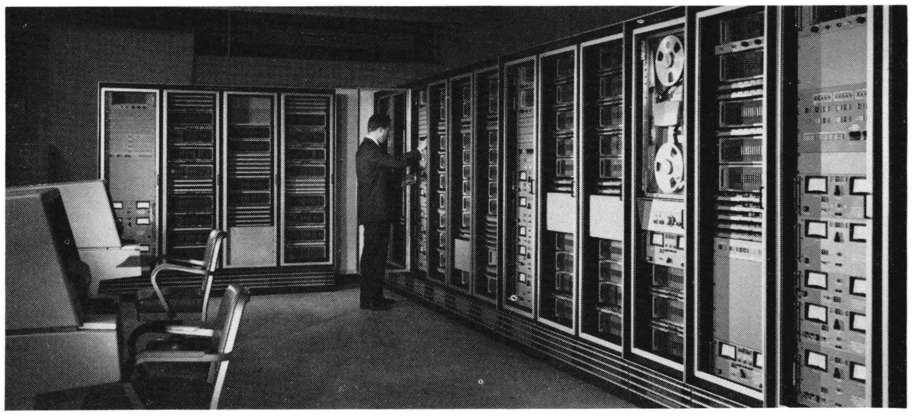
0	010101	110110	0	101110	101110
0	101011	100011	0	110001	101000
0	100011	100110	0	100100	001110
0	110001	100000	0	101100	101110
0	010101	110110	0	101110	101110
0	101011	100011	0	110001	101000
0	100011	100110	0	100100	001110
0	110001	100000	0	101100	101110
0	010101	110110	0	101110	101110
0	101011	100011	0	110001	101000
0	100011	100110	0	100100	001110
0	110001	100000	0	101100	101110
0	010101	110110	0	101110	101110
0	101011	100011	0	110001	101000
0	100011	100110	0	100100	001110
0	100001	100100	0	110001	110011
0	100111	100100	0	101011	100101
0	101000	101101	0	110010	110011
0	100100	101000	0	101101	001111
0	110001	100000	0	101100	101110
0	010101	110110	0	101110	101110
0	101011	100011	0	110001	101000
0	100011	100110	0	100100	001110

PROGRAMMING MANUAL RW-400 AN/FSQ-27

RAMO-WOOLDRIDGE, A DIVISION OF THOMPSON RAMO WOOLDRIDGE INC.
8433 FALLBROOK AVENUE, CANOGA PARK, CALIFORNIA



PROGRAMMING MANUAL AN/FSQ-27 (RW-400)

SECOND EDITION
FEBRUARY 1, 1961

PREPARED BY BERT HELFINSTEIN

This document was prepared in support of work being performed under Air Force Contract AF30 (602)—1814 with Rome Air Development Center.

DATA SYSTEMS PROJECT OFFICE

RAMO-WOOLDRIDGE
a division of
THOMPSON RAMO WOOLDRIDGE INC.
8433 FALLBROOK AVENUE
CANOGA PARK, CALIFORNIA

TABLE OF CONTENTS

Chapter I
INTRODUCTION

A. GENERAL

This manual has been designed to provide the programmer of the RW-400 with the basic facts of machine performance necessary to write programs. It is divided into twenty chapters: one on each of the modules of the system, one on intermodule communications, and this introductory chapter.

The chapter on each of the modules includes a discussion of the basic operation of the module, of the response of the module to various commands it may receive, and of the programming limitations and exceptions that apply to the module.

The chapter on intermodule communications (Chapter III) discusses the use of instructions by the Computer Module and the Buffer Module—the *controlling* modules of the system—in directing the activities of all the other modules. It gives the rules for getting the *controlling* modules connected to any of the *controlled* modules and for transferring data between *controlled* and *controlling* modules.

This chapter discusses the unique modular concept used in the RW-400 Data Processing Central, and gives a brief description of each module.

B. MODULAR CONCEPT

1. GENERAL

The RW-400 Data Processing Central is made up of a group of individual modules connected to each other through a central switching device. This technique allows the number and types of modules (including computers) connected to the system to be altered to fit the problem.

2. INTERCONNECTIONS

Each module in the Data Processing Central is connected to the Central Exchange (Interim Exchange in earlier systems) with a multiconductor cable. Fourteen of the conductors are used to carry input half-words plus parity (the system uses a 26-bit word, transferred as two 13-bit half-words with a parity bit associated with each half-word) and 14 are used to carry output half-words plus parity. Six lines are used for control signals—three input and three output. (These signals are machine generated and are not under program control but they may be tested by the program with a *Test Jump* instruction.) One line is used to carry incoming clock pulses and one is used to carry outgoing clock pulses. Several lines are used for carrying alert signals from *controlled* modules to *controlling* modules. The remaining lines are used for various purposes of little interest to the programmer.

3. STATUS

Each of the modules in the system can transmit a status half-word on its output data lines. These 13 bits are used to indicate to the *controlling* module the state of various conditions within the *controlled* module, such as whether the *controlled* module has detected any parity errors or clock failures.

4. CLOCKS

Each of the modules in the system contains one or more internal clock pulse generators. When a module is transferring data, it must accompany this data with clock pulses to synchronize the reception of the data at the receiving module. In addition, many of the modules have internal clocks. These generate pulses which are used for synchronizing the module's microprogram or are recorded on drum or tape for data synchronization.

5. INTERRUPT CAPABILITY

The interrupt capability of the system permits automatic interruption of a program, at the option of the program, when the computer receives an alert telling it that a condition requiring attention has arisen. These alerts cause one of 20 possible flip-flops to go into the *one* state. (See Figure 1.1 for a table of the various alert conditions and flip-flops which they affect.) These 20 flip-flops are called the I register for convenience, although they are not really a register. Any of these I-register flip-flops can be tested with a *Test Jump* instruction. However, if the programmer desires, he can cause one or more of these alerts to interrupt the program automatically by making the corresponding bit in the S register a one. For example, if the programmer wants the program interrupted when the Accumulator goes negative, he would make bit 14 of the S register (corresponding to bit 14 of the I register) a one. Then if the Accumulator becomes negative, the program will be interrupted.

Bit 20 of the S register serves as a master alert bit, so that whenever any interrupts are desired, bit 20 must be one as well as the bits corresponding to the conditions that are to cause interrupts.

Bit Number	Condition Causing Alert
I_{20}	Control Panel Alert Button has been operated.
I_{19}	1. Link Jump, Tally Jump, Compare Jump, Test Jump (other than test the control panel test light), or Character Transfer instruction has entered the X register and bits 19 and 20 of the S register are both one. The occurrence of this alert also turns on the control panel test light. 2. The control panel test light has been turned on by a *Command Output* instruction.
I_{18}	If a controlled module is connected to a Computer, is transmitting status, and bit 13 of that status transmission is one, I_{18} will be set to one. I_{18} will be zero at all other times.
I_{17}	The Program Error Indicator. This is set to one by: 1. A Command Output instruction which is not accepted and which does not contain a wait bit (a one in bit 20). 2. The execution of an illegal instruction code. 3. An attempt to use a Buffer operand when a Buffer is not connected or is not ready. 4. An attempt to communicate through the Central (or Interim) exchange when not connected to any module.
I_{16}	Input or Output Parity Error.
I_{15}	The Accumulator has overflowed.
I_{14}	The Accumulator is negative.
I_{13}-I_{12}	External Alerts—not yet assigned.
I_{11}	The Buffer connected to the cable whose IX connection address is 0001_2 is not in self-instruction mode.
I_{10}	The Buffer connected to the cable whose IX connection address is 0010_2 is not in self-instruction mode.
I_9-I_1	External Alerts—not yet assigned.

Note: All unassigned I lines are always one.

Figure 1.1. Alert Conditions and Associated Alert Bits

When an interrupt does occur during some instruction, control is transferred to memory location zero at the completion of the instruction. (Even if the instruction is a long data transfer, the interrupt will not occur until the transfer has been completed.) If the instruction in location zero is not a jump instruction, after its completion the P register is set to one.

Note: If an interrupt occurs during the execution of an **Insert S** instruction, the program will not be interrupted until after the instruction following the **Insert S.**

C. DESCRIPTION OF MODULES

(See Figures 1.2 through 1.4)

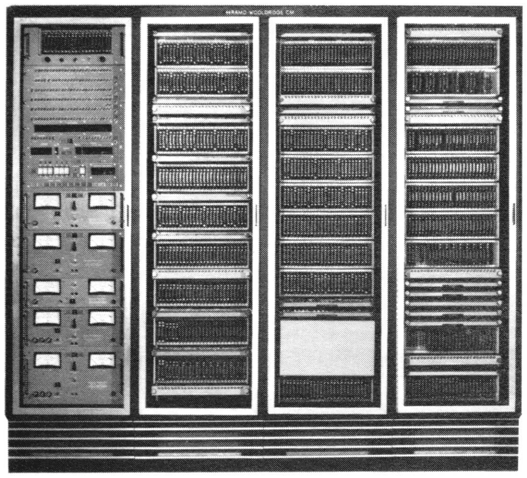
1. COMPUTER MODULE

The Computer Module is a high-speed, general-purpose, two-address, fixed-point, twos-complement arithmetic, digital computer with a 1024-word random-access magnetic core memory. It uses a 26-bit word plus two parity bits, each associated with a 13-bit half-word. There are 33 internal instructions available for arithmetic and logical operations and five external instructions available for controlling connected modules.

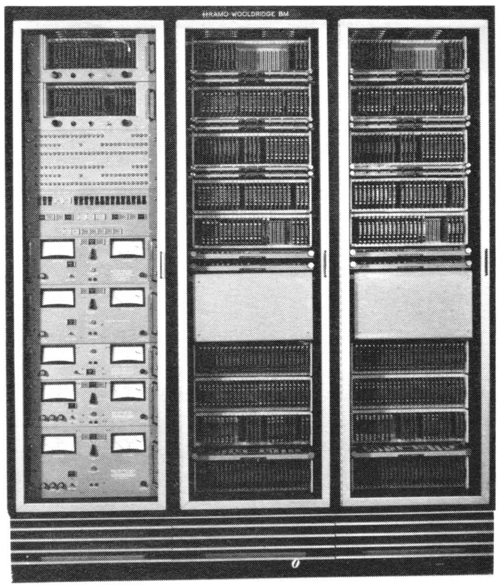
The Computer can direct the Central (or Interim) Exchange to connect it to any of the other modules in the system. It can then command the connected module to transmit or to accept data, or it can test the status of the connected module. Details of Computer programming are discussed in Chapters II and III.

2. BUFFER MODULE

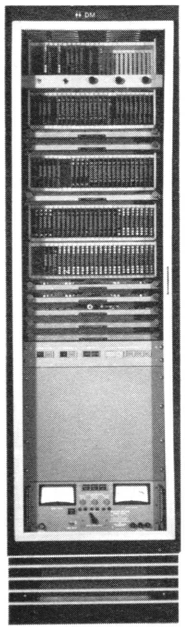
The Buffer Module is a high-speed, magnetic core buffer designed to operate in conjunction with the Computer. Each Buffer Module contains two independent 1024-word core memory Buffers. Either



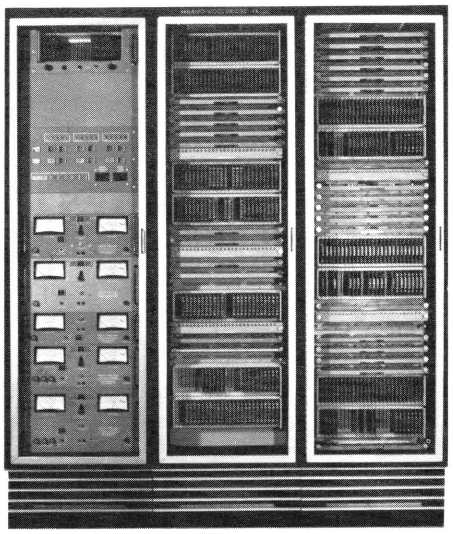
Computer Module



Buffer Module



Drum Module



Interim Exchange

Figure 1.2. RW-400 Modules

one can serve as a direct adjunct to the Computer core memory through a technique of indirect addressing described in Chapter II. Each Buffer has a limited repertoire of self-instructions which enables it to make tape and drum searches for a Computer and then rapidly to transfer the data to the Computer core memory, and to take data from the Computer core memory and transfer it to comparatively slow external devices. Details of Buffer programming are discussed in Chapter IV.

3. CENTRAL EXCHANGE

The Central Exchange is the central switching device for the RW-400. It is a 16-by-61 switch; that is it can accommodate up to 16 *controlling* modules and 61 *controlled* modules. It has its own magnetic core memory which allows the programs to use symbolic addressing and it provides for a Master Computer to be able to force disconnect other modules in the system. Switching is done at electronic speeds. The Central Exchange is discussed in Chapter V.

4. INTERIM EXCHANGE

The Interim Exchange is a 3-by-15 switch which is used in early systems. It can accommodate three *controlling* modules and up to 15 *controlled* modules. Any of the *controlling* modules can request a connection to any of the *controlled* modules with a *Command Output* instruction; if the requested module is available it can become connected to it at electronic speed.

A function called the Master Disconnect is included as part of the Interim Exchange. It provides any one of the three *controlling* modules with the capability of forcing either of the other *controlling* modules to disconnect from the module to which it is currently connected. Interim Exchange programming is discussed in Chapter VI.

5. DRUM MODULE

The Drum Module is designed to provide a moderate volume, relatively short access time memory for the system. It can store 8192 words. The module can be searched either for a particular word or for an address on the drum. Either way, the average access time to the first word of a block is 8-1/2 milliseconds and the transfer rate is then 60,000 words per second. Drum Module programming is discussed in Chapter VII.

6. DRUM MODULE B

Drum Module B is like the normal Drum Module in most respects. The principal difference is that Drum Module B has a much greater storage capacity (79,872 words) and has a different initial access time (13 milliseconds) and transfer rate (50,000 words per second). Drum Module B programming is discussed in Chapter VIII.

7. PERIPHERAL BUFFER

The Peripheral Buffer is designed to provide buffering between the high-speed Computer and Buffer Modules and the slow speed input-output devices such as Flexowriters, Plotters, and consoles with human operators. As many as 32 separate input-output devices can be connected to a Peripheral Buffer, which will store up to 256 six-bit characters from each of these devices.

The Peripheral Buffer will accept data from these devices at any rate up to 25 characters per second. It will transfer the data to a Computer or Buffer at a rate of 60,000 words per second with an 8-1/2 millisecond average access time to the first word. Peripheral Buffer programming is discussed in Chapter IX.

8. DISPLAY BUFFER

The Display Buffer provides a source of regenerative digital data for generating dot, line, symbol, or alphanumeric displays. The Display Buffer can service as many as eight independent displays simultaneously, and a Computer or Buffer Module can change a display at any time without affecting the other seven. A Computer or Buffer can transfer data to the Display Buffer at a 15,000-word-per-second rate. Display Buffer programming is discussed in Chapter X.

9. TAPE MODULE

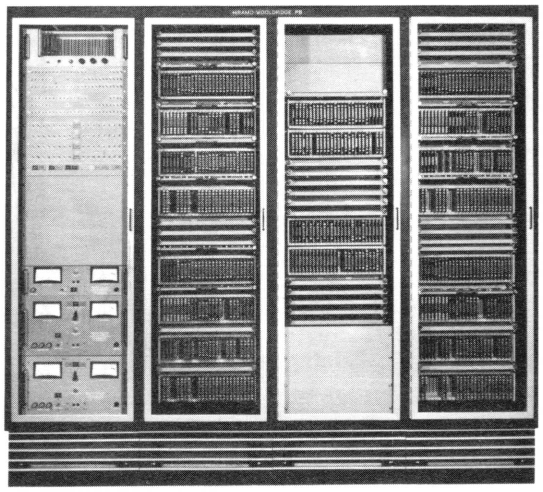
The Tape Module provides large-volume storage for the system. The tape transport is an Ampex FR-300 with a 90-inch-per-second normal tape speed and a 150-inch-per-second high speed. Words are stored on tape at a density of 139 words per inch. Tape is provided on 2500-foot reels. Tape Module programming is discussed in Chapter XI.

10. TAPE ADAPTER

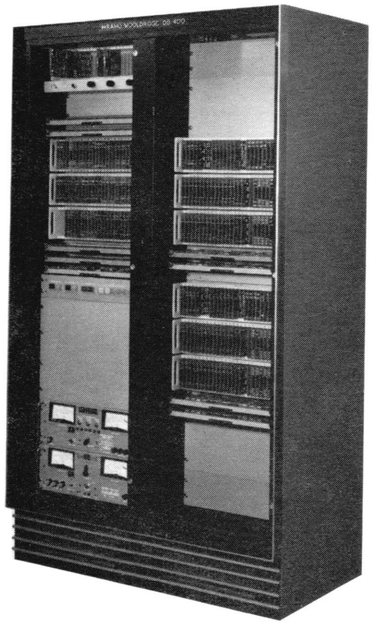
The Tape Adapter provides the capability of reading IBM tapes and of converting the information into RW format and of taking RW data and converting it into IBM format and writing it on IBM tape. The Tape Adapter makes and reads 1/2-inch tapes of the format used on IBM 727 and 729-I tape machines. Tape Adapter programming is discussed in Chapter XII.

11. PLOTTER

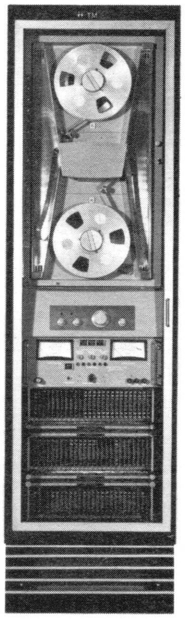
The Plotter gives the programmer the capability of getting a hard copy plot of any data he desires. The Plotter will plot points, lines, or symbols. It



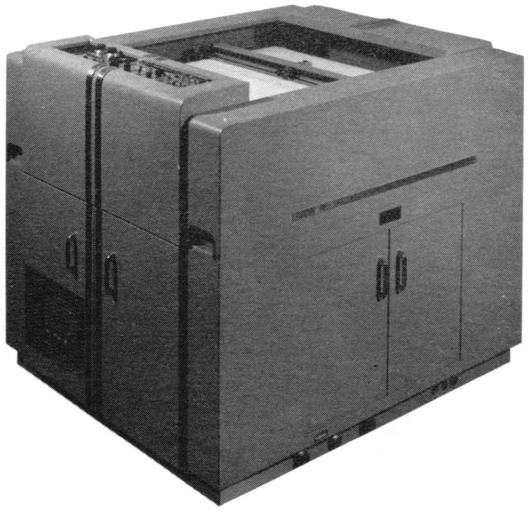
Peripheral Buffer



Display Buffer

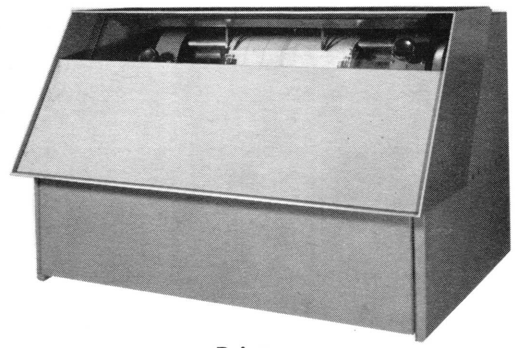


Tape Module



Plotter

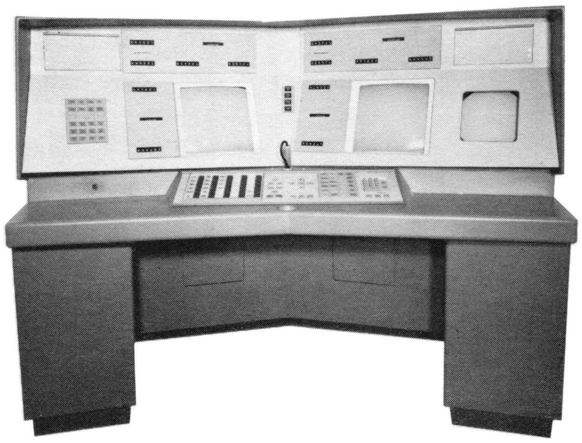
Figure 1.3. RW-400 Modules



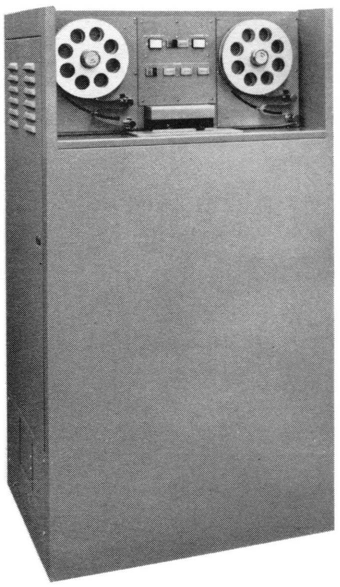
Printer

User Module

Computer Communication Console



Display and Analysis Console



Paper Tape Reader

Figure 1.4. RW-400 Modules

can plot as many as four different functions simultaneously, each to a different scale. The plots can be made on paper up to 30 inches by 30 inches in size. Plotter programming is discussed in Chapter XIII.

12. PRINTER

The Printer provides the ability to print data at relatively high speeds. It will print 80-character lines at a rate of 900 per minute and can make up to six legible copies when operating at this speed. Paper feed format is controlled with punched paper tape, permitting program selection of seven different line-skipping formats. Printer programming is discussed in Chapter XIV.

13. USER MODULE

The User Module is a Flexowriter in a special enclosure. It is connected to a Peripheral Buffer and will transmit or accept data at a rate of 10 characters per second. User Module programming is discussed in Chapter XV.

14. COMPUTER COMMUNICATIONS CONSOLE

The Computer Communications Console is a human-operated console that is used to facilitate communication between the operators of special-purpose devices and the Data Processing Central. Basically, the Console contains a group of keys, the depression of which generates various codes which will either insert data into or control functions of the Data Processing Central. The Console includes a 10-inch alphanumeric display tube on which the Data Processing Central can display messages for the operator. Console programming is discussed in Chapter XVI.

15. DISPLAY AND ANALYSIS CONSOLE

The Display and Analysis Console is a special-purpose console which allows a human operator to monitor and control functions of the Data Processing Central and to enter data into it. It has two 17-inch cathode-ray tube displays on which dot or symbol graphs of computed results can be automatically displayed. It has a 10-inch alphanumeric display tube on which any alphanumeric message of up to 640 characters in length can be displayed. It has a series of push buttons, similar to those on the Computer Communications Console, with which the operator can control functions of or enter data into the Data Processing Central. Display and Analysis Console programming is discussed in Chapter XVII.

16. PAPER TAPE READER

The Paper Tape Reader is a high-speed device for reading Flexowriter punched paper tapes. It can read up to 300 characters per second. Paper Tape Reader programming is discussed in Chapter XVIII.

17. DIGITAL CLOCK

The Digital Clock keeps time in terms of sixtieths of a second elapsed since the clock was set. A controlling module can connect to it through the Central (or Interim) Exchange and get the time. The Digital Clock counts up to 5,184,000 (24 hours) and then starts again. It may be reset at any time. Digital Clock programming is discussed in Chapter XX.

Chapter II

COMPUTER INTERNAL INSTRUCTIONS

A. GENERAL

There are 38 Computer instructions available to the programmer. Of these, 24 may be classified as arithmetic instructions, 9 as program control instructions, and 5 as external instructions. The arithmetic instructions and program control instructions are considered to be internal instructions; that is, instructions which cause computations or decisions involving only data from the Computer's own core memory or flip-flop registers. The external instructions are discussed in Chapter III. This chapter deals with the internal instructions.

B. DESCRIPTION OF THE COMPUTER

The Computer memory consists of 1024 words of magnetic core storage. Each word comprises 26 information bits and 2 parity bits. The bits of a word are numbered from 1 through 26 beginning with the least significant. The words are divided into two 13-bit half-words. The most significant half-word is composed of bits 26 through 14 and the least significant half-word, of bits 13 through 1. A machine-generated parity bit is associated with each half-word. In a data word, bit 26 is the sign bit, and bits 25 through 1 contain the magnitude of the number. (A one in bit 26 is a minus.) The binary point is between bits 26 and 25. In an instruction word, bits 1 through 10 are the h-field, bits 11 through 20 are the g-field, and bits 21 through 26 are the f-field.

The Arithmetic and Control section of the Computer consists of seven flip-flop registers: the Accumulator or A register, the accumulator extension or B register, the exchange or E register, the instruction or X register, the interrupt sensing or S register, the program counter or P register, and the word time counter or T register. The programmer deals directly with the A, B, P, and S registers.

The A register is the Accumulator and is used in all arithmetic and logical operations. It holds the result of all additions and subtractions, the quotient of all divisions, the roots of square root operations, and the most significant half of all products. It has 26 bit positions.

The B register, also a 26-bit register, is an extension of A. It holds the least significant half of products and the remainder of division and square root operations. It is also used as an extension register for double-length shifting.

The 20-bit S register, after having been loaded with an appropriate mask, is used to determine which of the alerts that come into the Computer will cause interrupts and which will not. (Interrupts are discussed in Chapter I.)

The 26-bit E register is used for temporary storage of operands during arithmetic operations and for transfer of information between the Computer and other modules of the system.

The X register is the instruction register. It receives the successive instructions from the Computer memory, causes the specified locations in the memory to be connected to the Arithmetic and Control section and then causes the instruction to be executed. It holds one 26-bit word.

P is a 10-bit register that carries the address of the next instruction to be placed into the X register. After each instruction is placed in X, P is incremented by one. A jump instruction can be used to put an entirely new instruction address into P.

The seven-bit T register controls the sequence of the Computer microprogram.

C. ARITHMETIC INSTRUCTIONS

1. OPERATIONS AND MODES

All but 3 of the 24 arithmetic instructions fit immediately into a symmetrical scheme of classification, consisting of seven basic operations and three modes for each operation. The seven operations are: add, subtract, absolute subtract, multiply, divide, square root, and insert. The three modes are: replace, hold, and store. These 21 instructions can be written in general form as follows:

Replace	$H*G \rightarrow H, A$
Hold	$H*G \rightarrow A$
Store	$A*G \rightarrow H, A$

Where:

G is the first operand (taken from address g)

H is the second operand (taken from address h)

A is the contents of the Accumulator

* denotes any arithmetic operation

$\rightarrow$ denotes replacement ($A \rightarrow B$ means the location containing B is made to contain A)

H, A means both H and A

From this, 21 of the arithmetic instructions can be deduced. For example:

Replace Add	$H + G \rightarrow H, A$
Hold Absolute Subtract	$\lvert H\rvert - \lvert G\rvert \rightarrow A$
Store Divide	$A \div G \rightarrow H, A$
Replace Square Root	$\sqrt{H + G} \rightarrow H, A$

The three remaining arithmetic instructions may be thought of as involving three additional basic instructions, each one existing in only one mode:

Add to Accumulator	$A + H + G \rightarrow A$ (hold mode)
Multiply Add	$A + (H \times G) \rightarrow A$ (hold mode)
Transmit	$G \rightarrow H$ (store mode)

2. ZERO ADDRESSES

The general format for these arithmetic instructions is as follows:

26 21	20 11	10 1
f	g	h

Where:

The f-field contains a code which identifies the instruction. (See Figure 2.2.)

The g-field contains the address of operand G.

The h-field contains the address of operand H.

If the address in g or h is made zero, instead of the contents of memory location zero being used as an operand, the number $+0$ is used. This means that the contents of location zero are not available as an operand in arithmetic instructions. The only exception to this among the arithmetic instructions is the *Transmit* instruction which uses address zero like any other address.

3. BUFFER ADDRESSING

One other address, like zero, produces special effects. For all arithmetic instructions (and several program control instructions), an address of 1777_8 in the g- or h-field addresses a Buffer Module. A Buffer, like a Computer, has a 1024-word core memory. Access to this memory is controlled by two 10-bit registers, the Read and Write registers. The contents of these registers, r and w, can be set by the Computer to any desired value with a *Command Output* instruction (refer to Chapter III for a discussion of *Command Output* instructions). When g or h in an arithmetic instruction is made 1777_8, either the Read or Write register (depending upon the instruction) is addressed and is used to read from or write into the Buffer memory, thus allowing the Computer to use the Buffer memory as an extension of its own. Therefore, no arithmetic instruction can address Computer core location 1777_8. If address 1777_8 is used and no Buffer is connected to the Computer, or if the connected Buffer is in the self-instruction mode or is transmitting status, the Program Error Indicator (bit 17 of the Computer's I register) is set. Figure 2.1 shows the rules for using the Buffer memory in Computer arithmetic instructions. The meaning of $g = Z$ in the table is that the g-field contains a 10-bit number which is neither 0 nor 1777_8. The meaning of $h = Z$ is that the h-field contains a 10-bit number which is neither 0 nor 1777_8. The meaning of g or $h = 0$ is that the g- or h-field is filled with zeros. R represents the contents of address r, W the contents of address w, G the contents of g, and H the contents of h.

4. DESCRIPTION OF ARITHMETIC INSTRUCTIONS

Each of the 24 arithmetic instructions can occur in any one of 9 different variations. This is because the address in the g- or h- field can be any one of three different cases: 0, Z, or 1777_8. Each of the 24 arithmetic instructions will be discussed, considering primarily the case of $g=Z$ and $h=Z$. In addition, a table of all 24 instructions, each in all nine variations, is included. (See Figure 2.2.)

Note: All addition, subtraction, division, and square root instructions can cause overflow, as can **Add to Accumulator** and **Multiply Add** instructions. Any overflow will result in the loss of the most significant bit and will set the Overflow Indicator. This indicator can be programmed to cause an interrupt if desired (refer to Chapter 1, page 1, for a discussion of interrupts).

In the following three add instructions, if the algebraic sum formed is zero, the sign will be minus when both operands are minus zero and plus when both operands are not minus zero.

a. *Replace Add.* The sum $H+G$ replaces H and A.

b. *Hold Add.* The sum $H+G$ replaces A. If either $g=0$ or $h=0$, a transmission to A of the non-zero operand occurs.

c. *Store Add.* The sum $A+G$ replaces H and A. If $g=0$ the contents of the accumulator replace H. If $h=0$, $A+G$ replaces A, H is not changed. If both $g=0$ and $h=0$, the instruction does not change any register.

Replace Mode H * G → H, A		
if $g=1777_8$ and $h=Z$	$H*R \to H, A$	$r+1 \to r$
if $g=Z$ and $h=1777_8$.	$R*G \to R, A$	$r+1 \to r$
if $g=1777_8$ and $h=1777_8$	$(r+1)*R \to (r+1), A$	$r+2 \to r$
if $g=1777_8$ and $h=0$	$(+0)*R \to A$	$r+1 \to r$
if $g=0$ and $h=1777_8$	$R*(+0) \to R, A$	$r+1 \to r$

Hold Mode H * G → A		
if $g=1777_8$ and $h=Z$	$H*R \to A$	$r+1 \to r$
if $g=Z$ and $h=1777_8$	$R*G \to A$	$r+1 \to r$
if $g=1777_8$ and $h=1777_8$	$(r+1)*R \to A$	$r+2 \to r$
if $g=1777_8$ and $h=0$	$(+0)*R \to A$	$r+1 \to r$
if $g=0$ and $h=1777_8$	$R*(+0) \to A$	$r+1 \to r$

Store Mode A * G → H, A		
if $g=1777_8$ and $h=Z$	$A*R \to H, A$	$r+1 \to r$
if $g=Z$ and $h=1777_8$	$A*G \to W, A$	$w+1 \to w$
if $g=1777_8$ and $h=1777_8$	$A*R \to W, A$	$r+1 \to r$ $w+1 \to w$
if $g=1777_8$ and $h=0$	$A*R \to A$	$r+1 \to r$
if $g=0$ and $h=1777_8$	$A*(+0) \to W, A$	$w+1 \to w$

Figure 2.1. Rules for Using Buffer Memory for Computer Arithmetic

Note: In the following three subtract instructions, if the algebraic difference formed is zero, the sign will be minus when the operand addressed by the g-field is plus zero and the operand addressed by the h-field is minus zero and the sign will be plus when either of these conditions is not met.

d. *Replace Subtract.* The difference $H-G$ replaces H and A.

e. *Hold Subtract.* The difference $H-G$ replaces A. If $h=0$, $-G$ is transmitted to the Accumulator.

f. *Store Subtract.* The difference $A-G$ replaces H and A. If $h=0$, $A-G$ replaces A, H is not changed.

g. *Replace Absolute Subtract.* The difference $|H|-|G|$ replaces H and A. If $g=0$, $|H|$ replaces H and A.

h. *Hold Absolute Subtract.* The difference $|H|-|G|$ replaces A. If $g=0$, $|H|$ replaces A. If $h=0$ then $-|G|$ replaces A.

i. *Store Absolute Subtract.* The difference $|A|-|G|$ replaces H and A. If $g=0$, $|A|$ replaces H and A. If $h=0$ $|A|-|G|$ replaces A. If $g=0$ and $h=0$, $|A|$ replaces A.

Note: In all arithmetic operations which give a double-length result, bit position 26 of both the A and B registers will contain the sign of the result.

j. *Replace Multiply.* (The subscript 'ms' refers to the most significant and 'ls' to the least significant portions of a double-length product.) The product $(H\times G)_{ms}$ replaces H and A. The product $(H\times G)_{ls}$ replaces B.

Note: In the following multiplication instructions the usual sign rule applies even when one operand is zero. Thus a minus zero may result.

k. *Hold Multiply.* The product $(H\times G)_{ms}$ replaces A. The product $(H\times G)_{ls}$ replaces B.

l. *Store Multiply.* The product $(A\times G)_{ms}$ replaces H and A. The product $(A\times G)_{ls}$ replaces B. If $h=0$, the product $(A\times G)_{ms}$ replaces A. H is not changed. If $g=0$, H, A, and B are all cleared.

Note: In the following divide instructions, either using a dividend equal to or greater than the divisor or allowing g to equal zero will result in an overflow. In the replace and hold modes G goes to B, H goes to A, and H is unaffected. In the case of the store mode, A and B are unchanged and G goes to H.

As a result of all divide operations where overflow does not occur, the sign of the remainder in B is made the same as the sign of the quotient in A.

m. *Replace Divide.* (The subscript q refers to the quotient, and the subscript r refers to the remainder resulting from a division or square root operation.) The quotient $(H\div G)_q$ replaces H and A, and the remainder $(H\div G)_r$ replaces B. $(H\div G)_q$ is rounded off. That is, the most significant numerical bit of B is added to the least significant end of A.

n. *Hold Divide.* The quotient $(H\div G)_q$ replaces A, and the remainder $(H\div G)_r$ replaces B. $(H\div G)_q$ is rounded off.

o. *Store Divide.* The quotient $(A\div G)_q$ replaces H and A, and the remainder $(A\div G)_r$ replaces B. If $h=0$, the quotient $(A\div G)_q$ replaces A, and H is not changed. $(A\div G)_q$ is not rounded off as it is in the hold and replace modes.

Note: In the following square root instructions, the remainder is always positive. When starting with a negative number, the root will be negative. The remainder has the same sign as the root.

p. *Replace Square Root.* The square root $\sqrt{H+G}$ replaces H and A and the remainder $(\sqrt{H+G})_r$ replaces B. The least significant bit of the root (in

Instruction (Octal f-Field Code)	Instruction Field		Register Contents		Stored Results				Execution Time (μSec)
	g	h	A	B	Computer Core Location h	Buffer Core Location r	Buffer Core Location r + 1	Buffer Core Location w	
REPLACE ADD (06)	Z	Z	H+G		H+G				63
	0000	Z	H						63
	Z	0000	G						63
	0000	0000	0						63
	1777_8	Z	H+R		H+R				67
	Z	1777_8	R+G			R+G			78
	1777_8	1777_8	(r+1)+R				(r+1)+R		81
	0000	1777_8	R						78
	1777_8	0000	R						67
HOLD ADD (07)	Z	Z	H+G						54
	0000	Z	H						52
	Z	0000	G						52
	0000	0000	0						52
	1777_8	Z	H+R						57
	Z	1777_8	R+G						55
	1777_8	1777_8	(r+1)+R						60
	0000	1777_8	R						55
	1777_8	0000	R						57
STORE ADD (05)	Z	Z	A+G		A+G				48
	0000	Z			A				48
	Z	0000	A+G						48
	0000	0000							48
	1777_8	Z	A+R		A+R				51
	Z	1777_8	A+G					A+G	58
	1777_8	1777_8	A+R					A+R	60
	0000	1777_8						A	58
	1777_8	0000	A+R						51
REPLACE SUB-TRACT (02)	Z	Z	H−G		H−G				63
	0000	Z	H						63
	Z	0000	−G						63
	0000	0000	0						63
	1777_8	Z	H−R		H−R				67
	Z	1777_8	R−G			R−G			78
	1777_8	1777_8	(r+1)−R				(r+1)−R		81
	0000	1777_8	R						78
	1777_8	0000	−R						67

Figure 2.2. Computer Module Arithmetic Instructions (Part 1 of 6)

Instruction (Octal f-Field Code)	Instruction Field		Register Contents		Stored Results				Execution Time (μSec)
	g	h	A	B	Computer Core Location h	Buffer Core Location r	Buffer Core Location r + 1	Buffer Core Location w	
HOLD SUB-TRACT (03)	Z	Z	H−G						52
	0000	Z	H						52
	Z	0000	−G						52
	0000	0000	0						52
	1777_8	Z	H−R						57
	Z	1777_8	R−G						55
	1777_8	1777_8	(r+1)−R						60
	0000	1777_8	R						55
	1777_8	0000	−R						57
STORE SUB-TRACT (01)	Z	Z	A−G		A−G				48
	0000	Z			A				48
	Z	0000	A−G						48
	0000	0000							48
	1777_8	Z	A−R		A−R				51
	Z	1777_8	A−G					A−G	58
	1777_8	1777_8	A−R					A−R	60
	0000	1777_8						A	58
	1777_8	0000	A−R						51
REPLACE ABSO-LUTE SUB-TRACT (12)	Z	Z	|H|−|G|		|H|−|G|				63
	0000	Z	|H|		|H|				63
	Z	0000	−|G|						63
	0000	0000	0						63
	1777_8	Z	|H|−|R|		|H|−|R|				67
	Z	1777_8	|R|−|G|			|R|−|G|			78
	1777_8	1777_8	|(r+1)|−R|				|(r+1)|−R|		81
	0000	1777_8	|R|			|R|			78
	1777_8	0000	−|R|						67
HOLD ABSO-LUTE SUB-TRACT (13)	Z	Z	|H|−|G|						52
	0000	Z	|H|						52
	Z	0000	−|G|						52
	0000	0000	0						52
	1777_8	Z	|H|−|R|						57
	Z	1777_8	|R|−|G|						55
	1777_8	1777_8	|(r+1)|−R|						60
	0000	1777_8	|R|						55
	1777_8	0000	−|R|						57

Figure 2.2. Computer Module Arithmetic Instructions (Part 2 of 6)

Instruction (Octal f-Field Code)	Instruction Field		Register Contents		Stored Results				Execution Time (μSec)
	g	h	A	B	Computer Core Location h	Buffer Core Location r	Buffer Core Location r + 1	Buffer Core Location w	
STORE ABSOLUTE SUBTRACT (11)	Z	Z	$\lvert A\rvert-\lvert G\rvert$		$\lvert A\rvert-\lvert G\rvert$				48
	0000	Z	$\lvert A\rvert$		$\lvert A\rvert$				48
	Z	0000	$\lvert A\rvert-\lvert G\rvert$						48
	0000	0000	$\lvert A\rvert$						48
	1777_8	Z	$\lvert A\rvert-\lvert R\rvert$		$\lvert A\rvert-\lvert R\rvert$				51
	Z	1777_8	$\lvert A\rvert-\lvert G\rvert$					$\lvert A\rvert-\lvert G\rvert$	58
	1777_8	1777_8	$\lvert A\rvert-\lvert R\rvert$					$\lvert A\rvert-\lvert R\rvert$	60
	0000	1777_8	$\lvert A\rvert$					$\lvert A\rvert$	58
	1777_8	0000	$\lvert A\rvert-\lvert R\rvert$						51
REPLACE MULTIPLY (26)	Z	Z	$(HxG)_{ms}$	$(HxG)_{ls}$	$(HxG)_{ms}$				124
	0000	Z	0	0	0				124
	Z	0000	0	0					124
	0000	0000	0	0					124
	1777_8	Z	$(HxR)_{ms}$	$(HxR)_{ls}$	$(HxR)_{ms}$				129
	Z	1777_8	$(RxG)_{ms}$	$(RxG)_{ls}$		$(RxG)_{ms}$			135
	1777_8	1777_8	$[(r+1)xR]_{ms}$	$[(r+1)xR]_{ls}$			$[(r+1)xR]_{ms}$		142
	0000	1777_8	0	0		0			136
	1777_8	0000	0	0					129
HOLD MULTIPLY (27)	Z	Z	$(HxG)_{ms}$	$(HxG)_{ls}$					120
	0000	Z	0	0					120
	Z	0000	0	0					120
	0000	0000	0	0					120
	1777_8	Z	$(HxR)_{ms}$	$(HxR)_{ls}$					124
	Z	1777_8	$(RxG)_{ms}$	$(RxG)_{ls}$					123
	1777_8	1777_8	$[(r+1)xR]_{ms}$	$[(r+1)xR]_{ls}$					127
	0000	1777_8	0	0					123
	1777_8	0000	0	0					124
STORE MULTIPLY (25)	Z	Z	$(AxG)_{ms}$	$(AxG)_{ls}$	$(AxG)_{ms}$				117
	0000	Z	0	0	0				117
	Z	0000	$(AxG)_{ms}$	$(AxG)_{ls}$					117
	0000	0000	0	0					117
	1777_8	Z	$(AxR)_{ms}$	$(AxR)_{ls}$	$(AxR)_{ms}$				120
	Z	1777_8	$(AxG)_{ms}$	$(AxG)_{ls}$				$(AxG)_{ms}$	127
	1777_8	1777_8	$(AxR)_{ms}$	$(AxR)_{ls}$				$(AxR)_{ms}$	129
	0000	1777_8	0	0				0	127
	1777_8	0000	$(AxR)_{ms}$	$(AxR)_{ls}$					120

Figure 2.2. Computer Module Arithmetic Instructions (Part 3 of 6)

Instruction	Instruction Field		Register Contents		Stored Results				Execution Time (μSec)
(Octal f-Field Code)	g	h	A	B	Computer Core Location h	Buffer Core Location r	Buffer Core Location r + 1	Buffer Core Location w	
REPLACE DIVIDE (22)	Z	Z	$(H \div G)_q$	$(H \div G)_r$	$(H \div G)_q$				192
	0000	Z	H, Overflow	0					63
	Z	0000	0	0					192
	0000	0000	0, Overflow	0					63
	1777_8	Z	$(H \div R)_q$	$(H \div R)_r$	$(H \div R)_q$				198
	Z	1777_8	$(R \div G)_q$	$(R \div G)_r$		$(R \div G)_q$			204
	1777_8	1777_8	$[(r+1) \div R]_q$	$[(r+1) \div R]_r$			$[(r+1) \div R]_q$		210
	0000	1777_8	R, Overflow	0					75
	1777_8	0000	0	0					198
HOLD DIVIDE (23)	Z	Z	$(H \div G)_q$	$(H \div G)_r$					192
	0000	Z	H, Overflow	0					63
	Z	0000	0	0					192
	0000	0000	0, Overflow	0					63
	1777_8	Z	$(H \div R)_q$	$(H \div R)_r$					198
	Z	1777_8	$(R \div G)_q$	$(R \div G)_r$					195
	1777_8	1777_8	$[(r+1) \div R]_q$	$[(r+1) \div R]_r$					200
	0000	1777_8	R, Overflow	0					64
	1777_8	0000	0	0					198
STORE DIVIDE (21)	Z	Z	$(A \div G)_q$	$(A \div G)_r$	$(A \div G)_q$				183
	0000	Z	Overflow		0				52
	Z	0000	$(A \div G)_q$	$(A \div G)_r$					183
	0000	0000	Overflow						52
	1777_8	Z	$(A \div R)_q$	$(A \div R)_r$	$(A \div R)_q$				184
	Z	1777_8	$(A \div G)_q$	$(A \div G)_r$				$(A \div G)_q$	192
	1777_8	1777_8	$(A \div R)_q$	$(A \div R)_r$				$(A \div R)_q$	195
	0000	1777_8	Overflow					0	63
	1777_8	0000	$(A \div R)_q$	$(A \div R)_r$					184
REPLACE SQUARE ROOT (16)	Z	Z	$\sqrt{H+G}$	$(\sqrt{H+G})_r$	$\sqrt{H+G}$				256
	0000	Z	$\sqrt{H}$	$(\sqrt{H})_r$	$\sqrt{H}$				256
	Z	0000	$\sqrt{G}$	$(\sqrt{G})_r$					256
	0000	0000	2^{-25}	0					256
	1777_8	Z	$\sqrt{H+R}$	$(\sqrt{H+R})_r$	$\sqrt{H+R}$				261
	Z	1777_8	$\sqrt{R+G}$	$(\sqrt{R+G})_r$		$\sqrt{R+G}$			268
	1777_8	1777_8	$\sqrt{(r+1)+R}$	$(\sqrt{(r+1)+R})_r$			$\sqrt{(r+1)+R}$		274
	0000	1777_8	$\sqrt{R}$	$(\sqrt{R})_r$		$\sqrt{R}$			268
	1777_8	0000	$\sqrt{R}$	$(\sqrt{R})_r$					261

Figure 2.2. Computer Module Arithmetic Instructions (Part 4 of 6)

Instruction (Octal f-Field Code)	Instruction Field		Register Contents		Stored Results				Execution Time (μSec)
	g	h	A	B	Computer Core Location h	Buffer Core Location r	Buffer Core Location r + 1	Buffer Core Location w	
HOLD SQUARE ROOT (17)	Z	Z	$\sqrt{H+G}$	$(\sqrt{H+G})_r$					252
	0000	Z	$\sqrt{H}$	$(\sqrt{H})_r$					252
	Z	0000	$\sqrt{G}$	$(\sqrt{G})_r$					252
	0000	0000	2^{-25}	0					252
	1777_8	Z	$\sqrt{H+R}$	$(\sqrt{H+R})_r$					256
	Z	1777_8	$\sqrt{R+G}$	$(\sqrt{R+G})_r$					254
	1777_8	1777_8	$\sqrt{(r+1)+R}$	$(\sqrt{(r+1)+R})_r$					259
	0000	1777_8	$\sqrt{R}$	$(\sqrt{R})_r$					254
	1777_8	0000	$\sqrt{R}$	$(\sqrt{R})_r$					256
STORE SQUARE ROOT (15)	Z	Z	$\sqrt{A+G}$	$(\sqrt{A+G})_r$	$\sqrt{A+G}$				241
	0000	Z	$\sqrt{A}$	$(\sqrt{A})_r$	$\sqrt{A}$				241
	Z	0000	$\sqrt{A+G}$	$(\sqrt{A+G})_r$					241
	0000	0000	$\sqrt{A}$	$(\sqrt{A})_r$					241
	1777_8	Z	$\sqrt{A+R}$	$(\sqrt{A+R})_r$	$\sqrt{A+R}$				244
	Z	1777_8	$\sqrt{A+G}$	$(\sqrt{A+G})_r$				$\sqrt{A+G}$	252
	1777_8	1777_8	$\sqrt{A+R}$	$(\sqrt{A+R})_r$				$\sqrt{A+R}$	253
	0000	1777_8	$\sqrt{A}$	$(\sqrt{A})_r$				$\sqrt{A}$	252
	1777_8	0000	$\sqrt{A+R}$	$(\sqrt{A+R})_r$					244
REPLACE INSERT (36)	Z	Z	HG'vAG	AG	HG'vAG				64
	0000	Z	H	0					64
	Z	0000	AG	AG					64
	0000	0000	0	0					64
	1777_8	Z	HR'vAR	AR	HR'vAR				70
	Z	1777_8	RG'vAG	AG		RG'vAG			75
	1777_8	1777_8	(r+1)R'vAR	AR			(r+1)R'vAR		82
	0000	1777_8	R	0					75
	1777_8	0000	AR	AR					70
HOLD INSERT (37)	Z	Z	HG'vAG	AG					64
	0000	Z	H	0					64
	Z	0000	AG	AG					64
	0000	0000	0	0					64
	1777_8	Z	HR'vAR	AR					70
	Z	1777_8	RG'vAG	AG					67
	1777_8	1777_8	(r+1)R'vAR	AR					72
	0000	1777_8	R	0					67
	1777_8	0000	AR	AR					70

Figure 2.2. Computer Module Arithmetic Instructions (Part 5 of 6)

Instruction	Instruction Field		Register Contents		Stored Results				Execution Time (μSec)
(Octal f-Field Code)	g	h	A	B	Computer Core Location h	Buffer Core Location r	Buffer Core Location r + 1	Buffer Core Location w	
STORE INSERT (35)	Z	Z	AG		AG				57
	0000	Z	0		0				57
	Z	0000	AG						57
	0000	0000	0						57
	1777_8	Z	AR		AR				63
	Z	1777_8	AG					AG	67
	1777_8	1777_8	AR					AR	69
	0000	1777_8	0					0	67
	1777_8	0000	AR						63
ADD TO ACCUMULATOR (04)	Z	Z	A+H+G						57
	0000	Z	A+H						57
	Z	0000	A+G						57
	0000	0000							57
	1777_8	Z	A+H+R						60
	Z	1777_8	A+R+G						60
	1777_8	1777_8	A+(r+1)+R						63
	0000	1777_8	A+R						60
	1777_8	0000	A+R						60
MULTIPLY ADD (24)	Z	Z	$A+(HxG)_{ms}$	$(HxG)_{ls}$					127
	0000	Z		0					127
	Z	0000		0					127
	0000	0000		0					127
	1777_8	Z	$A+(HxR)_{ms}$	$(HxR)_{ls}$					132
	Z	1777_8	$A+(RxG)_{ms}$	$(R+G)_{ls}$					130
	1777_8	1777_8	$A+[(r+1)xR]_{ms}$	$[(r+1)xR]_{ls}$					135
	0000	1777_8		0					130
	1777_8	0000		0					132
TRANSMIT (31)	Z	Z			G				48
	0000	Z			Word 0				48
	Z	0000			(G→Word 0)				48
	0000	0000							48
	1777_8	Z			R				49
	Z	1777_8						G	57
	1777_8	1777_8						R	60
	0000	1777_8						Word 0	57
	1777_8	0000			(R→Word 0)				49

Figure 2.2. Computer Module Arithmetic Instructions (Part 6 of 6)

all square root instructions) is always made a one. If g=0, then $\sqrt{H}$ replaces H and A and the remainder $(\sqrt{H})_r$ replaces B. If h=0, then $\sqrt{G}$ replaces A and $(\sqrt{G})_r$ replaces B.

q. *Hold Square Root.* The square root $\sqrt{H+G}$ replaces A, and the remainder $(\sqrt{H+G})_r$ replaces B. If h=0, then $\sqrt{G}$ replaces A, and the remainder $(\sqrt{G})_r$ replaces B. If g=0, then $\sqrt{H}$ replaces A, and the remainder $(\sqrt{H})_r$ replaces B.

r. *Store Square Root.* The square root $\sqrt{A+G}$ replaces H and A, and the remainder $(\sqrt{A+G})_r$ replaces B. If g=0, then $\sqrt{A}$ replaces H and A, and the remainder $(\sqrt{A})_r$ replaces B. If h=0, then $\sqrt{A+G}$ replaces A, and the remainder $(\sqrt{A+G})_r$ replaces B.

Note: The **insert** instructions which follow have been classified as arithmetic, because they use the g- and h-fields in exactly the same way as arithmetic instructions do. Actually, insertion is a logical operation and does not involve arithmetic operations. In the **insert** instructions the symbol v denotes the logical **or** function (a bit-by-bit sum without carries). Juxtaposition denotes the logical **and** function (a bit-by-bit product without carries). A prime mark denotes the ones complement of the function so marked. When entire words are subject to an operation such as an **and** operation, each bit is actually operated on individually.

s. *Replace Insert.* The quantity HG′ v AG replaces H and A, and the quantity AG replaces B. In effect, G is used as a selector to *merge* the words H and A. Whenever a zero occurs in G, a bit is selected from H, and whenever a one occurs in G, a bit is selected from A. This instruction may be used to perform other logical operations. To complement, for example, first set A=+0 by any convenient instruction, then choose h so that H has a one in every position of G that is to be complemented. If H is all ones, the function HG′ v AG replaces H, A reduces to $G' \rightarrow H$, A. To obtain an *or* with this instruction, first bring G to the Accumulator by any convenient instruction. HG′ v AG then becomes HG′ v GG=HG′ v G=HvG. An *and* can also be obtained from this instruction, but it is even more readily obtained with a *Store Insert* instruction.

t. *Hold Insert.* The quantity HG′ v AG replaces A, and the quantity AG replaces B. If g=0, H replaces A, and B is cleared. If g=0 and h=0, A and B are both cleared.

u. *Store Insert.* The quantity AG replaces H and A. With this instruction, selected bits of the word in h are replaced with the corresponding bits in A. The remaining bits in H are made zero. The word in g is used as the mask to select which of the bits in A are not used to replace bits in H. A one in any bit position of G causes the corresponding bit in H to be replaced from A. B is not changed.

v. *Add to Accumulator.* The sum A+H+G replaces A. If the algebraic sum is zero, its sign will be minus when all three operands are minus zero and plus when all three operands are not minus zero.

w. *Multiply Add.* The sum $A+(H \times G)_{ms}$ replaces A, and $(H \times G)_{ls}$ replaces B. The sign of H×G is determined in the same way as for the *Hold Multiply* instruction. If $|A| \geq |H \times G|$, the final sign of A will be the same as the sign of A before the instruction was executed. If $|A| < |H \times G|$, the final sign of A will be the same as the sign of H×G.

x. *Transmit.* G replaces H. If g=0, the contents of location 0 replace H. If h=0, G replaces the contents of location 0. If g=0 and h=0 or, in general, if g=h, a *no-op* results. This instruction is convenient for exchanging information with a Buffer Module and for moving words about in the Computer's own memory. It can be used to place a *Link Jump* instruction in location 0 for the interrupt linkage. The *Transmit* instruction does not affect the Accumulator in any way.

D. PROGRAM CONTROL INSTRUCTIONS

1. GENERAL

The CM-400 has nine program control instructions. Each of them is discussed in detail below. The octal f-field code of each program control instruction is shown in Figure 2.3.

Instruction	Octal f-Field Code
STORE A, B	34
LOAD A	33
INSERT S	77
STOP	00
LINK JUMP	72
COMPARE JUMP	73
TALLY JUMP	71
TEST JUMP	70
SHIFT	30

Figure 2.3. f-Field Codes for the Program Control Instructions

2. DESCRIPTION OF PROGRAM CONTROL INSTRUCTIONS

a. *Store A, B.* In this instruction the f-field contains the number 34_8, the g-field carries an address from 0 through 1777_8 and the h-field carries an address from 0 through 1777_8. The operation of the instruction is as follows: B replaces G and A replaces H. If $g=0$, A replaces H only. If $h=0$, B replaces G only. If $g=0$ and $h=0$, A and B are exchanged. Unless $g=0$ and $h=0$, A and B are unchanged. *Store A, B* is the only CM-400 instruction capable of storing a full word in the location specified in the g-field. Also note that if $g=1777_8$, and $h=1777_8$, the Buffer Module Write register is used twice and is, therefore, incremented between uses. This makes the instruction unique in being able to store in locations w and w+1. When either $g=1777_8$ or $h=1777_8$, $w+1 \rightarrow w$. When $g=1777_8$ and $h=1777_8$, $w+2 \rightarrow w$. This instruction is summarized in Figure 2.4.

g \ h	Z	O	1777_8
Z	B→G A→H 30 μsec	B→G 30 μsec	B→G A→W 36 μsec
O	A→H 30 μsec	A→B B→A 30 μsec	A→W 37 μsec
1777_8	B→W A→H 35 μsec	B→W 35 μsec	B→W A→W + 1 42 μsec

Note: The times indicated are execution times for the various cases.

Figure 2.4. Summary of the Store A, B Instruction

b. *Load A.* In the *Load A* instruction, the f-field contains the number 33_8, the g-field contains any number from 0 to 1777_8, and the h-field contains any number from 0 to 1777_8. The operation of the instruction is as follows: the g-field replaces bits 20-11 of A, and the h-field replaces bits 10-1 of A. Bits 26-21 of A are set to zero. This instruction does not change G, H, or B. If $g=0$ and $h=0$, the Accumulator is cleared. This is the fastest way of clearing the Accumulator. The execution of the instruction takes 15 microseconds.

c. *Insert S.* In the *Insert S* instruction, the f-field contains the number 77_8, the g- and h-fields are used as a mask to insert ones and zeros into the 20-bit sense register, S.

The *Insert S* instruction is similar to the *Hold Insert* instruction. The quantity $(A_{gh}X_{gh})$ v (SX'_{gh}) replaces S, unless X_{gh} is all zeros. (The subscripts g and h refer to the g- and h-fields of the register whose symbol is marked with the subscript. The symbol X refers to the contents of the Instruction Register.) If X_{gh} is all zeros, bit 20 of S is set to one and the remaining bits of S are unchanged, as are A and B.

If the contents of bits 1-20 of A have been specified previously (this may be done conveniently with a *Load A* instruction), the *Insert S* instruction may be used to specify the state (1 or 0) of each bit of S independently. Whenever X_{gh} contains a one, a bit from A will replace the corresponding bit in S. When X_{gh} contains a zero in a particular bit position, the corresponding bit in S will remain unchanged. To complete the *Insert S* instruction requires 27 microseconds.

After this instruction has been executed, S and bits 1 through 20 of A each contain $(A_{gh}X_{gh})$ v (SX'_{gh}). B contains $A_{gh}X_{gh}$ in bits 1 through 20. Both A and B contain zeros in bits 21 through 26.

d. *Stop.* In this instruction, the f-field contains the quantity 00_8. The g- and h-fields are not examined. The instruction causes the P register to be incremented by one. All further change in the Arithmetic and Control section then stops until an external start pulse is received. Twelve microseconds are required to complete the instruction.

The stop caused by this instruction is made just before an interruption would normally occur. (Refer to Chapter I for a discussion of interrupts.) Thus, if an interrupt comes in during the execution of a *Stop* instruction, the stop will occur just before transfer of control to location zero would normally take place. As soon as the Computer is restarted, the interrupt will occur unless some manual change in the flip-flops preventing interruptions is made during the stop. If an interrupt signal is received by the Computer while it is stopped and remains until the Computer starts, then, on starting, the Computer will execute one instruction and then interrupt. If an interrupt occurs while the Computer is stopped and goes off before the Computer starts, it is lost.

e. *Link Jump.* In this instruction, the contents of the P register plus one are stored in the h-field of the word in core location g $(P+1 \rightarrow G_{10-1})$. The number in the h-field of the instruction is used as the address of the next instruction to be executed ($h \rightarrow P$).

Placing 0 or 1777_8 in the g- or h-field of this instruction *will* address core memory location 0 or 1777_8. This instruction takes 30 microseconds to execute.

f. *Compare Jump*. In this instruction, A is compared algebraically with G. If $A \leqq G$, H is taken as the next instruction. If $A > G$, the next instruction is taken in sequence. If $g = 1777_8$, then A is compared with R instead of with G, and r is incremented by one. If $g = 0$ and $A = +0$, H is taken as the next instruction. If $g = 0$ and $A \neq +0$, the next instruction is taken in sequence. A is not changed. There is no indirect addressing with the h-field; h can address any location in the Computer memory including 0 and 1777_8.

This instruction takes 25 microseconds to execute except when $g = 1777_8$, in which case, it takes 27 microseconds.

g. *Tally Jump*. In this instruction, if $G = +0$ the next instruction is taken in sequence and G is not changed. If $G > +0$, then H is taken as the next instruction and one is subtracted from G. If $G = -0$, H is taken as the next instruction and G is not changed. If $G < -0$, the next instruction is taken in sequence and one is added to G. (Although -0 cannot occur as the result of a numerical CM-400 computation, it does occur as the result of a *Tally Jump* when $G = -2^{-25}$ at the beginning of the instruction.) If $g = 0$, an unconditional jump to H takes place. If $g = 1777_8$, R is used in the same manner as G above and $r + 1 \rightarrow r$. There is no indirect addressing with the h-field; h can address any location in the Computer memory including 0 and 1777_8.

The execution time for this instruction is 18 microseconds when $g = 0$, 38 microseconds when $g = Z$, and 46 microseconds when $g = 1777_8$.

h. *Test Jump*. The g-field of this instruction does not refer to a location in memory. It is an information word specifying the conditions of the test. The test consists of taking a word from one of several sources, placing it in the Accumulator, and comparing one bit of the word with bit 6 in the g-field of the instruction. If they are the same, there is a jump to h. (There is no indirect addressing with h.) If not, the next instruction is taken in sequence. The word to be put in the Accumulator is selected by bits 17 through 20 in the g-field of the instruction. The particular bit to be tested is selected by bits 11 through 15 in the g-field and, as mentioned above, bit 16 in the g-field is used as the basis of comparison. Whenever a source which consists of less than 26 bits is used to load the Accumulator, the source bits are placed in the least significant end of the Accumulator and all of the other Accumulator bits are set to zero.

The possible sources of test words and the arrangement of bits 17 through 20 of the g-field which select them are shown in Figure 2.5. The following definitions apply to the table: S is the 20-bit word in the Sense (S) Register. I is the set of 20 alert bits treated as a 20-bit word (refer to Chapter I for a discussion of alerts). N is a 16-bit word made up from the three control lines and 13 input lines coming into the Computer from the Central Exchange. (Refer to Chapter I, page 1, for a discussion of intermodule connections.) J is a word made of the eight Conditional Jump switches on the Computer Console. A is the contents of the Accumulator before execution of the *Test Jump* instruction. The symbol v denotes a logical *or* function, and juxtaposition denotes a logical *and* function. *Or* functions and *and* functions are explained on page 17. If N is tested by a Computer which is not connected to a controlled module, all of the 16 bits will be zero if the IX is being used, and all ones if the CX is being used.

As mentioned above, only one bit of the word placed in A by the *Test Jump* instruction is tested. The bit to be tested is selected by bits 11 through 15 in the g-field of the *Test Jump* instruction as indicated in Figure 2.6.

Note in Figure 2.6 that when the bits equal 27 through 31, no test word need be brought to the Accumulator since no bit in it is tested. However, if a test word is called for in bits 17 through 20, it will be brought to the Accumulator in any case.

$g = 0$ is a special case which results in an unconditional jump to h in 15 microseconds.

The time to execute the instruction for $g \neq 0$ depends on which one of the many tests is made. If the source number (see Figure 2.5) is from 0 through 11, the basic time is 32 microseconds. If the source number is from 12 through 15, the basic time is 37 microseconds. If the bit position number is from 1 through 26, the total time for the instruction is the basic time plus 5m/3 microseconds (where m is the bit position number). For other values of the bit position number, the time is just the basic time.

i. *Shift*. In this instruction, the contents of the Accumulator are shifted and then replace H. The g-field of the instruction is used to specify details of the shifting process. Bits 11 through 15 of the g-field contain a number m, from 0 through 30,

1+ 16 [illegible] Bit
Bit 11-15 Bit NO tested

JNA
JOA
JNB
JNB
JNX
JOX
JNS
JOS
JNI
JOI
JNSI
JØSI

Decimal Source Number	g-Field Bit Position				16 Source
	20	19	18	17	
0	0	0	0	0	0 1 A
1	0	0	0	1	0 1 J
2	0	0	1	0	0 1 N
3	0	0	1	1	0 1 N v J
4	0	1	0	0	0 1 S
5	0	1	0	1	0 1 S v J
6	0	1	1	0	0 1 S v N
7	0	1	1	1	0 1 S v N v J
8	1	0	0	0	0 1 I
9	1	0	0	1	0 1 I v J
10	1	0	1	0	0 1 I v N
11	1	0	1	1	I 0 1 v N v J
12	1	1	0	0	0 1 IS
13	1	1	0	1	0 1 (I v J)S
14	1	1	1	0	0 1 (I v N)S
15	1	1	1	1	(I 0 1 v N v J)S

Figure 2.5. Sources of Test Words for the Test Jump Instruction

if the shift is to be m places, or the number m=31 if a float operation is to take place. Each of the other five bits of g is used to specify a detail of the shifting process as indicated in Figure 2.7.

In a left shift, the bits are shifted toward the most significant end of the register. In a right shift, they are moved toward the least significant end.

In a logical shift, all 26 bits of the word are shifted. In a magnitude shift, the bit in the 26th (sign) position is not shifted with the rest of the bits.

In a double-length shift, A and B are coupled together to form one large shifting register, with

Decimal Representation of Bits 11 through 15 of the g-Field	Effect
0	No test. Only the input of the source to A occurs.
01-26	The corresponding bit in A is tested.
27	The Overflow Indicator is tested and reset.
28	The Parity Error Indicator is tested and reset.
29	The Program Error Indicator is tested and reset.
30	The Conditional Tape Read Indicator is tested and reset.
31	The Control Panel Test Light is tested and reset.

Figure 2.6. Test Bits Selected by the g-Field of the Test Jump Instruction

B as the least significant half. (In a double-length magnitude shift, the sign bits of the A and B registers are not moved.)

Whenever a non-zero bit is shifted out the left end of the shifting register and bit seven of the g-field of the shift instruction is a one, the Overflow Indicator is turned on.

Position of Bit in g-Field	Result if Bit is One	Result if Bit is Zero
16	left shift	right shift
17	overflow sets indicator	overflow does not set indicator
18	rounding takes place	rounding does not take place
19	logical shift	magnitude shift
20	double-length shift	single-length shift

Figure 2.7. Details of the Shift Instruction Specified by Bits 16 through 20 of the g-Field

Rounding always consists of adding a *round bit* to the magnitude of the shifted result in the Accumulator at the least significant bit position. The source of this *round bit* varies. In double-length shifts, it is the 26th bit of B in logical shifts or the 25th bit of B in magnitude shifts. In left single-length shifts, the *round bit* is always zero. In right single-length shifts, the source is the last bit shifted out of the Accumulator. In all right shifts, when m=0 the source is the last bit of the Accumulator itself. This leaves the last bit zero and rounds off the next to last bit. (In a right shift with m=0, if the rightmost 25 bits of A are all ones an overflow will result but the overflow indicator will not be set.)

In a float operation, the number in the Accumulator is shifted until a one appears in the last numerical bit position of A in the direction of the shift (position 1 on a right float, position 26 on a left logical float, and position 25 on a left magnitude float). A float operation will stop after 31 shifts, even if a one has not reached the last bit position.

As mentioned, after the shift operation the shifted number replaces H. If h=0, no storage takes place. If h=1777_8, the shifted number replaces W and w is incremented by one.

In shift operations, the shifted number remains in A (in addition to replacing H). In float operations, A is replaced with the absolute value of the number of shifts that took place.

The time for executing this instruction, whether shift or float, is 32+5m/3 microseconds unless h=1777_8, in which case it is 38+5m/3 microseconds.

Chapter III

EXTERNAL INSTRUCTIONS

A. GENERAL

Those instructions which deal with sending commands or data between the various modules of the RW-400 are classified as external instructions. There are three principal types: data input instructions, data output instructions, and command output instructions. The data input instructions deal with transmitting data from external units into the Computer or Buffer; the data output, with transmitting data from the Computer or Buffer into external units; and command output, with giving instructions to external units from the Computer or the Buffer. There are three variations of the data input type instruction: *Search Input*, *Direct Data Input*, and *Conditional Search Input*. There are two variations of the data output type instruction: *Search Output*, and *Direct Data Output*. There is only one command output type instruction, and it is called the *Command Output* instruction. In addition to these instructions, there is one called *Character Transfer*, which can be used for either data input or data output but only with a device that generates six-bit codes and is directly connected to a Computer Module.

The *Conditional Search Input* instruction is discussed in Chapter XI, Section D, since it is used primarily with Tape Modules. The *Character Transfer* instruction is discussed in Chapter XIX since it is used only with directly connected devices. The other external instructions, which have more general application, are discussed below.

B. COMMAND OUTPUT INSTRUCTIONS

The *Command Output* instruction enables a Computer or a Buffer to give a command to any external module to which it is connected, or to command the Central (or Interim) Exchange to connect it to or disconnect it from an external module, or, in the case of a Buffer in the self-instruction mode, to command itself.

The word format of this instruction is as follows:

Bit Position	26-21	20	19-18	17-14	13-1
Bit Use	f-Field	Wait or Not Wait	Function to be Performed	Category of Module Requested	Command or Requested Module

Where:

The f-field (when it contains octal code 42) identifies the word as a *Command Output* instruction.

Bit 20 dictates whether the device generating the instruction will wait for it to be accepted even if the receiving device is not immediately prepared to accept it (bit 20=1) or whether, if the command is not immediately accepted, it will set the Program Error Indicator and terminate the instruction (bit 20=0). This bit is often called the "wait" bit.

Note: If a **Command Output** instruction with bit 20=1 is permanently unacceptable the Computer will stall and will have to be manually reset. Bit 20 becomes meaningless and is ignored whenever a module is giving a command to itself.

The four possible bit combinations in positions 19 and 18 mean the following:

00 - The command in bits 13 through 1 is meant for the module currently connected to the Computer, providing it is not a Buffer Module. Or, the command in bits 13 through 1 is meant for the device currently connected to the Buffer (this latter, of course, when the *Command Output* instruction is being generated by a Buffer). The Master Disconnect of the Interim Exchange and the Master Coordinate of the Central Exchange are considered currently connected devices.

01 - The command in bits 13 through 1 is for the Computer itself. (The only current use for this is to allow the Computer to turn on the Test Light on its Control Panel.) If the Buffer is generating the command, it is a Buffer self-instruction command. (Buffer self-instructions are discussed in detail in Chapter IV.)

10 - Connect the controlling module to the unit indicated in bits 13 through 1, or, disconnect it from the unit to which it is now connected. (If the bits in positions 13 through 1, which indicate what unit it is desired to be connected to, are all zero, the instruction is

a *disconnect* instruction.) This command is directed to the Central (or Interim) Exchange.

11 - The command in bits 13 through 1 is meant for the Buffer to which the controlling module is now connected. (Only the Computer generates this code.)

The bits in positions 17 through 14 indicate the category of device to which a connection is desired when bits 19 and 18 are 10; i.e., when they request a connection to some unit, or when a disconnect is requested. The bits in positions 17 through 14 are not looked at when the bits in positions 19 and 18 are anything but 10. Figure 3.1 indicates the types of modules to which a connection can be made and the bit structure in bit positions 17 through 14 that identifies each of them. (These bits are only needed when the Computer Module is using the command. The Buffer Module does not need them.)

Category of Module Requested	Bit Position			
	17	16	15	14
BUFFER MODULE	0	0	0	0
TAPE MODULE	0	0	0	1
PAPER TAPE READER	0	0	0	1
DRUM MODULE	0	0	1	0
PERIPHERAL BUFFER	0	0	1	0
DISPLAY BUFFER	0	0	1	1
DIGITAL CLOCK	0	0	1	0
PRINTER	0	1	0	1
CENTRAL EXCHANGE MASTER COORDINATE	0	1	1	0
INTERIM EXCHANGE MASTER DISCONNECT	0	1	1	0
DISCONNECT REQUEST	1	1	1	1

Figure 3.1. Meaning of Bits 17 Through 14 in the Command Output Instruction

Bits 13 through 1 contain either the actual command which is being given to the module indicated in bits 19 and 18 (these commands are discussed in the chapters on each of the modules), or a code identifying the module to which a connection is requested (these codes are discussed in Chapters V and VI).

Briefly, the *Command Output* instruction is used by the Computer or Buffer to give a command to a module to which it is connected through the Central (or Interim) Exchange or to the Exchange itself. The most significant half-word (bits 26 through 14) identifies the word as a *Command Output* instruction, dictates whether or not the Computer or Buffer should wait for the command to be accepted, and indicates for what device or for what purpose the instruction is meant. The 13 least significant bits contain the command itself or, in the case of a connect instruction, a code representing the device to which a connection is desired. The various possible instructions in the 13 least significant bits are discussed in the chapters dealing with the modules to which the instructions would be addressed.

C. SEARCH INPUT INSTRUCTIONS

The *Search Input* instruction directs a module that is connected to a Computer or a Buffer through the Central (or Interim) Exchange to read information into the core memory of the Computer or Buffer. (A Computer or a Buffer can give instructions to any of the modules in the Data Processing Central and, in addition, a Computer can command a Buffer but a Buffer cannot command a Computer.) This instruction can only be used after a *Command Output* instruction has prepared the module being addressed to carry out the instruction. The *Command Output* instruction necessary before any particular type of module can accept a *Search Input* instruction is indicated in the chapter on that particular module.

The word format of a *Search Input* instruction is as follows:

26 21	20 11	10 1
f	g	h

Where:

The f-field (when it contains the octal code 41) identifies the word as a *Search Input* instruction.

The g-field carries the address of a search word, G, which should have been stored previously at address g.

The h-field contains the number of words to be transferred.

The first step in transferring data to a Computer or to a Buffer is a *Command Output* instruction to the connected module initiating the transfer of search data to the controlling module. This is followed by the *Search Input* instruction itself. The word in address g is compared with the search data, and when the two are equal data transfer starts with the first word transferred being placed in address g+1, the second in g+2, and so on. If G, the word at address g, is zero, no search is made and data is transferred unconditionally.

If g is made equal to zero and a Buffer is the controlling module, the contents of the W register will be used as the address of the search word, and the first transferred word will be stored in location w+1.

At the completion of the data transfer, if a Computer was the controlling module, the A register will contain the quantity $1777_8 - (h-n)$ and the B register will contain the search word G; if a Buffer was the controlling module, the L register will contain the quantity, h–n (h represents the quantity in the h-field and n represents the actual number of words transferred).

If the h-field is made zero, a search is made but no data is transferred. If the h-field is not equal to zero, data will be transferred until a completion signal is sent from the controlled module or until the number of words specified in the h-field has been transferred. This allows the programmer to avoid specifying in advance how many words are to be transferred. If a Computer is the controlling module and a completion signal is received before h is counted down to zero, one extra word is written into the core memory. This word is not tallied in the Accumulator. An extra word will not be written in this situation if a Buffer is the controlling module.

D. DIRECT DATA INPUT INSTRUCTIONS

The *Direct Data Input* instruction is available only to Buffers. Computers cannot use this instruction.

The format of the instruction is as follows:

26 21	20 11	10 1
f	g	h

Where:

The f-field contains the octal code 45.

The g-field specifies the Buffer address into which the first word will be transferred.

The h-field specifies the number of words to be transferred.

No search is performed with this instruction, and it is, therefore, not necessary to store a search word in the core memory.

E. SEARCH OUTPUT INSTRUCTIONS

The *Search Output* instruction directs a module that is connected to a Computer or a Buffer through the Central (or Interim) Exchange to record data transferred to it from the Computer, or Buffer, core memory. This instruction can only be used after a *Command Output* instruction has prepared the module being addressed to carry out the command. The *Command Output* instruction necessary before any particular type of module can accept a *Search Output* instruction is indicated in the chapter on that particular module.

The word format of a *Search Output* instruction is as follows:

26 21	20 11	10 1
f	g	h

Where:

The f-field contains the octal code 40.

The g-field carries the address of a search word G.

The h-field contains the number of words to be transferred.

The first step in transferring data from the Computer, or Buffer, is a *Command Output* instruction to the controlled module initiating the transfer of search data to the controlling module. This is followed by the *Search Output* instruction itself. The word in address g is compared with the search data, and when the two are equal information transfer starts with the word from address g+1 being the first transferred to the controlled module. If G, the word at address g, is zero, no search is made and data is transferred immediately.

At the completion of the data transfer, if a Computer was the controlling module, the A register will contain the quantity $1777_8 - (h-n)$, and the B register will contain the search word; if a

Buffer was the controlling module, the L register will contain the quantity h−n.

If g is made equal to zero and a Buffer is the controlling module, the contents of the W register will be used as the address of the search word, and the first transferred word will be taken from location w+1.

If h is zero and a Computer is the controlling module, a search is made but no data is transferred.

If h is zero and a Buffer is the controlling module, the contents of L are complemented and the result is taken as the number of words to be transferred.

F. DIRECT DATA OUTPUT INSTRUCTIONS

The *Direct Data Output* instruction is available only to Buffers. Computers cannot use this instruction. The format of the instruction is as follows:

26 21	20 11	10 1
f	g	h

Where:

The f-field contains the octal code 44.

The g-field specifies the Buffer address from which the first word will be transferred.

The h-field specifies the number of words to be transferred.

No search is performed with this instruction, and it is, therefore, not necessary to store a search word in the core memory.

G. REJECTED COMMANDS

Many types of programming errors such as illegal command codes, attempts to read from (or write on) tape before the previous operation has been completed, or attempts to connect to a busy module through the Exchange, will cause the controlled module or the Exchange to reject the command. Whenever a command is rejected the results are as follows:

If a Computer is the controlling module, bit I_{17} of the Alert register (the Computer's Program Error Indicator) is set to one and the next instruction is taken in sequence (unless the instruction is a *Command Output* with bit 20=1). This may be programmed to cause an alert. If bit 20 of the *Command Output* instruction is set to one, then the Computer will wait for the command to be executed and, if the command is one that is not a temporary delay like waiting for a tape to finish an operation, but is a permanent delay, like an illegal code, then the Computer will be stalled indefinitely and will have to be manually reset.

If a Buffer is in the self-instruction mode and one of its commands is rejected, then bit 18 of the Buffer test configuration (the Buffer's Program Error Indicator) will become one. If the error is an illegal operation code the Buffer leaves the self-instruction mode. Bit 18 can be tested with a Buffer Test instruction. Bit 20 has the same use in a Buffer as in a Computer except when a Buffer is commanding itself, in which case bit 20 has no meaning.

Chapter IV
BUFFER MODULE

A. GENERAL

The Buffer Module is a special purpose computing device which can operate either under Computer control as a controlled module or independently under its own control with the capability of controlling other modules or of executing an internally stored program. When controlling other modules, the Buffer uses the same *Command Output, Search Input,* and *Search Output* instructions that the Computer does, and, in addition has two other external instructions—*Direct Data Input* and *Direct Data Output.*

A Buffer Module contains two independent Buffers, each of which is capable of acting either as a *controlled* or a *controlling* device independent of the other. Each of these Buffers has a 1024-word core memory similar to the Computer's. Each Buffer has four registers associated with it.

The 26-bit N register is the exchange register. Words going to or from the Buffer core memory are stored temporarily in the exchange register. The N register is also the place where instructions are decoded.

The 10-bit L register counts the words being transmitted into or out of a Buffer during a data transfer.

The R register, or Read register, carries the address of the next word to be read from core memory. When in the self-instruction mode, the R register is used as a program counter, carrying the address of the next instruction.

The W register, or Write register, carries the core memory address into which the next word will be written. When in the self-instruction mode, the W register contains the Buffer core memory address into which or from which the next word will be transferred.

B. OPERATION UNDER COMPUTER CONTROL

1. GENERAL

A Computer can address a Buffer either directly with a *Command Output, Search Input,* or *Search Output* instruction or indirectly by using address 1777_8 in the g- or h-field of a Computer internal instruction. Indirect addressing is discussed in Chapter II. Direct addressing is discussed in the following paragraphs.

2. COMMAND OUTPUT INSTRUCTIONS

Figure 4.1 lists the *Command Output* instructions that a Buffer will accept. Some of these instructions the Buffer can give to itself (as indicated in the second column of the table). Buffer self-instructions are discussed in the next section, but for the sake of completeness each instruction in the table is discussed briefly in the following paragraphs:

In response to the *Send General Status* instruction, the Buffer puts the status information shown in Figure 4.2 on its data lines. (Whenever a Buffer has been in the self-instruction mode and stops, it automatically transmits general status.) In response to the *Send L, Send R,* or *Send W* command, the Buffer places the contents of the L, R, or W register on its data lines as shown in Figure 4.2. Note that each of the four commands mentioned previously in this paragraph causes the Buffer to transmit some form of status.

The *Reset Indicators* command causes any desired combination of the Branch, Program Error, or Parity Error Indicators to be reset, the combination depending on which of bit positions four, three, and one are made one. If all three of these bits are made zero in this instruction, a no-op occurs.

The *Exchange R and W* command causes the contents of the Read and Write registers to be exchanged.

The *Store W* command causes the contents of the W register to be stored in the core memory in the location specified by the current address in the R register (this will be the location directly after the one that the *Store W* command came from). The R register is incremented after W is stored. The contents of the W register are unchanged.

The *Buffer Link* command first causes the contents of the R and W registers to exchange and then proceeds exactly as in the *Store W* command.

The *Set Branch Indicator* command causes the Branch Indicator to be set. (The Branch Indicator can be used to remember which of two branches in a program was taken or is to be taken in the future. It can be tested at any time with a *Buffer Test* instruction.)

The *Set L, Set W,* and *Set R* instructions cause the Buffer to set its L, W, or R register to the value specified in bits 10 through 1 of the instruction.

Instruction Description	Source of Instruction S=Self C=Computer	Bit Positions													Time to Execute (Microseconds)	
		13	12	11	10	9	8	7	6	5	4	3	2	1	Computer Instruction	Self Instruction
SEND GENERAL STATUS	C	0	0	1	0	—	—	—	—	—	—	0	0	0	37	—
SEND L	C	0	0	1	0	—	—	—	—	—	—	0	0	1	37	—
SEND R	C	0	0	1	0	—	—	—	—	—	—	0	1	0	37	—
SEND W	C	0	0	1	0	—	—	—	—	—	—	0	1	1	37	—
RESET INDICATORS[1]	CS	0	0	1	1	0	—	—	—	—	X	X	—	X	37	21
EXCHANGE R AND W[2]	CS	0	0	1	1	1	—	—	—	—	X	—	0	1	37	21
STORE W[2]	S	0	0	1	1	1	—	—	—	—	X	—	1	0	—	49
BUFFER LINK[2]	S	0	0	1	1	1	—	—	—	—	X	—	1	1	—	49
SET BRANCH INDICATOR	CS	0	0	1	1	1	—	—	—	—	1	0	0	0	37	21
SET L[3]	CS	0	1	0	X	X	X	X	X	X	X	X	X	X	37	21
SET W[3]	CS	0	1	1	X	X	X	X	X	X	X	X	X	X	37	21
SET R[3]	CS	1	0	0	X	X	X	X	X	X	X	X	X	X	37	21
SWITCH CABLES AND SET R[3,4]	S	1	0	1	X	X	X	X	X	X	X	X	X	X	—	35
START SELF-INSTRUCTION AND SET R[3,4]	C	1	1	0	X	X	X	X	X	X	X	X	X	X	37	—
STOP SELF-INSTRUCTION AND SET R[3,4]	S	1	1	1	X	X	X	X	X	X	X	X	X	X	—	50

1. Bit 4 is the Branch Indicator, bit 3 the Program Error Indicator, and bit 1 the Parity Error Indicator. A one in any of these positions resets the associated indicator. Any combination of these bits is acceptable.
2. A one in bit position 4 in these instructions will set the Branch Indicator.
3. Bits 10 through 1 in these instructions are an arbitrary 10 bit number to be put into the indicated register.
4. A zero in bits 10 through 1 in these instructions will leave R unchanged.

Note: A Buffer command with a legal operation code and a configuration of bits in bit positions 1 through 13 other than one of those defined in the table above will be disregarded by the Buffer and the next instruction will be taken in sequence.

Figure 4.1. Buffer Command Output Instructions

Bit Position	General Status	L	R	W
1	0 Buffer Code	Contents of L Register	Contents of R Register	Contents of W Register
2	0			
3	0			
4	0			
5	—			
6	1 = Parity Error			
7	1 = Module Inoperative			
8	1 = Branch Indicator			
9	1 = Program Error			
10	—			
11	—	—	—	—
12	—	—	—	—
13	1 (mandatory)	1 (mandatory)	1 (mandatory)	1 (mandatory)

Figure 4.2. Buffer Status Formats

The two Buffers packaged in a Buffer Module, as previously indicated, are connected to the Central Exchange independently. The *Switch Cables with Associated Buffer and Set R* command causes the Buffers to reverse their cable connections and causes the R register in the Buffer that received the command to be set to the value in bits 10 through 1 of the instruction. Both Buffers must give this command before the switch will take place. When either Buffer has given this command, it will be indicated by a flip-flop register which the other Buffer can test with a *Buffer Test* instruction. Once a Buffer has given itself this command, it stalls until the other Buffer gives the command too so that, in general, one of the Buffers will be stalled for some time when this command is used.

With the *Start Self-Instruction and Set R* command the Computer causes the Buffer to go into the self-instruction mode. The command also causes the R-register to be set with the contents of bits 10 through 1 of the instruction.

The Buffer takes itself out of the self-instruction mode with the *Stop Self-Instruction and Set R* command. The command also causes the R-register to be set with the contents of bits 10 through 1 of the instruction.

3. SEARCH INPUT INSTRUCTIONS

The *Search Input* instruction the Computer can give the Buffer is the standard *Search Input* discussed in Chapter III. It will cause the transfer of data from the Buffer to the Computer.

The g-field of the instruction contains the address of a search word G. This search word must be zero since no search can be made. The first word of transferred data will be stored in address g+1. The h-field dictates the number of words to be transferred. If h=0, there is no transfer.

The Buffer R register contains the address of the first word in the Buffer to be transferred out. Succeeding words will be taken from r+1, r+2, etc.

When the transfer has been completed, the Buffer will automatically go into the self-instruction mode. It will put the quantity stored in register W into register R and then take its first instruction from the location addressed by the new quantity in R and succeeding instructions from r+1, r+2, etc. To stop the Buffer after the data transfer, the first instruction executed must be a *Stop and Set R* command.

4. SEARCH OUTPUT INSTRUCTIONS

The *Search Output* instruction the Computer can give the Buffer is the standard *Search Output* discussed in Chapter III. It will cause the transfer of a block of data from the Computer to the Buffer.

The g-field contains the address of a search word G. This search word must be zero, since no search can be made. The first word of transferred data will be taken from location g+1. (The contents of address g must be zero or the Computer will stall in an attempted search operation.) The h-field dictates the number of words to be transferred. If h=0, there is no transfer.

The W register in the Buffer contains the address into which the first transferred word will be stored. Succeeding words are stored in w+1, w+2, etc.

When the transfer is completed, the Buffer will automatically go into the self-instruction mode taking its first instruction from the location addressed by the R register and succeeding instructions from r+1, r+2, etc. As in the *Search Input* case, to stop the Buffer the first instruction executed must be a *Stop and Set R* command.

C. INDEPENDENT BUFFER OPERATION

1. SELF-INSTRUCTION MODE

A Buffer, when in the self-instruction mode, can either be giving instructions to itself or can be commanding other modules.

A Buffer enters the self-instruction mode either as a result of a *Start Self Instruction and Set R* command from the Computer or, automatically, after a data transfer between the Buffer and the Computer.

Whenever the Buffer connected to the cable whose IX connection address is 0001_2 is in the self-instruction mode, alert bit I_{11} in the Computer is zero, and when it is not in the self-instruction mode, I_{11} is one. The Buffer connected to the cable whose IX connection address is 0010_2 is similarly related to alert bit I_{10}.

A Computer cannot communicate with a Buffer which is in the self-instruction mode or which is transmitting any form of status. The execution of a *Test Jump* instruction which places the contents of the 13 data and 3 control lines in the Accumulator will cause the Buffer to stop the transmission of status (if it was being transmitted) following the transfer to the Accumulator. The Computer will then be able to communicate with the Buffer. It is not necessary to test the contents of the Accumulator after the transfer has occurred.

When a Buffer leaves the self-instruction mode or when a Computer has just connected to a Buffer, the first command the Computer gives the Buffer must be a *Test Jump* instruction bringing into the Accumulator the information on the control lines and the data lines of the Buffer. It is not necessary to test the information once it is in the Accumulator.

All of the commands which the Buffer can give to itself, except one, are listed above in the table in Figure 4.1 and are discussed in the paragraphs following the table. The exception is the *Buffer Test* instruction.

The *Buffer Test* instruction allows the programmer to test a number of conditions (these are listed in Figure 4.3) and to skip one instruction if the conditions tested are satisfied, or to go on to the next instruction if any of the conditions tested are not satisfied. A zero placed in any of the 20 least significant bits of the *Buffer Test* instruction, except 4 through 1 (see note 2 in Figure 4.3), means the Buffer should not skip unless the condition specified by that bit is met. A one placed in any of these 20 least significant (except 4 through 1) bits means the Buffer should not test the associated condition. Bits 26 through 21 of the instruction contain the octal f-field code 43. The execution time for this instruction is 28 microseconds.

A program error can occur only when a Buffer is communicating with a controlled module. The occurrence of an illegal operation code in a Buffer self-instruction program will cause the Buffer to stop self instruction.

When a Buffer is in the self-instruction mode, the occurrence of any f-field code other than 40_8, 41_8, 42_8, 43_8, 44_8, or 45_8 will cause the Buffer to stop self instruction.

Bit	Condition to be Tested[1]
20	Associated Buffer is ready to switch.
19	Branch Indicator has not been set.
18	There are no program errors, or sequence errors.
17	Not used.
16	There are no internal parity errors.
15	The L register = 0.
14	The connected module is sending status.
13-5[2]	The corresponding bit in the status message from the connected module is a zero.

1. When the bit of the instruction word associated with any condition is a zero and the condition is not satisfied, there will be no skip; otherwise, there will be a skip (except bits 4 through 1—see note 2).
2. Bits 13 through 1 are tested only when bit 14 of the command is zero. When bit 14 is zero, bits 4 through 1 of the command must be identical to the module identification code of the module that is supposed to be sending status, or there can be no jump.

Figure 4.3. Buffer Test Instruction

2. CONTROLLING OTHER MODULES

As far as the Buffer commanding other modules is concerned, the Buffer can give any *Command Output, Search Output,* or *Search Input* instruction to any module that the Computer can and will get the same results (except when $h=0$ as explained in the following paragraph) and, in addition, the Buffer can use the *Direct Data Input* and *Direct Data Output* instructions. Discussions of the use of these instructions appear in Chapter III. If the Buffer gives a *Command Output* instruction which is intended for a connected module, and no module is connected, the Buffer will stall and will have to be reset manually.

When $h=0$ in a *Search Output* or *Direct Data Output* command issued by the Buffer Module, the contents of the L register are complemented. The new quantity in L is used to dictate the number of words to be transferred. (This is useful in transferring an unknown number of words into the Buffer from some external module and then transferring them out. If the L register is set to 1777_8 for the incoming transfer, it will be counted down with each incoming word until, when n words have been transferred, $L=1777_8-n$. Complementing this gives $L=n$, the number of words to be transferred out.)

Chapter V
CENTRAL EXCHANGE

A. GENERAL

The Central Exchange is the central switching device for the RW-400. It is capable of servicing 16 controlling modules and 61 controlled modules. It has a magnetic core memory which allows the programmer to use symbolic addressing. It is so designed that only a Computer designated as Master Computer can load and unload the Symbolic File, thus controlling the connections permitted to the other controlling modules. The Master Computer has the capability of forcing disconnect of other controlling module-to-controlled-module connections, thus allowing it to forcibly implement priority requirements.

B. COMMAND OUTPUT INSTRUCTIONS

The Central Exchange (CX) will execute seven *Command Output* instructions. Five of the seven are given directly to the CX. The remaining two are given to the Master Coordinate of the CX and may be given only by the Master Computer, which must first request a connection to the Master Coordinate with the proper direct command and then must use the two master commands to address the Master Coordinate. The thirteen least significant bits of the seven commands are listed in Figure 5.1 and are discussed below. The thirteen most significant bits are discussed in Chapter III.

	Bit Position													Instruction Function
	13	12	11	10	9	8	7	6	5	4	3	2	1	
	Normal Commands													
1	0	0	0	0	0	-	X	X	X	X	X	X	X	Connect to symbolic coordinate X.
2	0	0	0	0	0	-	0	0	0	0	0	0	0	Self disconnect.
3	0	0	0	0	0	1	1	1	1	1	1	1	1	Connect to Master Coordinate.
4	0	0	0	1	1	-	X	X	X	X	X	X	X	Set alert X.
5	0	0	0	1	0	-	X	X	X	X	X	X	X	Clear alert X.
	Master Commands													
6	0	0	1	X	X	X	X	X	X	X	X	X	X	Set register R to address X.
7	0	0	0	0	0	0	0	0	0	X	X	X	X	Disconnect the controlling module at physical address X.

Figure 5.1. Central Exchange Commands

1. This command connects the requesting module to the controlled module at symbolic address X. Symbolic addresses 0 and 64 are not allowed. Symbolic address 255 is reserved for the Master Coordinate connection request. Be sure that if symbolic address 127 (1111111) is used, that the spare bit, bit 8, is zero. If the requested module is busy, the requesting module's program will go on unless bit 20 of the instruction is one, in which case it will wait for connection.

2. This command is used to request a disconnect from a controlled module. It is not necessary to use this if the controlling module is going to request

connection to another module or to set or clear an alert immediately after, because a new request will cause an automatic disconnect from the currently connected module.

3. This command causes the controlling module to be connected to the Master Coordinate. It should be used only by the designated Master Computer. Once this connection has been made commands 6 and 7 in the table may be used to load or unload the Symbolic File or to disconnect other modules. While the Master Computer has access to the Master Coordinate the Exchange will not accept commands from other controlling modules.

4. This command allows the controlling module to set any of 60 alerts with a symbolic address from 0 through 127 except that symbolic addresses 0 and 64 are not allowed. The alerts are discussed in more detail in paragraph E below.

5. This command allows the controlling module to clear any of 60 alerts with a symbolic address from 1 through 127 except that symbolic addresses 0 and 64 are not allowed.

6. This command allows the Master Computer, once connected to the Master Coordinate with Command No. 3 in Figure 5.1, to load or unload any part of the Symbolic File. With this command, register R on the Master Coordinate is set to the address in the CX core memory (the Symbolic File is stored in a 1024-word core memory) from which, or to which, data transfer is to start. (The organization of the Symbolic File is discussed in paragraph F below.) A *Search Output* or *Search Input* instruction may then be used to transfer the data. Note that a connection to the Master Coordinate is like a connection to any other controlled module and therefore commands No. 6 and No. 7 should have the function code 00 in bits 19 and 18.

7. This command allows the Master Computer, once connected to the Master Coordinate, to forcibly disconnect any controlling module by inserting its physical address (0-15) in the last four bits of the instruction code. This command must also be used by the Master Computer to disconnect itself from the Master Coordinate. To do this the Master Computer puts its own physical address in the four least significant bits of the instruction.

C. SEARCH INPUT INSTRUCTIONS

Command No. 6 in Figure 5.1 may be followed by either a *Search Input* or a *Search Output* instruction. The *Search Input* instruction should be the standard one with the h-field giving the number of words to be transferred and the g-field giving the address of the cell before the one where the first word of data is to be stored. The word at address g should be all zeros, otherwise the computer will wait for a search word which will never appear and thus will stall.

D. SEARCH OUTPUT INSTRUCTIONS

The *Search Output* instruction used to transfer data into the Symbolic File is the standard one discussed in Chapter III. The h-field specifies the number of words to be transferred and the g-field specifies the address preceding the one from which the first word is to be taken. The word in g should be zero since no search is to be made.

E. ALERTS

The Central Exchange provides for 63 fixed alerts and 60 programmable alerts. The fixed alerts are so arranged that one of them comes from each controlled module to a patch panel. At the patch panel they may be patched to any of the available alert lines (I_{13} - I_1) of any controlling Computer. The programmed alerts are 60 flip flops which may be set or cleared by putting their symbolic address in command No. 4 or No. 5 in Figure 5.1. These alerts may be patched to any Computer interrupt line with the patch panel. Of these 60 alerts, those with physical addresses 1-16, in addition to being able to be turned on with a *Command Output* instruction, are turned on automatically when the controlling module at physical address 0-15, respectively, has a format error in its command or if it tries to address an alert or a controlled module not assigned to it. (The Master Computer makes the assignments via the Symbolic File. See paragraph F for a discussion of the Symbolic File.) A format error occurs when an illegal command is sent to the CX.

In addition to these 60 programmable alerts, there are three more, with physical addresses 17, 18, and 19, which may be reset with a *Command Output* instruction but which are set by various error conditions in the system. These three alerts are designed to alert the Master Computer about conditions in the system and should be patched to its alert lines. The following conditions turn these alerts on:

A17	Parity Error in request. (There is no response to the requesting module and it will be stalled at the switch request instruction until the Master Computer intervenes and frees it.)
A18	Parity Error in Symbolic File.
A19	Program Error. (Any of the followlowing: format error in the request, requested module not assigned, no

Master Coordinate connection made after request for it was accepted, Master Coordinate connection made without request, format error in command to Master Coordinate.)

F. SYMBOLIC FILE

The Symbolic File is a 1024-word core memory, internal to the CX, which allows a Master Computer to store assignments for symbolic addressing of controlled modules and of alerts, and to store a history of the last request and action taken at each controlling module. The file is divided into 16 sectors, one associated with each controlling module. Words 0 through 63 are associated with the module at physical address 0, words 64 through 127 with the module at physical address 1 and so on. Each sector contains 64 26-bit words (plus 2 parity bits) organized as follows:

Bit	26	25 20	19 14	13	12 7	6 1
Word 63		(A127)	(X127)		(A63)	(X63)
Word 62		(A126)	(X126)		(A62)	(X62)
Word 61		(A125)	(X125)		(A61)	(X61)
Word 2		(A66)	(X66)		(A2)	(X2)
Word 1		(A65)	(X65)		(A1)	(X1)
Word 0	Alert Status Word			Connection Status Word		

Each word contains two six-bit physical connection addresses and two six-bit physical alert addresses. If a zero word is put into one of these cells it means that alert or connection control is not permitted at this address. Symbolic addresses 0 and 64 are forbidden for both connection requests and alerts because word zero (where these cells would be) is used for status. The formats of the two status configurations in word 0 are as follows:

Connection Status Configuration

13	12	11	10	9	8	7	6					1
–	–	X	X	X	X	X	X	X	X	X	X	X

Where:

Bits 13 and 12 are not used.

Bit 11 is one when a request is rejected because the module requested is busy.

Bit 10 is one when a request is rejected because the module requested is not assigned (the symbolic address is filled with zeros).

Bit 9 is one when the request is rejected because of a format error.

Bit 8 is one when the request is rejected because of a parity error in the request.

Bit 7 is one when the request is rejected because of a parity error in the Symbolic File.

Bits 6 through 1 contain the physical address of the last connection requested. If the last request was an alert request or if a self-disconnect was made, bits 6 through 1 will be zero.

The status word associated with a given coordinate is not altered by a master disconnect affecting that coordinate.

Alert Status Configuration

26	25	24	23	22	21	20	19					14
–	X	X	X	X	X	X	X	X	X	X	X	X

Where:

Bit 26 is not used.

Bit 25 is set to one when the request has been accepted and the alert cleared.

Bit 24 is set to one when the request has been accepted and the alert set.

Bit 23 is set to one when the request has been rejected because the alert requested is not assigned.

Bit 22 is set to one when the request is rejected because of a format error.

Bit 21 is set to one when the request is rejected because of a parity error in the request.

Bit 20 is set to one when the request is rejected because of a parity error in the Symbolic File.

Bits 19 through 14 contain the physical address of the last alert to be requested.

Chapter VI
INTERIM EXCHANGE

A. GENERAL

The Interim Exchange has been designed to serve as the central switching device for the RW-400 until the larger and more sophisticated Central Exchange is operational. The Interim Exchange will handle connections for three controlling modules (a Computer and the two halves of a Buffer Module or three Computers) and 15 controlled modules (two halves of a Buffer Module and 13 other modules, or 15 other modules).

A function called the Master Disconnect is included as part of the Interim Exchange. Its use is explained below.

B. COMMAND OUTPUT INSTRUCTIONS

The Interim Exchange will obey only two *Command Output* instructions: one is a request for a connection to a particular module (or disconnect from a module) and the other is a *Master Disconnect* command with which the module executing the command can cause the Master Disconnect device to force either of the other controlling modules to disconnect from the module to which it is currently connected.

The format for the most significant 13 bits of the *connect* command is discussed in Chapter III. The least significant 13 bits contain zeros in bit positions 13 through 5, and the code indicated in Figure 6.1 in bit positions 4 through 1. (Note that 0000 in bit positions 4 through 1 requests an unconditional disconnect from whatever module the requesting device is connected to. When this configuration is used in bit positions 4 through 1, the module category code in bit positions 17 through 14 must be all ones.)

To give a *Master Disconnect* command a controlling module must first request a connection to the Master Disconnect function using the usual *connect* instruction. Once connected to the Master Disconnect, the controlling module issues the *Master Disconnect Command Output* instruction, the 13 least significant bits of which are as follows:

13				9	8	7	6					1
0	0	0	0	0	X	X	1	1	0	1	1	0

where bits 8 and 7 indicate which of the three controlling modules is to be forced to disconnect:

$$\begin{array}{ccccc} 8 & \left.\begin{array}{c}0\\1\end{array}\right\}U & \left.\begin{array}{c}1\\0\end{array}\right\}V & \left.\begin{array}{c}1\\1\end{array}\right\}W \\ 7 & & & \end{array}$$

(U, V, and W are the arbitrary names given to the three connecting points on the Interim Exchange for controlling modules.) The other bits are mandatory as shown.

Module Requested	Bit Position			
	4	3	2	1
DISCONNECT	0	0	0	0
BUFFER A	0	0	0	1
BUFFER B	0	0	1	0
PAPER TAPE READER	0	0	1	1
PRINTER	0	1	0	0
DRUM MODULE	0	1	0	1
DISPLAY BUFFER	0	1	1	0
PERIPHERAL BUFFER	0	1	1	1
NOT USED	1	0	0	0
TAPE MODULE 1	1	0	0	1
TAPE MODULE 2	1	0	1	0
TAPE MODULE 3	1	0	1	1
TAPE MODULE 4	1	1	0	0
DRUM MODULE 2	1	1	0	1
DIGITAL CLOCK	1	1	1	0
MASTER DISCONNECT	1	1	1	1

Figure 6.1. Codes for Connecting to RW-400 Modules Through the Interim Exchange

The device code in bit positions 17 through 14 (refer to Chapter III) is not used in a *Master Disconnect* instruction.

When a Buffer in the self-instruction mode is forced to disconnect, it is removed from self-instruction.

When either a Computer or a Buffer is forced to disconnect, it will not be allowed to finish any data transfer it is engaged in before being disconnected.

C. STATUS

When a Computer or a Buffer connects to the Master Disconnect, a status message is available on the data lines which carries only one piece of information: if bit 7 has a one in it, the Master Disconnect device is in the panel mode. That is, it is out of the system and can only be operated manually from its control panel. It is possible for the Master Disconnect device to be in the panel mode while the Interim Exchange itself is in the system mode. The status configuration does not report the mode in which the Interim Exchange itself is operating.

Chapter VII

DRUM MODULE

A. GENERAL

The Drum Module DM-400 is designed to provide auxiliary storage space for Computer and Buffer Modules in the RW-400 Data Processing Central. The drum has a capacity of 8192 words. Data is transferred between a Drum Module and a Computer or Buffer at the rate of one word per 16.7 microseconds, or 60,000 words per second.

The drum is divided into eight bands numbered 0 through 7. Each band holds 1024 words numbered sequentially around the band from 0 to 1023 (1777_8). Words are stored on the drum as two 13-bit half words, each with a parity bit, one half-word right after the other on the drum, with a space following each pair of half words.

B. COMMAND OUTPUT INSTRUCTIONS

The Drum Module can perform three operations: *Search Read*, *Search Write*, and *Read Check*. The Drum Module requires a sequence of two commands to execute either of the first two operations—a *Command Output* instruction followed by either a *Search Input* or a *Search Output* instruction. The third operation, *Read Check*, requires only the correct *Command Output* instruction to cause its execution.

The least significant 13 bits of the five *Command Output* instructions which can address the Drum Module are shown in Figure 7.1. (*Command Output* instructions in general are discussed in Chapter III.)

Bits 13 through 11 in Figure 7.1 select the band to be read from or written into (0-7). Bits 10 and 9 are not used. Bit 8, when made a one will reset the parity error indicator and the monitor indicator. Bit 7, if it is a one, causes a search for a particular data word; if zero, a search for a particular address on the drum. A *Read Check* takes place if bit 6 is a one, regardless of the condition of bits 7 and 5. Bit 5 causes the Drum Module to read when it is zero and to write when it is one. Bits 4 through 1 are the mandatory Drum Module command code.

C. SEARCH INPUT INSTRUCTIONS

The first two *Command Output* instructions in Figure 7.1 must be followed by a *Search Input* instruction. (*Search Input* instructions are discussed in Chapter III.)

The g-field contains either the address of the search word or, in the case of a location search, a drum address. If G contains a drum address, it must be in the following format:

26 — 24	23 — 14	13 — 11	10 — 1
Band Number	Word Number	Band Number	Word Number

Both halves must be identical, and the band number must agree with that in the previous *Command Output* instruction or the controlling module will stall and will have to be reset manually.

When G is all zeros, data transfer starts immedi-

Instruction	Bit Position												
	13	12	11	10	9	8	7	6	5	4	3	2	1
LOCATION SEARCH, READ BLOCK OF DATA	X	X	X	–	–	X	0	0	0	0	0	1	0
DATA SEARCH, READ BLOCK OF DATA	X	X	X	–	–	X	1	0	0	0	0	1	0
LOCATION SEARCH, WRITE BLOCK OF DATA	X	X	X	–	–	X	0	0	1	0	0	1	0
DATA SEARCH, WRITE BLOCK OF DATA	X	X	X	–	–	X	1	0	1	0	0	1	0
READ CHECK	X	X	X	–	–	X	0	1	0	0	0	1	0

Figure 7.1. Command Output Instructions for the Drum Module

ately from the band specified in the previous *Command Output* instruction. This is true in both the *Location Search* and *Data Search* modes. In looking for a non-zero search word, when the search is satisfied, the first word transferred will be from drum location n+3 (where the search word is found at location n). In searching for a drum location (other than word zero, of band zero), when the search is satisfied, the first word transferred will be from drum location G—the search address.

The h-field specifies the number of words to be transferred. When h=0, no transfer takes place.

D. SEARCH OUTPUT INSTRUCTIONS

The third and fourth *Command Output* instructions in Figure 7.1 must be followed by *Search Output* instructions. *(Search Output* instructions are discussed in Chapter III.) The g-field contains either the address of a search word or, in the case of a location search, a drum address. If G contains a drum address, it must be in the form indicated above for *Search Input* instructions.

When G is all zeros, data transfer starts immediately to the band specified in the previous *Command Output* instruction. This is true in both the *Location Search* and *Data Search* modes. In searching for a search word, when the search is satisfied, the first word transferred will be written in drum location n+3 (where the search word is found at location n). In searching for a drum location (other than word zero of band zero), when the search is satisfied, the first word transferred will be to drum location G—the search address.

The h-field specifies the number of words to be transferred. When h=0 and a Computer is the controlling module, no transfer takes place. If a Buffer is the controlling module, L is complemented and the result is used as the number of words to transfer.

For all *data search* instructions, the search will start on the band specified in the *Command Output* instruction and will continue from band to band until either the whole drum has been searched, or the search word has been found. If the band in which the search word is located is unknown and a search of the whole drum is desired, it can be done most quickly if band 7 is specified in the *Command Output* instruction. (This is because the Drum Module's only way of knowing that it has read its entire contents is by reading word 32 of band zero twice. Since band zero follows band 7, the first complete band read will be band zero—there is no way to know where on band 7 the read heads will be when the search starts. The Drum Module will note that it has read word 32, go on and search the other seven bands and upon reaching word 32 of band zero again, it will stop. Naturally, if the search is satisfied before a complete search of the drum has been made, the search will end immediately.) If the search is unsuccessful, the Program Error Indicator is set.

There is no limit to the time that may be allowed to elapse between the sending of the *Command Output* instruction and a *Search Input* or *Search Output* instruction.

Whenever data is being transferred from or into a band of the drum and the end of the band is reached before data transfer is complete, the Drum Module will continue the transfer with the first word of the next higher band (this will be band zero when the transfer is started in band 7).

The Drum Module checks each incoming and each outgoing half-word of data for parity. If an error is detected, the Parity Error Indicator is set, and data transfer continues.

E. READ CHECK INSTRUCTION

The last instruction in Figure 7.1, *Read Check*, is used to cause the Drum Module to check its entire contents for parity errors. Once the operation is initiated with the proper *Command Output* instruction, the controlling module need not supervise the operation. The Drum Module will check each half-word in a sequence, starting with the band specified. If a parity check fails, the band counter is locked, the Parity Error Indicator is set on, and the *Read Check* operation is discontinued. *Read Check* is also discontinued by passing word 32 of band 1 twice or by giving a new command. A check of Drum Module status by the controlling module will determine if there was a parity error and, if so, in which band.

F. STATUS

The Drum Module sends status at all times, except when engaged in carrying out a command. Status is not transmitted during the time the drum is searching for a location or a word number. The data format on the information lines when status is being sent is as follows:

13 11	10	9	8	7	6	5	4 1
X X X	–	–	X	X	X	–	0 0 1 0

Where:

Bits 13 through 11 show the contents of the Band Counter (0-7).

Bits 10, 9, and 5 are not used.

Bit 8 is set to one when there is a monitor failure.

Bit 7 is set to one when the module is inoperative.

Bit 6 is set to one when there is a parity error.

Bits 4 through 1 are the mandatory Drum Module code.

Chapter VIII

DRUM MODULE B

A. GENERAL

Drum Module B (DM-B) provides auxiliary storage for the Computer and Buffer Modules of the RW-400. It is similar to the Drum Module DM-400 in many respects but the increase of storage space from 8192 words to 79,872 has necessitated some changes in programming.

The DM-B physically contains four magnetic drums but operationally they may be regarded as one drum with 96 bands (numbered from 0-95). Each band is divided into 13 blocks (numbered from 0-12) and each block contains 64 words (numbered from 0-63). The average access time to a word on the drum is 13 milliseconds. The data transfer rate is about 50,000 words per second.

B. COMMAND OUTPUT INSTRUCTIONS

The Drum Module B accepts two *Command Output* instructions: prepare to write and prepare to read. The format of the 13 least significant bits of these is as follows (the thirteen most significant bits of *Command Output* instructions are discussed in Chapter III):

13	12	11						5	4			1
X	X	X	X	X	X	X	X	X	X	X	X	X

Where:

Bit 13 when zero says read and when one says write.

Bit 12 when one resets the parity, monitor, and program error indicators.

Bits 11 through 5 contain the band number (0-95) in which reading or writing will take place.

Bits 4 through 1 contain the block number (0-12) in which reading or writing will take place.

Note that a number greater than 12 can be written in bits 4 through 1 although there is no block with a number greater than 12, and that a number greater than 95 can be written in bits 11 through 5 although the band numbers only go to 95. If a number greater than 12 is written in bits 4 through 1, or a number greater than 95 is written in bits 11 through 5, the command will be rejected.

There are four write inhibit switches which can be manually turned on at the DM-B. They prohibit writing in certain selected bands. Any attempt to write in a band for which the write inhibit switch has been turned on will be rejected and the Program Error Indicator will be turned on. A check of status will reveal which bands have been protected by write inhibit switches.

C. SEARCH INPUT INSTRUCTIONS

The *Search Input* instruction used to cause data to be read from the drum and transferred to a controlling module is the standard one discussed in Chapter III. The h-field specifies the number of words to be transferred. If h is greater than the number of words in the block, reading will continue from subsequent blocks of the same band and after block 12, from the following band. However, the Drum Module B will not read beyond the last block of band 95. Any attempt to do so will cause the Module to terminate the instruction and indicate a rejected command. The g-field of the instruction specifies the address of a search word in the Computer Memory. This search word is of the following format:

26												14
0	0	0	0	0	0	0	0	0	0	0	0	0

13							6					1
1	0	0	0	0	0	0	X	X	X	X	X	X

where bits 6 through 1 contain the word number (0-63) with which data transfer starts. The first word transferred goes into core location g + 1.

If the search word is all zeroes (including bit 13) data transfer starts in the band and block specified in the *Command Output* instruction with the first word of the block.

There are 4.5 milliseconds after the *Command Output* instruction in which to issue a *Search Input* instruction. If the delay is any longer than that, the command will be terminated and a program error will be indicated.

D. DIRECT DATA INPUT INSTRUCTIONS

The *Direct Data Input* instruction is the standard one discussed in Chapter III. It works like the all zero search word option of the Search Input except that the first word transferred goes into core location g instead of g + 1.

E. SEARCH OUTPUT INSTRUCTIONS

The *Search Output* instruction, used to cause Drum Module B to accept data from a controlling module and write it on the drum, is the standard *Search Output* instruction discussed in Chapter III. The h-field specifies the number of words to be transferred. If h > 64 writing will continue on subsequent blocks of the band and after block 12, on the next band. However, the drum will not go from the last block of band 95 to the first block of band zero. If it completes the last block of band 95 and still has more to write, it will terminate the instruction and indicate a program error. If $h \neq n\ 64$ (where n is any integer) then the word space after the last word will have an identifier written into it which will be read as a zero word with correct parity.

The g-field specifies the core location of a search word which must be all zeros. Data transfer starts from cell g + 1.

The first word is written in the band and block specified in the *Command Output* instruction in the zero-word position. There are 4.5 milliseconds after the *Command Output* instruction within which to issue a *Search Output* instruction. If the delay is any longer than that, the instruction will be terminated and a program error will be indicated.

F. DIRECT DATA OUTPUT INSTRUCTIONS

The *Direct Data Output* instruction works like the *Search Output* instruction except that the first word of transferred data comes from cell g instead of cell g + 1.

G. STATUS

The Drum Module B transmits status at all times except when executing an instruction.

The status format is as follows:

13	12	11	10	9	8	7	6	5	4			1
0	X	X	X	X	X	X	X	X	0	1	0	0

Where:

Bit 13 is a mandatory zero.

Bit 12 is one when a write inhibit switch has prohibited writing on bands 92 through 95.

Bit 11 is one when a write inhibit switch has prohibited writing on bands 88 through 91.

Bit 10 is one when a write inhibit switch has prohibited writing on bands 80 through 87.

Bit 9 is one when a write inhibit switch has prohibited writing on bands 72 through 79.

Bit 8 is one when there is a monitor failure.

Bit 7 is one when the Module is inoperative.

Bit 6 is one when the Module detects a parity error.

Bit 5 is one when there has been a program error.

Bits 4 through 1 are the mandatory Drum Module code.

Chapter IX

PERIPHERAL BUFFER

A. GENERAL

The Peripheral Buffer is designed to allow a Computer or Buffer Module to read from or write into a relatively slow-speed device, such as a Flexowriter or a Plotter, with a minimum loss of time. A Computer can, for example, unload information meant for a Flexowriter into a Peripheral Buffer at a much faster rate than the Flexowriter can accept it. The Peripheral Buffer can then transfer this information to the Flexowriter slowly while the Computer goes on to other tasks.

B. OPERATION

The Peripheral Buffer is a magnetic-drum type device like the Drum Module and the Display Buffer. The drum is divided into four bands, each containing a group of eight input sectors (loaded by an external module and in turn, unloaded by a connected Computer or Buffer), and a group of eight output sectors (loaded by a Computer or Buffer, and in turn, unloaded by an external module). Each of the 64 sectors has a capacity of 256 six-bit characters. Generally, an external module is connected to a specific input and a specific output sector. (In the case of a one-way module, like the Plotter, there is a connection to only one sector.) A Computer or Buffer can, of course, connect to any sector.

A set of 64 flip-flops—one for each sector of the drum—is used to indicate the availability of the sectors. A one in any of these Sector Availability flip-flops indicates that there is useful information in the sector it is associated with.

When a Computer Module or a Buffer Module loads an output sector of the Peripheral Buffer drum, it must make the last character of the message an end-of-message code (17_8) and it must follow this with a command to set the Availability flip-flop for that sector to one. This flip-flop (when set to one) tells the external module to get data from the Peripheral Buffer. The end-of-message code serves two functions: it tells the external module that it has received the entire message, and it causes the Availability flip-flop to be reset (set back to zero).

For information moving the other way, the situation is slightly different. When an external module loads an input sector, either the end-of-message code or the 256th character (if the message happens to be exactly that long) sets the Sector Availability flip-flop. This sets bit 13 of the general status format to one and, if a Computer is connected to the Peripheral Buffer, sets alert bit I_{18} to one. After the controlling module unloads the Peripheral Buffer, it must reset the Sector Availability flip-flop to zero with the *Set/Reset Availability Flip-Flop* command. If it is desired to reset the flip-flop without reading the message, 17 milliseconds must be allowed to elapse before the *Set/Reset Availability Flip-Flop* command will be effective.

An external module cannot write into the Peripheral Buffer when the Availability flip-flop for the sector to which it is tied is set to one. However, either a Computer Module or a Buffer Module can write into any sector even if the sector has information in it already and the Availability flip-flop is set. Caution should be taken to avoid this.

The Peripheral Buffer checks each half-word of data transferred to or from itself for odd parity. When it detects a parity error in a half-word it is preparing to transfer to a Computer or Buffer, it sets its Internal Parity Error Indicator but continues to transfer data. When it detects a parity error in a word coming from a Computer or a Buffer, it sets its External Parity Error Indicator and continues to accept the data.

An RW-400 word from a Computer or a Buffer, to be put into the Peripheral Buffer, must have the following format:

26	25 20	19 14	13	12 7	6 1
0	Character	Character	0	Character	Character

Characters received from external devices will be stored on the drum in this same format with the first character received placed in the most significant bit positions, the second character received placed in the next most significant bit positions, and so on.

Note that a 00_8 character, if stored in an output sector, will be ignored when the information in that sector is transmitted to an external device.

If an external module transmits a 00_8 to the Peripheral Buffer, the character will be accepted unless it has a parity error, in which case it will be ignored (no parity error indication will be set).

C. COMMAND OUTPUT INSTRUCTIONS

The Peripheral Buffer accepts five *Command Output* instructions. (The *Command Output* instruction in general is discussed in Chapter III.)

Instruction	Bit Position												
	13	12	11	10	9	8	7	6	5	4	3	2	1
SEND STATUS OF INDICATED GROUP (0–7)		Group											
	X	X	X	X	0	–	–	1	0	0	0	1	0
SEND GENERAL STATUS	X	X	X	X	1	–	–	1	0	0	0	1	0
SET/RESET AVAILABILITY FLIP-FLOP		Group			Sector								
	X	X	X	X	X	X	X	1	1	0	0	1	0
PREPARE TO READ FROM INDICATED BAND (0–3)		Band											
	X	X	X	–	–	–	0	0	0	0	0	1	0
PREPARE TO WRITE IN INDICATED BAND (0–3)		Band											
	X	X	X	–	–	–	0	0	1	0	0	1	0

Figure 9.1. Command Output Instructions for the Peripheral Buffer

The 13 least significant bits of each *Command Output* instruction are shown in Figure 9.1. A short discussion of each instruction follows.

In all five of the commands in Figure 9.1, bit 13, when set to one, causes the Internal and External Parity Error Indicators, and Monitor Error Indicator, to be reset. (The condition of these indicators is indicated in the general status format which is discussed below.) In all five of the commands bits 4 through 1 are the mandatory Peripheral Buffer code.

In the *Send Status of Indicated Group* command, bits 12 through 10 identify the group whose status is desired. (Groups 0 and 1 are in band zero, groups 2 and 3 are in band 1, 4 and 5 in band 2, and 6 and 7 in band 3. The odd-numbered groups are output sectors and the even-numbered groups are input sectors.) Bits 9, 6, and 5 are mandatory as shown. Bits 8 and 7 are not used.

In response to this command, the Peripheral Buffer puts a half-word on its output information lines and leaves it there until it receives a further command. The format of this half-word is as indicated in Figure 9.2.

In the *Send General Status* command, bits 12 through 9 and bits 6 and 5 are mandatory as shown. Bits 8 and 7 are not used.

In response to this command, the Peripheral Buffer puts a half-word on its output information lines with the format shown in Figure 9.2. The Peripheral Buffer sends general status continuously when not performing some other activity, unless it has been left in the *Send Status of Indicated Group* mode; in which case, it will continue to send group status.

In the *Set/Reset Availability Flip-Flop* command, bits 12 through 10 identify the group which has in it the sector whose Availability flip-flop is to be set or reset. Bits 9 through 7 identify the sector whose Availability flip-flop is to be set or reset. Bits 6 and 5 are mandatory as shown. If the sector being addressed with this command is an input sector, it will be reset to zero; if it is an output sector, it will be set to one.

In the *Prepare to Read from Indicated Band* command, bits 12 and 11 carry the number of the band to be read from (0-3). Bits 10 through 8 are not used. Bits 7 through 5 are mandatory as shown. This command prepares the Peripheral Buffer to read from a specified band. It must be followed by a *Search Input* instruction.

In the *Prepare to Write in Indicated Band* command, bits 12 and 11 carry the number of the band to be written into (0-3). Bits 10 through 8 are not used. Bits 7 through 5 are mandatory as shown. This command prepares the Peripheral Buffer to write in a specified band. It must be followed by a *Search Output* instruction.

D. SEARCH INPUT INSTRUCTIONS

As indicated above, the *Prepare to Read from Indicated Band* instruction must be followed by a *Search Input* instruction. (Refer to Chapter III for a general discussion of *Search Input* instructions.) There is no time limit within which a

Bit Position	General Status		Group Status	
	Bit*	Meaning	Bit*	Meaning
13	1	One of the Input Availability flip-flops is set to 1.	1	The Availability flip-flop of one of the sectors of the group is set to 1.
12	—	Not used.	1	The Availability flip-flop of sector 7 is set to 1.
11	—		1	The Availability flip-flop of sector 6 is set to 1.
10	—		1	The Availability flip-flop of sector 5 is set to 1.
9	—		1	The Availability flip-flop of sector 4 is set to 1.
8	1	There has been a monitor error.	1	The Availability flip-flop of sector 3 is set to 1.
7	1	The module is inoperative.	1	The Availability flip-flop of sector 2 is set to 1.
6	1	There has been an internal parity error.	1	The Availability flip-flop of sector 1 is set to 1.
5	1	There has been an external parity error.	1	The Availability flip-flop of sector 0 is set to 1.
4	0	Mandatory Drum command code.	0	Mandatory Drum command code.
3	0		0	
2	1		1	
1	0		0	

*In bits 13 through 5 in these columns, a zero means that the condition indicated under 'Meaning' does not exist.

Figure 9.2 Peripheral Buffer Status Formats

Search Input instruction must follow the *Prepare to Read from Indicated Band* instruction, but no other instruction should be given to the Peripheral Buffer between the two.

The search word in G, addressed by the g-field of the *Search Input* instruction, must have the following format:

26	25 24	23 14	13	12 11	10 1
1	Band Number	Word Number	1	Band Number	Word Number

Both halves of the word must be identical, and the band number specified must be the same as that in the preceding *Command Output* instruction. If these conditions are not met, the controlling module will stall and will have to be reset manually. The first word to be transferred is the word in the drum location specified in G. It is stored in core location g+1. (See Figures 9.3 and 9.4 for drum locations.)

The number of words to be transferred, as specified by h, may have any value from 0 to 1777_8.

However, if during the transfer an end-of-message character is encountered, the word in the Computer core memory following the one where the end-of-message character is stored is destroyed and the instruction is terminated. If the end of the band is reached during a transfer, the next word is taken from location zero of the same band.

E. SEARCH OUTPUT INSTRUCTIONS

The *Prepare to Write in Indicated Band* instruction must be followed by a *Search Output* instruction. (Refer to Chapter III for a general discussion of *Search Output* instructions.) There is no time limit within which the *Search Output* instruction must follow the *Prepare to Write* instruction, but no other instruction should be given to the Peripheral Buffer between the two.

The search word in G, addressed by the g-field of the *Search Output* command, should have the same format, and has the same restrictions, as it does for a *Search Input* command. The first word to be transferred is taken from core location g+1 and is stored in the drum location specified in G. (See Figures 9.3 and 9.4 for drum locations.)

The number of words to be transferred, specified by h, may have any value from 0 to 1023. Normally, it will not exceed 64. If the end of the band is reached while data is being transferred, the next word will be written into location zero of the same band.

	Band / Sector	0	1	2	3
INPUT	0	0000-0063	0000-0063	0000-0063	0000-0063
	1 00	100 0064-0127	0064-0127	0064-0127	0064-0127
	2 00	200 0128-0191	0128-0191	0128-0191	0128-0191
	3 00	300 0192-0255	0192-0255	0192-0255	0192-0255
	4 00	0256-0319	0256-0319	0256-0319	0256-0319
	5 00	0320-0383	0320-0383	0320-0383	0320-0383
	6 00	0384-0447	0384-0447	0384-0447	0384-0447
	7 00 8	0448-0511	0448-0511	0448-0511	0448-0511
OUTPUT	0 00	0512-0575	0512-0575	0512-0575	0512-0575
	1 00	0576-0639	0576-0639	0576-0639	0576-0639
	2 00	0640-0703	0640-0703	0640-0703	0640-0703
	3 00	0704-0767	0704-0767	0704-0767	0704-0767
	4 00	0768-0831	0768-0831	0768-0831	0768-0831
	5 00	0832-0895	0832-0895	0832-0895	0832-0895
	6 00	0896-0959	0896-0959	0896-0959	0896-0959
	7 00 [illegible]	0960-1023	0960-1023	0960-1023	0960-1023

Figure 9.3. Assignment of Peripheral Buffer Words (Decimal Notation)

	Band / Sector	0	1	2	3
INPUT	0	0000-0077	0000-0077	0000-0077	0000-0077
	1	0100-0177	0100-0177	0100-0177	0100-0177
	2	0200-0277	0200-0277	0200-0277	0200-0277
	3	0300-0377	0300-0377	0300-0377	0300-0377
	4	0400-0477	0400-0477	0400-0477	0400-0477
	5	0500-0577	0500-0577	0500-0577	0500-0577
	6	0600-0677	0600-0677	0600-0677	0600-0677
	7	0700-0777	0700-0777	0700-0777	0700-0777
OUTPUT	0	1000-1077	1000-1077	1000-1077	1000-1077
	1	1100-1177	1100-1177	1100-1177	1100-1177
	2	1200-1277	1200-1277	1200-1277	1200-1277
	3	1300-1377	1300-1377	1300-1377	1300-1377
	4	1400-1477	1400-1477	1400-1477	1400-1477
	5	1500-1577	1500-1577	1500-1577	1500-1577
	6	1600-1677	1600-1677	1600-1677	1600-1677
	7	1700-1777	1700-1777	1700-1777	1700-1777

Figure 9.4. Assignment of Peripheral Buffer Words (Octal Notation)

Chapter X
DISPLAY BUFFER

A. GENERAL

The Display Buffer is used to provide recirculating digital information for external systems, such as cathode-ray tube displays. The Display Buffer can serve as many as eight external displays simultaneously with 1024 digital words each. It takes 67 milliseconds to read 1024 words to an external device, or, in other words, any given word in a display occurs 15 times per second.

The drum is arranged into eight bands—one for each output sector. The information on any band is constantly fed out as long as there is an external unit connected to that band. Words are placed on the band as two adjacent half-words. The pairs of half-words are interlaced—word one being four spaces from zero and word 256 being right after word zero. Thus, it takes four revolutions of the drum to completely load or unload a band. The drum speed is 3600 rpm.

B. OPERATION

The Display Buffer will perform only one operation—to write information on its drum. The commands for this operation can come from either a Computer Module or a Buffer Module. A combination of a *Command Output* instruction and a *Search Output* instruction is required to cause the operation to be carried out.

The 13 least significant bits of the *Command Output* instruction are as follows (refer to Chapter III for a general discussion of *Command Output* instructions):

13 11	10	9	8	7	6	5	4 1
Band Number	–	–	X	–	–	1	0 0 1 1

Where:

Bits 13 through 11 name the band (0-7) to be written on.

Bits 10, 9, 7, and 6 are not used.

Bit 8 causes the Parity Error Indicators to be reset when it is a one.

Bit 5 causes the Display Buffer to search and write when it is one.

Bits 4 through 1 are the mandatory Display Buffer code.

C. SEARCH OUTPUT INSTRUCTION

The g-field of the *Search Output* instruction must contain the address of a search word which has this format:

26 24	23 14	13 11	10 1
Band Number	Word Number	Band Number	Word Number

Both halves must be identical, and the band number must agree with that specified in the preceding *Command Output* instruction or the controlling module will stall and will have to be reset manually. The first word of data transferred will be stored in the location specified in the search word.

If the search word, G, is equal to zero, data transfer starts immediately to the band which was specified in the previous *Command Output* instruction.

The h-field of the instruction specifies the number of words to be transferred. If h=0, there is no transfer of data.

The time that may be permitted to elapse between the *Command Output* instruction and the *Search Output* instruction is not critical.

If the Display Buffer detects a parity error in incoming data, it will set its External Parity Error Indicator but data transfer will continue. If it detects a parity error in the information being transmitted out to one of the display units, it will set its Internal Parity Error Indicator and go on transmitting.

D. STATUS

The Display Buffer transmits status on its information lines, whenever it is not carrying out an instruction. The format of the status information is as follows:

13 11	10	9	8	7	6	5	4 1
– – –	X	X	X	X	X	X	0 0 1 1

Where:

Bits 13 through 11 are not used.

Bit 10 is set to one when there is an internal parity error in bands 4 through 7.

Bit 9 is set to one when there is an internal parity error in bands 3 through 0.

Bit 8 is set to one when there is a Circuit Monitor error.

Bit 7 is set to one when the module becomes inoperative.

Bit 6 is set to one when there is an internal parity error.

Bit 5 is set to one when there is an external parity error.

Bits 4 through 1 are the mandatory Display Buffer code.

Chapter XI

TAPE MODULE

A. GENERAL

The Tape Module (TM) provides the RW-400 with a large volume storage medium. The module includes an Ampex FR-300 tape transport, transport-control electronics, logical and clock circuits, amplifiers, and power supplies.

Data is recorded on 2500-foot reels of mylar tape. A row of 16 bits is recorded across the width of the tape: a 13-bit half-word, a parity bit, and two clock bits. Two hundred and seventy-eight of these rows (139 full RW-400 words) are recorded per inch of tape. The words are grouped into blocks. The size of a block is limited to 1023 words because that is the number of words a controlling module can transfer in one group. Each block is separated from the previous one by approximately 0.6 inches of blank tape called the interblock gap. The first word of each block may be used as an identifying block number.

A number of blocks may be grouped together as a file. A special end-of-file mark is used to separate one file from another. This mark is a zero-length block (i.e., just the keys that go before and after a block written back to back) and is most conveniently written by giving the Tape Module a *Command Output* instruction to prepare to write and then by following this with an instruction that involves tape motion and that has a one in bit 20. If the end-of-file is the last operation, the controlling module can merely disconnect and the TM will write the end-of-file and stop.

B. COMMAND OUTPUT INSTRUCTIONS

A Tape Module may be caused to perform various operations by making variations in the combinations of the 13 least significant bits of the *Command Output* instruction. The meaning of each of these bits is given below. This is followed by a discussion of the limitations on making these combinations.

The 13 least significant bits of the *Command Output* instruction are as follows:

13	12	11	10	9	8	7	6	5	4	3	2	1
X	X	X	X	X	X	X	X	X	X	0	0	1

Where:

Bit 13, when one, causes the tape to move at high speed (135 inches per second); when zero, causes the tape to move at normal speed (90 inches per second).

Bit 12, when one, puts the Tape Module in the *full tape* mode; when zero, puts the Tape Module in the *single block* mode.

Bit 11, when one, stops the tape.

Bit 10, when one, starts the tape.

Bit ~~9~~ 7, when one, causes the tape to move backwards.

Bit 8, when one, causes the tape to move forward.

Bit ~~7~~ 9, when zero, puts the Tape Module into the *location search* mode (searching only the first word of each block); when one, puts the Tape Module into the *data search* mode (searching each word of a block including the first).

Bits 6 and 5 indicate either read (00), erase (01), replace (10), or write (11).

Bit 4, when one, resets the error indicators.

Bits 3 through 1 are the mandatory Tape Module code. (001).

In order to read, write, or replace, the *Command Output* instruction should be followed by an appropriate instruction to cause data transfer such as *Search Input* or *Search Output.* There are time limits within which data transfer type instructions must follow *Command Output* instructions. These limits are discussed below in the section on timing.

The Tape Module cannot erase, replace, or write in the reverse direction. Any command to perform one of these operations will be rejected. It can perform any read backwards operation, and therefore can move back any desired number of blocks or files.

When commanding a tape to move where no data transfer is required, the Tape Module should be in the read mode (bits 6 and 5 should be 00).

When a tape is commanded to move in the *single block* mode, it will move to the next end-of-block gap if a read command is being executed, or it will write one block if a write command followed by an output instruction is being executed. When a tape is commanded to move in the *full tape* mode, if the Tape Module is reading or searching, the tape will move to the interblock gap following the next end-of-file mark (unless commanded to stop); if the Tape Module is writing, it will continue until it receives a *Stop* command or until it writes an end-of-file or comes to the end-of-tape mark. When reading in the *single block* mode, an end of file mark is regarded as a block.

If a command is issued before the one previous to it has been completed, it will be rejected. (If bit 20 is one, the command will wait for the previous one to be executed and will then be accepted.)

When a tape is moving at high speed, the Tape Module cannot read, write, replace, or erase. In addition it will reject all commands including *Stop* and *Reset Errors.* It cannot recognize end-of-block gaps or end-of-file marks and will, therefore, move the tape to its physical beginning or end. A command to rewind will be accepted any time the tape is not in motion even if the tape is already rewound.

When a tape is at its far end, any command requesting the tape to move forward will be rejected. (If bit 20 of the instruction is one, the controlling module will stall and will have to be reset manually.)

A command to stop will be accepted independent of the setting of bits 5 and 6, but will not be accepted when the tape is moving at high speed. If the command to stop is issued during the reading or writing of a block or within 300 microseconds after status appears at the end of the block, the tape will stop after that block. If the command comes later than that, the tape will stop after the next block. If the keys are misread coming out of a block, status will appear 300 microseconds late and a *Stop* command issued after this status appears will not stop the tape until the following interblock gap. Whenever these keys are misread a sequence error is indicated in status.

If bits 9 and 8 are both zero in an instruction, the tape will move the way it did in the previous instruction.

If bits 10 and 11 are both zero in an instruction, the tape motion will remain the same (stopped or moving) as it was before the command was given.

Bit 4, which resets the error indicators, may be set to one in combination with any other command.

When replacing, the new block must be equal to or shorter in length than the old. However, if it is too long it will not be permitted to destroy the subsequent block on the tape; it will be terminated when space runs out and sequence errors will be indicated. If the new block is just slightly longer than the old it will be written correctly. No matter how short the new blocks are, only one new block can be put in the space previously occupied by another block.

When an erase command is given in the forward direction, it will never be accepted in the *full tape* mode. When given in the *single block* mode, it will erase approximately 9 inches of tape and then stop regardless of the length of any block on the tape. This area of blank tape will be treated as a long interblock gap. No subsequent command will be accepted while an erase is being done.

When searching in the *single block* mode, if the search is unsuccessful, the Program Error Indicator will be set, the *Search Input* instruction will be ignored and the program will be continued. When searching in the full tape mode, if an end-of-file mark is reached with the search incomplete, a program error is indicated, the *Search Input* instruction that is in the controlling module instruction register is terminated, and the next instruction is taken in sequence.

If a *Command Output* instruction is given to a stopped tape and the instruction does not have a one in the start bit (bit 10), the instruction will be accepted but the tape will not move. A *Search Input* or *Search Output* or *Direct Input* or *Direct Output* instruction following this will cause the controlling module to stall and it will have to be reset manually.

A tape cannot be written on or erased unless a Write-Enable ring is on the tape reel. Any command requesting a write, overwrite, or erase operation given to a tape without a Write-Enable ring on the tape reel will result in a rejected command.

The following combinations of bits in a command will never be accepted: bits 9 and 8 both one, or bits 11 and 10 both one. In addition, the Tape Module will not accept a command to move the tape at high speed while in the single-block mode.

If a controlling module is disconnected from a Tape Module while the Tape Module is performing an operation, the Tape Module will complete the operation and stop.

C. SEARCH INPUT AND DIRECT DATA INPUT INSTRUCTIONS

The *Search Input* or *Direct Data Input* instruction that may follow all read type *Command Output* instructions is the standard *Search Input* or

Direct Data Input instruction discussed in Chapter III. The first word of transferred data goes into core location g + 1 in the *Search Input* instruction or location g in the *Direct Data Input* instruction. In the *Search Input* instruction, if the search word is non-zero, the first word transferred will be the second word after the search word in the *data search* mode or the word directly after the block number in the *location search* mode. In the *Direct Input* or in the *Search Input* instruction if the search word is zero, the first word transferred will be the first word of the block (block number) whether in the *location search* or the *data search* mode.

The h-field of the *Data Input* instruction contains the number of words to be transferred. If h = 0, there will be no transfer. If h = n, where n is the number of words in the block, then n words will be transferred. If h > n, transfer will continue until an end-of-block gap is encountered. In this case (h > n), if the controlling module is a Computer, an all zero word will be written into the memory cell following that in which the last data word transferred was stored.

D. CONDITIONAL SEARCH INPUT INSTRUCTIONS

The Conditional Search Input instruction that may follow all read type *Command Output* instructions may be given to the Tape Module only by a Computer Module. It has the same results and restrictions as the *Search Input* instruction as long as the search is satisfied by the first transmitted identification word (this identification word will be the block number). If the first word examined does not satisfy the search, the Conditional Tape Read Indicator will be turned on and the controlling module will be released. If the first thing read from the tape is an end-of-file mark, the Computer thinks the search was satisfied and the Conditional Tape Read Indicator is not turned on. The octal f-field code for this instruction is 43_8.

Note: The Command Output instruction preceding a Conditional Search Input instruction must put the Tape Module in the location search mode or the Conditional Tape Read Indicator will be turned on, but the search will act like a normal search of data.

E. SEARCH OUTPUT AND DIRECT DATA OUTPUT INSTRUCTIONS

The *Search Output* or *Direct Data Output* instruction which must follow all write or replace *Command Output* instructions is the standard *Search Output* and *Direct Data Output* discussed in Chapter III. In the *Search Output* instruction the contents of address g (i.e., G) must be all zeros or the controlling module will stall and have to be reset manually. No searching is possible when the write mode has been selected.

If h = 0 and a Computer is the controlling module, the instruction is ignored and the Tape Module accepts the next instruction. If h = 0 and a Buffer is the controlling module, the contents of the L register is complemented and the results are used to specify the number of words transferred. If the contents of the L register is 1777_8 the instruction is ignored and the next instruction is accepted.

F. STATUS

The Tape Module transmits status whenever any of the following conditions prevail: the tape is moving at high speed, the Tape Module is reading, writing, or searching and is in an interblock gap, or the Tape Module is idle. Following a write operation there is a short period (4.3 milliseconds) when there is no status. In these cases, if a Computer is controlling, check alert bit 18 to tell when status is available; if a Buffer is controlling, use the *Buffer Test* instruction and test bit 14.

The status format is as follows:

13	12	11	10	9	8	7	6	5	4			1
X	X	X	X	X	X	X	X	X	0	0	0	1

Where:

A one appears in bit 13 whenever the read head is in an interblock gap. (Bit 13 stays one when tape is moving at high speed.)

A one in bit 12 indicates that the Tape Module is stopped, and ready to accept any allowable command.

A one in bit 11 indicates that the physical beginning-of-tape or physical end-of-tape has been reached.

A one in bit 10 indicates that the physical end-of-tape is about to be reached. After bit 10 is made one, it will remain in that state until the tape motion has been reversed and the sensing point has been passed in the backward direction. While bit 10 is one, it is possible to write or replace one more short block before the physical end of tape is reached (bit 11 of status). This block should not be more than 100 words in length to ensure its proper recording. If the block is too long or if more than one block is written, the Tape

Module will stop at the end of tape, but will finish the transfer, without actually recording the information. Status will indicate a sequence error.

A one in bit 9 indicates that the Tape Module has reached an end-of-file mark on the tape. Bit 9 will stay one during the entire gap following the end-of-file even if the Tape Module stops, and will go to zero again when the next block is reached.

A one in bit 8 indicates a sequence error. (A sequence error occurs when the tape starts or stops reading from somewhere other than an interblock gap, or when the read heads misread the machine-written keys at the beginning or end of a block.)

A one in bit 7 indicates that the module is inoperative.

A one in bit 6 indicates a transient error.

A one in bit 5 indicates a data parity error.

Bits 4 through 1 are the mandatory Tape Module code.

If a sequence error, transient error, or parity error is detected during an operation, the Tape Module will continue the operation and then display the error in status during the gap.

G. TIMING

Note: All times given here are times within which the programmer should work. Actual design times are for the most part greater but cannot be safely relied upon due to individual component differences.

The maximum time available between a *Command Output* instruction commanding the Tape Module to prepare to write (in the *single block* or *full tape* mode) and a *Search Output* or *Direct Data Output* instruction is 5.9 milliseconds. If a *Search Output* or *Direct Data Output* instruction follows later than this, an end-of-file mark may be written stopping the tape. If no *Search Output* or *Direct Output* follows this command an end-of-file will be written and the tape will stop. If the Tape Module is in the *full tape* mode, there are 10.2 milliseconds available after each succeeding *Data Output* instruction in which to issue another instruction (except a *Stop* command which must be issued within 4.3 milliseconds after the data transfer). If a period of time greater than this is allowed to elapse, an end-of-file mark may be written stopping the tape.

The maximum time available between a *Command Output* instruction commanding the Tape Module to prepare to replace (in the *single block* or *full tape* mode) and a *Search Output* or *Direct Data Output* instruction is 4.3 milliseconds. If there is no *Search Output* or *Direct Data Output* instruction within 4.3 milliseconds of the *Command Output* instruction, an end-of-file mark may be written stopping the tape. If the Tape Module is in the *full tape* mode, there are 10.2 milliseconds available after each succeeding *Search Output* or *Direct Data Output* instruction (assuming the new block is the same length as the old) in which to issue another *Search Output* or *Direct Data Output* instruction (a *Stop* command must be issued within 4.3 milliseconds of the completion of an original-length transfer). If a period of time greater than this is allowed to elapse, an end-of-file mark may be written.

The time available between a *Command Output* instruction commanding the Tape Module to prepare to read (in the *single block* or the *full tape* mode) and a *Search Input, Direct Data Input,* or *Conditional Search Input* instruction is 8.6 milliseconds. (This time applies to reading in the forward direction; when reading backwards the time is only 1 millisecond.) If more than the stipulated amount of time is allowed to elapse, the search data may be missed. If the search data is missed (or the search is not satisfied for any other reason) and the Tape Module is in the *single block* mode, the Program Error Indicator will be set, the instruction will be terminated, and the program will go on. If the search data is missed and the Tape Module is in the *full tape* mode, the tape will move to the next end-of-file mark or if none, to the end of tape and stop. Then the Program Error Indicator is set, the instruction terminated, and the next instruction taken in sequence. If. however, the *Search Input* instruction is used and G = 0, or if the *Direct Data Input* is used, data transfer will start with the first block read.

If the Tape Module is in the *full tape* mode, 10.2 milliseconds are available after each block has been read in which to issue another *Data Input* instruction.

After the Tape Module has completed an instruction in the *single block* mode, there is still the possibility of giving a new *Command Output* instruction before the tape commences stopping. The time available to give a new *Command Output* instruction to a still-moving tape after completion of a previous read instruction is 300 microseconds. (The previous instruction is not considered finished until the entire block has been read, even though the number of words requested by the controlling module is less than the number of words in the block.) The time available to give a new *Command Output* instruction to a still-moving tape after completion of a previous write instruction is 4.3 milliseconds; and after a previous replace instruction 4.3 milliseconds (after a data transfer of the original length). This technique may only be used to follow an instruction with another of the same type. That is, read must follow read, write must follow write, and so on—any violation of this order will result in a rejected command. If bit 20 of the instruction is one, there will be a delay of 2.5 milliseconds and then the command will be accepted.

Between 150 and 250 milliseconds are necessary for the mechanization of a reversal of tape direction. After a *Command Output* instruction requesting a reversal has been accepted, a command to transfer data will not be accepted until this delay has elapsed. A stopped tape can be given a command calling for a tape reversal without any tape motion. The program can then go on to other things for 250 milliseconds, and come back and start the tape in the new direction.

There is a 2.5 millisecond delay between the execution of a stop (whether commanded, or caused by reading an end-of-file mark) and the execution of a succeeding start command.

Tape motion cannot occur for approximately six seconds following the completion of a high-speed rewind or fast forward movement of the tape.

Chapter XII

TAPE ADAPTER

A. GENERAL

The Tape Adapter provides the RW-400 with the ability to read magnetic tapes of the type produced on IBM 727 and 729-I tape units and to assemble them into RW-400 half-words, or to do the reverse, that is, to translate RW-400 half-words into IBM 727 and 729-I format and store them on magnetic tape.

Data is recorded on half-inch Mylar tape. Tape speed is 150 inches per second. Seven bits are recorded across the width of the tape: six data bits and a parity bit, called lateral parity. In addition to the parity stored with each IBM character, there is a parity word stored after each block. Each bit in this word, called longitudinal parity, makes even parity with the sum of the equivalent bits in all of the words in the block. The length of a block is limited only by the size of the core memory in the controlling device. Data is stored in the blocks at a density of 183.3 bits per inch. The interblock gap is approximately 0.75 inches. The blocks are grouped into files. There may be any number of blocks in a file. The files are separated by 3.75 inches. In this interfile gap is an end-of-file mark consisting of the octal character 17 written with even parity.

When storing RW-400 words on tape in IBM format, bits 13 and 26 of the RW-400 word are lost, as indicated in Figure 12.1. This presents no problem when the RW data is in binary coded decimal format because the two lost bit positions carry no data. If the RW data is in binary form, care must be taken to rearrange the data in such a way that none of it is lost before it is transferred to the Tape Adapter.

Figure 12.2 shows how IBM characters are assembled into RW words. Note that bits 13 and 26 contain a special tag from the Tape Adapter. This bit is a zero when there is a parity error in the least significant character of the half-word or when there is no parity error in the half-word. This bit is one when there is a parity error in the most significant character in the half-word or when there is a parity error in both characters in the half-word. Bits 27 and 28, the parity bits associated with each half-word, will make odd parity with the 12 bits of the associated half-word that contain the data when there is no parity error or when there is a parity error in both characters of a half-word. These bits will make even parity with the 12 bits of the associated half-word that contain the data when there is a parity error in one of the characters in the half-word.

Figure 12.3 summarizes the use of bits 13 and 26 and bits 27 and 28.

B. COMMAND OUTPUT INSTRUCTIONS

The following table lists the uses of the various bits in the least significant half-word of the Tape Adapter *Command Output* instruction. The most significant half-word is the same as for all *Command Output* instructions and is discussed in Chapter III.

13	12	11	10	9	8	7	6	5	4	3		1
X	X	X	X	X	X	X	X	X	X	0	0	1

Where:

Bit 13, when one, says move tape at high speed; when zero, move tape at normal speed.

Bit 12, when one, says move tape one file; when zero, move tape one block.

Bit 11, when one, says stop tape.

Bit 10, when one, says start or continue tape motion.

Bit 9, when one, says set reverse actuator.

Bit 8, when one, says set forward actuator.

Bit 7, when one, says odd parity operation; when zero, even parity operation.

Bits 6 and 5, when *00* says read searching on block number; when *10,* read searching on data; when *01,* send special status; and when *11,* write.

Bit 4, when one, says reset error indicators.

Bits 3 through 1 are the mandatory tape adapter code.

Note that there are two read instructions. One provides for reading in the *data search* mode (searching each word of a block) and the other in the *location search* mode (searching only the first word of each block). In the *data search* mode when the search word is found, the word following it is skipped and then data transfer starts. In the *location search* mode, the word directly after the search word is the first word transferred. If the instruction is a *Direct Data Input* instruction or a *Search Input* instruction with a zero search word, transfer starts with the first word of the block in either mode.

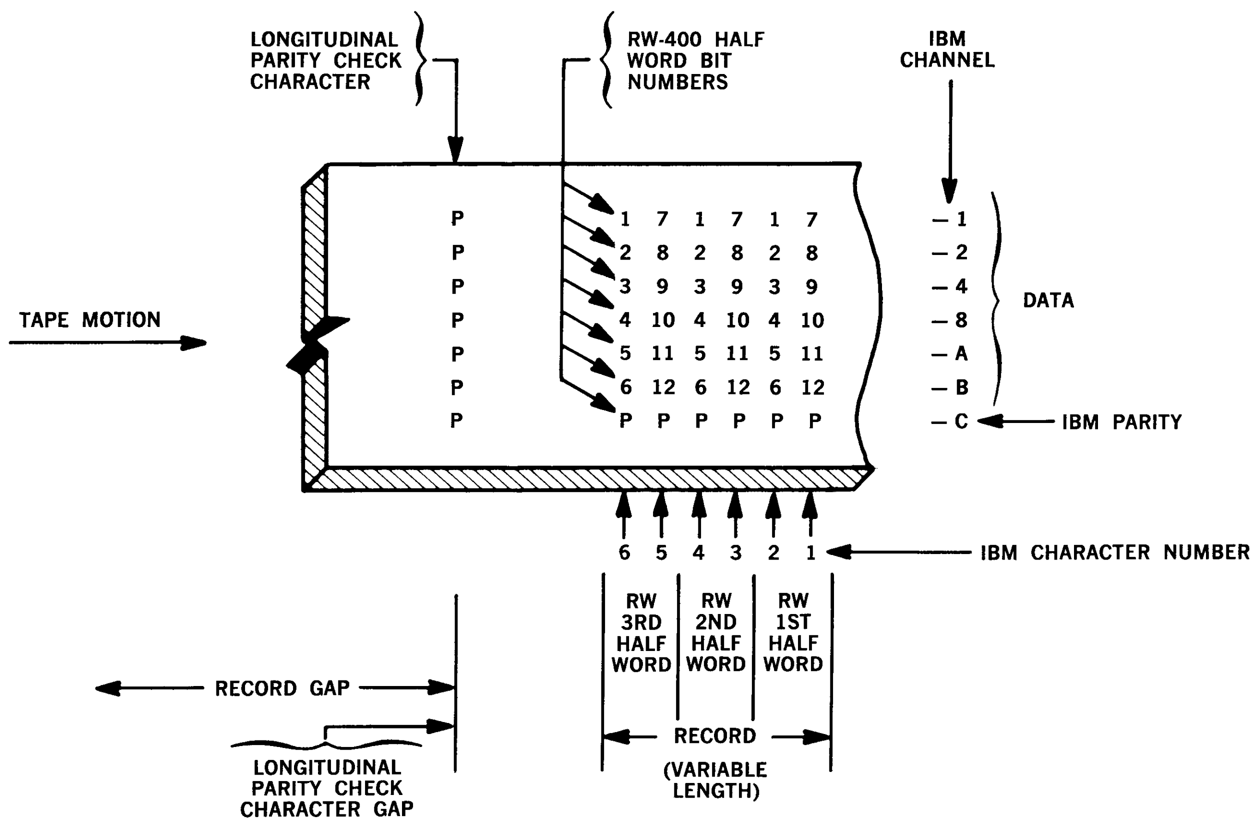


Figure 12.1. RW-400 Half Words Disassembled into IBM-727 and 729-I Tape Characters

The laterial parity on the IBM Tape will be even when the data is stored as BCD and odd when the data is in binary form. When reading, bit 7 must correspond to the parity system used on the tape or else parity errors will be generated.

When writing, be sure to specify the even parity option when writing in BCD format and the odd parity option when writing in binary format.

If while writing in BCD format, a character of octal format 17 (17_8) is written, the post-write verification will read this as an end-of-file indication and set bit 9 of status to one; however, the Tape Adapter will not stop. Subsequently, anytime the Tape Adapter reads this tape the end-of-file indication will show in status. Anytime this occurs the end-of-file indication will be reset when the tape leaves the block in which the 17_8 character appears.

Certain combinations of bits in the *Command Output* instruction will not work and will result in rejected commands. These combinations are as follows:

Bits 9 and 8 both one.

Bits 11 and 10 both one.

Bits 6 and 5 both one (the write option) in combination with either bit 13 one (write at high speed) or bit 9 one (write in the reverse direction).

Bits 6 and 5 as 01_2 (send special status) in combination with bit 10 as one.

Bit 13 as a one in combination with bit 12 as zero (move in the single block mode at high speed).

In addition to the forbidden bit combinations listed above, the following additional actions will cause a rejected command: writing without a Write-Enable ring on the tape reel, giving the tape a forward command once the end of tape is reached, attempting to reverse tape direction while the tape is in motion, and changing from odd parity to even parity without stopping the tape.

The following restrictions also apply:

A *Stop* command must be a stop-write command if the tape was writing previous to the command and a stop-read command in the appropriate read option if the tape was reading before the command. After the end-of-tape mark causes bit 10 of status to go to one, writing or reading may continue for

Half Word	Bit Position												
	14	13	12	11	10	9	8	7	6	5	4	3	2	1
FIRST	P1	T1	(CHARACTER NO. 1)						(CHARACTER NO. 2)					
SECOND	P2	T2	(CHARACTER NO. 3)						(CHARACTER NO. 4)					

P=RW-400 PARITY T=TAG (TAG OUTPUT FROM TA-400)

Figure 12.2. IBM Characters Assembled in RW-400 Words

500 milliseconds after which the Tape Adapter will terminate the operation by signaling a rejected command.

If bits 8 and 9 in an instruction are both zero, there will be no tape motion.

IBM tapes sometimes have zero-length blocks on them that are readable as such on IBM 727 or 729-I tape machines. The Tape Adapter reads these as long interblock gaps and ignores them.

C. SEARCH INPUT AND DIRECT DATA INPUT INSTRUCTIONS

The *Search Input* or *Direct Data Input* instruction used to cause the Tape Adapter to read data and transfer it to the controlling module is the standard one discussed in Chapter III. The first word of transferred data goes into core location g + 1 in the *Search Input* instruction or into location g in the *Direct Data Input* instruction. In the *Search Input* instruction, the first word transferred will be the second word after the search word in the *data search* mode and the first word after the search word in the *location search* mode. In the *Direct Data Input* or in the *Search Input* instruction, if the search word (g) is zero, the first word transferred will be the first word of the block in the *location search* mode or in the *data search* mode.

The h-field of the instruction contains the number of words to be transferred. If h = 0 there is no transfer. If h = n where n < the number of words in the block, then n words will be transferred. If h is greater than the number of words in the block, transfer will continue until an end-of-block gap is encountered. In this case, if the controlling module is a Computer, an all zero word will be written into the memory cell following the one in which the last data word transferred was stored.

D. SEARCH OUTPUT AND DIRECT DATA OUTPUT INSTRUCTIONS

The *Search Output* or *Direct Data Output* instruction which is used to cause the Tape Adapter to accept data from a controlling module and record it on tape is the standard one discussed in Chapter III. In the *Search Output* instruction, the contents of address g must be all zeros or the controlling module will stall and have to be reset manually. The first word of transferred data comes from address g + 1 in the *Search Output* instruction and address g in the *Direct Output Instruction.*

The h-field specifies the number of words to be transferred. If h = 0 and a Computer is the controlling module, there is no data transfer and an end-of-file mark is written. If h = 0 and a Buffer is the controlling module, the contents of the L

RW-400 Half-Word			RW-400 Parity Bit
Most significant IBM character	Least significant IBM character	Bit 13 or 26	Bit 27 or 28
No parity error	No parity error	0	Odd parity
Parity error	No parity error	1	Even parity
No parity error	Parity error	0	Even parity
Parity error	Parity error	1	Odd parity

Figure 12.3. Summary of the Use of Bits 13, 26, 27, and 28

register are complemented and the results are used to specify the number of words transferred.

E. STATUS

The Tape Adapter transmits status whenever any of the following conditions prevail:

The Tape Adapter is idle, the tape is moving forward at high speed, or the tape is moving at normal speed and is in an interblock gap.

The format of status is as follows:

13	12	11	10	9	8	7	6	5	4			1
X	X	X	X	X	X	X	X	X	0	0	0	1

Where:

Bit 13 is one when the tape is in an interblock gap.

Bit 12 is one when the tape will accept a command without a wait bit.

Bit 11 is one when the tape is at its physical beginning or end.

Bit 10 is one when the end-of-tape warning is passed.

Bit 9 is one when an end-of-file mark is read.

Bit 8 is one when there is a double parity error (more than one word in the block has a parity error).

Bit 7 is one when the module is inoperative.

Bit 6 is one when there is a sequence error.

Bit 5 is one when a parity error is read from tape.

Bits 4 through 1 are the mandatory Tape Adapter code.

F. SPECIAL STATUS

When bits 6 and 5 of a *Command Output* instrution to the Tape Adapter have the code *01* in them, special status appears on the data lines. Bits 6 through 1 are the same as general status. Bits 7 through 12, when any one of them is one, indicate a longitudinal parity error in channel 1 through 6, respectively. Bit 13 is the parity bit of the longitudinal parity word.

G. TIMING

Note: All times given here are times within which the programmer should work. Actual design times are for the most part greater but cannot be safely relied upon due to individual component differences.

The maximum time available between a *Command Output* instruction causing the Tape Adapter to prepare to write (in the *single block* or the *full tape* mode) and a *Search Output* or *Direct Data Output* instruction is 2.4 milliseconds. If a *Search Output* or *Direct Data Output* instruction follows after this, an end-of-file mark will be written stopping the tape. If the Tape Adapter is in the *full tape* mode, there are 5 milliseconds available after each succeeding *Data Output* instruction in which to issue another instruction (except a *Stop* instruction which must be issued within 2 milliseconds after the data transfer). If a period of time greater than this is allowed to elapse an end-of-file mark will be written stopping the tape.

The time available between a *Command Output* instruction commanding the Tape Adapter to prepare to read (in the *single block* or the *full tape* mode), and a *Search Input* or *Direct Data Input* instruction is 2.2 milliseconds. If more than the stipulated amount of time is allowed to elapse, the search data may be missed. If the search data is missed and the Tape Adapter is in the *single block* mode, the tape will move to the next interblock gap and stop. The controlling module will not stall because the Tape Adapter will signal command rejected. If the search data is missed and the Tape Adapter is in the *full tape* mode, the tape will move to the next end-of-file mark and stop and the controlling module will be released. If, however, the *Search Input* instruction is used with G = 0, or if the *Direct Data Input* instruction is used, and more than the stipulated amount of time is allowed to elapse, data transfer will start immediately and the data may be out of half-word synchronization.

If the Tape Adapter is in the *full tape* mode, one millisecond is available after each block has been read in which to issue another *Data Input* instruction.

After the Tape Adapter has completed an instruction in the *single block* mode, there is still the possibility of giving a new *Command Output* instruction before the tape stops. The time available to do this is one millisecond (the previous instruction is not considered finished until the entire block has been read, even though the number of words requested by the controlling module is less than the number of words in the block).

The time available to give a new *Command Output* instruction to a still moving tape after completion of a previous write instruction is 3.6 milliseconds.

Two hundred fifty milliseconds are necessary for the mechanization of a reversal of tape direction. After a *Command Output* instruction requesting a reversal has been accepted, a command to transfer data will not be accepted until the 250 milliseconds have elapsed. A stopped tape can be given a command calling for a tape reversal without any tape motion. The program can then go on to other things for 250 milliseconds, and then come back and start the tape in the new direction. If the technique is used, bits 12 and 13 of the *Command Output* instruction reversing tape direction must be zero.

There is an 18-millisecond delay between the execution of a stop (whether commanded, or caused by reading an end-of-file mark) and the execution of a succeeding start command.

Tape motion cannot occur for approximately six seconds following the completion of a high-speed rewind or fast forward movement of the tape.

Chapter XIII

PLOTTER

A. GENERAL

The Plotter provides the RW-400 with the ability to plot computed results. The Plotter can accept data from the Peripheral Buffer, from punched paper tape, or from a manually operated keyboard. It can also accept analog data from voltage-generating devices, such as digital-to-analog converters. It can produce graphs of up to four functions simultaneously. The graphs may be continuous curves (for analog inputs), point plots, or straight-line plots. The programmer may order various symbols to be printed at specific points, to distinguish curves from one another. The Plotter receives data from a Computer or Buffer Module through a Peripheral Buffer.

The plotting surface is a 30-by-30-inch horizontal board. There are two recording heads available for the Plotter: one is a simple pen which can plot points or lines, and the other is a symbol head which includes a print wheel with 12 characters on it. The wheel can be made to advance to any desired character by the program.

The Plotter has an adjustable parallax factor which allows the operator to set the origin of the axes anywhere on the plotting surface, between 3 inches and 30 inches from the bottom and left edges. The distances between the origin and the right edge and the origin and the top are divided into 10,000 units each to make the X and Y scales.

B. PROGRAMMING

Before the Plotter can be operated by the program, the Plotter operator has to preset the desired parallax factor and has to set a switch indicating the type of plot to be made. Specifically, the operator must know:

1. The X and Y parallax values for the curve to be plotted.
2. The kind of plot—that is, *line plot* or *point plot* and, if a *point plot,* whether symbols or simple points will be plotted. In a *line plot,* the Plotter will draw straight line segments from point to point. These should be no longer than 6 inches if possible. In a *point plot,* the Plotter will plot points or symbols (depending on which head is in place) at each coordinate indicated.

The messages to the Plotter are made up of Flexowriter characters stored on the Peripheral Buffer drum in the usual manner. Figure 13.1 gives the codes the Plotter will accept and explains the Plotter interpretation of each code.

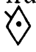
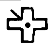
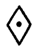
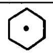
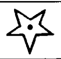
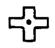
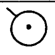
As indicated, either a series of points (or symbols) or else straight line segments can be drawn. When plotting in the *line plot* mode the *raise pen* code must be given first. After this, the first set of coordinates must be entered and followed by a *P* code which will cause the pen to move to these coordinates. Then a command to lower the pen is given. Following this with sets of coordinates alternated with *plot* commands (*P*) will cause the pen to move along the paper from point to point drawing a line between each point. After the last plot code has been entered, a *U* code followed by an *L* code will raise the pen and stop the Plotter.

When plotting in the *point plot* mode, the first set of coordinates is entered and then followed with a *plot* code. Succeeding sets of coordinates followed by *plot* codes will cause additional points to be plotted. An *L* code will stop the Plotter, after the last point is plotted.

To plot symbols in the *point plot* mode it is necessary to precede each set of coordinates for which a symbol change is desired with an *S* code followed by the code for the desired symbol.

Transmitting a set of coordinates consists of sending an *X* code followed by a plus or minus sign and then four digits. This is followed by a *Y* code, a plus or minus sign, and four digits.

The Plotter will operate at a rate of from 40 to 80 plots per minute depending upon the distance between the points.

Flexowriter Characters	Flexowriter Code Bit Positions	Plotter Interpretations or Responses
–	0 0 1 1 0 0 1	Sign of four numerical characters which follow
•	0 0 1 0 1 1 0	Sign of four numerical characters which follow or plot symbol  *
+	0 0 1 0 1 0 1	Plot symbol
0	0 1 0 0 0 0 0	0 or Plot symbol  *
1	0 1 0 0 0 1 1	1 or Plot symbol *
2	0 1 0 0 1 0 1	2 or Plot symbol *
3	0 1 0 0 1 1 0	3 or Plot symbol  *
4	0 1 0 1 0 0 1	4 or Plot symbol *
5	0 1 0 1 0 1 0	5 or Plot symbol  *
6	0 1 0 1 1 0 0	6 or Plot symbol *
7	0 1 0 1 1 1 1	7 or Plot symbol  *
8	0 1 1 0 0 0 1	8 or Plot symbol  *
9	0 1 1 0 0 1 0	9 or Plot symbol  *
D	1 0 0 0 1 1 0	Lower pen (responds only if PL-400 is in the line-plot mode).**
L	1 0 1 0 1 1 1	Stop plot.
P	1 0 1 1 1 1 0	Move record head to point specified by preceding coordinate pair, and when symbol head is on, plot a point.
S	1 1 0 0 1 0 0	Prepare to interpret next character as a symbol selector.
U	1 1 0 1 0 0 0	Raise pen (responds only if in the line-plot mode).**
X	1 1 0 1 1 1 0	Prepare to accept a signed, four-digit X coordinate.
Y	1 1 1 0 0 0 0	Prepare to accept a signed, four-digit Y coordinate.

*These Codes are interpreted as symbols only when immediately preceded by an S code.

**In the line plot mode, the pen must be ordered up and down. In the point plot mode when a new set of coordinates followed by a P code are received, the head moves to the new location and automatically plots the point (or symbol) and raises off the paper.

Note: Any code other than those listed in Figure 13.1 may result in some unpredictable excursion of the pen.

Figure 13.1. Plotter Codes

Chapter XIV

PRINTER

A. GENERAL

The Printer provides hard copy of data at 600 or 900 80-character lines per minute. It has 50 different characters available, each one called for by a six-bit binary code.

Data can be fed to the Printer by either a Computer Module or a Buffer Module connected to it through the Central (or Interim) Exchange. When a Buffer is communicating with a Printer, a Program Error Indication is caused by the execution of each *Search Output* operation to the Printer.

B. PROGRAMMING

The standard *Command Output* instruction discussed in Chapter III must be used to address the Printer. The 13 least significant bits are as follows:

13	12	11	10	9 7	6	5	4 1
X	X	–	–	X X X	–	–	0 1 0 1

Where:

Bit 13, when one, causes the Parity Error Indicator to be reset.

Bit 12, when one, inhibits paper feeding.

Bits 11, 10, 6, and 5 are not used.

Bits 9 through 7 indicate which of the seven paper-feed channels will control the amount of paper fed. (These three bits are only used when bit 12 is a zero.)

Bits 4 through 1 are the mandatory code identifying the Printer.

Ordinarily, paper is fed at the end of each line, but bits 12, 9, 8, and 7 provide the capability of advancing paper before sending any data.

After receiving this *Command Output* instruction, the Printer will, when it is ready to accept data, make bit 8 of its status format a zero. This is a signal to the controlling module indicating that the Printer is ready to accept data. This signal will remain for 200 microseconds. During this time, the Computer or Buffer can test the Printer status with a *Test Jump* or *Buffer Test* instruction and, when bit 8 is found to be zero, a *Search Output* or *Direct Data Output* instruction must be given or the first character on the print wheel will be missed. (These are the standard *Search Output* and *Direct Data Output* instructions discussed in Chapter III.)

Note that when using a *Search Output* instruction, the word at location g must be all zeros to avoid having the controlling module attempt an impossible search.

The data from the controlling module, in order to be printed, must have the following format:

26	25 20	19 14	13	12 7	6 1
x	xxxxxx	xxxxxx	x	xxxxxx	xxxxxx

Bits 26 and 13 are zero suppression bits. When a suppression bit is made one, all zero characters that follow it are regarded by the Printer as blanks. The zero suppression ends when the first non-zero character appears. The remaining bits make up four characters. (The codes associated with each of the 50 available characters are shown in Figure 14.1.) The most significant character is printed at the extreme left of the line, the next most significant character just to the right of that, and so on. Since the Printer will handle up to 80 characters on a line, a message for a complete line would consist of 21 RW-400 words. Twenty of these will contain the 80 characters, and the last one will contain an end-of-line character and paper feed data. This information must be included at the end of each line of data. If there are to be any fewer than 80 characters, the word containing the end-of-line character and the paper feed data will follow the last word containing a useful character. The format of this final word of each line is as follows:

26	25 20	19 14	13	12 10	9 7	6 1
–	111111	XXXXXX	–	---	XXX	------

Where:

Bits 26, 13, 12 through 10, and 6 through 1 are not used.

Bits 25 through 20 are the end-of-line code.

Bits 19 through 14, when they contain another end-of-line code, are interpreted as an end-of-block code. When they contain anything else, they are ignored. The only effect

Octal Printer Codes	Character
00	not used
01	′
02	)
03	″
04	−
05	+
06	°
07	=
10	*
11	(
12	,
13	±
14	?
15	/
16	not used
17	.
20	0
21	1
22	2
23	3
24	4
25	5
26	6
27	7
30	8
31	9
32	not used
33	not used
34	not used
35	not used
36	not used
37	not used

Octal Printer Codes	Character
40	A
41	B
42	C
43	D
44	E
45	F
46	G
47	H
50	I
51	J
52	K
53	L
54	M
55	N
56	O
57	P
60	Q
61	R
62	S
63	T
64	U
65	V
66	W
67	X
70	Y
71	Z
72	not used
73	not used
74	not used
75	not used
76	not used
77	end-of-line

Figure 14.1. Printer Codes

of the end-of-block code is to set bit 11 of the status transmission to 1. If an attempt is made to send more than 80 characters for a line, the Printer will print only 80 characters and then terminate the transfer.

Bits 9 through 7 indicate which of the seven paper-feed channels will control the amount of paper fed. (Paper feeding is discussed below.)

The Printer operates in such a manner that it first prints all the A's on a line, then all of the B's, and so on. Because of this, it must examine the 21 RW-400 words that make up a full line 50 times before the line is completed.

The Printer takes all of the 80 characters and examines them during the time the print wheels are moving from one character position to the next. Then it decides which of the 80 match the one coming up on the print wheels, and causes a hammer to strike each print wheel where the character in the message matches the one coming up on the wheel. For example, if all the A's have just been printed and the B's are coming up on the print wheels, then if characters 2, 17, and 79 on a line are B's, when the B's on the print wheels are in position, hammers 2, 17, and 79 will strike their respective wheels.

Following the printing of each character, bit 8 of status becomes zero. Another *Search Output* or *Direct Data Output* must be given within 610 microseconds of this or the next character will be missed.

After this cycle has been repeated 50 times, bit 9 of the status format will become one indicating that the last character has been printed. Between the last character on the print wheel and the first, there is a gap which takes 15 milliseconds to pass over. During this gap, paper is advanced. If more than two lines of paper are advanced, it will take more time than is available in the gap and status bit 8 will remain one. This will result in losing the time of a complete revolution or more of the print wheel, depending upon how much paper is fed. Note that bit 8 will only become zero again for the 200 microseconds before the first character, and will not become zero between each character. This is true as long as bit 9 is a one, indicating that the first character has not been written.

A seven-channel, punched-paper tape installed in the Printer gives the programmer the option of seven different paper feed formats. There is a row on the paper tape for each line on the Printer paper. On any one of the channels on the tape a hole is punched in each row on which it is desired to stop the paper. For example, if in channel 3 on the tape every other hole is punched, then, if channel 3 is specified in the paper feed data (i.e., bits 9 through 7 of the last word of a line of data are 011), the paper will advance two lines. If channel 6 on the tape is punched in every fourth line position and channel 6 is specified at the end of a line of characters, the paper will advance four lines. Or, generally speaking, when any channel is specified, the Printer paper will advance until the next hole in that tape channel is sensed.

In bits 9 through 7, code 001 specifies channel 1 on the paper tape, 010 channel 2 and so on until 111, which specifies channel 7. If 000 appears in these bit positions, it will be regarded as 001.

On these paper tapes, it is required that channel 1 have a punch in every line position so that whenever 000 or 001 appears in bits 9 through 7 the paper will advance only one line. The circuitry associated with channel 4 on the paper tape is so arranged that every time a punch comes up on channel 4, a first-line signal appears in status bit 10 (whether or not channel 4 was specified in the line-feed data). Therefore, channel 4 should only be punched with a hole corresponding to the first line on each page, unless the programmer wants to have a first-line signal at other places on the page. The convention being used on the tape in the Printer provides that channel 1 gives a single space, channel 2 a double space, channel 3 a triple space and channel 4 a new page. No convention has been established for channels 5, 6, and 7.

C. STATUS

The Printer transmits status whenever it is not actually engaged in accepting data from the controlling module. The status format is as follows:

13	12	11	10	9	8	7	6	5	4			1
X	X	X	X	X	X	X	0	X	0	1	0	1

Where:

Bit 13, when one, indicates that the Printer is ready to accept data.

Bit 12, when one, indicates that there is no paper in the Printer. (If a *Search Output* or *Direct Data Output* instruction is given when bit 12 is one, there will be a Command Rejected signal.)

Bit 11, when one, indicates that the Printer has received an end of block code.

Bit 10, when one, indicates that the paper is at the first line of a new page. (The test of bit 10 should only be made when bit 8 is zero.)

Bit 9, when one, indicates that the last character on the print wheel has been printed.

Bit 8, when zero, indicates that the Printer is ready to accept data. (This is always zero when 13 is one.)

Bit 7, when one, indicates that the Printer is in the Panel Mode.

Bit 6 must be zero.

Bit 5 indicates that the Printer has detected a parity error in the data it has received. Bit 5 is reset automatically after the 50th character has been printed. (The Printer will print regardless of parity errors sensed.)

Bits 4 through 1 are the mandatory code identifying the Printer.

Chapter XV
USER MODULE

A. GENERAL

The User Module is a Flexowriter mounted in an acoustical enclosure. It is a slow-speed, input-output device for alphanumeric data. Each User Module is assigned an input and an output sector of a Peripheral Buffer drum through which it can communicate with the Data Processing Central. The Flexowriter is an electric typewriter equipped to punch and read paper tape and edge cards. It reads, records, or transfers data at a rate of 10 characters per second. It uses a 7-bit code (six data bits and a parity bit) to represent each character.

B. MODES OF OPERATION

The User Module can operate in any one of three modes: *transmit, receive,* or *manual.* In each mode, several operations can be performed simultaneously. Figure 15.1 summarizes the possible simultaneous operations for each mode.

1. TRANSMIT MODE

When the User Module is in the *transmit* mode, data is transmitted to the Peripheral Buffer. To operate in this mode, two conditions must be satisfied: the User Module's TRANSMIT-MANUAL-RECEIVE switch must be in the TRANSMIT position and the input sector of the Peripheral Buffer must have its Availability flip-flop set to receive data. When the Availability flip-flop indicates that the Peripheral Buffer is set to receive data, a signal sent from the Peripheral Buffer to the User Module causes a lamp, labeled T, located on the User Module, to be lit. The signal also unlocks the Flexowriter keyboard. The operator can then transmit data from either the keyboard or the Card-Tape Reader.

Data must be transmitted in messages of 256 characters or fewer. (That is the capacity of one sector of the Peripheral Buffer drum.) When the Peripheral Buffer receives the 256th character, further data transfer is automatically prevented. The Computer or the Buffer Module can then process the message and reset the Availability flip-flop so that another message may be sent. (It takes a Computer or Buffer Module only one millisecond to unload the Peripheral Buffer and reset the Availability flip-flop. Therefore, the Flexowriter operator will not be aware of any time delay between the several 256-character segments of a long message if the controlling module accepts the data from the Peripheral Buffer immediately.) Messages shorter than 256 characters, or the last segment of a message longer than 256 characters, must be terminated by an end-of-message code. This code is generated by pushing the Flexowriter key labeled END RECORD. The end-of-message code notifies the Peripheral Buffer that it has the entire message. Further data transfer is prevented until either the Computer or the Buffer Module has reset the Availability flip-flop.

If the operator makes a mistake while entering data and has not yet pressed the END RECORD key, he can erase the entire message from the Peripheral Buffer by pressing the key labeled ERASE. If a mistake is made while punching paper tape or edge cards, the tape or card can be back spaced manually to the mistake and the erroneous character deleted by pressing the CODE DELETE key. This punches a hole in each of the 7-bit positions on the card or tape. When the card or tape is read, this code is ignored by the Card-

Operation		Mode													
		Transmit						Manual					Receive		
Typewriter	Keyboard	X	X					X	X						
	Print	X	X			X	X	X	X		X	X	X		X
Paper Tape or Edge Cards	Read			X	X	X	X			X	X	X			
	Punch		X		X		X		X	X		X		X	X

Figure 15.1. User Module Simultaneous Operations

Tape Reader. The arrangement of the characters *on the Peripheral Buffer drum is discussed in* Chapter VII.

2. RECEIVE MODE

When the UM-400 is in the *receive* mode, data is transferred from the Peripheral Buffer to the User Module. When the Peripheral Buffer has a message on an output sector connected to a User Module, it sends a signal to the User Module. This signal causes a light on the Flexowriter, labeled R, to be lit. With this light on and the mode selector switch set to RECEIVE, data from the Peripheral Buffer output sector will be accepted by the User Module. Data received may be printed by the typewriter, punched by the Card-Tape Punch, or both simultaneously, depending upon how the control switches are set.

If the operator does not wish to receive the entire *message, he can press the ERASE key and the* message will be terminated.

3. MANUAL MODE

When the User Module is in the *manual* mode, it is disconnected from the Peripheral Buffer and may be used as a card-tape reader, a punch, or a typewriter.

C. KEYBOARD LAYOUT

The User Module keyboard is shown in Figure 15.2. All keys and switches except the end-of-message key may be classified, according to their function, into one of three categories: Control

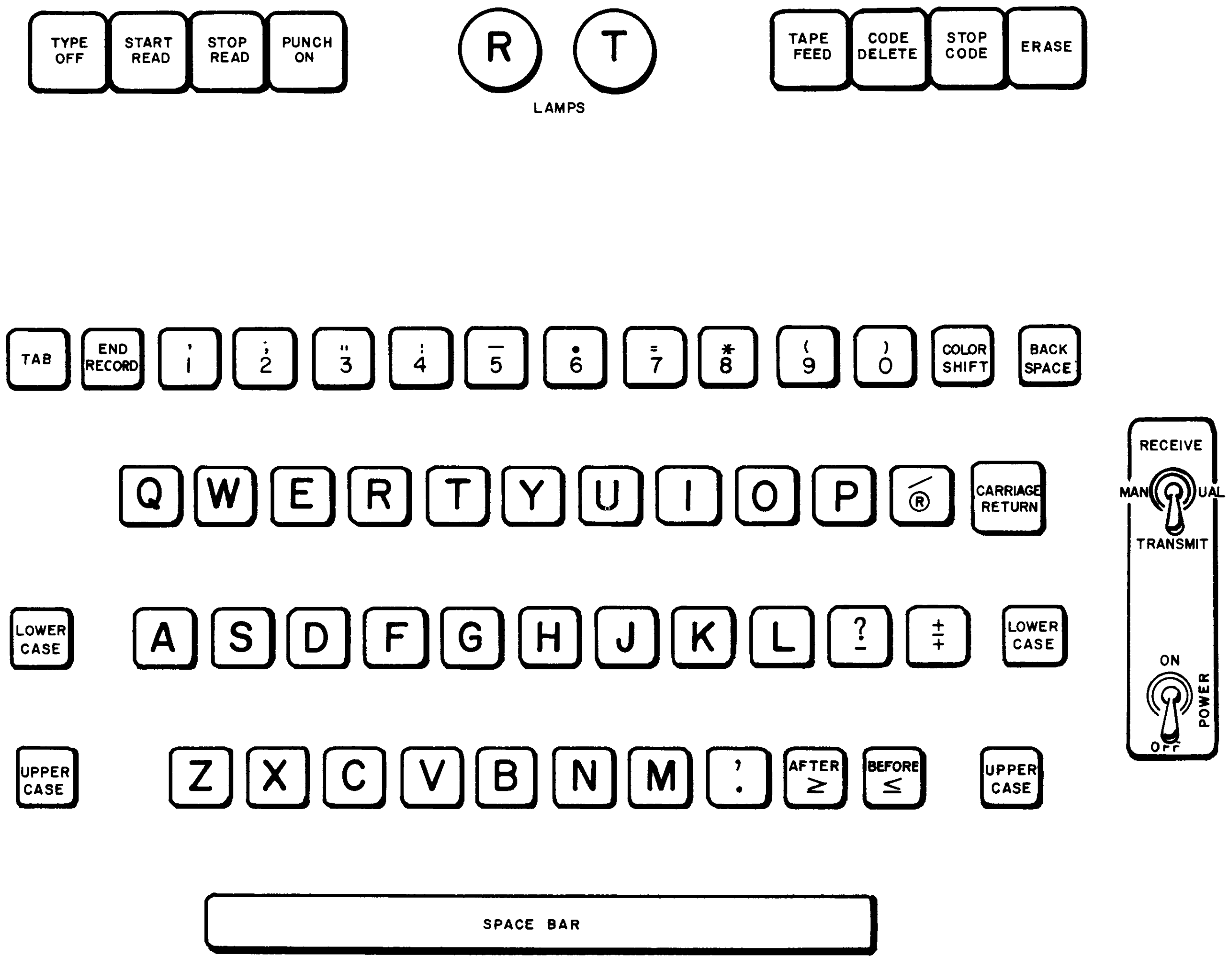


Figure 15.2. Flexowriter Keyboard

Keys and Switches, Format Control Keys, and Type Keys. Each category is discussed below.

1. CONTROL KEYS AND SWITCHES

The Control Keys and Switches are: TYPE OFF, START READ, STOP READ, PUNCH ON, TAPE FEED, CODE DELETE, STOP CODE, ERASE, TRANSMIT-MANUAL-RECEIVE, and POWER ON-OFF. The TYPE OFF key when depressed keeps the Flexowriter from typing any

Characters		**Bit Positions**							
Lower Case	**Upper Case**	**1**	**2**	**3**	**4**	**.**	**5**	**6**	**7**
Unspecified		0	0	0	0		0	0	1
COLOR SHIFT		0	0	0	0		0	1	0
TAB		0	0	0	0		1	0	0
CARRIAGE RETURN		0	0	0	0		1	1	1
BACK SPACE		0	0	0	1		0	0	0
UPPER CASE		0	0	0	1		0	1	1
LOWER CASE		0	0	0	1		1	0	1
$\geq$	(blank)	0	0	0	1		1	1	0
$\leq$	(blank)	0	0	1	0		0	0	0
STOP CODE		0	0	1	0		0	1	1
. (period)	, (comma)	0	0	1	0		1	0	1
+	$\pm$	0	0	1	0		1	1	0
−	?	0	0	1	1		0	0	1
R	/	0	0	1	1		0	1	0
SPACE BAR		0	0	1	1		1	0	0
END RECORD		0	0	1	1		1	1	1
0	)	0	1	0	0		0	0	0
1	′ (min)	0	1	0	0		0	1	1
2	; (semi-	0	1	0	0		1	0	1
	colon)								
3	″ (sec)	0	1	0	0		1	1	0
4	: (colon)	0	1	0	1		0	0	1
5	-	0	1	0	1		0	1	0
6	° (deg)	0	1	0	1		1	0	0
7	=	0	1	0	1		1	1	1
8	*	0	1	1	0		0	0	1
9	(	0	1	1	0		0	1	0
Reserved		0	1	1	0		1	0	0
Reserved		0	1	1	0		1	1	1
Reserved		0	1	1	1		0	0	0
Reserved		0	1	1	1		0	1	1
Reserved		0	1	1	1		1	0	1
Reserved		0	1	1	1		1	1	0

Characters		**Bit Positions**							
Lower Case	**Upper Case**	**1**	**2**	**3**	**4**	**.**	**5**	**6**	**7**
a	A	1	0	0	0		0	0	0
b	B	1	0	0	0		0	1	1
c	C	1	0	0	0		1	0	1
d	D	1	0	0	0		1	1	0
e	E	1	0	0	1		0	0	1
f	F	1	0	0	1		0	1	0
g	G	1	0	0	1		1	0	0
h	H	1	0	0	1		1	1	1
i	I	1	0	1	0		0	0	1
j	J	1	0	1	0		0	1	0
k	K	1	0	1	0		1	0	0
l	L	1	0	1	0		1	1	1
m	M	1	0	1	1		0	0	0
n	N	1	0	1	1		0	1	1
o	O	1	0	1	1		1	0	1
p	P	1	0	1	1		1	1	0
q	Q	1	1	0	0		0	0	1
r	R	1	1	0	0		0	1	0
s	S	1	1	0	0		1	0	0
t	T	1	1	0	0		1	1	1
u	U	1	1	0	1		0	0	0
v	V	1	1	0	1		0	1	1
w	W	1	1	0	1		1	0	1
x	X	1	1	0	1		1	1	0
y	Y	1	1	1	0		0	0	0
z	Z	1	1	1	0		0	1	1
Unspecified		1	1	1	0		1	0	1
Unspecified		1	1	1	0		1	1	0
Unspecified		1	1	1	1		0	0	1
Unspecified		1	1	1	1		0	1	0
Unspecified		1	1	1	1		1	0	0
CODE DELETE		1	1	1	1		1	1	1
Blank		0	0	0	0		0	0	0

Note: If the User Module is in the **receive** mode and the type is off and the punch is on, anything in the table will be punched except **Blank**. If the type is on (with the punch on or off) the following will not be typed (or punched): **Unspecified, Stop Code, Reserved, Code Delete,** and **Blank.**

If the User Module is in the **transmit** mode and type is off, all codes will be transmitted except **Code Delete** or **Blank**. If type is on, the following will not be transmitted: **Unspecified, Stop Code, Reserved, Code Delete,** and **Blank.**

Figure 15.3. Flexowriter Character Codes

information it may be receiving, or any information it may be sending from the Card-Tape Reader. The START READ key when depressed starts the Card-Tape Reader. The STOP READ key stops the Card-Tape Reader when it is depressed. When the PUNCH ON key is depressed, it causes all data being typed, received, or transmitted to be punched by the Card-Tape Punch. The TAPE FEED switch gives a continuous feed of paper tape while it is depressed. The CODE DELETE key is used to punch over an erroneous character with a code that the Card-Tape Reader will not read, thus deleting the error. The STOP CODE key causes a code to be punched which, when read, will stop the Card-Tape Reader. The ERASE key will cause the contents of the sector of the Peripheral Buffer to which the User Module is connected to be erased. With the TRANSMIT-MANUAL-RECEIVE switch, the operator can select the mode in which the User Module is to operate. The POWER ON-OFF switch is used to turn the User Module on and off.

2. FORMAT CONTROL KEYS

Codes generated by the Format Control Keys will be punched if the punch is on and will be sent to the Peripheral Buffer if the User Module is in the *transmit* mode.

If the User Module is in the *receive* mode, these codes will cause the indicated Flexowriter response. The Format Control Keys are:

Key	**Response**
LOWER CASE	Type basket locks in the lower case position.
UPPER CASE	Type basket locks in the upper case position.
BACK SPACE	Carriage moves one space to the right.
TAB	Carriage moves to the next tab position. (Ten tab stops are provided. They can be set a minimum of 2 spaces apart.)
CARRIAGE RETURN	Carriage returns to the extreme right. (Automatic carriage return is also provided. The point at which the carriage will automatically return is adjustable.)
COLOR SHIFT	Ribbon is shifted. If a two-color ribbon is used, the type color is changed.
SPACE BAR	Carriage moves one space to the left.

3. TYPE KEYS.

The Type Keys include all the letters of the alphabet, the decimal digits, and 20 special symbols (shown in Figure 15.3).

4. END OF MESSAGE KEY

This key, labeled END RECORD, has the function of generating a code which notifies the Peripheral Buffer that the last character of the current message has been sent.

Chapter XVI

COMPUTER COMMUNICATIONS CONSOLE

A. GENERAL

The Computer Communications Console is used primarily to facilitate communications between special purpose console operators and the Data Processing Central.

An operator can enter data either by using an alphanumeric keyboard or by depressing any one of a large number of keys, each of which sends a unique code with a predetermined meaning to the Data Processing Central. The Data Processing Central can reply to the operator by causing a display to appear on a 10-inch alphanumeric display tube or by causing one of the transilluminated push buttons to light.

The Console receives data from the Data Processing Central through a Display Buffer and enters data into the Data Processing Central through a Peripheral Buffer. It may also be connected directly to a special purpose console.

The following discussion includes a description of the Console and a tabulation of all the messages it can send and receive.

B. DESCRIPTION

The following paragraphs describe the elements of the Computer Communications Console that are concerned with receiving or transmitting data.

1. PROCESS STEP KEYS AND OVERLAYS

The Process Step Keys (see Figures 16.1 and 16.2) are a group of 30 push buttons, each one of which generates a unique code when depressed. A plastic overlay with holes in it for the keys to pass through and with a label on it associated with each key can be placed over the keys. There are 62 such overlays available. When any one of the overlays is placed over the keys, it automatically sets a group of switches which will generate a unique code identifying the overlay when the first of the

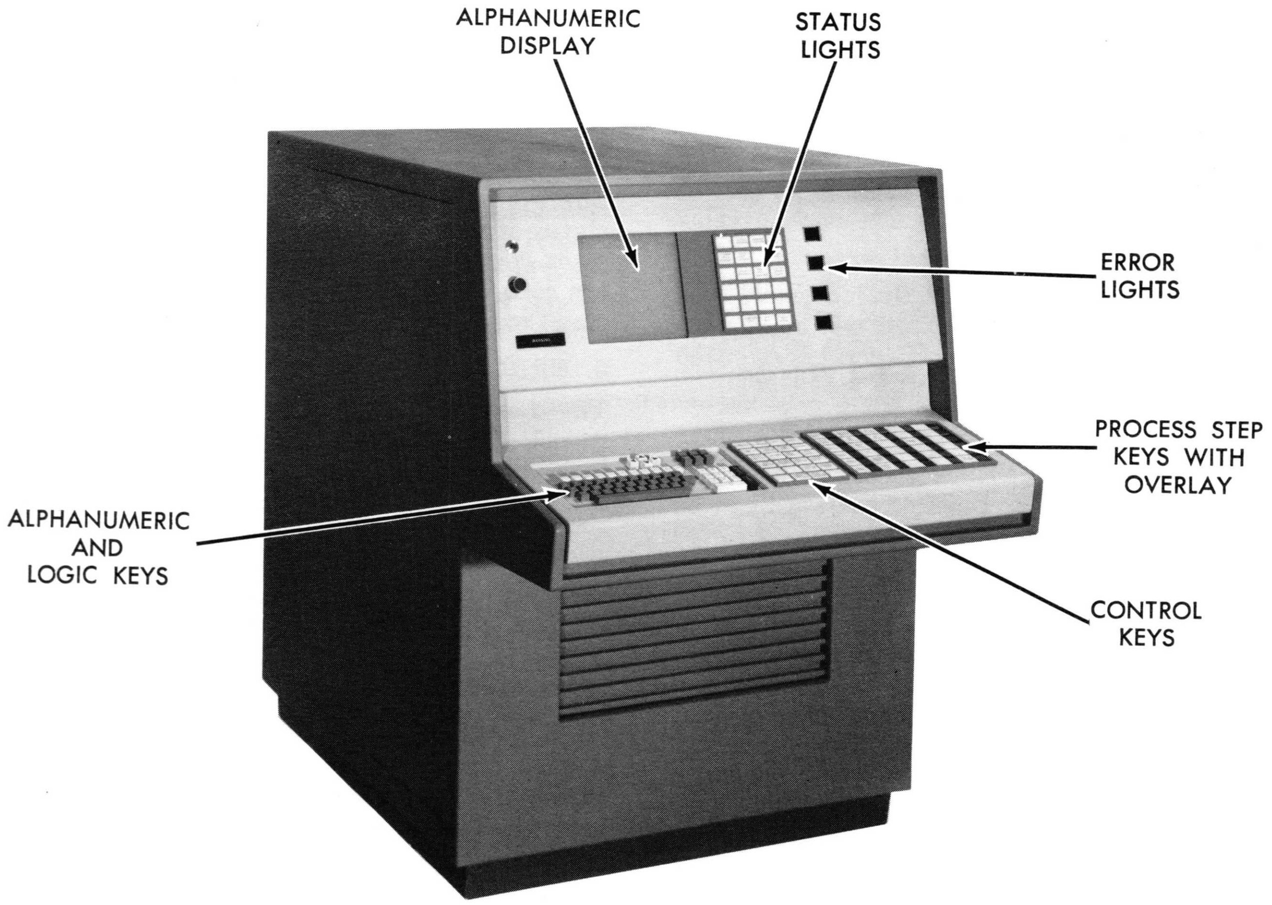


Figure 16.1. Computer Communications Console

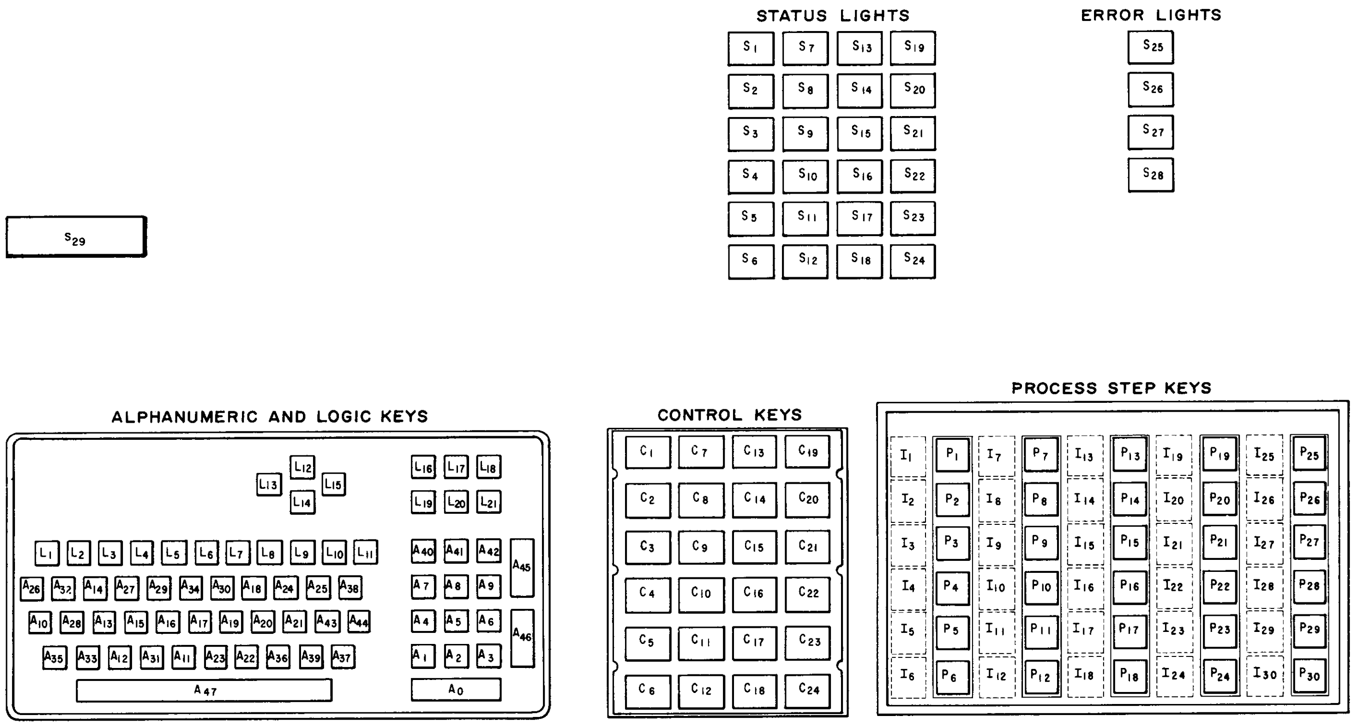


Figure 16.2. Computer Communications Console Control Panel Diagram

30 keys (key P1) is operated. Thus, the code corresponding to each of the other 29 keys can be assigned a different meaning for each overlay available.

Each of the 30 keys and each of the label slots next to the keys can be illuminated from behind. A message from the Data Processing Central is required to turn on most of these lights.

2. CONTROL KEYS

Each of the 24 Control Keys generates a unique message when operated. Each of the keys can be lit from behind at the direction of the program.

3. ALPHANUMERIC KEYS

The alphanumeric keyboard consists of 32 Alphanumeric Keys arranged like a typewriter keyboard, a space bar, an ENTER key, a CANCEL key, and 13 keys which include the digits 0 through 9, a decimal point, a plus sign, and a minus sign.

4. LOGIC KEYS

Each of these keys when pressed generates a complete and unique message to the Data Processing Central. These messages may be assigned any arbitrary meaning.

5. STATUS LIGHTS

There are 24 indicator lamps which indicate the status of various system components and of processes going on in the system. All but one of these are controlled by messages from the Data Processing Central.

6. ERROR LIGHTS

The four Error Indicators, S_{25} through S_{28}, when lit indicate errors in parity or procedure. S_{25}, the Peripheral Buffer Parity Error Indicator, is normally turned on by the program when a parity error is detected in the Peripheral Buffer. When this key is depressed, it generates a message informing the Data Processing Central that the Computer Communications Console operator is aware of the error. The program should then turn off the lamp and accept the next Computer Communications Console message.

S_{26}, the Display Buffer Parity Error Indicator, is turned on by the Computer Communications Console when it detects a parity error in the data being received from the Display Buffer. Pressing the key sends a message to the Peripheral Buffer indicating that the parity error has been found.

S_{27} is used to tell the operator that the Peripheral Buffer is not ready to accept data. It is locally controlled.

S_{28}, the Procedure Error Indicator, is turned on by the program and usually is used for signaling an unacceptable Computer Communications Console procedure, but may be assigned any function.

7. ALPHANUMERIC DISPLAY TUBE

Up to 640 alphanumeric characters can be displayed on the 10-inch alphanumeric cathode-ray display tube.

C. PROGRAMMING

1. GENERAL

The following discussion covers all the possible messages that can pass between the Computer Communications Console and the Data Processing Central. The first section discusses data flow from the Computer Communications Console to the Data Processing Central (through the Peripheral Buffer), and the second section discusses data flow from the Data Processing Central to the Computer Communications Console (through the Display Buffer).

2. DATA FROM THE COMPUTER COMMUNICATIONS CONSOLE TO THE DATA PROCESSING CENTRAL

a. *General.* All data from the Computer Communications Console is stored on a preselected sector of the Peripheral Buffer. This sector is established by the cabling between the two modules and is not changeable by programming. The programmer must, of course, know what sector of the Peripheral Buffer the Computer Communications Console is connected to.

b. *Message Format.* All messages from the Computer Communications Console to the Data Processing Central consist of three parts: a signature, the message body, and an end-of-message code. The signature identifies the source of the message. Figure 16.3 is a list of all the signature codes generated by the Computer Communications Console. The message body consists of one or more code words resulting from operating any of the various keys on the Computer Communications Console. They are discussed in detail in the following paragraphs. The end-of-message code is appended to all messages. It is a six-bit character, represented in octal notation by 17.

c. *Messages from Process Step Key P_1.* The message generated when Process Step Key P_1 is depressed consists of three 6-bit codes. The first is the signature code for Process Step Key P_1 (02_8),

Octal Signature Code	Meaning
02	Overlay (Process Step Key - P_1)
04	Process Step Keys P_2 - P_{30}
05	Control Keys (C_2, C_5 - C_{24})
40	Control Key C_1
20	Control Key C_3
21	Control Key C_4
10	Alphanumeric Keys
11	Logic Keys

Figure 16.3. Signature Code Assignments

the second is the code identifying the overlay in place as indicated in Figure 16.4, and the third is the end-of-message code (17_8). These three codes are all generated automatically when P_1 is depressed. Keys P_2 through P_{30} and lights I_2 through I_{30} will not work until P_1 has been operated.

d. *Messages from Process Step Keys P_2 through P_{30}.* The message generated when any of the 29 Process Step Keys, P_2 through P_{30}, is depressed consists of three 6-bit codes. The first is the signature code for Process Step Keys P_2 through P_{30} (04_8), the second is the code corresponding to the particular key depressed as indicated in Figure 16.5, and the third is the end-of-message code (17_8). (The meaning of a code associated with a particular Process Step Key depends on which overlay is in place. The programmer will have to obtain a list of the meanings assigned to the various keys associated with a particular overlay when he is preparing to write a program connected with that overlay.) In addition to sending the indicated codes, pressing P_{29} or P_{30} deactivates the overlay and P_1 must be operated again to re-energize it.

e. *Messages from the Control Keys.* The message generated when any of the 24 Display Control Keys, C_1 through C_{24} except C_1, C_3, and C_4 is depressed, consists of three 6-bit codes. The first is the signature code for the Control Keys (05_8), the second is the code associated with the particular Control Key depressed as indicated in Figure 16.6, and the third is the end-of-message code (17_8). Keys C_1, C_3, and C_4 send different messages as indicated in the next paragraphs.

Octal Overlay Code	Meaning
00	No overlay is in place.
01	Overlay number 1 is in place.
02	Overlay number 2 is in place.
03	Overlay number 3 is in place.
04	Overlay number 4 is in place.
05	Overlay number 5 is in place.
06	Overlay number 6 is in place.
07	Overlay number 7 is in place.
10	Overlay number 8 is in place.
11	Overlay number 9 is in place.
12	Overlay number 10 is in place.
13	Overlay number 11 is in place.
14	Overlay number 12 is in place.
15	Overlay number 13 is in place.
16	Overlay number 14 is in place.
17	Not used (reserved for end-of-message code)
20	Overlay number 15 is in place.
21	Overlay number 16 is in place.
—	
—	
76	Overlay number 61 is in place.
77	Overlay number 62 is in place.

Figure 16.4. Overlay Code Assignments

f. *Messages from Control Key C_1.* The message generated when Control Key C_1 is depressed consists of 10 six-bit codes. The first is the signature code (40_8). The next 8 make up a message from a special purpose console which may be connected to the Computer Communications Console, and the last is the end-of-message code (17_8).

g. *Messages from Control Key C_3.* The message generated when Control Key C_3 is depressed consists of 12 six-bit codes. The first is the signature code (20_8). The next 10 make up a message from the special purpose console, and the last is the end-of-message code (17_8).

Octal Process Step Key Code	Process Step Key Generating The Code
02	P_2
03	P_3
04	P_4
05	P_5
06	P_6
07	P_7
10	P_8
11	P_9
12	P_{10}
13	P_{11}
14	P_{12}
15	P_{13}
16	P_{14}
57	P_{15}
20	P_{16}
21	P_{17}
22	P_{18}
23	P_{19}
24	P_{20}
25	P_{21}
26	P_{22}
27	P_{23}
30	P_{24}
31	P_{25}
32	P_{26}
33	P_{27}
34	P_{28}
35	P_{29}
36	P_{30}

Figure 16.5. Process Step Key Code Assignments

h. *Messages from Control Key C_4.* This is another message from the special purpose console in exactly the same form as that generated when C_3 is operated, except for the signature code, which is 21_8.

Octal Control Key Code	Control Key Number
See Sec. C.1.f	C_1
02	C_2
See Sec. C.1.g	C_3
See Sec. C.1.h	C_4
05	C_5
06	C_6
07	C_7
10	C_8
11	C_9
12	C_{10}
13	C_{11}
14	C_{12}
15	C_{13}
16	C_{14}
57	C_{15}
20	C_{16}
21	C_{17}
22	C_{18}
23	C_{19}
24	C_{20}
25	C_{21}
26	C_{22}
27	C_{23}
30	C_{24}

Figure 16.6. Control Key Code Assignments

i. *Messages from the Alphanumeric Keys.* The first depression of one of the alphanumeric keys results in a message of two 6-bit codes. The first is the Alphanumeric Keyboard signature code (10_8), and the second is the code associated with the particular key depressed as indicated in Figure 16.7. Subsequent depressions of Alphanumeric Keys result in one 6-bit code being generated for each key depressed. This is the code associated with the key operated. Depressing the ENTER key transmits the end-of-message code (17_8). If another Alphanumeric Key were depressed after the ENTER key, it would transmit the signature code along with its own code indicating that a new message was starting from the keyboard. (The CANCEL key on the Alphanumeric Keyboard does not send a message, but causes the entire Peripheral Buffer sector connected to the Computer Communications Console to be erased.)

j. *Messages from the Logic Keys.* Depressing any of the 21 Logic Keys results in a message of three 6-bit codes. The first is the Logic Key signature code (11_8), the second is the code associated with the particular Logic Key being operated as indicated in Figure 16.8, and the third is the end-of-message code (17_8).

k. *Messages from the Peripheral Buffer and Display Buffer Parity Error Indicators.* The message generated when either one of these keys is depressed consists of three 6-bit codes: the first is the signature code (05_8—the same as the signature for the Control Keys), next is the message body (31_8 for the Peripheral Buffer and 32_8 for the Display Buffer), and last is the end-of-message code (17_8).

3. DATA FROM THE DATA PROCESSING CENTRAL TO THE COMPUTER COMMUNICATIONS CONSOLE

a. *General.* One channel of the Display Buffer is required to control all the displays on the Computer Communications Console. The particular Display Buffer channel used is determined by the cabling and is not under program control.

b. *Use of Words Feeding the Alphanumeric Display.*

Note: Some models of the Computer Communications Console do not have character generators. For these, the alphanumeric tube should be programmed like that in the Display Analysis Console, as explained in Chapter XVII, Sec. C.3.

The display area on the alphanumeric tube is divided into 20 lines of 36 characters each. This display is refreshed 60 times per second.

Words 0000 through 0959 of the Display Buffer band feeding this tube are used for creating this display. Each word contains three 6-bit character codes. Bits 25 through 20 of word 0000 contain the code of the character which will appear in the leftmost column on the first line. Bits 19 through 14 of this word contain the code of the character which will appear in the first line just to the right of the first character. Bits 12 through 7 contain the code of the character which will appear in the first line in the third position from the left. The rest of the word is not used. Succeeding words each provide three character codes which are added from left to right and in succeeding lines. It takes 12 words to fill one line and 240 words to complete the display. The display should be repeated four times on the drum: from word 0000 to word 0239; from word 0240 to word 0479; from word 0480 to 0719; and from word 0720 to 0959. Since the drum speed allows for a refresh rate of 15 times a second for any character, the four repetitions of the display on the drum give a refresh rate of 60 times a second.

The first character on each line must be a blank (code 00_8). Therefore, the most significant 6-bit code of words 0000, 0012, 0024, 0036. . . .0940 should be 00_8.

The format of the display is as follows:

Blank	2	3.	36
Blank	38	39.	72
	.	.	.
	.	.	.
	.	.	.
Blank	221	222.	240

Where characters 1(blank), 2 and 3, come from the word stored at drum locations 0000, 0240, 0480, and 0720; characters 4, 5, and 6 come from

Key Designation	Key Number	Octal Key Code
0	A_0	20
1	A_1	21
2	A_2	22
3	A_3	23
4	A_4	24
5	A_5	25
6	A_6	26
7	A_7	27
8	A_8	30
9	A_9	31
A	A_{10}	40
B	A_{11}	41
C	A_{12}	42
D	A_{13}	43
E	A_{14}	44
F	A_{15}	45
G	A_{16}	46
H	A_{17}	47
I	A_{18}	50
J	A_{19}	51
K	A_{20}	52
L	A_{21}	53
M	A_{22}	54
N	A_{23}	55
O	A_{24}	56
P	A_{25}	57
Q	A_{26}	60
R	A_{27}	61
S	A_{28}	62
T	A_{29}	63
U	A_{30}	64
V	A_{31}	65
W	A_{32}	66
X	A_{33}	67
Y	A_{34}	70
Z	A_{35}	71
Comma	A_{36}	32
Question Mark	A_{37}	33
Slash	A_{38}	15
Period	A_{39}	12
Decimal Point	A_{40}	12
Plus Sign	A_{41}	13
Minus Sign	A_{42}	14
Blank Key	A_{43}	Unassigned
—	A_{44}	73
Cancel	A_{45}	*
Enter	A_{46}	**
Space Bar	A_{47}	**

* Erases Peripheral Buffer
** Sends end-of-message code

Figure 16.7. Alphanumeric Keyboard Code Assignments

the word stored at drum locations 0001, 0241, 0481, and 0720; and characters 238, 239, and 240 come from the word stored at drum locations 0239, 0479, 0719, and 0959.

The 6-bit codes available for use along with the characters they cause to be generated are shown in figures 16.9 and 16.10.

c. *Use of Words 0971 and 0972.* These words are used for sending messages through the Computer Communications Console to the special purpose module which may be connected to it.

d. *Use of Words 0973 through 0977.* These words are used to store control signals for all the Display and Control Panel lamps. There is a bit in one of these words corresponding to each lamp. When the bit is made a one, the lamp is turned on. Figure 16.11 gives the bit assignments for these words.

Key Number	Octal Logic Key Code
L_1	01
L_2	02
L_3	03
L_4	04
L_5	05
L_6	06
L_7	07
L_8	10
L_9	11
L_{10}	12
L_{11}	13

Key Number	Octal Logic Key Code
L_{12}	14
L_{13}	15
L_{14}	16
L_{15}	57
L_{16}	20
L_{17}	21
L_{18}	22
L_{19}	23
L_{20}	24
L_{21}	25

Figure 16.8. Logic Key Code Assignments

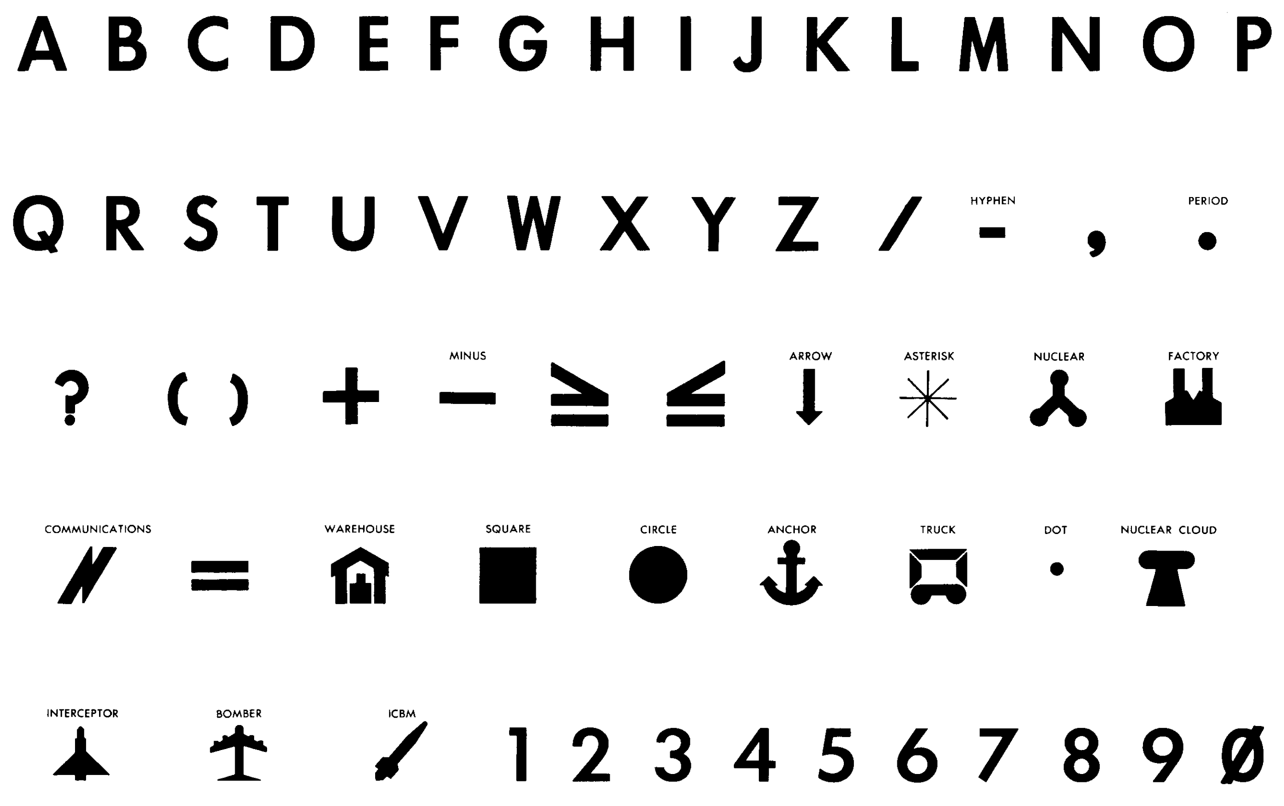


Figure 16.9. Display Tube Characters for the Computer Communications Console

Character	Octal Code
Asterisk	01
Nuclear	02
Factory	03
Bomber	04
Communications	05
Nuclear Cloud	06
$\geq$	07
$\leq$	10
ICBM	11
. (period)	12
+	13
− (minus)	14
/ (slash)	15
Circle	16
Anchor	17
$\not\geq$	20
1	21
2	22
3	23
4	24
5	25
6	26
7	27
8	30
9	31
, (comma)	32
?	33
Arrow	34
Warehouse	35
Square	36
Dot	37
A	40
B	41
C	42
D	43
E	44
F	45
G	46
H	47
I	50
J	51
K	52
L	53
M	54
N	55
O	56
P	57
Q	60
R	61
S	62
T	63
U	64
V	65
W	66
X	67
Y	70
Z	71
=	72
Hyphen	73
Truck	74
Interceptor	75
((parenthesis)	76
) (parenthesis)	77
Blank	00

Figure 16.10. Octal Codes for Display Tube Characters

Assigned Illuminator	Word No.	Bit No.
S 2	0976	24
S 3	0976	23
S 4	0976	22
S 5	0976	21
S 6	0976	20
S 7	0976	19
S 8	0976	18
S 9	0976	17
S10	0976	16
S11	0976	15
S12	0976	14
S13	0976	12
S14	0976	11
S15	0976	10
S16	0976	9
S17	0976	8
S18	0976	7
S19	0976	6
S20	0976	5
S21	0976	4
S22	0976	3
S23	0976	2
S24	0976	1
S25	0975	12
S28	0975	11
I 1	0973	25
I 2	0973	23
I 3	0973	21
I 4	0973	19
I 5	0973	17
I 6	0973	15
I 7	0973	12
I 8	0973	10
I 9	0973	8
I10	0973	6
I11	0973	4
I12	0973	2
I13	0974	25
I14	0974	23
I15	0974	21
I16	0974	19
I17	0974	17
I18	0974	15

Assigned Illuminator	Word No.	Bit No.
I19	0974	12
I20	0974	10
I21	0974	8
I22	0974	6
I23	0974	4
I24	0974	2
I25	0975	25
I26	0975	23
I27	0975	21
I28	0975	19
I29	0975	17
I30	0975	15
P 1	0973	24
P 2	0973	22
P 3	0973	20
P 4	0973	18
P 5	0973	16
P 6	0973	14
P 7	0973	11
P 8	0973	9
P 9	0973	7
P10	0973	5
P11	0973	3
P12	0973	1
P13	0974	24
P14	0974	22
P15	0974	20
P16	0974	18
P17	0974	16
P18	0974	14
P19	0974	11
P20	0974	9
P21	0974	7
P22	0974	5
P23	0974	3
P24	0974	1
P25	0975	24
P26	0975	22
P27	0975	20
P28	0975	18
P29	0975	16
P30	0975	14
C 1	0977	25

Figure 16.11. Assignments for Display Buffer Words 973-977 (Part 1 of 2)

Assigned Illuminator	Word No.	Bit No.
C 2	0977	24
C 3	0977	23
C 4	0977	22
C 5	0977	21
C 6	0977	20
C 7	0977	19
C 8	0977	18
C 9	0977	17
C10	0977	16
C11	0977	15
C12	0977	14
C13	0977	12

Assigned Illuminator	Word No.	Bit No.
C14	0977	11
C15	0977	10
C16	0977	9
C17	0977	8
C18	0977	7
C19	0977	6
C20	0977	5
C21	0977	4
C22	0977	3
C23	0977	2
C24	0977	1

Figure 16.11. Assignments for Display Buffer Words 973-977 (Part 2 of 2)

Chapter XVII
DISPLAY AND ANALYSIS CONSOLE

A. GENERAL

The Display and Analysis Console provides the capability of displaying data graphically on cathode-ray tubes and of allowing an operator to interrogate the Data Processing Central with a large selection of special purpose, pre-prepared codes. The operator can enter data into the Data Processing Central either with a numerical keyboard or by pressing any one of a large number of keys, each of which, when operated, transmits a unique code with a pre-determined meaning. The Data Processing Central in turn can reply to the operator by causing a display to appear on either of two 17-inch cathode-ray tubes or on a 10-inch alphanumeric display tube, or by causing one of the transilluminated push buttons to light.

The Display and Analysis Console receives data from the Data Processing Central through the Display Buffer and enters data into the Data Processing Central through the Peripheral Buffer.

The following discussion includes a description of the Console and a tabulation of all the messages the Console can send and receive.

B. DESCRIPTION

The following paragraphs describe the elements of the Display and Analysis Console that are concerned with receiving or transmitting data.

1. PROCESS STEP KEYS AND OVERLAYS

The Process Step Keys (see Figures 17.1 and 17.2) are a group of 30 push buttons, each one of which generates a unique code when depressed. A plastic overlay with holes in it for the keys to pass through and with a label on it associated with each key can be placed over the keys. There are 62 such overlays available. When any one of the overlays is placed over the keys, it automatically sets a group of switches which will generate a unique code identifying the overlay when the first of the 30 keys is operated. Thus, the code corresponding to each of the other 29 keys can be assigned a different meaning for each overlay

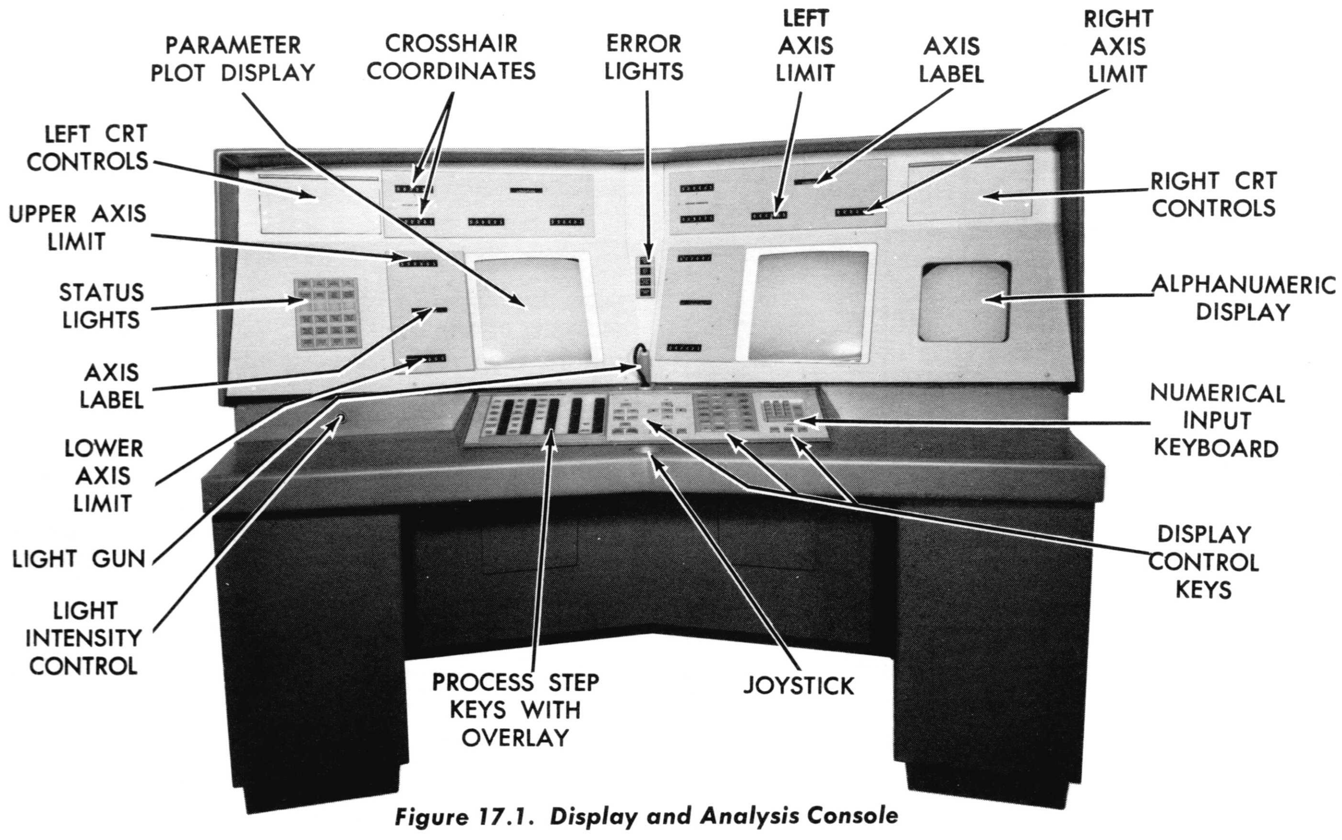


Figure 17.1. Display and Analysis Console

available. A message from one of these keys must always be preceded by the message identifying the overlay in place.

Each of the 30 keys and label slots next to the keys can be illuminated from behind. A message from the Data Processing Central is required to turn on most of these lights.

2. DISPLAY CONTROL KEYS

Each of the 41 Display Control Keys generates a unique message when operated. Each of the keys can be illuminated from behind. The lights for all but three of them are controlled by messages from the Data Processing Central.

3. DATA ENTRY KEYBOARD

There are 15 Data Entry Keys: digits 0 through 9, a plus sign, a minus sign, a decimal point, an ENTER key, and a CANCEL key. All of these except the CANCEL key generate codes.

4. JOYSTICK LEVER AND KEYS J_1 AND J_2

Pressing key J_1 causes a set of locally generated crosshairs to appear on one or both of the 17-inch cathode-ray tubes. (Which tube the crosshairs will appear on is determined by selecting one of the Display Control Keys labeled LEFT, RIGHT, or BOTH.) The Joystick is then used to position the crosshairs. Pressing J_2 causes the crosshairs to disappear and a message to be generated which specifies the coordinates of the point at which the crosshairs intersected.

5. LIGHT GUN

The light gun emits a small beam of light which can be directed at any point on either of the 17-inch cathode-ray tubes. When the trigger on the light gun is pressed, a message is generated which specifies the address on the Display Buffer drum containing the coordinates of the selected point.

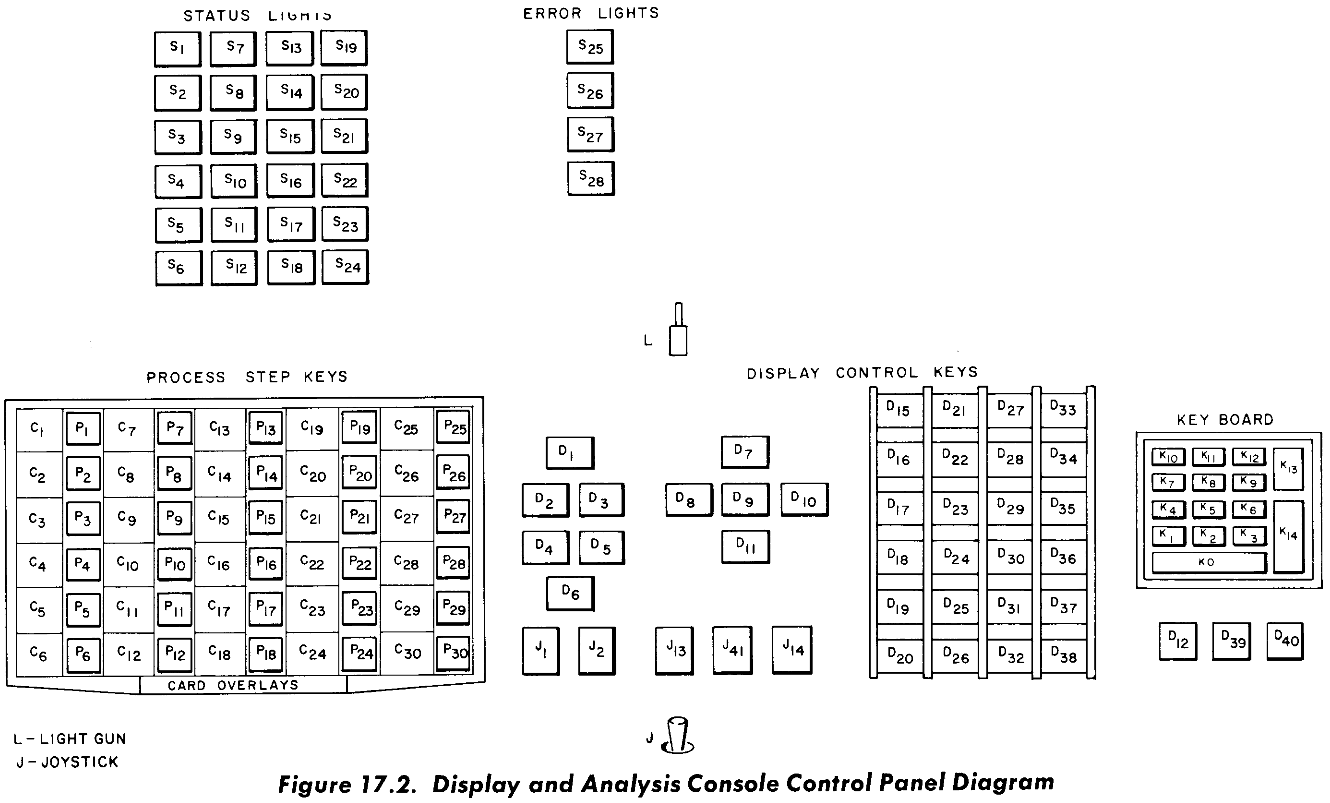
6. STATUS LIGHTS

There are 24 indicator lamps which indicate the status of various system components and of processes going on in the system. All but one of these are controlled by messages from the Data Processing Central.

7. ERROR INDICATORS

The four Error Indicators, S_{25} through S_{28}, when lit indicate errors in parity or procedure.

S_{25}, the Peripheral Buffer Parity Error Indicator, is normally turned on by the program when a parity error is detected in the Peripheral Buffer. When this key is depressed, it generates a message informing the Data Processing Central that the Console operator is aware of the error. The pro-



STATUS LIGHTS
ERROR LIGHTS
S1 S7 S13 S19 S2 S8 S14 S20 S3 S9 S15 S21 S4 S10 S16 S22 S5 S11 S17 S23 S6 S12 S18 S24
S25 S26 S27 S28
L
PROCESS STEP KEYS
C1 P1 C7 P7 C13 P13 C19 P19 C25 P25
C2 P2 C8 P8 C14 P14 C20 P20 C26 P26
C3 P3 C9 P9 C15 P15 C21 P21 C27 P27
C4 P4 C10 P10 C16 P16 C22 P22 C28 P28
C5 P5 C11 P11 C17 P17 C23 P23 C29 P29
C6 P6 C12 P12 C18 P18 C24 P24 C30 P30
CARD OVERLAYS
DISPLAY CONTROL KEYS
D1 D2 D3 D4 D5 D6 D7 D8 D9 D10 D11
D15 D21 D27 D33 D16 D22 D28 D34 D17 D23 D29 D35 D18 D24 D30 D36 D19 D25 D31 D37 D20 D26 D32 D38
J1 J2 J13 J41 J14
KEY BOARD
K10 K11 K12 K13 K7 K8 K9 K4 K5 K6 K14 K1 K2 K3 KO
D12 D39 D40
J
L - LIGHT GUN
J - JOYSTICK

Figure 17.2. Display and Analysis Console Control Panel Diagram

gram should then turn off the lamp and accept the next Console message.

S_{26}, the Display Buffer Parity Error Indicator, is turned on by the Console when it detects a parity error in the data being received from the Display Buffer. Pressing the key sends a message to the Peripheral Buffer indicating that the parity error has been found.

S_{27} is used to notify the operator that the Peripheral Buffer is not ready to accept data. It is locally controlled.

S_{28}, the Procedure Error Indicator, is turned on by the program and usually is used to signal an unacceptable Console procedure but may be assigned any function.

8. 17-INCH CATHODE-RAY TUBES

The 17-inch cathode-ray tubes are used to display dots, symbols, and lines. Up to 1020 points or 510 symbols can be displayed on each tube at once with a refresh rate of 15 times per second.

9. ALPHANUMERIC DISPLAY TUBE

Up to 640 alphanumeric characters can be displayed on the 10-inch alphanumeric cathode-ray tube.

10. SCALE LIMIT DISPLAY

The Scale Limit display is a six-digit numerical indicator which gives the value of the scale limits of data plotted on the 17-inch cathode-ray tubes. There are eight Scale Limit displays—one for the upper limit and one for the lower limit of both the ordinate and the abscissa of each 17-inch display tube.

11. PARAMETER LABELS

The Parameter Labels are used to label both axes of each 17-inch display tube. There are 16 possible labels—selected by the program—for each of the four axes.

12. CROSSHAIR READOUT

The Crosshair Readout consists of two 6-digit numerical indicators which give the coordinates of a crosshair on the display. There is one Crosshair Readout for each of the 17-inch display tubes.

C. PROGRAMMING

1. GENERAL

The following discussion covers all the possible messages that can pass between the Display and Analysis Console and the Data Processing Central. The first section discusses data flow from the Display and Analysis Console to the Data Processing Central (through the Peripheral Buffer), and the second section discusses data flow from the Data Processing Central to the Display and Analysis Console (through the Display Buffer).

2. DATA FROM THE DISPLAY AND ANALYSIS CONSOLE TO THE DATA PROCESSING CENTRAL

a. *General.* All data from the Display and Analysis Console is stored on a preselected sector of the Peripheral Buffer. This sector is established by the cabling between the two modules and is not changeable by programming. The programmer must, of course, know what sector of the Peripheral Buffer the Console is connected to.

b. *Message Format.* All messages from the Display and Analysis Console to the Data Processing Central consist of three parts: a signature, the message body, and an end-of-message code. The signature identifies the source of the message. Figure 17.3 is a list of all the signature codes gen-

Octal Signature Code	Meaning
01	The following code resulted from depressing Process Step Key P_1 and identifies one of the 62 overlays.
02	The following code resulted from depressing one of the Process Step Keys (P_2-P_{30}).
03	The following codes resulted from depressing one of the Data Entry Keys (K_0-K_{12}).
04	The following code resulted from depressing one of the Display Control Keys or the Peripheral Buffer or Display Buffer Parity Error Keys.
05	The following code resulted from depressing key J_2 and represents the coordinates at which the crosshairs intersect.
06	The following code resulted from triggering the Light Gun while pointed at the left display. The code represents the Display Buffer address containing the coordinates of the target selected by the Light Gun.
46	The same as 06 but for the right display.

Figure 17.3. Signature Code Assignments

erated by the Console. The message body consists of one or more code words resulting from operating any of the various keys on the Console. They are discussed in detail in the following paragraphs. The end-of-message code is appended to all messages. It is a six-bit character, represented in octal notation by 17.

c. *Messages from Process Step Key P_1.* The message generated when Process Step Key P_1 is depressed consists of three 6-bit codes. The first is the signature code for Process Step Key P_1 (01_8), the second is the code identifying the overlay in place as indicated in Figure 17.4, and the third is the end-of-message code (17_8). These three codes are all generated automatically when P_1 is depressed. Keys P_2 through P_{30} and lights I_2 through I_{30} will not work until P_1 has been operated.

d. *Messages from Process Step Keys P_2 through P_{30}.* The message generated when any of the 29 Process Step Keys, P_2 through P_{30}, is depressed consists of three 6-bit codes. The first is the signature code for Process Step Keys P_2 through P_{30} (02_8), the second is the code corresponding to the particular key depressed as indicated in Figure 17.5, and the third is the end-of-message code (17_8). (The meaning of a code associated with a particular Process Step Key depends on which overlay is in place. The programmer will have to obtain a list of the meanings assigned to the various keys associated with a particular overlay when he is preparing to write a program connected with that overlay.) In addition to sending the indicated codes, pressing P_{29} or P_{30} deactivates the overlay and P_1 must be operated again to reenergize it.

Octal Overlay Code	Meaning
00	No overlay is in place.
01	Overlay number 1 is in place.
02	Overlay number 2 is in place.
03	Overlay number 3 is in place.
04	Overlay number 4 is in place.
05	Overlay number 5 is in place.
06	Overlay number 6 is in place.
07	Overlay number 7 is in place.
10	Overlay number 8 is in place.
11	Overlay number 9 is in place.
12	Overlay number 10 is in place.
13	Overlay number 11 is in place.
14	Overlay number 12 is in place.
15	Overlay number 13 is in place.
16	Overlay number 14 is in place.
17	Not used (reserved for end-of-message code).
20	Overlay number 15 is in place.
21	Overlay number 16 is in place.
—	
—	
76	Overlay number 61 is in place.
77	Overlay number 62 is in place.

Figure 17.4. Overlay Code Assignments

Octal Process Step Key Code	Process Step Key Generating The Code
02	P_2
03	P_3
04	P_4
05	P_5
06	P_6
07	P_7
10	P_8
11	P_9
12	P_{10}
13	P_{11}
14	P_{12}
15	P_{13}
16	P_{14}
17	Not used
20	P_{15}
21	P_{16}
22	P_{17}
23	P_{18}
24	P_{19}
25	P_{20}
26	P_{21}
27	P_{22}
30	P_{23}
31	P_{24}
32	P_{25}
33	P_{26}
34	P_{27}
35	P_{28}
36	P_{29}
37	P_{30}

Figure 17.5. Process Step Key Code Assignments

e. *Messages from the Display Control Keys.* The message generated when any of the 41 Display Control Keys, D_1 through D_{41}, is depressed consists of three 6-bit codes. The first is the signature code for the Display Control Keys (04_8), the second is the code associated with the particular Display Control Key depressed as indicated in Figure 17.6, and the third is the end-of-message code (17_8). Note in Figure 17.6 that key D_{12} generates two codes. It generates the codes alternately as it is depressed.

f. *Messages from the Data Entry Keyboard.* The first depression of one of the Data Entry Keys results in a message of two 6-bit codes: the first is the Data Entry Keyboard signature code (03_8), and the second is the code associated with the particular key depressed as indicated in Figure 17.7. Subsequent depressions of Data Entry Keys result in one 6-bit code being generated for each key depressed. This is the code associated with the key operated. Depressing the ENTER key transmits the end-of-message code (17_8). If another Data Entry Key were depressed after the ENTER key, it would transmit the signature code along with its own code indicating a new message was starting from the Data Entry Keyboard. (The CANCEL key on the Data Entry Keyboard does not send a message but causes the entire Peripheral Buffer sector connected to the Display and Analysis Console to be erased.)

g. *Messages from Joystick Key J_2.* The message generated when key J_2 is depressed consists of six 6-bit characters. The first is the signature code for Key J_2 (05_8). The next four are the body of the message and specify the X and Y coordinates of the point at which the crosshairs intersect. The first two represent the X coordinate, and the second two the Y coordinate. Each coordinate consists of a total of 10 bits so that the most significant bit of each character is not needed (it is made a one). The most significant half of each coordinate is transmitted first. The last character of the six transmitted when J_2 is operated is the end-of-message code (17_8).

h. *Messages from Light Gun.* The message generated when the trigger on the Light Gun is depressed consists of four 6-bit characters. The first is the Light Gun signature (06_8 or 46_8 depending upon which display the gun is aimed at). (See Figure 17.3.) The next two form the body of the message and specify the address on the Peripheral Buffer drum containing the target at which the

Octal Display Control Key Code	Display Control Key Number	*Display Control Key Label
01	D1	Expand xy
02	D2	Expand x
03	D3	Expand y
04	D4	Contract x
05	D5	Contract y
06	D6	Contract xy
07	D7	Shift Up
10	D8	Shift Left
11	D9	Center
12	D10	Shift Right
13	D11	Shift Down
14	D12a	Disconnect
15	D12b	Reconnect
52	D13	L (Left)
53	D14	R (Right)
16	D15	X-coordinate
17	Not used (end-of-message code)	—
20	D16	Y-coordinate
21	D17	not assigned
23	D18	not assigned
—	—	—
—	—	—
50	D40	not assigned
51	D41	Both

* The labels assigned to these keys are not fixed and can be changed for various applications.

Figure 17.6. Display Control Key Code Assignments

light gun is pointed. The five least significant bits of each of the two 6-bit characters are used to make up the 10-bit address. The most significant half of the address is transmitted first. The last of the four codes generated when the Light Gun trigger is operated is the end-of-message code (17_8).

i. *Messages from the Peripheral Buffer and Display Buffer Parity Error Indicators.* The message generated when either of these keys is depressed consists of three 6-bit codes. The first is the signature (04_8—the same as the signature for the Display Console Keys). The next is the message body (54_8 for the Peripheral Buffer and 55_8 for the Display Buffer), and the last is the end-of-message code (17_8).

3. DATA FROM THE DATA PROCESSING CENTRAL TO THE DISPLAY AND ANALYSIS CONSOLE

a. *General.* Three bands of a Display Buffer are required to supply data to a Display and Analysis Console: one band for each of the two 17-inch cathode-ray tube displays and one band for the alphanumeric display and the various key and indicator lights. Each of these bands holds 1024 26-bit words. The following discussion explains the use of each bit of each word. The words on the two bands feeding the 17-inch displays are used identically, so the two bands are discussed as one.

Octal Data Entry Code	Meaning of Associated Data Entry Key
00	0
01	1
02	2
03	3
04	4
05	5
06	6
07	7
10	8
11	9
12	Decimal Point
13	Plus
43	Minus
17	Enter (end-of-message)

Figure 17.7. Data Entry Keyboard Code Assignments

b. *Use of Words on the Display Buffer Bands Feeding the 17-inch Cathode-Ray Tubes.* Words 0 through 3 inclusive must each contain a word of zeros—this time is reserved for locally generated crosshairs. Words 4 through 1023 are used to display dots, symbols, and lines on the CRT.

The meaning of each bit in words 4 through 1023 is as follows:

Bit	Meaning
26	unblanking*
25	not assigned
24–21	symbol definition (see Figure 17.8)
20–11	X-coordinate of symbol**
10– 1	Y-coordinate of symbol**

* When there is a one in this bit, a line is drawn on the CRT between the previous dot and the dot defined by the current word. The line should be no longer than seven inches. (On some consoles the points defined by these two sets of coordinates must be such that a line drawn between them will be at some increment of 90° from the horizontal. In these cases the line may be longer than seven inches.) When bits 24-21 call for a symbol bit 26 should contain a zero. That is, a line should not be drawn to a symbol, only to a dot. However a line can be drawn from a symbol.

** The CRT is divided into 1024 units on the ordinate and on the abcissa. The units start from the lower left and count up for the ordinate and to the right for the abcissa.

Whenever possible, unused words on these two bands should have the following quantity in them:

f	g	h
00_8	1000_8	1000_8

This puts the smallest demand on the circuits in the Display and Analysis Console.

When bits 24 through 21 in a word define a symbol, the following word on the Display Buffer channel is ignored.

c. *Use of Words on the Display Buffer Band Feeding the Alphanumeric Display, the Keys, and Indicators. (Some Display Analysis Consoles have Character Generator units feeding the alphanumeric CRT. These are programmed in the same way as the alphanumeric CRT in the Computer Communications Console — refer to Chapter XVI, Sec. C.3.b. — except that some of the symbols and symbol codes are different. These are given in Figures 17.9 and 17.10.)*

1) *Words 0 through 959*—Words 0 through 959 are used for the alphanumeric display. Two modes of operation of the alphanumeric dis-

play are available. In one mode, there are 640 character positions on the screen with 20 lines of 32 characters each displayed 15 times per second. The character positions appear on the display as follows:

```
  1. . . . . . .  32
 33. . . . . . .  64
  .               .
  .               .
  .               .
609. . . . . . . 640
```

In the second mode of operation, there are 320 character positions with 16 lines of 20 characters each displayed 30 times per second. The character positions appear on the display as follows:

```
  1. . . . . . .  20
 21. . . . . . .  40
  .               .
  .               .
  .               .
301. . . . . . . 320
```

In the 320-character mode, each character must be written on the Display Buffer in two different locations, while each character is written in only one location in the 640-character mode. The choice of modes is determined by the program. On this same band, word 975, bit number 1, is used to inform the Console which operation mode is to be used. A zero in this bit position indicates that the 20 x 16 format is to be used, while a one indicates that the 32 x 20 format is to be used.

When writing words in two drum locations for the 320-character mode, a word at location n (n=0 through 479) should be rewritten at location n+480.

The bits in the words feeding the alphanumeric display are used to form characters by associating each bit with a corresponding matrix dot position. The matrix dot positions are numbered as follows:

1	2	3	4	5
6	7	8	9	10
11	12	13	14	15
16	17	18	19	20
21	22	23	24	25
26	27	28	29	30
31	32	33	34	35

Since each word has 26 bit positions and the matrix requires 35 bits, more than one word is required to generate one display character. The correspondence between computer word bit position and matrix dot position is as follows (bit positions 26 and 13 are not used and are ignored by the Display Console):

	Bit Position	**Dot Position**	
word n	25	1	1st character formed
	24	2	
	.	.	
	.	.	
	.	.	
	14	12	
	12	13	
	.	.	
	.	.	
	.	.	
	1	24	
word n+1	25	25	
	24	26	
	.	.	
	.	.	
	.	.	
	15	35	
	14	1	2nd character formed
	12	2	
	.	.	
	.	.	
	.	.	
	1	13	
word n+2	25	14	
	.	.	
	.	.	
	.	.	
	14	25	
	12	26	
	.	.	
	.	.	
	.	.	
	3	35	

Bit positions 2 and 1 of word n+2 are not used.

A character of any arbitrary shape may be formed by programming ones in the appropriate positions. A one in any bit position causes the corresponding dot in the 5 x 7 matrix to be illuminated.

2) *Words 960 through 971*—Words 960 through 971 are used to store display data for the Axis Scale Limit displays and the Crosshair Coordinate displays for the two 17-inch cathode-ray tubes. Figure 17.11 shows these word assignments. Each of these displays is a six-digit number. The six digits come out of an RW-400 word in the following way:

26	25-22	21-18	17-14	13	12-9	8-5	4-1
–	most significant digit	second most significant digit	third most significant digit	–	fourth most significant digit	fifth most significant digit	least significant digit

Bit Number 24 23 22 21	Symbol	Size of Symbol (S-small; L-large)
0 0 0 0	Repression	
0 0 0 1	Dot	
0 0 1 0	O	S
0 0 1 1	O	L
0 1 0 0	+	S
0 1 0 1	+	L
0 1 1 0	X	S
0 1 1 1	X	L

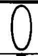
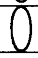
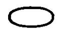
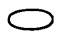
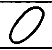
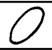
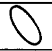
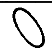
Bit Number 24 23 22 21	Symbol	Size of Symbol (S-small; L-large)
1 0 0 0		S
1 0 0 1		L
1 0 1 0		S
1 0 1 1		L
1 1 0 0		S
1 1 0 1		L
1 1 1 0		S
1 1 1 1		L

Figure 17.8. Definition of Symbols Displayed on the 17-inch Cathode-Ray Tubes

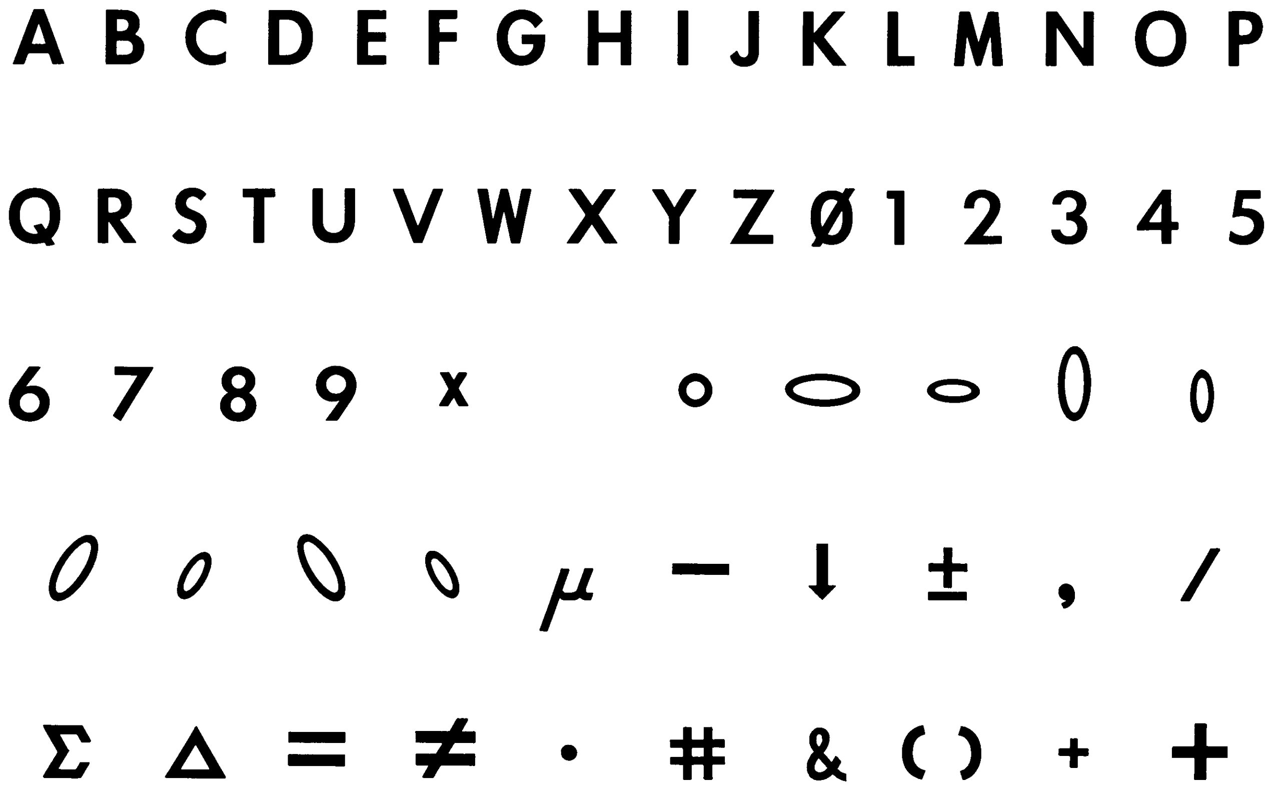

A B C D E F G H I J K L M N O P
Q R S T U V W X Y Z Ø 1 2 3 4 5
6 7 8 9 x
μ — ↓ ± , /
Σ Δ = ≠ · # & ( ) + +

Figure 17.9. Alphanumeric Display Tube Characters for the Display and Analysis Console

The decimal digits represented by the four bits in each digit position are as indicated in Figure 17.12.

3) *Word 972*—Word 972 is used to control the X and Y axes indicator devices. Each of these devices has 16 display faces, each of which has a pre-printed label on it. The bits in word 972 apply to the four labels in the following manner:

25, 24, 23, 22	X axis label, left CRT
21, 20, 19, 18	Y axis label, left CRT
12, 11, 10, 9	X axis label, right CRT
8, 7, 6, 5	Y axis label, right CRT

The binary number (0000-1111) made up of one of these 4-bit groups designates which of the faces (1 through 16) will appear on the Console. (Number 0000 says face 1, 0001, face 2 . . . . 1111, face 16.)

4) *Words 973-978*—Words 973-978 are used to store control signals for all the panel lights. A one in a bit position of one of these words turns the associated light on. Figure 17.13 shows the assignments for the bits in each of these words.

5) *Words 979-1023*—Words 979 through 1023 have no assigned use.

Character	Octal Code
○	01
⬭	02
⬯	03
0	04
0	05
0	06
o	07
⬮	10
◦	11
. (period)	12
+ (large)	13
−	14
/	15
x (small)	16
μ	17
$\not{\phi}$	20
1	21
2	22
3	23
4	24
5	25
6	26
7	27
8	30
9	31
, (comma)	32
+ (small)	33
↓	34
±	35
Σ	36
Δ	37
A	40
B	41
C	42
D	43
E	44
F	45
G	46
H	47
I	50
J	51
K	52
L	53
M	54
N	55
O	56
P	57
Q	60
R	61
S	62
T	63
U	64
V	65
W	66
X	67
Y	70
Z	71
=	72
≠	73
#	74
&	75
((parenthesis)	76
) (parenthesis)	77
Blank	00

Figure 17.10. Octal Codes for Alphanumeric Display Tube Characters for the Display and Analysis Console

Word Number	Display	Location
0960	Left Display	Left X-axis limit
0961	Left Display	Right X-axis limit
0962	Left Display	Lower Y-axis limit
0963	Left Display	Upper Y-axis limit
0964	Left Display	X-crosshair coordinate
0965	Left Display	Y-crosshair coordinate
0966	Right Display	Left X-axis limit
0967	Right Display	Right X-axis limit
0968	Right Display	Lower Y-axis limit
0969	Right Display	Upper Y-axis limit
0970	Right Display	X-crosshair coordinate
0971	Right Display	Y-crosshair coordinate

Figure 17.11. Assignments for Words 960 Through 971

Code	Character
0000	0
0001	1
0010	2
0011	3
0100	4
0101	5
0110	6
0111	7
1000	8
1001	9
1010	Decimal Point
1011	Minus Sign

Figure 17.12. Characters Represented by the Bits in Words 960 Through 971

Assigned Illuminator	Word No.	Bit No.
C 1	0973	25
C 2	0973	23
C 3	0973	21
C 4	0973	19
C 5	0973	17
C 6	0973	15
C 7	0973	12
C 8	0973	10
C 9	0973	8
C10	0973	6
C11	0973	4
C12	0973	2
C13	0974	25
C14	0974	23
C15	0974	21
C16	0974	19
C17	0974	17
C18	0974	15
C19	0974	12
C20	0974	10
C21	0974	8
C22	0974	6
C23	0974	4
C24	0974	2
C25	0975	25
C26	0975	23
C27	0975	21
C28	0975	19
C29	0975	17
C30	0975	15
P 1	0973	24
P 2	0973	22
P 3	0973	20
P 4	0973	18
P 5	0973	16
P 6	0973	14
P 7	0973	11
P 8	0973	9
P 9	0973	7
P10	0973	5
P11	0973	3

Figure 17.13. Assignments for Words 973-978 (Part 1 of 2)

Assigned Illuminator	Word No.	Bit No.
P12	0973	1
P13	0974	24
P14	0974	22
P15	0974	20
P16	0974	18
P17	0974	16
P18	0974	14
P19	0974	11
P20	0974	9
P21	0974	7
P22	0974	5
P23	0974	3
P24	0974	1
P25	0975	24
P26	0975	22
P27	0975	20
P28	0975	18
P29	0975	16
P30	0975	14
D 1	0977	25
D 2	0977	24
D 3	0977	23
D 4	0977	22
D 5	0977	21
D 6	0977	20
D 7	0977	19
D 8	0977	18
D 9	0977	17
D10	0977	16
D11	0977	15
D12	0977	14
D15	0977	10
D16	0977	9
D17	0977	8
D18	0977	7
D19	0977	6
D20	0977	5
D21	0977	4
D22	0977	3
D23	0977	2
D24	0977	1
D25	0978	25

Assigned Illuminator	Word No.	Bit No.
D26	0978	24
D27	0978	23
D28	0978	22
D29	0978	21
D30	0978	20
D31	0978	19
D32	0978	18
D33	0978	17
D34	0978	16
D35	0978	15
D36	0978	14
D37	0978	12
D38	0978	11
D39	0978	10
D40	0978	9
S 2	0976	24
S 3	0976	23
S 4	0976	22
S 5	0976	21
S 6	0976	20
S 7	0976	19
S 8	0976	18
S 9	0976	17
S10	0976	16
S11	0976	15
S12	0976	14
S13	0976	12
S14	0976	11
S15	0976	10
S16	0976	9
S17	0976	8
S18	0976	7
S19	0976	6
S20	0976	5
S21	0976	4
S22	0976	3
S23	0976	2
S24	0976	1
S25	0975	12
S28	0975	9
J 1	0975	8
J 2	0975	7

Figure 17.13. Assignments for Words 973-978 (Part 2 of 2)

Chapter XVIII
PAPER TAPE READER

A. GENERAL

The Paper Tape Reader is a photo-electric device for reading Flexowriter paper tapes. It can be connected to the Central (or Interim) Exchange like any other module and addressed by a Computer or a Buffer with a *Command Output* instruction, or it can be connected directly to a Computer and operated with *Character Transfer* instructions like the directly connected Flexowriter. The Paper-Tape Reader will read a standard Flexowriter tape and convert the characters into RW-400 words for transmission to a Computer or a Buffer.

B. PROGRAMMING

The *Command Output* instruction necessary to cause the Paper Tape Reader to transfer data to a Computer or a Buffer is the standard one discussed in Chapter III. The thirteen least significant bits of this instruction are as follows:

13	12	11	10	9	8	7	6	5	4	3 1
0	0	X	X	0	1	1	X	X	X	0 0 1

Where:

Bits 13, 12, 9, 8, and 7 are mandatory as shown.

Bit 11, when one, says prepare to read.

Bit 10, when one, says start.

Bit 6, when zero, says use format I and, when one, says use format II (see Figure 18.1).

Bit 5, when zero, says ignore all zero zone characters (characters with a zero in both bit one and bit two) and ignore the code delete character. Bit 5, when one, says use all characters except code delete.

Bit 4, when one, says reset the parity error indicator.

Bits 3 through 1 are the code identifying the Paper Tape Reader.

Note: When bit 6 is zero and bit 5 is one a Rejected Command Signal results.

The *Command Output* instruction may be followed by a *Search Input* instruction from a Computer or by a *Search Input* or *Direct Data Input* instruction from a Buffer. These are the standard *Search Input* and *Direct Data Input* instructions discussed in Chapter III.

The tape will stop after the last character of the hth word has been read. If a Flexowriter Stop character is encountered before h words have been transferred the tape will stop at the end of the RW-400 word in which the Stop code appears. If bit 5 is a zero the Stop code will not form part of

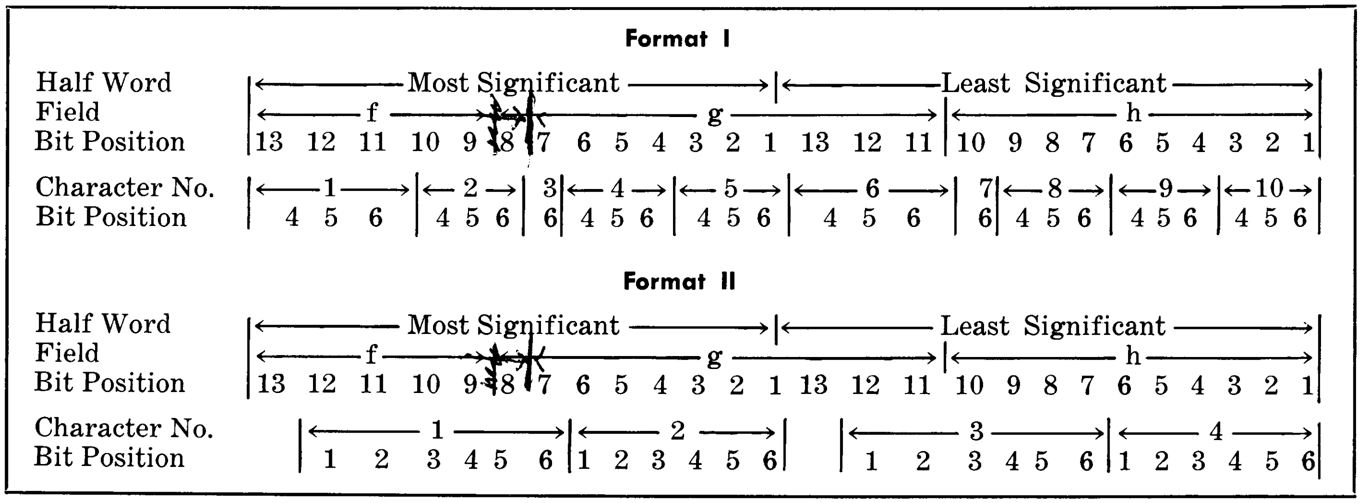


Figure 18.1. Paper Tape Reader Format

the word (since stop is a zero zone character) but if bit 5 is a one the Stop character will form part of the RW-400 word.

After stopping because of a Stop code the Paper Tape Reader will transmit status and if the stop code was part of a word rather than a separate code coming after a word, a parity error will be indicated in the status format because of the Stop code. The only exception to this is if the Stop is the last character of a word in format II when bit 5 is a one. In this case the Stop will be part of the word and no parity error will be indicated.

Note: Whenever the tape is started, at least two blank frames must occur before the first punched character appears. If an additional segment of data is to be read subsequently the last word of the previous transmission must be followed by at least two frames of blank tape before any other character occurs.

C. STATUS

Whenever it is not actually reading, the Paper Tape Reader is sending status. The status format is as follows:

13	12	11	10	9	8	7	6	5	4			1
–	–	–	–	–	–	X	X	–	0	0	0	1

Where:

Bits 13 through 8 and bit 5 are not used.

Bit 7 when one indicates that the Paper Tape Reader is in the panel mode.

Bit 6 when one indicates that there has been a parity error or that the Paper Tape Reader has been stopped by a misplaced Stop code.

Bits 4 through 1 are the mandatory Paper Tape Reader code.

Chapter XIX
DIRECTLY CONNECTED FLEXOWRITER

A. GENERAL

A directly connected Flexowriter is a Flexowriter connected directly to a Computer Module, thus avoiding going through a Peripheral Buffer and the Central Exchange. It can be used to enter data directly into a Computer or to receive data directly from a Computer.

B. PROGRAMMING

The only instruction a Computer can use in addressing a directly connected Flexowriter is the *Character Transfer* instruction. The format is as follows:

26 21	20 11	10 1
f	g	h

Where:

The f-field code is 75_8.

The most significant bit of the g-field indicates input if it is a one and output if it is a zero. The rest of the g-field is not used.

The h-field is a jump address.

Upon receipt of the instruction if bit 20 is a one, a 6-bit character is added to a special register, the F register in the Computer. When the next character is transferred, it will enter the F register and the character in the F register will enter the Accumulator. The sign of the Accumulator is not changed.

If bit 20 is a zero, a six-bit character is transferred out of the least significant end of the Accumulator. (The F register is not used in transferring data out.) The Accumulator remains unchanged. If bits 7-13 of the Accumulator contain an odd number of ones, then no character will be transferred, and the next instruction in sequence will be taken.

If the Flexowriter is not ready when a *Character Transfer* instruction is given, the next instruction is taken from h.

Execution of the *Character Transfer* instruction requires 18 microseconds.

Chapter XX

DIGITAL CLOCK

A. GENERAL

The Digital Clock is designed to provide the time of day to any Computer or Buffer Module in the Data Processing Central. The requesting module must connect to the Clock through the Central (or Interim) Exchange. Once connected, the requesting module can ask for the time and will get it in terms of the number of sixtieths of a second that have elapsed since the Clock was set.

B. PROGRAMMING

To get the time from the Digital Clock, the requesting module must first give a *Command Output* instruction requesting the time and then must follow this with a one-word *Search Input* or *Direct Data Input* instruction. The 13 least significant bits of the *Command Output* instruction are as follows: *(Command Output* instructions are discussed in Chapter III.)

13 8	7 5	4 1
– – – – – – –	1 0 0	0 0 1 0

Where:

Bits 13 through 8 are not used.

Bits 7 through 5 are mandatory as shown.

Bits 4 through 1 identify the type module being addressed.

There is no limit to the amount of time that may be permitted to elapse between the *Command Output* instruction and the *Direct Data Input* or *Search Input* instruction. When the *Direct Data Input* or *Search Input* instruction is accepted (these instructions are discussed in Chapter III) the Digital Clock will respond with a 26-bit word which is the number of sixtieth-of-a-second intervals that have elapsed since the Clock was originally set. The Clock will count to 5,184,000 (24 hours) and then go back to 1 and count over again. It will repeat this 24-hour cycle indefinitely.

C. STATUS

When not transferring data, the Clock sends status. The only useful information is in bit 7 which, when one, indicates that the Clock is in the Panel Mode. Bits 4 through 1 contain the identifier 0 0 1 0.

GLOSSARY

g	The contents of the g-field of a Computer or a Buffer instruction—often a core memory address.
G	The 26-bit word stored at location g.
h	The contents of the h-field of a Computer or a Buffer instruction—often a core memory address.
H, (h)	The 26-bit word stored at location h.
r	The contents of the 10-bit Read register in a Buffer—often a core memory addres.
R, (r)	The 26-bit word stored at location r.
w	The core memory address stored in the 10-bit Write register in a Buffer.
W, (w)	The 26-bit word stored at location w.
Z	A number $0<Z<1023$.
*	Any of the following operations: add, subtract, absolute subtract, multiply, divide, square root, or insert.
$\rightarrow$	Replacement. ($A \rightarrow B$ means A replaces B.)
v	The logical *or* process. (A bit-by-bit sum without carries.)
AB (juxtaposition)	Juxtaposition in logical expressions denotes a logical *and* function. (A bit-by-bit product without carries.)
A′ (prime)	A prime mark means the ones complement of the function so marked. (A′ means the ones complement of A.)
Controlled module	A module that is accepting commands from another module. All modules except Computers and Buffer Modules are controlled modules. A Buffer Module may be a controlled module.
Controlling module	A module that is giving commands to other modules. Computers are controlling modules, and Buffer Modules may be controlling modules.
External module	A module other than a Computer or a Buffer.
Panel Mode	A mode of operation in which a module can be controlled from its control panel but cannot be controlled by the program.
Program Error Indicator	Bit I_{17} of the Computer Alert register or bit 18 of the Buffer Test configuration.

Subscripts:

$_8$	Indicates that a number is in octal notation.
$_g$	The g-field part of a word (bits 20 through 11).
$_h$	The h-field part of a word (bits 10 through 1).
$_{ms}$	Most significant half of a double-length result.
$_{ls}$	Least significant half of a double-length result.
$_q$	Quotient.
$_r$	Remainder.

Note: Numbers in the text are decimal unless they are obviously binary or indicated to be octal..

RW-400 OCTAL INSTRUCTION LIST

	Octal f-Field Code	Instruction Name	Page Reference
OP	00	STOP	18
SSB	01	STORE SUBTRACT	10
RSB	02	REPLACE SUBTRACT	10
HSB	03	HOLD SUBTRACT	10
AAC	04	ADD TO ACCUMULATOR	17
SAD	05	STORE ADD	9
RAD	06	REPLACE ADD	9
HAD	07	HOLD ADD	9
SSA	11 9	STORE ABSOLUTE SUBTRACT	10
RSA	12 10	REPLACE ABSOLUTE SUBTRACT	10
ISA	13 11	HOLD ABSOLUTE SUBTRACT	10
SQ	15 13	STORE SQUARE ROOT	17
SQ	16 14	REPLACE SQUARE ROOT	17
ISQ	17 15	HOLD SQUARE ROOT	17
DV	21 17	STORE DIVIDE	10
DV	22 18	REPLACE DIVIDE	10
DV	23 19	HOLD DIVIDE	10
MAD	24 20	MULTIPLY ADD	17
MU	25 21	STORE MULTIPLY	10
RMU	26 22	REPLACE MULTIPLY	10
HMU	27 23	HOLD MULTIPLY	10
*	30 24	SHIFT	19
TRM	31 25	TRANSMIT	17
LA	33 27	LOAD A	18
DAB	34 28	STORE A, B	18
SIN	35 29	STORE INSERT	17
RIN	36 30	REPLACE INSERT	17
HIN	37 31	HOLD INSERT	17
*	40 32	SEARCH OUTPUT	24
	41 33	SEARCH INPUT	23
*	42 34	COMMAND OUTPUT	22
	43 35	CONDITIONAL SEARCH INPUT	50
*	44 36	DIRECT DATA OUTPUT	25
	45 37	DIRECT DATA INPUT	24
* *	70	TEST JUMP	19
TJP	71	TALLY JUMP	19
	72	LINK JUMP	18
	73	COMPARE JUMP	19
	75	CHARACTER TRANSFER	90
INS	77	INSERT S	18

Note: Any instruction with an f-field code other than those listed in the table will be ignored; the Program Error Indicator will be set and the next instruction taken in sequence.

RW-400 ALPHABETICAL INSTRUCTION LIST

Instruction Name	Octal f-Field Code	Page Reference
ADD TO ACCUMULATOR	04	17
CHARACTER TRANSFER	75	90
COMMAND OUTPUT	42	22
COMPARE JUMP	73	19
CONDITIONAL SEARCH INPUT	43	50
DIRECT DATA INPUT	45	24
DIRECT DATA OUTPUT	44	25
HOLD ABSOLUTE SUBTRACT	13	10
HOLD ADD	07	9
HOLD DIVIDE	23	10
HOLD INSERT	37	17
HOLD MULTIPLY	27	10
HOLD SUBTRACT	03	10
HOLD SQUARE ROOT	17	17
INSERT S	77	18
LINK JUMP	72	18
LOAD A	33	18
MULTIPLY ADD	24	17
REPLACE ABSOLUTE SUBTRACT	12	10
REPLACE ADD	06	9
REPLACE DIVIDE	22	10
REPLACE INSERT	36	17
REPLACE MULTIPLY	26	10
REPLACE SQUARE ROOT	16	17
REPLACE SUBTRACT	02	10
SEARCH INPUT	41	23
SEARCH OUTPUT	40	24
SHIFT	30	19
STOP	00	18
STORE A ,B	34	18
STORE ABSOLUTE SUBTRACT	11	10
STORE ADD	05	9
STORE DIVIDE	21	10
STORE INSERT	35	17
STORE MULTIPLY	25	10
STORE SQUARE ROOT	15	17
STORE SUBTRACT	01	10
TALLY JUMP	71	19
TEST JUMP	70	19
TRANSMIT	31	17

Note: Any instruction with an f-field code other than those listed in the table will be ignored; the Program Error Indicator will be set and the next instruction taken in sequence.