the rear of the right wing of the console. Punch tape feeds out of a slot in the compartment door; the chad box is just inside the door.

- 1) To ensure proper performance of the punch, always keep the chad box clean.
- 2) Set Punch switch to SELECT (control panel) and check for sufficient paper in reel.

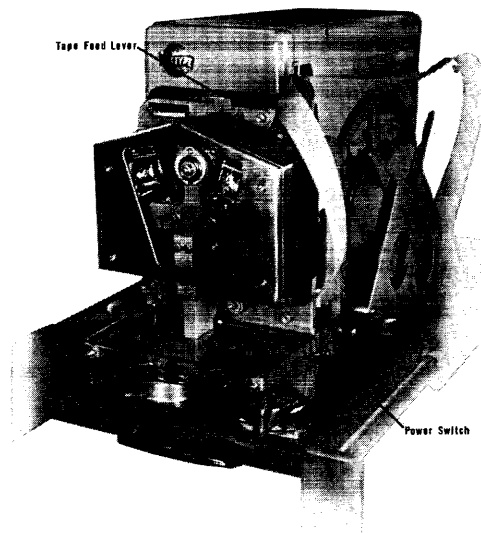


Figure 4-6. Paper Tape Punch

- 3) If the punch has been used, generate a foot of leader by pressing tape feed; remove feed; remove tape and wind it up.
- 4) Perform the following steps to replace a tape roll at punch.
  - a) Remove the tape reel from cradle at side of punch.
  - b) Unscrew tape hold-down assembly, remove old roll, and place new roll on reel. Replace hold-down assembly and mount reel in cradle.
  - c) Thread tape as shown in figure 4-6. Bring tape around lower roller and into guides leading to punch block.
  - d) Turn on punch motor and advance tape through the punch block by pressing the tape feed-out lever (top of punch block).
  - e) Bring leader out through slot in door. Swing punch back into compartment.

## **TYPEWRITER**

The typewriter has all of the characters and functions of a standard electric machine. As a keyboard entry device the typewriter is used only in the character mode. After the program selects keyboard and initiates an input buffer, each striking of a key causes a 6-bit coded character to be entered into the lower six positions of a computer word. The remaining bits of the word are all "0". If the keyboard is selected along with an interrupt feature, each carriage return or tab sends an interrupt signal to the computer. This notifies the program of the entry of data from the keyboard.

When the typewriter is used as an output device certain conditions cause it to hang up until the space bar is struck: receipt of an illegal typewriter code, a code to shift up when the carriage is already up, or a code to shift down when the carriage is already down.

If the typewriter is to be used:

- a) Place paper in it.
- b) Set the switch beneath the righthand corner to ON.

## **TAPE TRANSPORTS**

The tape transports which can be used with 1604 are CONTROL DATA 606 and CONTROL DATA 1607. To use the 606, the CONTROL DATA 1615 Adapter is needed. The codes for the adapter and the tape unit are given in appendix VI.



## 606 TAPE UNIT

### Controls and Indicators

The manual controls and indicators for operating each tape unit are mounted on a panel located below the front door of the unit (figure 4-7). The functions of the controls are described in table 4-4.

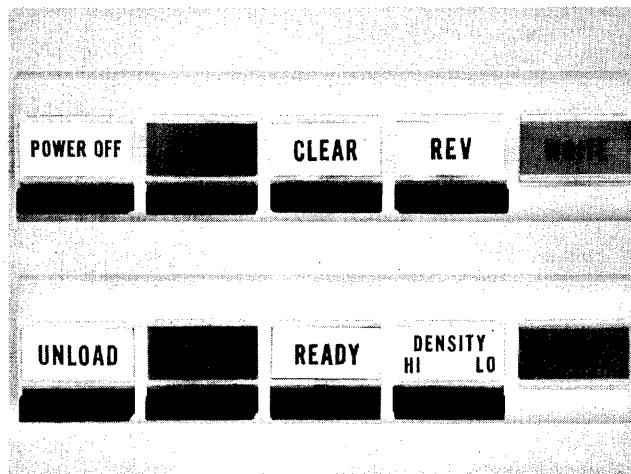


Figure 4-7. Operator Control Panel

TABLE 4-4. 606 CONTROLS AND INDICATORS

NAME		FUNCTION
POWER OFF	*S	Removes power from all components and power supplies.
	**I	Power is available to components and power supplies.
FWD	S	Moves tape forward at 150 ips. Motion stops when end of tape marker is sensed.
	I	Tape is moving forward at 150 ips.
CLEAR	S	Master clears all previous settings and conditions. Stops tape motion immediately. New Manual selections are necessary to reselect tape unit and/or operation required.
	I	606 is cleared.

\*Switch  
\*\*Indicator

TABLE 4-4. 606 CONTROLS AND INDICATORS (CONT'D)

NAME		FUNCTION
REV	*S	Rewinds tape at 225 ips. Motion stops when load point marker is sensed.
	**I	Tape is moving in reverse direction at 150 or 225 ips.
WRITE	I	Write operation is in progress.
UNLOAD	S	Moves tape at 225 ips to unload position (all tape on supply reel). Tape load procedure must be performed to resume operation.
	I	Tape is in unload status.
LOAD POINT	S	Moves tape forward at 150 ips to load point marker. Motion stops when marker is sensed.
	I	Tape is at load point marker.
READY	S	Places 606 under external control.
DENSITY	I	Unit is under external control.
	S	Changes density mode selection.
	I (Hi) I (Low)	High density mode selected. Low density mode selected.
READ	I	Read operation is in progress (not on when reading for horizontal checking during write operation).
UNIT SELECTION	S	10-position switch; 0-7 provide input designation while two standby positions disconnect unit from external control.
	I (White)	Show selected number.
	I (Red)	Fault Condition (power failure, tape not in columns, etc.).
OVERHEAD LIGHTS	I	File protection ring is on reel (unit can write) and tape unit is not in the unload position.

\*Switch  
\*\*Indicator

### Tape Load Procedure

- 1) Make sure that tape unit is properly energized.
- 2) Slide front glass door down to lowest position (figure 4-8).
- 3) Check that supply reel has been file protected as necessary.
- 4) Mount reel on supply reel hub and tighten hub knob. For proper alignment, push reel firmly against hub stop before tightening knob.
- 5) Make sure that tape load arms are in up position.
- 6) Pull sufficient tape from supply reel to reach take-up reel. Thread tape on the outside of the supply tape load arm, over the head assembly, around the outside of the take-up load arm and over the top of the take-up reel hub two or three times.
- 7) Slide tape under head assembly.
- 8) Snap tape load arms down.

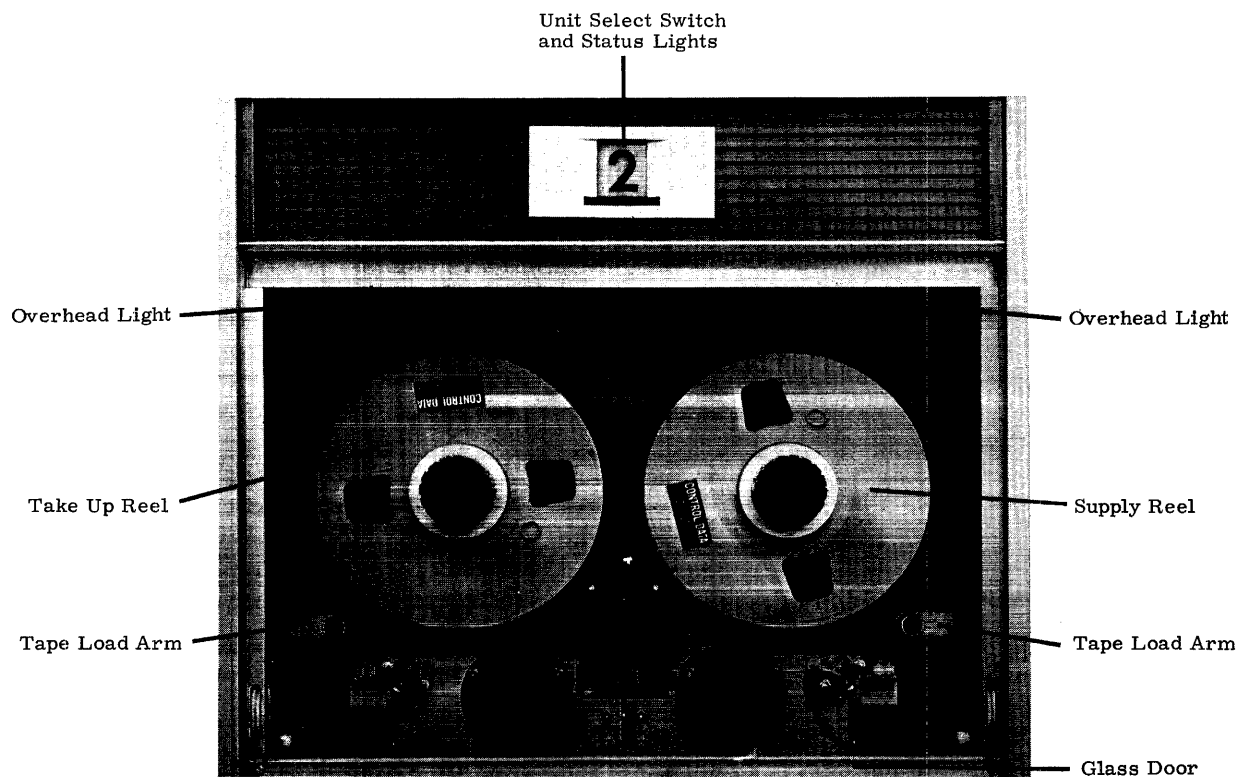


Figure 4-8. 606 Tape Load and Unload Mechanics

- 9) Set Unit Selection switch to one of ten positions (0-7 or standby) to assign a logical program selection number.
- 10) Press Clear switch.
- 11) Press Load Point switch. Tape will drop in columns, move forward, and stop on load point marker. The Load Point light will turn on. (If the light does not turn on, notify maintenance.) If tape continues moving forward for more than 3 or 4 seconds, it indicates either no load point marker was placed on the tape or the operator manually wound the marker onto the take-up reel during step 6.
- 12) If the unit is to be controlled, press the Ready switch. If it is to be manually operated and the Ready switch has been pushed, press the Clear switch.
- 13) Raise the front glass door completely.

If the supply reel contains a file protection ring, the overhead lights should be on, indicating that a write operation may be performed. If the lights are not on, notify maintenance.

#### Tape Unload Procedure

- 1) Press Clear switch.
- 2) Press Unload switch. All tape will automatically be drawn from the take-up reel and wound on the supply reel. The Unload indicator will light.
- 3) Slide down front door.
- 4) Loosen supply reel hub knob and remove supply reel.
- 5) Check if reel needs to be file protected and if it is labeled adequately prior to storage.

#### Special Instructions

In order to simulate an unload condition without removing all tape from the take-up reel, simultaneously press the Clear and Unload switches. The unload condition will be simulated but tape will not move. In order to place the unit in operational status, remove all tape from the vacuum columns by revolving the take-up reel clockwise and the supply reel counterclockwise. Snap the tape load arms down and press the Load Point switch. The tape will move forward and stop on the nearest load point marker. The Load Point indicator will turn on.

If all tape is unwound from the supply reel:

- 1) Snap tape load arms up, if necessary.
- 2) Guide tape around the tape load arms, over the head assembly, and wrap approximately ten turns around the supply reel.
- 3) Slide tape under head assembly.
- 4) Press the Load Point switch.
- 5) As soon as the Forward light turns on, press the Clear switch and then the Reverse switch. Tape will rewind on the nearest load point marker.

The following information is applicable when a number of load point or end of tape markers are used on a single tape.

To move forward from a reflective marker and stop at nearest end of tape marker, press the Forward switch.

To move forward off a reflective marker and stop at nearest load point or end of tape marker, press the Forward and then the Load Point switches. Load Point indicator will light if motion stops at load point marker.

To reverse from a reflection marker and stop at nearest load point marker, press the Unload, Clear, and Reverse switches, in that order.

Tape motion may be stopped at any time by pressing the Clear switch. An unload operation may be performed by pressing the Unload switch.

## **1607 Tape Unit**

### Controls and Indicators

Each tape unit is provided with push buttons for manual operation. These controls are mounted on a panel above the front door (figure 4-9, table 4-5).

### Tape Load Procedure

- 1) Open door to handler.
- 2) Check that file reel to be loaded has been file protected as necessary.
- 3) Mount the reel on the file reel hub and tighten the hub knob. To insure proper reel alignment push the reel firmly against the reel hub stop before tightening the knob. If the file protection ring has been removed from the reel, check that the Write Lockout lamp turns on when the reel is loaded. If the lamp does not turn on call maintenance.

TABLE 4-5. 1607 CONTROLS AND INDICATORS

CONTROL		FUNCTION
REWIND	*S	Controls manual rewind to load point.
	**I	Indicates rewind in progress.
CHANGE TAPE	S	Drops any manual selection and places tape unit in automatic or program control mode.
	I	When lighted, indicates tape rewound under program control and interlocked at load point. The interlock prevents operation of the tape unit until the Stop Manual switch is operated.
WRITE LOCKOUT	S	Drops power from unit and removes program designation.
	I	When lighted, indicates that tape unit is loaded with a reel which does not contain a file protection ring. The tape cannot be written as long as the light is on, but may be read.
1, 2, 3 or 4	S	Designates program selection of unit and applies power to unit. Each new unit designation cancels an existing designation.
	I	Indicates unit selection and power-on condition.
REVERSE	S	Initiates reverse tape motion during manual operation.
	I	Indicates reverse tape motion.
STOP MANUAL	S	Drops unit from program control or drops forward or reverse selection and places unit in manual mode.
	I	Indicates manual mode and ready.
FORWARD	S	Initiates forward tape motion during manual mode.
	I	Indicates forward tape motion.

\*Switch  
 \*\*Indicator

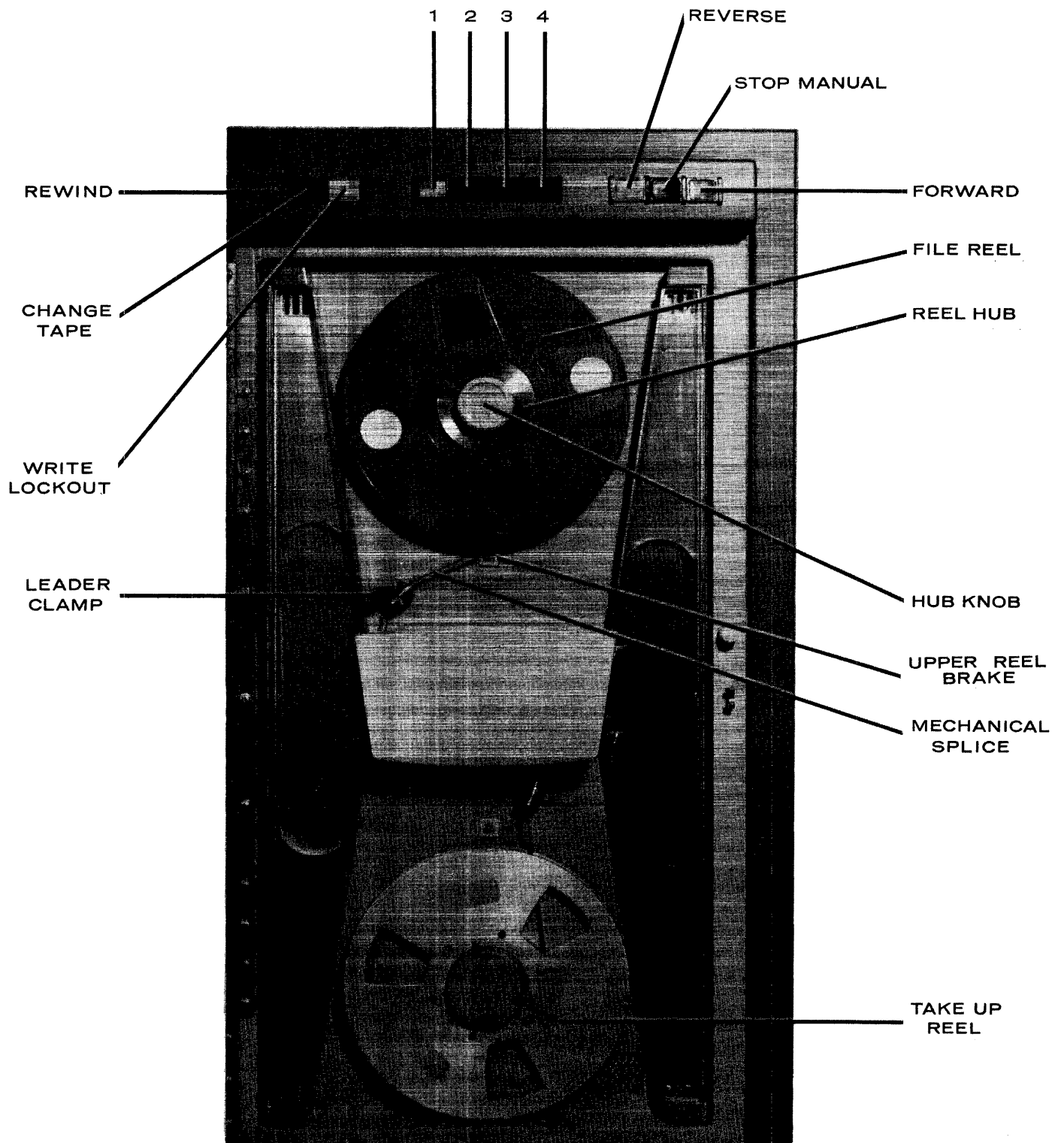


Figure 4-9. 1607 Tape Unit

- 4) Press upper Reel Brake pushbutton to release mechanical brake and check that pulling tape from reel causes it to rotate clockwise. Pull sufficient tape from reel to reach end of permanent machine leader held by leader clamp.
- 5) Connect file tab to permanent machine leader.
- 6) Take up slack by turning file reel while pressing upper Reel Brake pushbutton
- 7) Lift leader clamp and close door.
- 8) Press one of the unit selection switches (1, 2, 3, 4) to apply power to the unit and assign the unit a logical program selection number. Wait two minutes. The Stop Manual lamp should turn on; if not, call maintenance.
- 9) Press Stop Manual.
- 10) Press Rewind button. Unit is ready when Rewind lamp turns off. If Stop Manual lamp remains on, unit is not ready; call maintenance.

#### Tape Unload Procedure

- 1) Press Stop Manual button to select manual mode.
- 2) Press Reverse button to move tape backwards to change tape position.
- 3) Open front door of tape unit.
- 4) To secure tape, lower leader clamp.
- 5) Press the upper Reel Brake button to release the mechanical brake and pull tape from file reel to provide slack.
- 6) Unfasten mechanical splice which connects the file tab to the permanent machine leader.
- 7) Loosen file reel hub knob and remove the file reel.
- 8) Check if reel needs to be file protected and also if it is labeled adequately prior to storage.

#### **FILE PROTECTION RING**

The back of the file reel has a slot near the hub which accepts a plastic file protection ring (figure 4-10). Writing on a tape is possible only when the reel contains a file protection ring. The ring should be removed from the reel after writing is completed to avoid accidental rewriting. Tape may be read either with the ring in place or without it. On the 606 the overhead lights go on immediately after the tape load procedure is



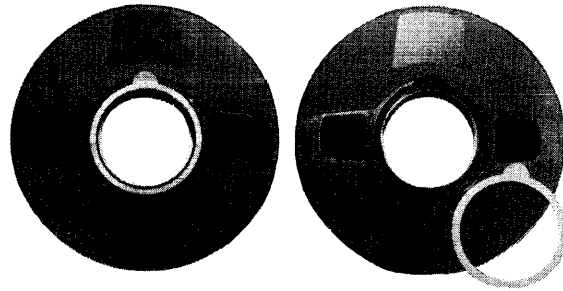


Figure 4-10. File Protection Ring

executed if the file protection ring is in place. The Write Lockout light on the 1607 is off if the file protection ring is in place.

**EMERGENCY PROCEDURE**

A fault indication, or a warning signal from the buzzer, may call for special procedures on the part of the operator.

TABLE 4-6. EMERGENCY PROCEDURE

CONDITION	PROCEDURE
Punch out of tape	Load new roll of tape in punch at end of current operation.
Odd Storage Fault	Master clear. Restart program.
Even Storage Fault	Master clear. Restart program.
Deep End	If all I/O operation has ceased, master clear and restart program. If condition persists, notify maintenance.
Sweep	Place Mode switch in neutral position.
Buzzer Signal	Notify maintenance engineer immediately.

Faults for which the program provides corrective action are: Divide, Shift, Overflow and Exponent Faults. (Refer to appendix.)

# **APPENDIX SECTION**

## APPENDIX I NUMBER SYSTEMS

Any number system may be defined by two characteristics, the radix or base and the modulus. The radix or base is the number of unique symbols used in the system. The decimal system has ten symbols, 0 through 9. Modulus is the number of unique quantities or magnitudes a given system can distinguish. For example, an adding machine with ten digits, or counting wheels, would have a modulus of  $10^{10}-1$ . The decimal system has no modulus because an infinite number of digits can be written, but the adding machine has a modulus because the highest number which can be expressed is 9,999,999,999.

Most number systems are positional, that is, the relative position of a symbol determines its magnitude. In the decimal system, a 5 in the units column represents a different quantity than a 5 in the tens column. Quantities equal to or greater than 1 may be represented by using the 10 symbols as coefficients of ascending powers of the base 10. The number  $984_{10}$  is:

$$\begin{array}{r} 9 \times 10^2 = 9 \times 100 = 900 \\ +8 \times 10^1 = 8 \times 10 = 80 \\ +4 \times 10^0 = 4 \times 1 = \underline{4} \\ \hline 984_{10} \end{array}$$

Quantities less than 1 may be represented by using the 10 symbols as coefficients of ascending negative powers of the base 10. The number  $0.593_{10}$  may be represented as:

$$\begin{array}{r} 5 \times 10^{-1} = 5 \times .1 = .5 \\ +9 \times 10^{-2} = 9 \times .01 = .09 \\ +3 \times 10^{-3} = 3 \times .001 = \underline{.003} \\ \hline 0.593_{10} \end{array}$$

### BINARY NUMBER SYSTEM

Computers operate faster and more efficiently by using the binary number system. There are only two symbols 0 and 1; the base = 2. The following shows the positional value.

. . .	$2^5$	$2^4$	$2^3$	$2^2$	$2^1$	$2^0$	
	=32	=16	=8	=4	=2	=1	Binary point

The binary number 0 1 1 0 1 0 represents:

$$\begin{array}{r}
 0 \times 2^5 = 0 \times 32 = 0 \\
 +1 \times 2^4 = 1 \times 16 = 16 \\
 +1 \times 2^3 = 1 \times 8 = 8 \\
 +0 \times 2^2 = 0 \times 4 = 0 \\
 +1 \times 2^1 = 1 \times 2 = 2 \\
 +0 \times 2^0 = 0 \times 1 = \underline{0} \\
 \hline
 26_{10}
 \end{array}$$

Fractional binary numbers may be represented by using the symbols as coefficients of ascending negative powers of the base.

$$\begin{array}{cccccc}
 & 2^{-1} & 2^{-2} & 2^{-3} & 2^{-4} & 2^{-5} \dots \\
 \text{Binary Point} & . & =1/2 & =1/4 & =1/8 & =1/16 & =1/32
 \end{array}$$

The binary number 0.10 110 may be represented as:

$$\begin{array}{r}
 1 \times 2^{-1} = 1 \times 1/2 = 1/2 = 8/16 \\
 +0 \times 2^{-2} = 0 \times 1/4 = 0 = 0 \\
 +1 \times 2^{-3} = 1 \times 1/8 = 1/8 = 2/16 \\
 +1 \times 2^{-4} = 1 \times 1/16 = 1/16 = \underline{1/16} \\
 \hline
 11/16_{10}
 \end{array}$$

### OCTAL NUMBER SYSTEM

The octal number system uses eight discrete symbols, 0 through 7. With the base eight the positional value is:

$$\begin{array}{ccccccc}
 \dots & 8^5 & 8^4 & 8^3 & 8^2 & 8^1 & 8^0 \\
 & 32,768 & 4,096 & 512 & 64 & 8 & 1
 \end{array}$$

The octal number  $513_8$  represents:

$$\begin{array}{r}
 5 \times 8^2 = 5 \times 64 = 320 \\
 +1 \times 8^1 = 1 \times 8 = 8 \\
 +3 \times 8^0 = 3 \times 1 = \underline{3} \\
 \hline
 331_{10}
 \end{array}$$

Fractional octal numbers may be represented by using the symbols as coefficients of ascending negative powers of the base.

$$\begin{array}{ccccccc}
 8^{-1} & & 8^{-2} & & 8^{-3} & & 8^{-4} & \dots \\
 1/8 & & 1/64 & & 1/512 & & 1/4096 & 
 \end{array}$$

The octal number 0.4520 represents:

$$\begin{array}{r}
 4 \times 8^{-1} = 4 \times 1/8 = 4/8 = 256/512 \\
 +5 \times 8^{-2} = 5 \times 1/64 = 5/64 = 40/512 \\
 +2 \times 8^{-3} = 2 \times 1/512 = 2/512 = \underline{2/512} \\
 \hline
 298/512 = 149/256_{10}
 \end{array}$$

### ARITHMETIC

#### ADDITION AND SUBTRACTION

Binary numbers are added according to the following rules:

$$\begin{array}{l}
 0 + 0 = 0 \\
 0 + 1 = 1 \\
 1 + 0 = 1 \\
 1 + 1 = 0 \text{ with a carry of } 1
 \end{array}$$

The addition of two binary numbers proceeds as follows (the decimal equivalents verify the result):

$$\begin{array}{r}
 \text{Augend} \qquad \qquad 0111 \qquad (7) \\
 \text{Addend} \qquad \qquad \underline{+0100} \qquad + (4) \\
 \text{Partial Sum} \qquad \qquad 0011 \\
 \text{Carry} \qquad \qquad \underline{1} \\
 \text{Sum} \qquad \qquad \underline{1011} \qquad (11)
 \end{array}$$

Subtraction may be performed as an addition:

$$\begin{array}{r}
 8 \text{ (minuend)} \\
 \underline{-6 \text{ (subtrahend)}} \\
 2 \text{ (difference)}
 \end{array}
 \quad \text{or} \quad
 \begin{array}{r}
 8 \text{ (minuend)} \\
 \underline{+4 \text{ (10's complement or subtrahend)}} \\
 2 \text{ (difference - omit carry)}
 \end{array}$$

The second method shows subtraction performed by the "adding the complement" method. The omission of the carry in the illustration has the effect of reducing the result by 10.

### One's Complement

The 1604 performs all arithmetic operations in the binary one's complement mode. In this system, positive numbers are represented by the binary equivalent and negative numbers in one's complement notation.

The one's complement representation of a number is found by subtracting each bit of the number from 1. For example:

$$\begin{array}{r} 1111 \\ -1001 \\ \hline 0110 \end{array} \quad \begin{array}{l} 9 \\ \\ \text{(one's complement of 9)} \end{array}$$

This representation of a negative binary quantity may also be obtained by substituting "1's" for "0's" and "0's" for "1's".

The value zero can be represented in one's complement notation in two ways:

$$\begin{array}{ll} 0000 \rightarrow 00_2 & \text{Positive (+) Zero} \\ 1111 \rightarrow 11_2 & \text{Negative (-) Zero} \end{array}$$

The rules regarding the use of these two forms for computation are:

- 1) Both positive and negative zero are acceptable as arithmetic operands.
- 2) If the result of an arithmetic operation is zero, it will be expressed as positive zero. The one exception to this rule is when negative zero is added to negative zero.\* In this case, the result is negative zero.

One's complement notation applies not only to arithmetic operations performed in A, but also to the modification of execution addresses in the  $U^2$  register. During address modification, the modified address will equal  $77777_8$  only if the unmodified execution address equals  $77777_8$  and  $b = 0$  or  $(B^b) = 77777_8$ .

### Two's Complement

The counters in the computer use two's complement arithmetic. A counter is a register with provisions for increasing its contents by one if it is additive (P register) or decreasing its contents by one if it is subtractive (R register). A two's complement counter is open-ended; there is no end-around carry or borrow.

---

\* When a 1604 instruction calls for subtracting positive zero from negative zero, the computer complements the subtrahend and adds so that the actual operation is the addition of negative zero to negative zero and the result is negative zero.

Positive numbers have the same representation in both systems while negative values differ by one count.

Count	2's comp. rep.	1's comp. rep.
+2	00010	00010
+1	00001	00001
0	00000	00000
-1	11111	11110
-2	11110	11101

The difference in the representation of negative values in these two systems is due to the skipping of the "all one's" count in one's complement notation. In the one's complement system the end-around-carry feature of the register automatically changes a count of all one's to all zeros. (Note exception under one's complement.)

As an example, if the content of a subtractive counter is positive seven (0111) and is to be reduced by one, add the two's complement expression of negative one, (1111), to 0111 as shown below. The result is six.

$$\begin{array}{r}
 0111 \\
 +1111 \\
 \hline
 0110
 \end{array}$$

Note that the two's complement expression for a negative number may also be formed by adding one to the one's complement representation of the number.

## MULTIPLICATION

Binary multiplication proceeds according to the following rules:

$$\begin{array}{l}
 0 \times 0 = 0 \\
 0 \times 1 = 0 \\
 1 \times 0 = 0 \\
 1 \times 1 = 1
 \end{array}$$

Multiplication is always performed on a bit-by-bit basis. Carries do not result from multiplication, since the product of any two bits is always a single bit.

Decimal example:

multiplicand	14	
multiplier	<u>12</u>	
partial products	28	
	<u>14</u>	(shifted one place left)
product	168	<sub>10</sub>

The shift of the second partial product is a shorthand method for writing the true value 140.

Binary example:

multiplicand	(14)	1110	
multiplier	(12)	<u>1100</u>	
partial products		0000	
		0000	shift to place digits
		1110	in proper columns
		<u>1110</u>	
product	(168) <sub>10</sub>	10101000	<sub>2</sub>

The computer determines the running subtotal of the partial products. Rather than shifting the partial product to the left to position it correctly, the computer right shifts the summation of the partial products one place before the next addition is made. When the multiplier bit is "1", the multiplicand is added to the running total and the results are shifted to the right one place. When the multiplier bit is "0", the partial product subtotal is shifted to the right (in effect, the quantity has been multiplied by  $10_2$ ).

### DIVISION

The following example shows the familiar method of decimal division:

divisor	13	14	quotient
		<u>185</u>	dividend
		13	
		55	partial dividend
		<u>52</u>	
		3	remainder



The computer performs division in a similar manner (using binary equivalents):

divisor	1101	$\begin{array}{r} 1110 \\ \hline 10111001 \\ \hline 1101 \\ \hline 10100 \\ \hline 1101 \\ \hline 1110 \\ \hline 1101 \\ \hline 11 \end{array}$	quotient (14) dividend   partial dividends   remainder (3)
---------	------	---	---

However, instead of shifting the divisor right to position it for subtraction from the partial dividend (shown above), the computer shifts the partial dividend left, accomplishing the same purpose and permitting the arithmetic to be performed in the A register. The computer counts the number of shifts, which is the number of quotient digits to be obtained; after the correct number of counts, the routine is terminated.

### CONVERSIONS

The procedures that may be used when converting from one number system to another are power addition, double dabble, and substitution.

Recommended Conversion Procedures (Integer and Fractional)

Conversion	Recommended Method
Binary to Decimal	Power Addition
Octal to Decimal	Power Addition
Decimal to Binary	Double Dabble
Decimal to Octal	Double Dabble
Binary to Octal	Substitution
Octal to Binary	Substitution
<b>GENERAL RULES</b>  $r_i > r_f$ : use Double Dabble, Substitution $r_i < r_f$ : use Power Addition, Substitution $r_i$ = Radix of initial system $r_f$ = Radix of final system	

## POWER ADDITION

To convert a number from  $r_i$  to  $r_f$  ( $r_i < r_f$ ) write the number in its expanded  $r_i$  polynomial form and simplify using  $r_f$  arithmetic.

### EXAMPLE 1 Binary to Decimal (Integer)

$$\begin{aligned} 010111_2 &= 1(2^4) + 0(2^3) + 1(2^2) + 1(2^1) + 1(2^0) \\ &= 1(16) + 0(8) + 1(4) + 1(2) + 1(1) \\ &= 16 + 0 + 4 + 2 + 1 \\ &= 23_{10} \end{aligned}$$

### EXAMPLE 2 Binary to Decimal (Fractional)

$$\begin{aligned} .0101_2 &= 0(2^{-1}) + 1(2^{-2}) + 0(2^{-3}) + 1(2^{-4}) \\ &= 0 + 1/4 + 0 + 1/16 \\ &= 5/16_{10} \end{aligned}$$

### EXAMPLE 3 Octal to Decimal (Integer)

$$\begin{aligned} 324_8 &= 3(8^2) + 2(8^1) + 4(8^0) \\ &= 3(64) + 2(8) + 4(1) \\ &= 192 + 16 + 4 \\ &= 212_{10} \end{aligned}$$

### EXAMPLE 4 Octal to Decimal (Fractional)

$$\begin{aligned} .44_8 &= 4(8^{-1}) + 4(8^{-2}) \\ &= 4/8 + 4/64 \\ &= 36/64_{10} \end{aligned}$$

## DOUBLE DABBLE

To convert a whole number from  $r_i$  to  $r_f$  ( $r_i > r_f$ ):

- 1) Divide  $r_i$  by  $r_f$  using  $r_i$  arithmetic
- 2) The remainder is the lowest order bit in the new expression
- 3) Divide the integral part from the previous operation by  $r_f$
- 4) The remainder is the next higher order bit in the new expression
- 5) The process continues until the division produces only a remainder which will be the highest order bit in the  $r_f$  expression.

To convert a fractional number from  $r_i$  to  $r_f$ :

- 1) Multiply  $r_i$  by  $r_f$  using  $r_i$  arithmetic
- 2) The integral part is the highest order bit in the new expression
- 3) Multiply the fractional part from the previous operation by  $r_f$
- 4) The integral part is the next lower order bit in the new expression
- 5) The process continues until sufficient precision is achieved or the process terminates.

**EXAMPLE 1**                  Decimal to Binary (Integer)

$45 \div 2 = 22$ remainder 1; record	1
$22 \div 2 = 11$ remainder 0; record	0
$11 \div 2 = 5$ remainder 1; record	1
$5 \div 2 = 2$ remainder 1; record	1
$2 \div 2 = 1$ remainder 0; record	0
$1 \div 2 = 0$ remainder 1; record	1
	101101

Thus:  $45_{10} = 101101_2$

**EXAMPLE 2**                  Decimal to Binary (Fractional)

$.25 \times 2 = 0.5$ ; record	0
$.5 \times 2 = 1.0$ ; record	1
$.0 \times 2 = 0.0$ ; record	0
	.010

Thus:  $.25_{10} = .010_2$

**EXAMPLE 3**                  Decimal to Octal (Integer)

$273 \div 8 = 34$ remainder 1; record	1
$34 \div 8 = 4$ remainder 2; record	2
$4 \div 8 = 0$ remainder 4; record	4
	421

Thus:  $273_{10} = 421_8$

**EXAMPLE 4**          Decimal to Octal (Fractional)

$$\begin{array}{r} .55 \times 8 = 4.4; \text{ record} \qquad 4 \\ .4 \times 8 = 3.2; \text{ record} \qquad 3 \\ .2 \times 8 = 1.6; \text{ record} \qquad 1 \\ -- \quad -- \qquad \qquad \qquad - \\ -- \quad -- \qquad \qquad \qquad - \\ \hline .431\dots \end{array}$$

Thus:  $.55_{10} = .431\dots_8$

**SUBSTITUTION**

This method permits easy conversion between octal and binary representations of a number. If a number in binary notation is partitioned into triplets to the right and left of the binary point, each triplet may be converted into an octal digit. Similarly each octal digit may be converted into a triplet of binary digits.

**EXAMPLE 1**          Binary to Octal

$$\begin{array}{l} \text{Binary} = 110\ 000 . 001\ 010 \\ \text{Octal} = 6\ 0 . 1\ 2 \end{array}$$

**EXAMPLE 2**          Octal to Binary

$$\begin{array}{l} \text{Octal} = 6\ 5\ 0 . 2\ 2\ 7 \\ \text{Binary} = 110\ 101\ 000 . 010\ 010\ 111 \end{array}$$

### COMMON PURE NOTATIONS

Decimal	Binary	Octal
00	00000	00
01	00001	01
02	00010	02
03	00011	03
04	00100	04
05	00101	05
06	00110	06
07	00111	07
08	01000	10
09	01001	11
10	01010	12
11	01011	13
12	01100	14
13	01101	15
14	01110	16
15	01111	17
16	10000	20
17	10001	21

### POWERS OF COMMON NUMBER SYSTEMS

$2^0 = 1$	$8^0 = 1$	$10^0 = 1$
$2^1 = 2$	$8^1 = 8$	$10^1 = 10$
$2^2 = 4$	$8^2 = 64$	$10^2 = 100$
$2^3 = 8$	$8^3 = 512$	$10^3 = 1,000$
$2^4 = 16$	$8^4 = 4,096$	$10^4 = 10,000$
$2^5 = 32$	$8^5 = 32,768$	$10^5 = 100,000$
$2^6 = 64$	$8^6 = 262,144$	$10^6 = 1,000,000$
$2^7 = 128$	$8^7 = 2,097,152$	
$2^8 = 256$	$8^8 = 16,777,216$	
$2^9 = 512$		
$2^{10} = 1,024$		

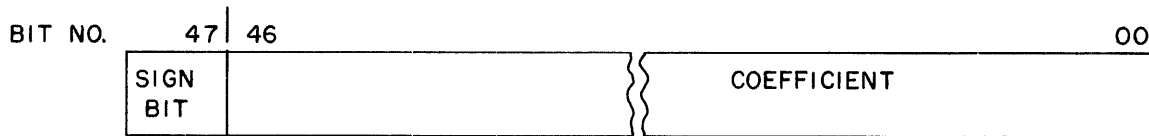
## FIXED POINT AND FLOATING POINT NUMBERS

Any number may be expressed in the form  $kB^n$ , where  $k$  is a coefficient,  $B$  a base number, and the exponent  $n$  the power to which the base number is raised.

A fixed point number assumes:

- 1) The exponent  $n = 0$  for all fixed point numbers.
- 2) The coefficient,  $k$ , occupies the same bit positions within the computer word for all fixed point numbers.
- 3) The radix (binary) point remains fixed with respect to one end of the expression.

A 1604 fixed point number consists of a sign bit and coefficient as shown below. The upper bit of any 1604 fixed point number designates the sign of the coefficient (47 lower order bits). If the bit is "1", the quantity is negative since negative numbers are represented in one's complement notation; a "0" sign bit signifies a positive coefficient.



The coefficient may be an integer or fraction. The radix (binary) point, in the case of an integer, is assumed to be immediately to the right of the lowest order bit (00). In the case of the fraction, the point is just to the right of the sign bit.

In many instances, the values in a fixed point operation may be too large or too small to be expressed by the computer. The programmer must position the numbers within the word format so they can be represented with sufficient precision. The process, called scaling, consists of shifting the values a predetermined number of places. The numbers must be positioned far enough to the right in the register to prevent overflow but far enough to the left to maintain precision. The scale factor (number of places shifted) is expressed as the power of the base. For example,  $5,100,000_{10}$  may be expressed as  $0.51 \times 10^7$ ,  $0.051 \times 10^8$ ,  $0.0051 \times 10^9$ , etc. The scale factors are 7, 8, and 9.

Since only the coefficient is used by the computer, the programmer is responsible for remembering the scale factors. Also, the possibility of an overflow during intermediate operations must be considered. For example, if two fractions in fixed point format are

multiplied, the result is a number  $< 1$ . If the same two fractions are added, subtracted, or divided, the result may be greater than one and an overflow will occur. Similarly, if two integers are multiplied, divided, subtracted or added, the likelihood of an overflow is apparent.

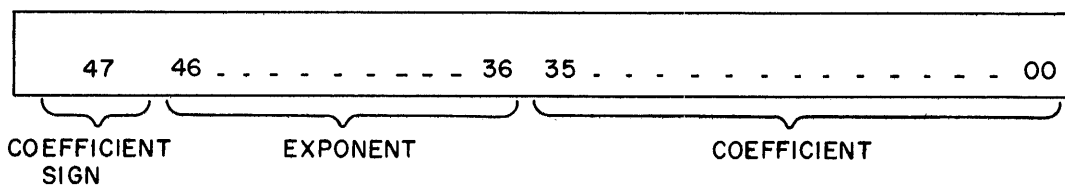
As an alternative to fixed point operation, a method involving a variable radix point, called floating point, is used. This significantly reduces the amount of bookkeeping required on the part of the programmer.

By shifting the radix point and increasing or decreasing the value of the exponent, widely varying quantities which do not exceed the capacity of the machine may be handled.

Floating point numbers within the computer are represented in a form similar to that used in "scientific" notation, that is, a coefficient or fraction multiplied by a number raised to a power. Since the computer uses only binary numbers, the numbers are multiplied by powers of two.

$$F \cdot 2^E \quad \text{where: } F = \text{fraction} \\ E = \text{exponent}$$

In floating point, different coefficients need not relate to the same power of the base as they do in fixed point format. Therefore, the construction of a floating point number includes not only the coefficient but also the exponent.



Coefficient

The coefficient consists of a 36-bit fraction in the 36 lower-order positions of the floating point word. The coefficient is a normalized fraction; it is equal to or greater than 1/2 but less than 1. The highest order bit position (47) is occupied by the sign bit of the coefficient. If the sign bit is a "0", the coefficient is positive; a "1" bit denotes a negative fraction (negative fractions are represented in one's complement notation).

Exponent

The floating point exponent is expressed as an 11-bit quantity with a value ranging from  $0000_8$  to  $3777_8$ . It is formed by adding a true positive exponent and a bias of  $2000_8$  or a true negative exponent and a bias of  $1777_8$ . This results in a range of biased exponents as shown below.

True Positive Exponent	Biased Exponent	True Negative Exponent	Biased Exponent
+0	2000	-0	2000*
+1	2001	-1	1776
+2	2002	-2	1775
--	----	--	----
--	----	--	----
+1776	3776	-1776	0001
+1777 <sub>8</sub>	3777 <sub>8</sub>	-1777 <sub>8</sub>	0000 <sub>8</sub>

The exponent is biased so that floating point operands can be compared with each other in the normal fixed point mode.

As an example, compare the unbiased exponents of  $+52_8$  and  $+0.02_8$  (Example 1).

<b>EXAMPLE 1</b>	Number = +52	
0	0 0 000 000 110	(36 bits)
Coefficient Sign	Exponent	Coefficient
	Number = +0.02	
0	1 1 111 111 011	(36 bits)
Coefficient Sign	Exponent	Coefficient

In this case  $+0.02$  appears to be larger than  $+52$  because of the larger exponent. If, however, both exponents are biased, (Example 2) changing the sign of both exponents makes  $+52$  greater than  $+0.02$ .

\* Minus zero is sensed as positive zero by the computer and is therefore biased by  $2000_8$  rather than  $1777_8$ .



EXAMPLE 2	Number = +52 <sub>8</sub>				
0	1 0	000	000	110	(36 bits)
Coefficient Sign	Exponent			Coefficient	
	Number = +0.02 <sub>8</sub>				
0	0 1	111	111	011	(36 bits)
Coefficient Sign	Exponent			Coefficient	

When bias is used with the exponent floating-point operation is more versatile since floating-point operands can be compared with each other in the normal fixed point mode.

### CONVERSION PROCEDURES

#### Fixed Point to Floating Point

- 1) Express the number in binary.
- 2) Normalize the number. A normalized number has the most significant 1 positioned immediately to the right of the binary point and is expressed in the range  $1/2 \leq k < 1$ .
- 3) Inspect the sign of the true exponent. If the sign is positive add 2000<sub>8</sub> (bias) to the true exponent of the normalized number. If the sign is negative add the bias 1777<sub>8</sub> to the true exponent of the normalized number. In either case, the resulting exponent is the biased exponent.
- 4) Assemble the number in floating point.
- 5) Inspect the sign of the coefficient. If negative, complement the assembled floating point number to obtain the true floating point representation of the number. If the sign of the coefficient is positive the assembled floating point number is the true representation.

EXAMPLE 1 Convert +4.0 to floating point

- 1) The number is expressed in octal.
- 2) Normalize.  $4.0 = 4.0 \times 8^0 = 0.100 \times 2^3$ .
- 3) Since the sign of the true exponent is positive, add 2000<sub>8</sub> (bias) to the true exponent. Biased exponent = 2000 + 3.

- 4) Assemble number in floating point format.  
 Coefficient = 400 000 000 000<sub>8</sub>  
 Biased Exponent = 2003<sub>8</sub>  
 Assembled word = 2003 400 000 000 000<sub>8</sub>
- 5) Since the sign of the coefficient is positive, the floating point representation of +4.0 is as shown. If, however, the sign of the coefficient were negative, it would be necessary to complement the entire floating point word.

**EXAMPLE 2** Convert -4.0 to floating point

- 1) The number is expressed in octal.
- 2) Normalize.  $-4.0 = -4.0 \times 8^0 = -0.100 \times 2^3$
- 3) Since the sign of the true exponent is positive, add 2000<sub>8</sub> (bias) to the true exponent. Biased exponent = 2000 + 3.
- 4) Assemble number in floating point format.  
 Coefficient = 400 000 000 000<sub>8</sub>  
 Biased Exponent = 2003<sub>8</sub>  
 Assembled word = 2003 400 000 000 000<sub>8</sub>
- 5) Since the sign of the coefficient is negative, the assembled floating point word must be complemented. Therefore, the true floating point representation for -4.0 = 5774 377 777 777 777<sub>8</sub>

**EXAMPLE 3** Convert 0.5<sub>10</sub> to floating point

- 1) Convert to octal.  $0.5_{10} = 0.4_8$
- 2) Normalize.  $0.4 = 0.4 \times 8^0 = 0.100 \times 2^0$
- 3) Since the sign of the true exponent is positive, add 2000<sub>8</sub> (bias) to the true exponent. Biased exponent = 2000 + 0.
- 4) Assemble number in floating point format.  
 Coefficient = 400 000 000 000<sub>8</sub>  
 Biased Exponent = 2000<sub>8</sub>  
 Assembled word = 2000 400 000 000 000<sub>8</sub>
- 5) Since the sign of the coefficient is positive, the floating point representation of +0.5<sub>10</sub> is as shown. If, however, the sign of the coefficient were negative, it would be necessary to complement the entire floating point word. This example is a special case of floating point since the exponent of the normalized number is 0 and could be represented as -0. The exponent would then be biased by 1777<sub>8</sub> instead of 2000<sub>8</sub> because of the negative exponent. The 1604, however, recognizes -0 as +0 and biases the exponent by 2000<sub>8</sub>.

**EXAMPLE 4** Convert  $0.04_8$  to floating point

- 1) The number is expressed in octal.
- 2) Normalize.  $0.04 = 0.04 \times 8^0 = 0.4 \times 8^{-1} = 0.100 \times 2^{-3}$ .
- 3) Since the sign of the true exponent is negative, add  $1777_8$  (bias) to the true exponent. Biased exponent =  $1777_8 + (-3) = 1774_8$ .
- 4) Assemble number in floating point format.  
Coefficient =  $400\ 000\ 000\ 000_8$   
Biased Exponent =  $1774_8$   
Assembled word =  $1774\ 400\ 000\ 000\ 000_8$
- 5) Since the sign of the coefficient is positive, the floating point representation of  $0.04_8$  is as shown. If, however, the sign of the coefficient were negative, it would be necessary to complement the entire floating point word.

Floating Point to Fixed Point Format

- 1) If the floating point number is negative, complement the entire floating point word and record the fact that the quantity is negative. The exponent is now in a true biased form.
- 2) If the biased exponent is equal to or greater than  $2000_8$  subtract  $2000_8$  to obtain the true exponent. If less than  $2000_8$  subtract  $1777_8$  to obtain true exponent.
- 3) Separate the coefficient and exponent. If the true exponent is negative the binary point should be moved to the left the number of bit positions indicated by the true exponent. If the true exponent is positive, the binary point should be moved to the right the number of bit positions indicated by the true exponent.
- 4) The coefficient has now been converted to fixed binary. The sign of the coefficient will be negative if the floating point number was complemented in step one. (The sign bit must be extended if the quantity is placed in a register.)
- 5) Represent the fixed binary number in fixed octal notation.

**EXAMPLE 1** Convert floating point number  $2003\ 400\ 000\ 000\ 000_8$  to fixed octal

- 1) The floating point number is positive and remains uncomplemented.
- 2) The biased exponent  $> 2000_8$ , therefore subtract  $2000_8$  from the biased exponent to obtain the true exponent of the number.  $2003 - 2000 = +3$
- 3) Coefficient =  $400\ 000\ 000\ 000_8 = .100_2$ . Move binary point to the right 3 places.  
Coefficient =  $100.0_2$

- 4) The sign of the coefficient is positive because the floating point number was not complemented in step one.
- 5) Represent in fixed octal notation.  $100.0 \times 2^0 = 4.0 \times 8^0$

**EXAMPLE 2** Convert floating point number  $5774\ 377\ 777\ 777\ 777_8$  to fixed octal

- 1) The sign of the coefficient is negative, therefore, complement the floating point number.

$$\text{Complement} = 2003\ 400\ 000\ 000\ 000_8$$

- 2) The biased exponent (in complemented form)  $> 2000_8$ , therefore subtract  $2000_8$  from the biased exponent to obtain the true exponent of the number.  
 $2003 - 2000 = +3$

- 3) Coefficient =  $4000\ 000\ 000\ 000_8 = 0.100_2$   
Move binary point to the right 3 places.

$$\text{Coefficient} = 100.0_2$$

- 4) The sign of the coefficient will be negative because the floating point number was originally complemented.
- 5) Convert to fixed octal.  $-100.0_2 = -4.0_8$

**EXAMPLE 3** Convert floating point number  $1774\ 400\ 000\ 000\ 000_8$  to fixed octal

- 1) The floating point number is positive and remains uncomplemented.
- 2) The biased exponent  $< 2000_8$ , therefore subtract  $1777_8$  from the biased exponent to obtain the true exponent of the number.  $1774_8 - 1777_8 = -3$

- 3) Coefficient =  $400\ 000\ 000\ 000_8 = .100_2$   
Move binary point to the left 3 places.

$$\text{Coefficient} = .000100_2$$

- 4) The sign of the coefficient is positive because the floating point number was not complemented in step one.
- 5) Represent in fixed octal notation.  $.000100_2 = .04_8$

## APPENDIX II

### FAULTS

Certain fault conditions may occur in the execution of a computer program which may be sensed by EXF instructions. The occurrence of the fault does not stop operation but sets an indicator that can be sensed. A fault is visually indicated on the console.

#### Shift Fault

Any attempt to shift a register more than  $127_{10}$  ( $177_8$ ) places right or left results in a shift fault. If the fault exists, the indicator is set prior to execution of the shift instruction and the shift fault background light on the console display panel is lighted. The shifts will be performed regardless of the status of the fault indicator. If an interrupt has been selected, the main program will be interrupted after executing the shift instruction. The shift fault may be sensed by 74 7 00120, 1.

#### Divide Fault

A divide fault occurs in fixed point divide instructions (25 and 27) when the divisor is zero or the required quotient exceeds the 47-bit capacity of the quotient register, Q. The sign bit of Q is examined at the end of the division phase. If it is equal to "1", a divide fault has occurred. If an interrupt has been selected, the main program will be interrupted after the divide instruction is completed. A divide fault is sensed by a 74 0 00110, 1.

#### Overflow Fault

An overflow fault is produced when the capacity of the A register ( $> 2^{47}-1$ ) is exceeded. The fault is sensed after the arithmetic operation causing the overflow is completed. An overflow may be sensed by a 74 7 00130, 1.

If an interrupt has been selected, the main program will not be halted until one or two of the next instructions have been executed because the scanner may not immediately recognize the interrupt. If the scanner is within four scanning positions of the interrupt, after executing the first instruction, the second instruction will not be executed. If it is farther than four positions from the interrupt, the second instruction could also be executed.

A pass or sense overflow instruction could be programmed after the arithmetic instruction which causes the overflow. The interrupt would, in this case, be recognized before executing succeeding instructions which might alter the contents of the A register.

### Exponent (Floating Point Range) Fault

The exponent fault occurs during floating point instructions when the exponent of the result, after rounding and normalizing, is  $\geq 2^{+10}$  (overflow) or  $\leq 2^{-10}$  (underflow). The exponent fault is sensed by a 74 7 00140, 1.

If an interrupt has been selected on exponent fault, the main program may not be interrupted until one and possibly two of the next instructions are executed.

By using a pass or sense exponent fault instruction following the floating point instruction, the interrupt is recognized prior to executing succeeding instructions.

### Even and Odd Storage Faults

These faults indicate a failure in computer storage and turn on background lights on the console display. The indicators may be cleared by an internal master clear. If a storage fault is produced, maintenance should be notified.

# APPENDIX III

## TABLE OF POWERS OF TWO

$2^n$	$n$	$2^{-n}$															
1	0	1.0															
2	1	0.5															
4	2	0.25															
8	3	0.125															
16	4	0.062 5															
32	5	0.031 25															
64	6	0.015 625															
128	7	0.007 812 5															
256	8	0.003 906 25															
512	9	0.001 953 125															
1 024	10	0.000 976 562 5															
2 048	11	0.000 488 281 25															
4 096	12	0.000 244 140 625															
8 192	13	0.000 122 070 312 5															
16 384	14	0.000 061 035 156 25															
32 768	15	0.000 030 517 578 125															
65 536	16	0.000 015 258 789 062 5															
131 072	17	0.000 007 629 394 531 25															
262 144	18	0.000 003 814 697 265 625															
524 288	19	0.000 001 907 348 632 812 5															
1 048 576	20	0.000 000 953 674 316 406 25															
2 097 152	21	0.000 000 476 837 158 203 125															
4 194 304	22	0.000 000 238 418 579 101 562 5															
8 388 608	23	0.000 000 119 209 289 550 781 25															
16 777 216	24	0.000 000 059 604 644 775 390 625															
33 554 432	25	0.000 000 029 802 322 387 695 312 5															
67 108 864	26	0.000 000 014 901 161 193 847 656 25															
134 217 728	27	0.000 000 007 450 580 596 923 828 125															
268 435 456	28	0.000 000 003 725 290 298 461 914 062 5															
536 870 912	29	0.000 000 001 862 645 149 230 957 031 25															
1 073 741 824	30	0.000 000 000 931 322 574 615 478 515 625															
2 147 483 648	31	0.000 000 000 465 661 287 307 739 257 812 5															
4 294 967 296	32	0.000 000 000 232 830 643 653 869 628 906 25															
8 589 934 592	33	0.000 000 000 116 415 321 826 934 814 453 125															
17 179 869 184	34	0.000 000 000 058 207 660 913 467 407 226 562 5															
34 359 738 368	35	0.000 000 000 029 103 830 456 733 703 613 281 25															
68 719 476 736	36	0.000 000 000 014 551 915 228 366 851 806 640 625															
137 438 953 472	37	0.000 000 000 007 275 957 614 183 425 903 320 312 5															
274 877 906 944	38	0.000 000 000 003 637 978 807 091 712 951 660 156 25															
549 755 813 888	39	0.000 000 000 001 818 989 403 545 856 475 830 078 125															
1 099 511 627 776	40	0.000 000 000 000 909 494 701 772 928 237 915 039 062 5															
2 199 023 255 552	41	0.000 000 000 000 454 747 350 886 464 118 957 519 531 25															
4 398 046 511 104	42	0.000 000 000 000 227 373 675 443 232 059 478 759 765 625															
8 796 093 022 208	43	0.000 000 000 000 113 686 837 721 616 029 739 379 882 812 5															
17 592 186 044 416	44	0.000 000 000 000 056 843 418 860 808 014 869 689 941 406 25															
35 184 372 088 832	45	0.000 000 000 000 028 421 709 430 404 007 434 844 970 703 125															
70 368 744 177 664	46	0.000 000 000 000 014 210 854 715 202 003 717 422 485 351 562 5															
140 737 488 355 328	47	0.000 000 000 000 007 105 427 357 601 001 858 711 242 675 781 25															
281 474 976 710 656	48	0.000 000 000 000 003 552 713 678 800 500 929 355 621 337 890 625															
562 949 953 421 312	49	0.000 000 000 000 001 776 356 839 400 250 464 677 810 668 945 312 5															
1 125 899 906 842 624	50	0.000 000 000 000 000 888 178 419 700 125 232 338 905 334 472 656 25															
2 251 799 813 685 248	51	0.000 000 000 000 000 444 089 209 850 062 616 169 452 667 236 328 125															
4 503 599 627 370 496	52	0.000 000 000 000 000 222 044 604 925 031 308 084 726 333 618 164 062 5															
9 007 199 254 740 992	53	0.000 000 000 000 000 111 022 302 462 515 654 042 363 166 809 082 031 25															
18 014 398 509 481 984	54	0.000 000 000 000 000 055 511 151 231 257 827 021 181 583 404 541 015 625															
36 028 797 018 963 968	55	0.000 000 000 000 000 027 755 575 615 628 913 510 590 791 702 270 507 812 5															
72 057 594 037 927 936	56	0.000 000 000 000 000 013 877 787 807 814 456 755 295 395 851 135 253 906 25															
144 115 188 075 855 872	57	0.000 000 000 000 000 006 938 893 903 907 228 377 647 697 925 567 626 953 125															
288 230 376 151 711 744	58	0.000 000 000 000 000 003 469 446 951 953 614 188 823 848 962 783 813 476 562 5															
576 460 752 303 423 488	59	0.000 000 000 000 000 001 734 723 475 976 807 094 411 924 481 391 906 738 281 25															
1 152 921 504 606 846 976	60	0.000 000 000 000 000 000 867 361 737 988 403 547 205 962 240 695 953 369 140 625															











# APPENDIX V

## OCTAL-DECIMAL FRACTION CONVERSION TABLE

OCTAL	DEC.	OCTAL	DEC.	OCTAL	DEC.	OCTAL	DEC.
.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 (Cont'd)

OCTAL	DEC.	OCTAL	DEC.	OCTAL	DEC.	OCTAL	DEC.
.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 (Cont'd)

OCTAL	DEC.	OCTAL	DEC.	OCTAL	DEC.	OCTAL	DEC.
.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

## APPENDIX VI

### EXF AND CHARACTER CODES

1604 EXF CODES

#### SELECT INTERNAL

74 0	000C0	Allow Interrupt on Channel C inactive
	000C1	Disallow Interrupt on Channel C inactive
	01000	Start Real-Time Clock
	02000	Stop Real-Time Clock
	00070	Clear Arithmetic Faults and Clock Overflow
	C0000	Clear All Channel C Selections

C = 1-6

#### SELECT INTERRUPTS

74 0	00100	Allow Interrupt on Internal (Arithmetic) Faults or Clock Overflow
	00101	Disallow Interrupt on Internal (Arithmetic) Faults or Clock Overflow
74 0	04000	Allow Selected Interrupts
	04001	Disallow (Mask) All Interrupts

#### SENSE INTERNAL

74 7	000C0	Exit on Channel C Active
	000C1	Exit on Channel C Inactive

C = 1-6

	001A0	Exit on Arithmetic Fault A
	001A1	Exit on No Arithmetic Fault A

A = 1 : Divide  
2 : Shift  
3 : Overflow  
4 : Exponent Overflow  
5 : Exponent Underflow

CONSOLE EQUIPMENT  
(CHANNEL PAIR 1 and 2)

SELECT

INPUT	74 0	<u>11</u> 140	Select Typewriter for Input, and Interrupt on Carriage Return
		100	Select Typewriter for Input, and No Interrupt on C.R.
		200	Select Paper Tape Reader, and No Interrupt on End of Tape
		210	Select Paper Tape Reader and Set End of Tape Indicator
		220	Select Paper Tape Reader, and Interrupt on End of Tape
OUTPUT	74 0	<u>21</u> 100	Select Typewriter for Output, Assembly Mode
		110	Select Typewriter for Output, Character Mode
		200	Select Paper Tape Punch, Assembly Mode
		210	Select Paper Tape Punch, Character Mode
		240	Turn Paper Tape Punch Motor Off

SENSE

INPUT	74 7	<u>11</u> 200	Exit on Paper Tape Reader, End of Tape
		201	Exit on Paper Tape Reader, No End of Tape
		210	Exit on Paper Tape Reader in Assembly Mode
		211	Exit on Paper Tape Reader in Character Mode
		140	Exit on Typewriter in Lower Case
		141	Exit on Typewriter in Upper Case
		100	Exit on Carriage Return or Tab from Typewriter
		101	Exit on No Carriage Return or Tab from Typewriter
OUTPUT	74 7	<u>21</u> 200	Exit on Paper Tape Punch Out of Tape
		201	Exit on Paper Tape Punch Not Out of Tape



1607 EXF CODES

(CHANNEL C, CABINET 2)\*

SELECT

INPUT	74 0	<u>C20N1</u>	Select Read Tape N, Binary Mode
		0N2	Select Read Tape N, Coded Mode
		001	Read Selected Tape, Binary Mode
		002	Read Selected Tape, Coded Mode
		004	Interrupt When Selected Tape Ready
		005	Rewind Selected Tape
		006	Backspace Selected Tape
		007	Rewind Selected Tape with Interlock
OUTPUT	74 0	<u>C20N1</u>	Select Write Tape N, Binary Mode
		0N2	Select Write Tape N, Coded Mode
		001	Write Selected Tape, Binary Mode
		002	Write Selected Tape, Coded Mode
		003	Write End of File Mark on Selected Tape
		004	Interrupt When Selected Tape Ready
		005	Rewind Selected Tape
		007	Rewind Selected Tape with Interlock

SENSE

INPUT	74 7	<u>C2000</u>	Exit on Ready to Read
		001	Exit on Not Ready to Read
		002	Exit on Read Parity Error
		003	Exit on No Read Parity Error
		004	Exit on Read Length Error
		005	Exit on No Read Length Error
		006	Exit on End of File Mark
		007	Exit on No End of File Mark

---

\* The equipment designator (fourth octal digit from the right) in 1607 EXF codes may be either 2 or 3. A switch in the rear of the 1607 cabinet determines which number will be recognized as the designator for that cabinet.

OUTPUT	74 7	<u>C2000</u>	Exit on Ready to Write
		001	Exit on Not Ready to Write
		002	Exit on Write Reply Parity Error
		003	Exit on No Write Reply Parity Error
		004	Exit on Write Reply Length Error
		005	Exit on No Write Reply Length Error
		006	Exit on End of Tape Marker
		007	Exit on No End of Tape Marker

1608 EXF CODES

(CHANNEL C)

C = 1-6

SELECT

INPUT	74 0	<u>C77N1</u>	Select Read Tape N, Binary Mode
		7N2	Select Read Tape N, Coded Mode
		001	Read Selected Tape, Binary Mode
		002	Read Selected Tape, Coded Mode
		004	Interrupt When Selected Tape Ready
		005	Rewind Selected Tape
		006	Backspace Selected Tape
		007	Rewind Selected Tape with Interlock
		101	Turn Off "Tape Indicator" on Read Unit
		102	Set Low Density on Read Unit
		103	Set High Density on Read Unit
		104	Search File Mark Forward on Read Unit
		105	Search File Mark Backward on Read Unit
		106	Remove Interrupt Selection on Read Unit
OUTPUT	74 0	<u>C77N1</u>	Select Write Tape N, Binary Mode
		7N2	Select Write Tape N, Coded Mode
		001	Write Selected Tape, Binary Mode
		002	Write Selected Tape, Coded Mode
		003	Write End of File Mark on Selected Tape

OUTPUT	74 0	<u>C7004</u>	Interrupt When Selected Tape Ready
		005	Rewind Selected Tape
		006	Backspace Selected Tape
		007	Rewind Selected Tape with Interlock
		101	Turn Off "Tape Indicator" on Write Unit
		102	Set Low Density on Write Unit
		103	Set High Density on Write Unit
		104	Skip Bad Spot on Selected Write Unit
		106	Remove Interrupt on Write Unit

SENSE

INPUT	74 7	<u>C7000</u>	Exit on Ready to Read
		001	Exit on Not Ready to Read
		002	Exit on Read Parity Error
		003	Exit on No Read Parity Error
		004	Exit on Read Length Error
		005	Exit on No Read Length Error
		006	Exit on End of File Mark
		007	Exit on No End of File Mark
		106	Exit When Read Unit is Rewinding or at Load Point
		107	Exit When Read Unit is Not Rewinding or is at Load Point

OUTPUT	74 7	<u>C7000</u>	Exit on Ready to Write
		001	Exit on Not Ready to Write
		002	Exit on Write Reply Parity Error
		003	Exit on No Write Reply Parity Error
		004	Exit on Write Reply Length Error
		005	Exit on No Write Reply Length Error
		006	Exit on End of Tape Marker
		007	Exit on No End of Tape Marker
		106	Exit when Write Unit is Rewinding or at Load Point
		107	Exit when Write Unit is Not Rewinding or is at Load Point

1610 EXF CODES

(CHANNEL C)

C = 1-6

SELECT

INPUT	74 0	<u>C4001</u>	Select Primary Read Station
		002	Select Secondary Read Station
		003	Select Primary and Secondary Read Stations
		005	Select Primary Read Station and Interrupt
		006	Select Secondary Read Station and Interrupt
		007	Select Primary and Secondary Read Stations and Interrupt

OUTPUT	74 0	<u>C4001</u>	Select Printer
		002	Select Punch
		005	Select Printer and Interrupt
		006	Select Punch and Interrupt

SENSE

INPUT	74 7	<u>C4002</u>	Exit on Reader Ready
		003	Exit on Reader Not Ready
		004	Exit on 1604 Selected
		005	Exit on 1604 Not Selected

OUTPUT	74 7	<u>C4002</u>	Exit on Printer Ready
		003	Exit on Printer Not Ready
		004	Exit on Punch Ready
		005	Exit on Punch Not Ready
		010	Exit on 1604 Selected
		011	Exit on 1604 Not Selected

1612 EXF CODES

(CHANNEL C)

C = 1-6

SELECT

OUTPUT (ONLY)	74 0	<u>C6000</u>	Select Printer
		001	Single Space the Printer
		002	Double Space the Printer
		003	Select Format Channel 7
		004	Select Format Channel 8
		010	Clear Monitor Channels 1 - 6
		01N	Select Monitor Channel N : N = 1 - 6

SENSE

OUTPUT (ONLY)	74 7	<u>C6000</u>	Exit on Printer Ready
		001	Exit on Printer Not Ready

1615 FUNCTION CODES  
(CHANNEL C, CABINET 2)\*  
(N =  $1_8$  -  $10_8$ )

OUTPUT

74 0 C	20N1	Select Tape <u>N</u> To Write Binary
	20N2	Select Tape <u>N</u> To Write Coded
	2001	Prepare Selected Tape To Write Binary
	2002	Prepare Selected Tape To Write Coded
	2003	Write End-Of-File Mark On Selected Tape
	2004	Select Interrupt When Write Tape Next Ready
	2005	Rewind Selected Write Tape
	2006	Backspace Selected Write Tape
	2007	Rewind-Unload Selected Write Tape
	2400	Clear Interrupt Selections On Write Tape
	2401	Set Low Density On Selected Write Tape
	2402	Set High Density On Selected Write Tape
	2403	Skip Bad Spot On Selected Write Tape
	2404	Select Interrupt On Next Error

---

\* The equipment designator (fourth octal digit from the right) in 1615 EXF codes may be either 2 or 3. A switch in the 1615 cabinet determines which number will be recognized as the designator for that cabinet.

SENSE

74 7 C      2000      Exit On Ready To Write  
              2001      Exit On Not Ready To Write  
              2002      Exit On Write Reply Parity Error  
              2003      Exit On No Write Reply Parity Error  
              2004      Exit On Write Reply Length Error  
              2005      Exit On No Write Reply Length Error  
              2006      Exit On End Of Tape Marker  
              2007      Exit On Not End Of Tape Marker  
              2400      Exit On Ready To Select  
              2401      Exit On Not Ready To Select  
              2402      Exit On Load Point  
              2403      Exit On Not Load Point  
              2404      Exit On Interrupt On Write Tape  
              2405      Exit On No Interrupt On Write Tape  
              2406      Exit On Write Program Error  
              2407      Exit On No Write Program Error

INPUT

74 0 C      20N1      Select Tape N To Read Binary One Record  
              20N2      Select Tape N To Read Coded One Record  
              22N1      Select Tape N To Read Binary One File  
              22N2      Select Tape N To Read Coded One File  
              2001      Prepare Selected Tape To Read Binary One Record  
              2002      Prepare Selected Tape To Read Coded One Record  
              2201      Prepare Selected Tape To Read Binary One File  
              2202      Prepare Selected Tape To Read Coded One File  
              2003      Move Selected Read Tape Forward One Record  
              2203      Search File Mark Forward  
              2004      Select Interrupt When Read Tape Next Ready  
              2005      Rewind Selected Read Tape  
              2006      Backspace Selected Read Tape  
              2206      Search File Mark Backward  
              2007      Rewind-Unload Selected Read Tape  
              2400      Clear Interrupt Selections On Read Tape  
              2401      Set Low Density On Selected Read Tape  
              2402      Set High Density On Selected Read Tape  
              2404      Select Interrupt On Next Error

SENSE

74 7 C	2000	Exit On Ready To Read
	2001	Exit On Not Ready To Read
	2002	Exit On Read Parity Error
	2003	Exit On No Read Parity Error
	2004	Exit On Read Length Error
	2005	Exit On No Read Length Error
	2006	Exit On End Of Tape Marker
	2007	Exit On Not End Of Tape Marker
	2400	Exit On Ready To Select
	2401	Exit On Not Ready To Select
	2402	Exit On Load Point
	2403	Exit On Not Load Point
	2404	Exit On Interrupt On Read Tape
	2405	Exit On No Interrupt On Read Tape
	2406	Exit On Read Program Error
	2407	Exit On No Read Program Error

SATELLITE EXTERNAL FUNCTION CODES

1604 EXTERNAL FUNCTION CODES

OUTPUT SELECT

74 0 C	2500	Release Direct Selections
	2501	Select Write Control For 160
	2502	Release Write Control To 1604
	2503	Select Direct 1604 To 160
	2504	Select Action Request
	2520	Clear Communication Flag 2
	2540	Set Communication Flag 1
	2560	Clear Communication Flag 1

OUTPUT SENSE

74 7 C	2500	Exit On Write Control Available
	2501	Exit On Write Control Not Available
	2520	Exit On Communications Flag 2 Set
	2521	Exit On Communications Flag 2 Not Set
	2560	Exit On Communications Flag 1 Set
	2561	Exit On Communications Flag 1 Not Set

INPUT SELECT

74 0 C	2501	Select Read Control For 160
	2520	Release Read Control To 1604
	2503	Select Direct 160 To 1604
	2505	Release Interrupt

INPUT SENSE

74 7 C	2500	Exit On Read Control Available
	2501	Exit On Read Control Not Available
	2504	Exit On 160 Interrupt
	2505	Exit On No 160 Interrupt

160 EXTERNAL FUNCTION CODES

WRITE SELECT

6050	Release Action Request
6051	Set Communications Flag 2
6052	Release Write Control To 1604
6055	Clear Communications Flag 1
6056	Clear Communications Flag 2

READ SELECT

5051	Set Communications Flag 1
5052	Release Read Control To 1604
5053	Select Interrupt

STATUS RESPONSE\*

4XXX	Read Control Available
2XXX	Write Control Available
1XXX	Direct 160 To 1604
X4XX	Direct 1604 To 160
XXX2	160 Action Request
XXX1	Communications Flag 1 Set

---

\* Bits may be superimposed; e.g., 6XXX means both Read Control and Write Control available.



## APPENDIX VII

### Magnetic Tape BCD Codes

Character	Code (Octal)	Character	Code (Octal)
A	61	2	02
B	62	3	03
C	63	4	04
D	64	5	05
E	65	6	06
F	66	7	07
G	67	8	10
H	70	9	11
I	71	&	60
J	41	-	40
K	42	(blank)	20
L	43	/	21
M	44	. (period)	73
N	45	\$	53
O	46	*	54
P	47	, (comma)	33
Q	50	%	34
R	51	#	13
S	22	@	14
T	23	⌘	74
U	24	0 (numerical zero)	12
V	25	record mark	32
W	26	0 (minus zero)	52
X	27	0 (plus zero)	72
Y	30	group mark	77
Z	31	tape mark	17
0	12		
1	01		

## APPENDIX VIII

### Flexowriter Codes

UC	LC	CODE	UC	LC	CODE
A	a	30	Y	y	25
B	b	23	Z	a	21
C	c	16	o	0	56
D	d	22	1	1	74
E	e	20	2	2	70
F	f	26	3	3	64
G	g	13	4	4	62
H	h	05	5	5	66
I	i	14	6	6	72
J	j	32	7	7	60
K	k	36	8	8	33
L	l	11	9	9	37
M	m	07	-	-	52
N	n	06	/	/	44
O	o	03	(	)	54
P	p	15	+	,	46
Q	q	35	=	.	42
R	r	12	:	;	50
S	s	24	CR		45
T	t	01	Upper Case (UC)		47
			Lower Case (LC)		57
			Back Space (BS)		61
U	u	34	Color Shift (CS)		02
V	v	17	Tabulate (TAB)		51
			Stop		43
W	w	31	Space		04
			Tape Feed		00
X	x	27	Delete		77

- Note:
- 1) Leader - Blank tape, Delete - Deleted character  
Stop - Stop Flexowriter reader,
  - 2) 10, 40, 41, 53, 55, 63, 65, 67, 71, 73, 75, and 76 - illegal

# APPENDIX IX

## Punched Card Codes

Char	Card	BCD	Char	Card	BCD	Char	Card	BCD	Char	Card	BCD
			+	1 <sup>2</sup>	60	---	1 <sup>1</sup>	40			20
1	1	01	A	1 <sup>2</sup> <sub>1</sub>	61	J	1 <sup>1</sup> <sub>1</sub>	41	/	0 <sub>1</sub>	21
2	2	02	B	1 <sup>2</sup> <sub>2</sub>	62	K	1 <sup>1</sup> <sub>2</sub>	42	S	0 <sub>2</sub>	22
3	3	03	C	1 <sup>2</sup> <sub>3</sub>	63	L	1 <sup>1</sup> <sub>3</sub>	43	T	0 <sub>3</sub>	23
4	4	04	D	1 <sup>2</sup> <sub>4</sub>	64	M	1 <sup>1</sup> <sub>4</sub>	44	U	0 <sub>4</sub>	24
5	5	05	E	1 <sup>2</sup> <sub>5</sub>	65	N	1 <sup>1</sup> <sub>5</sub>	45	V	0 <sub>5</sub>	25
6	6	06	F	1 <sup>2</sup> <sub>6</sub>	66	O	1 <sup>1</sup> <sub>6</sub>	46	W	0 <sub>6</sub>	26
7	7	07	G	1 <sup>2</sup> <sub>7</sub>	67	P	1 <sup>1</sup> <sub>7</sub>	47	X	0 <sub>7</sub>	27
8	8	10	H	1 <sup>2</sup> <sub>8</sub>	70	Q	1 <sup>1</sup> <sub>8</sub>	50	Y	0 <sub>8</sub>	30
9	9	11	I	1 <sup>2</sup> <sub>9</sub>	71	R	1 <sup>1</sup> <sub>9</sub>	51	Z	0 <sub>9</sub>	31
0	0	12	†	1 <sup>2</sup> <sub>0</sub>	72	-	1 <sup>1</sup> <sub>0</sub>	52			
=	8,3	13	.	1 <sup>2</sup> <sub>8,3</sub>	73	⌘	1 <sup>1</sup> <sub>8,3</sub>	53	,	0 <sub>8,3</sub>	33
-	8,4	14	)	1 <sup>2</sup> <sub>8,4</sub>	74	*	1 <sup>1</sup> <sub>8,4</sub>	54	(	0 <sub>8,4</sub>	34

## APPENDIX X

### Input/Output Typewriter Codes

CHARACTERS		CODE	CHARACTERS		CODE
UC	LC		UC	LC	
A	a	30	X	x	27
B	b	23	Y	y	25
C	c	16	Z	z	21
D	d	22	)	0	56
E	e	20	*	1	74
F	f	26	@	2	70
G	g	13	#	3	64
H	h	05	\$	4	62
I	i	14	%	5	66
J	j	32	¢	6	72
K	k	36	&	7	60
L	l	11	½	8	33
M	m	07	(	9	37
N	n	06	-	-	52
O	o	03	?	/	44
P	p	15	"	'	54
Q	q	35	°	+	46
R	r	12	.	.	42
S	s	24	:	;	50
T	t	01	,	,	40
U	u	34	÷	=	02
V	v	17	tab	tab	51
W	w	31	space		04
Backspace		61	Carriage Return		45
Lower Case		57	Upper Case		47

# APPENDIX XI

## 1612 Printer Codes

CHAR	CODE	CHAR	CODE	CHAR	CODE	CHAR	CODE
Blank	20	F	66	V	25	≤	15
0	12	G	67	W	26	†	16
1	01	H	70	X	27	□	17
2	02	I	71	Y	30	⊔	32
3	03	J	41	Z	31	→	35
4	04	K	42	.	73	≡	36
5	05	L	43	-	40	~∧	37
6	06	M	44	+	60	% or √	52
7	07	N	45	=	13	\$ or ⊐	53
8	10	O	46	(	34	†	55
9	11	P	47	)	74	‡	56
A	61	Q	50	/	21	>	57
B	62	R	51	*	54	<	72
C	63	S	22	,	33	≥	75
D	64	T	23	:	00	?	76
E	65	U	24	‡	14	;	77

In last column, codes ~ % \$ appear if business application, ∧ √ ⊐ for scientific application.

# **GLOSSARY**

ABSOLUTE ADDRESS	A specific storage location; contrast with relative address.
ACCESS TIME	The time needed to perform a storage reference, either read or write. In effect, the access time of a computer is one storage reference cycle.
ACCUMULATOR	A register with provisions for the addition of another quantity to its content. It is also the name of the A register.
ADDER	A device capable of forming the sum of two or more quantities.
ADDRESS	A 15-bit quantity which identifies a particular storage location.
ALPHABETIC CODING	A system of abbreviation used in preparing information for input into a computer, e.g., Q Right Shift would be QRS.
AND FUNCTION	A logical function in Boolean algebra that is satisfied (has the value "1") only when all of its terms are "1's". For any other combination of values it is not satisfied and its value is "0".
A REGISTER	Principal arithmetic register; operates as a 48-bit subtractive accumulator (modulus: $2^{48}-1$ ).
BASE	A quantity which defines some system of representing numbers by positional notation; radix.
BIT	Binary digit, either "1" or "0".
BLOCK	A group of words transported in and out of storage as a unit.
BOOTSTRAP	The coded instructions at the beginning of an input tape, together with the manually entered instructions.
BORROW	In a subtractive counter or accumulator, a signal indicating that in stage n, a "1" was subtracted from a "0". The signal is sent to stage n+1 which it complements.
BRANCH	A conditional jump.

BREAKPOINT	A point in a routine at which the computer may be stopped by a manual switch for a visual check of progress.
B <sup>1</sup> -B <sup>6</sup> REGISTERS	Index registers used primarily for modification of execution address.
BUFFER	A device in which data is stored temporarily in the course of transmission from one point to another. To store data temporarily. The operation in which either a word from storage is sent to an external equipment via an output channel (output buffer), or a word is sent from an external equipment to storage via an input channel (input buffer).
CAPACITY	The upper and lower limits of the numbers which may be processed in a register or the quantity of information which may be stored in a storage unit. If the capacity of a register is exceeded, an overflow is generated.
CARRY	In an additive counter or accumulator, a signal indicating that in stage n, a "1" was added to a "1". The signal is sent to stage n+1, which it complements.
CHANNEL	A transmission path that connects the computer to an external equipment.
CHARACTER	Two types of information handled by the computer: 1) A group of 6 bits which represents a digit, letter or symbol. In the assembly mode, eight 6-bit characters make up a computer word. 2) A group of 7 bits which represents an item of information. In the character mode, this item is one 7-bit character and "0's" in the remaining (upper) 41 bits.
CLEAR	A command that removes a quantity from a register by placing every stage of the register in the "0" state.
CLOCK PHASE	One of two outputs from the master clock, "even" or "odd".



COMMAND	A signal that performs a unit operation, such as transmitting the content of one register to another, shifting a register one place to the left or setting a FF.
COMPILER	A routine which automatically produces a specific program for a particular problem. The routine determines the meaning of information expressed in a psuedo-code, selects or generates the required subroutine, transforms the subroutine into specific coding, assigns storage registers, and enters the information as an element of the problem program.
COMPLEMENT	Noun: see One's Complement or Two's Complement. Verb: a command which produces the one's complement of a given quantity.
CONTENT	The quantity or word held in a register or storage location.
CORE	A ferromagnetic toroid used as the bistable device for storing a bit in a memory plane.
COUNTER	A register with provisions for increasing or decreasing its content by 1.
EVEN STORAGE	The storage unit which contains the 16,384 even addresses.
EXECUTION ADDRESS	The lower 15 bits of a 24-bit instruction. Most often used to specify the storage address of an operand. Sometimes used as the operand.
EXIT	Initiation of a second control sequence by the first, occurring when the first is near completion; the circuit involved in exiting.
EXTERNAL FUNCTION	1) External Function Select (74.0) sends a code to an external equipment to direct its operation. 2) External Function Sense (74.7) sends a code to an external equipment to sense its operating condition.
FAULT	Operational difficulty which stops operation or sets an indicator.

FIXED POINT	A notation or system of arithmetic in which all numerical quantities are expressed by a predetermined number of digits with the binary point implicitly located at some predetermined position; contrasted with floating point.
FLIP-FLOP (FF)	A bistable storage device. A "1" input to the set side puts the FF in the "1" state; a "1" input to the clear side puts the FF in the "0" state. The FF remains in a state indicative of its last "1" input. A stage of a register consists of a FF.
FLOATING POINT	A means of expressing a number X by a pair of numbers, Y and Z, such that $X = Yn^Z$ . Z is an integer, called the exponent or characteristic; n is a base, usually 2 or 10; and Y is called the fraction or mantissa.
FUNCTION CODE	The upper 9 bits of a 24-bit instruction consisting of the operation and index codes.
INDEX CODE	A 3-bit quantity, bits 15, 16, and 17 of an instruction; usually specifies an index register whose contents are added to the execution address; sometimes specifies the conditions for executing the instruction.
INSTRUCTION	A 24-bit quantity consisting of a function code, execution address, and index designator.
INTERRUPT REQUEST	A signal received from an external equipment that may cause a special sequence of instructions to be executed.
INVERTER	A circuit which provides as an output a signal that is opposite to its input. An inverter output is "1" only if all the separate OR inputs are "0".
JUMP	An instruction which alters the normal sequence control of the computer and, conditionally or unconditionally, specifies the location of the next instruction.
LOAD	To place a quantity from storage in a register.

LOCATION	A storage position holding one computer word, usually designated by a specific address.
LOGICAL PRODUCT	In Boolean algebra, the AND function of several terms. The product is "1" only when all the terms are "1"; otherwise it is "0". Sometimes referred to as the result of "bit-by-bit" multiplication.
LOGICAL SUM	In Boolean algebra, the OR function of several terms. The sum is "1" when any or all of the terms are "1"; it is "0" only when all are "0".
LOOP	Repetition of a group of instructions in a routine.
LOWER ADDRESS	The execution address portion of a lower instruction; bits 0 through 14 of a 48-bit register or storage location.
LOWER INSTRUCTION	See Program Step.
MASK	In some instructions, one quantity may determine what part of the other quantity is to be considered. If the first quantity, the mask, contains a "1", the corresponding bit of the second quantity is considered.
MASTER CLOCK	The source of standard signals required for sequencing computer operation. The clock determines the basic frequency of the computer.
MASTER CLEAR (MC)	A general command produced by placing the CLEAR switch up (external MC) or down (computer MC) which clears all the crucial registers and control FFs.
MNEMONIC CODE	A three-letter code which represents the function or purpose of an instruction. Also called Alphabetic Code.
MODULUS	An integer which describes certain arithmetic characteristics of registers, especially counters and accumulators, within a digital computer. The modulus of a device is defined by $r^n$ for an open-ended device and $r^n-1$ for a closed (end-around) device, where $r$ is

the base of the number system used and  $n$  is the number of digit positions (stages) in the device. Generally, devices with modulus  $r^n$  use two's complement arithmetic; devices with modulus  $r^n - 1$  use one's complement.

NORMALIZE	To adjust the exponent and mantissa of a floating-point result so that the mantissa lies in the prescribed standard (normal) range.
NORMAL JUMP	An instruction that jumps from one sequence of instructions to a second, and makes no preparation for returning to the first sequence.
NUMERIC CODING	A system of abbreviation in which all information is reduced to numerical quantities.
ODD STORAGE	The storage unit which contains the 16,384 odd addresses.
ONE'S COMPLEMENT	With reference to a binary number, that number which results from subtracting each bit of the given number from "1". The one's complement of a number is formed by complementing each bit of it individually, that is, changing a "1" to "0" and a "0" to a "1". A negative number is expressed by the one's complement of the corresponding positive number.
ON-LINE OPERATION	A type of system application in which the input data to the system is fed directly from the external equipment to the computer.
OPERAND	Usually refers to the quantity specified by the execution address. This quantity is operated upon in the execution of the instruction.
OPERATIONAL REGISTERS	Registers which are displayed on the operator's console ( $B^1 - B^6$ , A, Q, P, $U^1$ ).
OPERATION CODE	The upper 6 bits of a 24-bit instruction which identify the instruction. After the code is translated, it conditions the computer for execution of the specified instruction. This code, which is expressed by two octal digits, is designated by the letter f.
$O^1 - O^4$ REGISTERS	Output registers $O^1, 2, 3$ are used for output buffer operations; $O^4$ handles all high-speed output transfer operations.

OR FUNCTION	A logical function in Boolean algebra that is satisfied (has the value "1") when any of its terms are "1". It is <u>not</u> satisfied when all terms are "0". Often called the 'inclusive' OR function.
OVERFLOW	The capacity of a register is exceeded.
PARITY CHECK	A summation check in which the binary digits in a character are added and the sum checked against a previously computed parity digit; i. e., a check which tests whether the number of ones is odd or even.
PARTIAL ADD	An addition without carries. Accomplished by toggling each bit of the augend where the corresponding bit of the addend is a "1".
P REGISTER	The Program Address Counter is a two's complement additive register (Modulus $2^{15}$ ) which generates in sequential order the storage addresses containing the individual program steps.
PROGRAM	A precise sequence of instructions that accomplishes a computer routine; a plan for the solution of a problem.
PROGRAM STEP	Two 24-bit instructions contained in one 48-bit storage address; the higher-order 24 bits are the upper instruction, lower-order 24 bits, the lower instruction. A pair of instructions is read from storage, and the upper instruction is executed first. The lower one is then executed, except when the upper one provides for skipping the lower one.
Q REGISTER	Auxiliary arithmetic register which assists the A register in the more complicated arithmetic operations (Modulus: $2^{48}-1$ ).
RANDOM ACCESS	Access to storage under conditions in which the next position from which information is to be obtained is in no way dependent on the previous one.
R REGISTER	Address Buffer register. Two's complement subtractive register (Modulus: $2^{15}$ ) which acts as an exchange register for transmissions involving index registers.

READ	To remove a quantity from a storage location.
READY	<ol style="list-style-type: none"> <li>1) The input-output control signal sent by the computer or an external equipment. The ready signal indicates that a word or character is available for transmission.</li> <li>2) A status reponse indicating that the external device being addressed is ready for operation.</li> </ol>
RELATIVE ADDRESS	Identifies a word in a subroutine or routine with respect to its position. Relative addresses are translated into absolute addresses by the addition of some specific reference address, usually that at which the first word of the routine is stored.
REPLACE	In the title of an instruction, the result of the execution of the instruction is stored in the location from which the initial operand was obtained.
RESUME	The input-output control signal sent by either the computer or an external equipment to indicate that it is prepared to receive another word (48 bits) or character (usually 6 bits). The resume signal is thus a request for data.
RETURN JUMP	An instruction that jumps from a sequence of instructions to initiate a second sequence and prepares for continuing the first sequence after the second is completed.
ROUTINE	The sequence of operations which the computer performs under the direction of a program.
S <sup>1</sup> REGISTER	Storage Address register (even storage). Selects the storage address specified by the contents of the P register.
S <sup>2</sup> REGISTER	Storage Address register (odd storage). Selects the storage address specified by the contents of the P register.
SCALE FACTOR	One or more coefficients by which quantities are multiplied or divided so that they lie in a given range of magnitude.

SCANNER	That portion of the computer which automatically samples the state of the buffer channels, interrupt line, and real time clock and initiates action in accordance with the information obtained.
SECONDARY REGISTERS	Transient registers not displayed on the console ( $U^2$ , $S^{1,2}$ , $Z^{1,2}$ , $R$ , $X$ , $O^1-O^6$ ).
SHIFT	To move the bits of a quantity right or left.
SIGN BIT	In registers where a quantity is treated as signed by use of one's complement notation, the bit in the highest-order stage of the register. If the bit is "1", the quantity is negative; if the bit is "0", the quantity is positive.
SIGN EXTENSION	The duplication of the sign bit in the higher-order stages of a register.
SKIP	To omit the execution of a lower instruction in a program; occurs only if the upper instruction provides for skipping on a specified condition, and the condition is met.
STAGE	The FF's and inverters associated with a bit position of a register.
STORE	To transmit information to a device from which the unaltered information can later be obtained.
SUBINSTRUCTION	The index code specifies one of eight forms of the instruction indicated by the operation code. Such forms are called "sub-instructions". Thus, 74.0 is a subinstruction of instruction 74.
TOGGLE	To complement each bit of a quantity as a result of an individual condition.
TRANSFER	High-speed data input-output transmission under direct program control.

TRANSMISSION, FORCED	A transfer of bits into a register which has not been cleared previously.
TWO'S COMPLEMENT	Number that results from subtracting each bit of a number from "0". The two's complement may be formed by complementing each bit of the given number and then adding one to the result, performing the required carries.
U <sup>1</sup> REGISTER	Program Control register. Holds a program step while the two instructions contained in it are executed.
U <sup>2</sup> REGISTER	Auxiliary Program Control register. A 15-bit subtractive accumulator (Modulus: $2^{15}-1$ ) used primarily for modification of the base execution address.
UPPER ADDRESS	The execution address portion of an upper instruction; bit positions 24 through 38 of a 48-bit register or storage address.
UPPER INSTRUCTION	See Program Step
WORD	A unit of information which has been coded for use in the computer as a series of bits. The normal word length is 48 bits.
WRITE	To enter a quantity into a storage location.
X REGISTER	Exchange register. All internal transmissions between the arithmetic section or the input-output section and the rest of the computer are made through X.
Z <sup>1</sup> REGISTER	Storage Restoration register (even storage). Holds the word to be written into a given storage location.
Z <sup>2</sup> REGISTER	Storage Restoration register (odd storage). Holds the word to be written into a given storage location.



## COMMENT SHEET

MANUAL TITLE CONTROL DATA 1604-C COMPUTER

Reference Manual

PUBLICATION NO. 60016700 REVISION F

**FROM:** NAME: \_\_\_\_\_

BUSINESS  
ADDRESS: \_\_\_\_\_

### COMMENTS:

This form is not intended to be used as an order blank. Your evaluation of this manual will be welcomed by Control Data Corporation. Any errors, suggested additions or deletions, or general comments may be made below. Please include page number references and fill in publication revision level as shown by the last entry on the Record of Revision page at the front of the manual. Customer engineers are urged to use the TAR.

CUT ALONG LINE

PRINTED IN U.S.A.

AA3419 REV. 11/69

NO POSTAGE STAMP NECESSARY IF MAILED IN U. S. A.

FOLD ON DOTTED LINES AND STAPLE

STAPLE

STAPLE

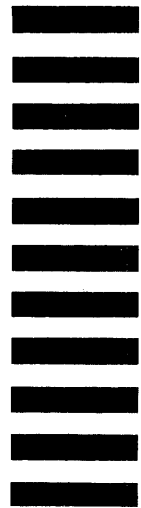
FOLD

FOLD

FIRST CLASS  
PERMIT NO. 8241  
MINNEAPOLIS, MINN.

**BUSINESS REPLY MAIL**  
NO POSTAGE STAMP NECESSARY IF MAILED IN U.S.A.

POSTAGE WILL BE PAID BY  
**CONTROL DATA CORPORATION**  
Technical Publications Department  
4201 North Lexington Avenue  
Arden Hills, Minnesota 55112



CUT ALONG LINE

FOLD

FOLD

## 1604 INSTRUCTIONS

		Page				Page	
ADD	Add	14	2 - 18	MUF	Multiply Fractional	26	2 - 21
ADL	Add Logical	45	2 - 36	MUI	Multiply Integer	24	2 - 19
AJP	A Jump	22	2 - 28, 31	OUT	Output Transfer	63	2 - 41
ALS	A Left Shift	05	2 - 14	QJP	Q Jump	23	2 - 29, 32
ARS	A Right Shift	01	2 - 14	QLS	Q Left Shift	06	2 - 15
DVF	Divide Fractional	27	2 - 21	QRS	Q Right Shift	02	2 - 14
DVI	Divide Integer	25	2 - 20	RAD	Replace Add	70	2 - 39
ENA	Enter A	10	2 - 26	RAO	Replace Add One	72	2 - 39
ENI	Enter Index	50	2 - 27	RSB	Replace Subtract	71	2 - 39
ENQ	Enter Q	04	2 - 26	RSO	Replace Subtract One	73	2 - 40
EQS	Equality Search	64	2 - 37	SAL	Substitute Address, L	61	2 - 16
EXF	External Function	74	3 - 3	SAU	Substitute Address, U	60	2 - 16
FAD	Floating Add	30	2 - 21	SBL	Subtract Logical	46	2 - 36
FDV	Floating Divide	33	2 - 24	SCA	Scale A	34	2 - 25
FMU	Floating Multiply	32	2 - 23	SCL	Selective Clear	41	2 - 35
FSB	Floating Subtract	31	2 - 22	SCM	Selective Complement	42	2 - 34
IJP	Index Jump	55	2 - 17	SCQ	Scale AQ	35	2 - 25
INA	Increase A	11	2 - 26	SEV	(not used)	77	
INI	Increase Index	51	2 - 27	SIL	Store Index, L	57	2 - 13
INT	Input Transfer	62	2 - 41	SIU	Store Index, U	56	2 - 13
ISK	Index Skip	54	2 - 17	SLJ	Selective Jump	75	2 - 30, 32
LAC	Load A, Complement	13	2 - 11	SLS	Selective Stop	76	2 - 30, 32
LDA	Load A	12	2 - 11	SSH	Storage Shift	37	2 - 33
LDL	Load Logical	44	2 - 36	SSK	Storage Skip	36	2 - 33
LDQ	Load Q	16	2 - 11	SST	Selective Set	40	2 - 34
LIL	Load Index, L	53	2 - 13	SSU	Selective Substitute	43	2 - 36
LIU	Load Index, U	52	2 - 27	STA	Store A	20	2 - 12
LLS	AQ Left Shift	07	2 - 15	STL	Store Logical	47	2 - 36
LQC	Load Q, Complement	17	2 - 11	STQ	Store Q	21	2 - 12
LRS	AQ Right Shift	03	2 - 14	SUB	Subtract	15	2 - 18
MEQ	Masked Equality	66	2 - 38	THS	Threshold Search	65	2 - 37
MTH	Masked Threshold	67	2 - 38	ZRO	(not used)	00	



>>> CUT OUT FOR USE AS LOOSE-LEAF BINDER TITLE TAB

**CONTROL DATA**  
CORPORATION

8100 34th AVE. SO., MINNEAPOLIS, MINN. 55440

LITHO IN U.S.A.