SPERRY RAND

UNIVAC®
1218
MILITARY
COMPUTER

PROGRAMMERS
REFERENCE
MANUAL

UNIVAC 1218

UNIVAC 1218

UNIVAC
FEDERAL SYSTEMS DIVISION
St. Paul, Minnesota
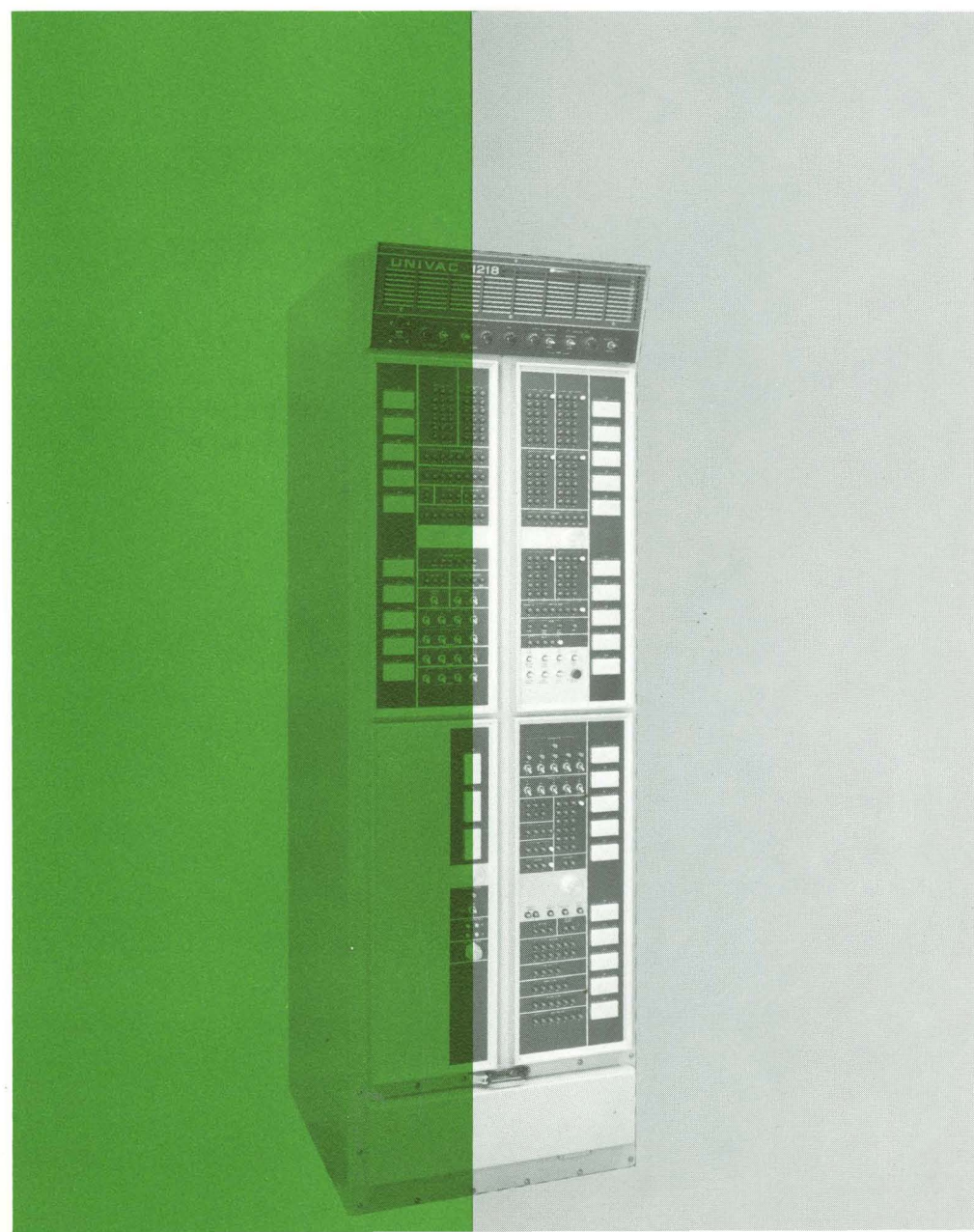
November, 1969

CHANGE 2

To: All holders of the 1218 Programmers Reference Manual, PX 2910.

From: Advanced Programming Department

Insert the changes as follows: (with 33 enclosures)

1. Title page: Remove and replace with new title page attached.

2. Remove page iii and replace with new page iii attached.

3. Remove Table of Contents pages ix/x thru xxv and replace with new pages ix/x thru xxvii attached.

4. Remove page II-D-23/II-D-24 and replace with new page II-D-23/II-D-24 attached.

5. Following Section II-F insert new tab, "9200/9300 Subsystem," and new pages II-G-1/11-G-2 thru II-G-39.

PROGRAMMERS REFERENCE MANUAL

FOR

UNIVAC 1218 COMPUTER

PX 2910

REVISION C

CHANGE 2

NOVEMBER 1969

UNIVAC

FEDERAL SYSTEMS DIVISION

CHANGE RECORD

Change No.	Date	Reason for Change
Rev C	August 1968	Retyped and extensively revised entire manual.
Rev C Change 1	July 1969	Updated and revised manual. Deleted section on 1469 High-Speed Printer.
Change 2	November 1969	Added Section II-G which describes the use of the 9200/9300 Computer as a peripheral device. Minor corrections.

LIST OF EFFECTIVE PAGES

Page No.	Change in Effect	Page No.	Change in Effect
Title Page	Rev C, Change 2	III-C-27	Rev C, Change 1
ii thru iii	Rev C, Change 2	III-C-28 thru III-C-30	Rev C
v	Rev C, Change 1	III-C-31 thru III-C-42	Rev C, Change 1
Letter	Rev C	IV-1	Rev C, Change 1
ix	Rev C, Change 2	IV-A-1 thru IV-A-14	Rev C
xv thru xxvii	Rev C, Change 2	IV-B-1 thru IV-B-25	Rev C
I-1	Rev C	IV-B-26 thru IV-B-28	Rev C, Change 1
I-A-1 thru I-A-15	Rev C	IV-B-29	Rev C
I-B-1 thru I-B-46	Rev C	IV-B-30 thru IV-B-34	Rev C, Change 1
I-C-1 thru I-C-17	Rev C	IV-B-35 thru IV-B-40	Rev C
II-1	Rev C	IV-C-1 thru IV-C-6	Rev C
II-A-1 thru II-A-12	Rev C	IV-D-1	Rev C, Change 1
II-B-1 thru II-B-8	Rev C	IV-D-2 thru IV-D-6	Rev C
II-C-1 thru II-C-7	Rev C	IV-D-7 thru IV-D-8	Rev C, Change 1
II-C-8	Rev C, Change 1	IV-D-9 thru IV-D-10	Rev C
II-C-9 thru II-C-23	Rev C	IV-E-1	Rev C, Change 1
II-D-1 thru II-D-23	Rev C	IV-E-2	Rev C
II-D-24	Rev C, Change 2	IV-E-3 thru IV-E-4	Rev C, Change 1
II-D-25 thru II-D-29	Rev C	IV-F-1	Rev C, Change 1
II-E-1	Rev C, Change 1	IV-F-2 thru IV-F-11	Rev C
II-F-1 thru II-F-10	Rev C	IV-F-12 thru IV-F-14	Rev C, Change 1
II-F-11	Rev C, Change 1	V-1	Rev C
II-G-1 thru II-G-39	Rev C, Change 2	V-A-1 thru V-A-16	Rev C
III-1 thru III-2	Rev C	V-B-1 thru V-B-2	Rev C
III-A-1 thru III-A-13	Rev C	V-C-1 thru V-C-4	Rev C
III-B-1 thru III-B-26	Rev C	A-1 thru A-4	Rev C
III-C-1 thru III-C-26	Rev C		

PREFACE

The UNIVAC 1218 System includes the UNIVAC ® 1218 Military Computer, standard peripheral equipment, and a set of standard computer software. This manual was written to meet the needs and requirements of the programmer. It gives general information about the computer, peripheral equipment, and the standard software available with the computer. The principal software packages consists of:

1) Assemblers.
2) Operator service routines.
3) Programmer service subroutines.

Organizationally, the manual is comprised of five major sections, each of which has several parts. Section I contains general information concerning the computer hardware and functional capabilities of the computer. It gives the reader logical and functional characteristics of the instruction repertoire and describes the Input/Output (I/O) characteristics of the computer in detail. In addition, hardware features are included such as size, power, and cooling requirements, as well as compatibility between peripheral equipment and other computers.

Section II contains functional information concerning peripheral equipments commonly used in a 1218 System.

Section III describes the family of TRIM assemblers available with the computer. Since computer memory size and peripheral equipment differ from site to site, three assemblers have been written and are available to satisfy the user's needs.

Section IV describes operator service routines. Operator routines are those routines used by a computer operator, under manual control, to perform computing-center operations. Typical routines of this category are paper tape load, magnetic tape handler, program trace, memory dump, and so forth. Such routines perform a service to the user, but they do not become integrated into his programs. Most of these routines may also be operated under program control.

Section V describes programmer service subroutines. The standard package of programmer service subroutines saves the user time since it contains general subroutines that are used often. These subroutines are supplied in assembler source language for easy integration into the user's program. Subroutines in this category include mathematical, conversion, and assembler support subroutines.

The information contained in this manual is generally written from the standpoint of a UNIVAC 1218 Computer operating in the 1218 mode with exceptions for other modes specifically noted. Therefore, this manual also serves as the programming manual for the UNIVAC 1218 Computer.

The following page is a pre-addressed application for future revisions to this manual. To receive future revisions to the manual, the end user need only fill in the necessary data and return the application by mail.

TABLE OF CONTENTS

CHANGE 1

TABLE OF CONTENTS (CONT.)

TABLE OF CONTENTS (CONT.)

TABLE OF CONTENTS (CONT.)

TABLE OF CONTENTS (CONT.)

TABLE OF CONTENTS (CONT.)

TABLE OF CONTENTS (CONT.)

(This section has been intentionally omitted.)

TABLE OF CONTENTS (CONT.)

TABLE OF CONTENTS (CONT.)

TABLE OF CONTENTS (CONT.)

TABLE OF CONTENTS (CONT.)

TABLE OF CONTENTS (CONT.)

TABLE OF CONTENTS (CONT.)

TABLE OF CONTENTS (CONT.)

TABLE OF CONTENTS (CONT.)

TABLE OF CONTENTS (CONT.)

LIST OF TABLES

LIST OF ILLUSTRATIONS

LIST OF ILLUSTRATIONS (CONT.)

UNIVAC | FEDERAL SYSTEMS DIVISION

APPLICATION FOR MANUAL REVISIONS

Upon receipt of this manual, please fill in the necessary data. It is impor-
tant that the addressee be the end user so that the operating personnel will
receive all revisions to the manual.

Comments concerning this manual may be set to Univac using the same address
which appears on the reverse side of this page.

EQUIPMENT NAME _____

SERIAL NO. _____ MODEL OR PART NO. _____

MANUAL TITLE _____

_____ MANUAL NUMBER _____

PURCHASING AGENCY _____

NAME OF USER _____

ADDRESS OF USER _____

_____

_____

ATTN: _____

CUT

FOLD

Postage
Will Be Paid
by
Addressee

No
Postage Stamp
Necessary
If Mailed in the
United States

BUSINESS REPLY MAIL
FIRST CLASS PERMIT NO. 1145   SAINT PAUL, MINNESOTA

UNIVAC
FEDERAL SYSTEMS DIVISION

UNIVAC PARK
P.O. BOX 3525
ST. PAUL, MINN. 55101

ATTN: ENGINEERING AND PROGRAMMING SUPPORT SERVICES DEPT., M.S. 8631

FOLD

SECTION I. UNIVAC 1218 MILITARY COMPUTER

This section provides an introduction to the UNIVAC 1218 Computer. It is not intended to serve as a detailed technical description of the UNIVAC 1218 Computer, but rather as a presentation of information that is essential to programming the UNIVAC 1218 Computer.

The section consists of three subsections which describe the following aspects of the UNIVAC 1218 Computer:

1) The general physical and functional characteristics of the computer.
2) The format and execution characteristics of all instructions.
3) The functional input/output characteristics.

SECTION I-A. DESCRIPTION OF COMPUTER

1. GENERAL CHARACTERISTICS

The UNIVAC ® 1218 Military Computer is a medium-scale, stored-program, general-purpose computer designed to provide high reliability under adverse operating environments. The UNIVAC 1218, hereinafter called simply the computer, utilizes modular design concepts in both the memory section and the input/output (I/O) section.

The computer is equipped with a 4-microsecond internal random access core memory in sizes of 4,096, 8,192, 16,384, and 32,768 18-bit words with a read access time of 1.8 microseconds. In addition to this, other random-access storage devices connected to I/O channels provide unlimited memory capacities. A portion (32 word locations) of core memory has a characteristic non-destructive feature which stores constants and instructions for automatic recovery from fault situations and for an initial load of routines.

The computer is designed with a modular I/O section which provides the option of either 4 or 8 I/O channels for communication with peripheral equipment or other computers. Each 4-channel module is available in either of two types of interface design. I/O communication is normally accomplished in an 18-bit parallel mode; however, single channels from two modules can be combined by switch setting into one 36-bit I/O channel.

Arithmetic operations can be performed on the basis of a single-length 18-bit word, or a double length 36-bit word if greater precision is required for compatibility with other computers. The repertoire of 98 instructions allows complete programming freedom in mathematical and logical computations, as well as full control of I/O buffer transfers and of real-time, on-line operations. The computer features buffered parallel transfers, one's complement binary arithmetic, direct addressing, and program-controlled automatic address or operand modification via eight control-memory-contained index registers.

The ability of the computer to process various applications concurrently is implemented by a program intervention system called interrupts. These interrupts may originate at some remote external device (external interrupts) or they may originate within the computer (internal interrupts). Since more than one may occur at the same time, the computer possesses a priority scheme with decision-making qualities so that it can select the branch of operation for solving the problem requiring the most urgent attention. Under program control the other interrupts may be honored, in turn, according to the next highest priority or they may be ignored. With this interrupt feature, real-time problem solution and maximum processing potential of the system is realized since less important routines can occupy the computer's surplus time.

2. PHYSICAL DESCRIPTION

The physical characteristics of the computer depend upon the size of memory and the number of I/O channels. The computer is housed in a single cabinet that contains the power supply, logic circuits, core memory, maintenance and control

panel, and a cooling system. Logic modules are encapsulated printed circuit
cards which plug into the wired chassis of easily accessible vertical pull-out
drawers. The cabinet contains either four or six of these drawers, as shown in
Figures I-A-1 and I-A-2, depending upon the memory and I/O channel options. The
front of each drawer is the associated portion of the computer control panel.
The power supply is mounted in a horizontal drawer at the bottom of the cabinet.
A list of the other physical characteristics follows.

2.1 APPROXIMATE SIZE AND WEIGHT

Height - 72 inches
Width - 25 inches (4-drawer)
       - 38 inches (6-drawer)
Depth - 30 inches
Weight - 834 pounds (4-drawer)
         1180 pounds (6-drawer)

2.2 ENVIRONMENT

Operating temperatures - 0°C to 50°C
Non-operating temperatures - -62°C to $+75^{o}$C
Humidity - relative humidity to 95 per cent

2.3 COOLING

Blower forced ambient air (water cooling optional).

2.4 POWER REQUIREMENTS

115-volt, $\pm$ 5 per cent, 3-phase, 400 cps, 1250 watts maximum and 115-volt, $\pm$ 10
per cent, single-phase, 60-cps, 208 watts.

3. FUNCTIONAL DESCRIPTION

Figure I-A-3 is a general block diagram of the computer. As indicated, the
computer has four major functional sections: control, memory, arithmetic, and
I/O.

3.1 CONTROL

The control section contains circuitry necessary to procure, modify, and execute
the single address instructions of a program stored in the core memory of the
computer. It controls parallel transfers of instructions and data. Direct or
indirect addressing capabilities and automatic address and operand modification
are directed by the control section translators and the timing of the synchron-
ous electronic master clock. This section controls all arithmetic, logical,
and sequential operations of the computer except those assigned to the I/O sec-
tion. It has facilities to permit an interruption of the running program when
certain real-time events require such interventions.
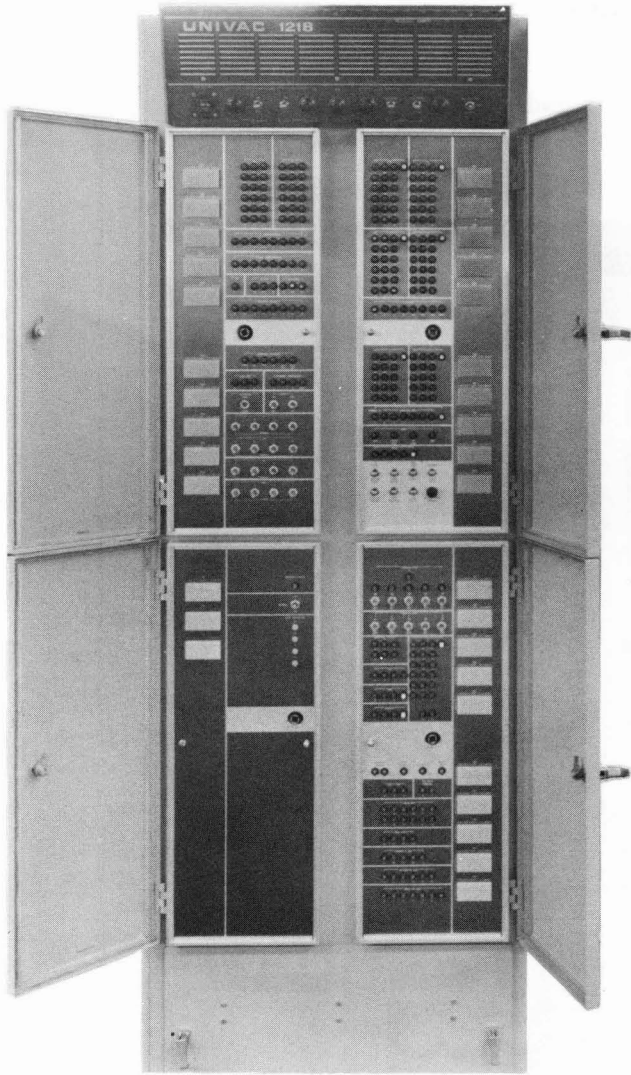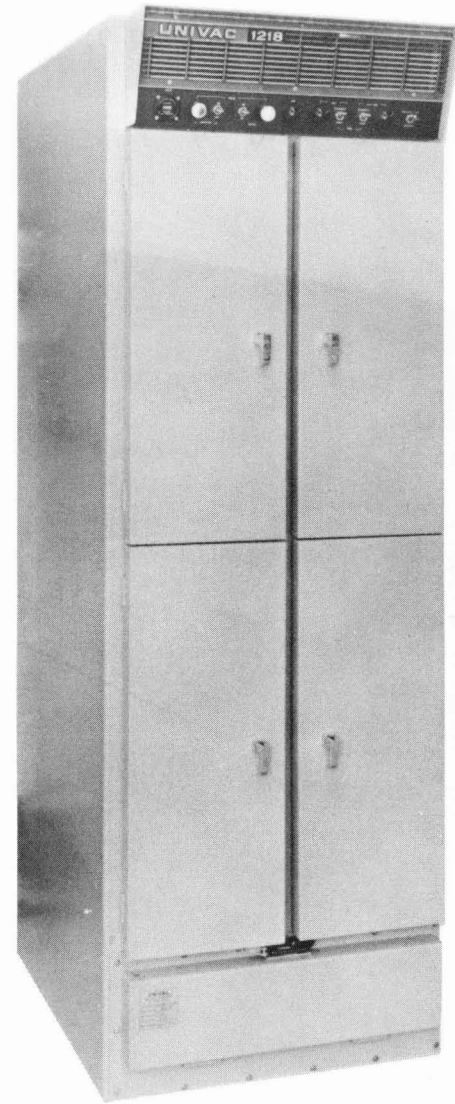
a. Doors Open

b. Doors Closed

Figure I-A-1. UNIVAC 1218 Computer (4 Drawer)
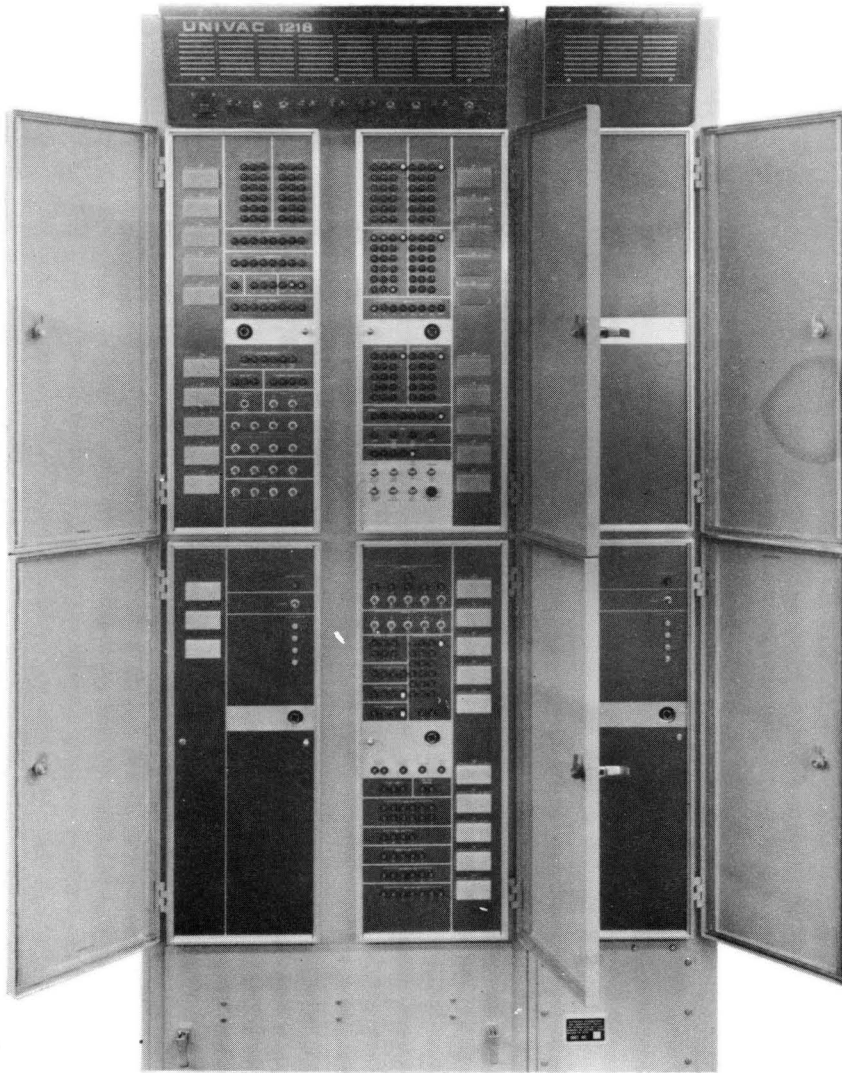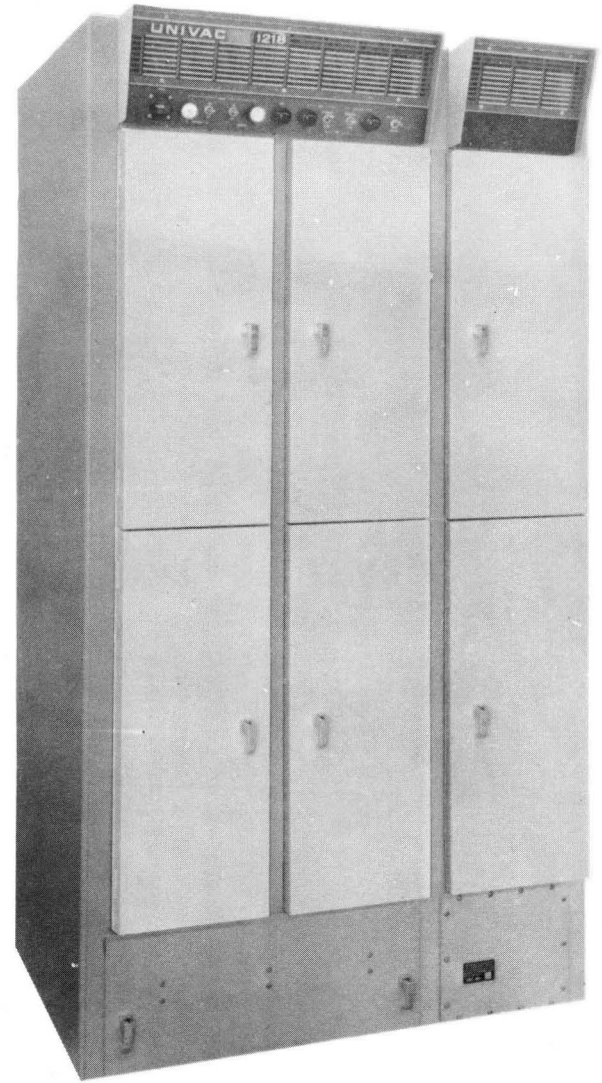
a) Doors Open   b. Doors Closed

Figure I-A-2.   UNIVAC 1218 Computer (6 Drawers)

```
      +---------------------+         +---------------------+
      |                     |         |                     |
      |                     |         |                     |
      |      Control        |         |     Arithmetic      |
      |      Section        |<------->|      Section        |
      |                     |         |                     |
      |                     |         |                     |
      |  Timing Function    |         | Mathematical and    |
      |                     |         | Logical Operations  |
      | Operation    Code   |         |                     |
      | Control   Translation|        |                     |
      +---------------------+         +---------------------+
```
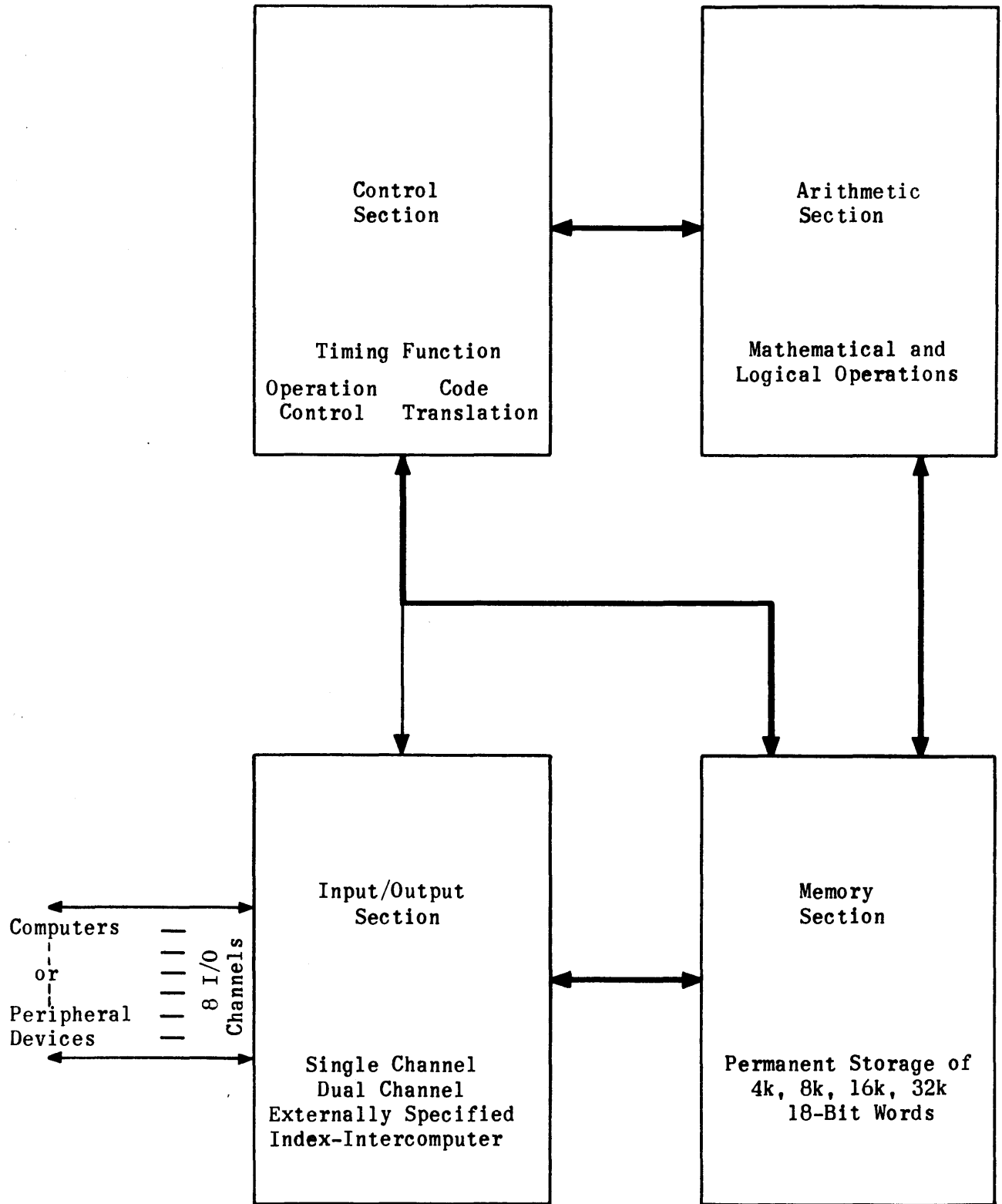

Figure I-A-3. Computer General Block Diagram

I-A-5

3.2 MEMORY

The computer memory consists of up to 32,768 18-bit words of addressable storage locations divided into two distinct sections in a continuous addressing structure. The two sections are bootstrap memory and main memory. The master clock in the computer controls and synchronizes all operations performed by the various sections through the electronic timing chains allotted to them. The read/restore cycle time of main memory is 4 microseconds. All control and timing sequences for the various functions the computer performs are based on this 4-microsecond cycle. An instruction from main memory storage can be transferred to the control section for execution in approximately 1.8 microseconds.

3.2.1 BOOTSTRAP MEMORY

The computer is provided with 32_{10} nondestructive readout memory locations (00200_8 through 00237_8) which contain computer instructions and constants for an initial load program (bootstrap). This provides the ability to enter an initial package of utility routines that may be used to load and/or debug more sophisticated programs. These memory locations have unique characteristics since they are transformer cores which operate in a special type of non-destructive readout mode. They are not accessible to the programmer for store-type instructions.

3.2.2 MAIN MEMORY

Main memory consists of 18-bit addressable core storage locations with a read-restore cycle time of 2 microseconds. All locations are accessible to the programmer at random and to all sections of the computer on a time shared basis. Some locations are allocated for specific purposes, as defined in Table I-A-1. The number of main memory locations is equal to the total memory size minus the locations used for bootstrap memory.

3.3 ARITHMETIC

The arithmetic section, which contains a subtractive type adder, performs all the arithmetic and logical operations for the computer under direction of the function code translator and I/O control. The arithmetic section and memory are shared by the control section and the I/O section. After an instruction is executed, the I/O section may gain control of memory for an input or output transfer on an active channel. It is also supplied to the S register of memory control for the reference required to transfer the word. The arithmetic section is used by control for any address or operand modification requested by an instruction and for overflow detection if overflow exists at the completion of any arithmetic instruction except multiply.

3.4 INPUT/OUTPUT (I/O)

The I/O section of the computer controls the communications between the computer and the peripheral devices connected in a system. There may be up to 8 peripheral equipments connected directly to the computer. Communication, the passing of one computer word from or to the computer, can be carried on with only one peripheral device at a time; however, communications on several channels may be interlaced.

The computer may also communicate with other computers. To do so, the connecting I/O channel must be manually switched into intercomputer operation. Communication with computers or peripheral devices that have word lengths greater than 18 bits (36-bit maximum) is possible by using two adjacent I/O channels in combination.

Besides handling all information transfer between the computer and other devices, the I/O section of the computer handles all interrupts. Interrupts provide the means to intervene in program operation, thus giving the computer real-time and fault detection and correction capabilities.

Because a complete understanding of the I/O operations is essential to programming the computer, a separate section entitled Input/Output (I/O) Characteristics is included in this document.

3.5 REGISTERS AND THEIR CONTENTS

All registers in the computer may be classified as addressable or non-addressable. Only the registers separate from the normal core storage registers are discussed here. Addressable registers are directly available to the programmer through computer instructions. The other functional registers are non-addressable.

3.5.1 ADDRESSABLE REGISTERS

A   A 36-bit arithmetic accumulator which:

1) Contains the product of two 18-bit quantities.
2) Contains the 36-bit dividend for a divide instruction.
3) Is used as an accumulator for double length arithmetic and logical functions.
4) Has shifting capabilities and complementing capabilities.

AU   The upper accumulator (most significant 18 bits) of A which:

1) Contains a mask for logical instructions.
2) Captures the remainder for the divide process.
3) Has shifting capabilities.
4) Has complementing capabilities.

AL   The lower accumulator (least significant 18 bits) of A which:

1) Is used as the main accumulator for the arithmetic section for all functions.
2) Contains quotient for the divide process; contains sum for add.
3) Has shifting capabilities.
4) Has complementing capabilities.

B   The contents of the active index register in control memory which are used to modify y to form an address or an operand in every odd-numbered instruction less than 50_8. B is an 18-bit one's complement number that may be used to increment or decrement. When the quantity, $y + B$, is used as an address, only the number of lower order bits sufficient to fill the S register is transmitted.

ICR   An index control register (3 bits) contains the index register identifier currently active in address or operand modification requested by instructions. Any one of eight index registers may be selected by the numerical value entered into this register by the program.

(P)   The contents of the program address register, P, that is, the address of the instruction currently being entered for execution, is incremented by one in the arithmetic section as soon as the instruction is transferred from memory. If the computer is stopped, the P register exhibits the address of the next instruction, (P) + 1. This is incremented by one again if the condition stated by a SKIP instruction is satisfied. When the current instruction is a return jump, (P) + 1 is stored in the core location specified by the instruction, and the entrance address of the new routine is entered into the program address register.

     When the return jump is the result of an interrupt, (P) is stored in the core location specified by the instruction since the interrupt condition does not initiate the (P) + 1 sequence.

SR   A 4-bit special register, SR, through which the program has control of the 4,096-word modules in core memory (in all instructions numbered under 50_8 except jump and enter constant or add constant instructions). When the 2^3 bit contains one, the remaining bits of SR are used to extend u for an address instead of the upper bits of P. If the 2^3 bit of SR is zero, the most significant bits of P extend u for the address. Therefore, y (the address) equal to u_P or u_{SR} is determined by the 2^3 bit of SR (active if = 1), refer to Section I-B.

3.5.2   NON-ADDRESSABLE REGISTERS

CO
and
CE   Two 18-bit output buffer registers for transferring data or instruction words (external function) to external devices which may include other computers. The CO register is the buffer register for the odd-numbered channels (1, 3, 5, and 7) and the CE register is for the even-numbered channels (0, 2, 4, and 6). These two output registers may be linked in consecutive even-odd pairs to permit 36-bit parallel output transfers when words larger than 18 bits are desired.

D   An 18-bit arithmetic exchange register holds an operand for the adder during arithmetic operations.

F   A 7-bit function register holds the function code of the instruction being executed. The low order six bits hold the function code (f for Format I instructions and m for Format II instructions). The most significant bit is set for Format II instructions only. Computer control is directed from this register.

S   An address register receives the address of a memory location at the beginning of a memory cycle and holds it to control the translators and circuitry throughout the read/write cycle. The S register may receive its address from the I/O section (which generates certain assigned addresses), the control section, the arithmetic section, or from an input channel connected to a device capable of specifying an address.

X An 18-bit exchange or communication register in the arithmetic section which receives operands for arithmetic and logical instructions.

Z An 18-bit main memory buffer register for all transfers to and from core memory. The Z register communicates with all other sections of the computer since core memory may contain instructions, control words, and data.

4. SUMMARY OF TECHNICAL CHARACTERISTICS

4.1 MEMORY

Cycle Time:	4 microseconds (1.8 microsecond access)
Capacity:	4,096; 8,192; 16,384; or 32,768 18-bit words
Type:	Coincident current, magnetic core
Organization:	Addresses 00000 through 00177 allocated to index registers, I/O control registers, and interrupt registers
	Addresses 00200 through 00237 allocated to bootstrap memory; a 32-word transformer core, nondestructive readout memory used for boot-strap (initial load) program storage. Addresses 00240 to end of memory are used for program and data storage

4.2 INPUT/OUTPUT (I/O)

4.2.1 CHANNELS

Four input and four output or eight input and eight output channels are available. Each channel provides 18 parallel data lines plus necessary control lines. Channels can be paired to form 36-bit dual channels.

4.2.2 BUFFERED TRANSFERS

All input/output transfers are fully buffered, do not require program attention, and operate asynchronously at the rate required by the external device. Control words guide the active buffers by defining the memory location, buffer direction and monitor.

4.2.3 OPERATING MODES

Normal Single Channel:	18-bit parallel data transfers.
Normal Dual Channel:	36-bit parallel data transfers.

Externally Specified Index
(Dual Channel):   18-bit parallel data transfers with data
                                       storage index address specified by external
                                       device; can be used to automatically multiplex
                                       data to or from unique buffer locations.

Intercomputer Single Channel: 18-bit parallel data transfers allow
                                       communication with other UNIVAC 18-bit
                                       computers (direct cable connection).

Intercomputer Dual Channel: 30/36 bit parallel data transfers allow
                                       communication with the other computers such
                                       as the UNIVAC ® 1230, 1107, and 490 Computers.

4.2.4 TRANSFER TIMES

Maximum Input/Output
Transfer Rate: 100 kc, 18-bit words
                                       16 usec per 18-bit word (input or output)
                                       20 usec per 30/36-bit word (input or output)

4.2.5 INTERRUPTS

33 unique interrupts are provided as follows:

1 synchronizing interrupt (not channel associated)
8 external interrupts (one per channel)
8 external function monitor interrupts (one per channel)
8 output monitor interrupts (one per channel)
8 input monitor interrupts (one per channel)

4.2.6 PRIORITY

Priority of interrupts is according to function as listed above. Sub-priority
is established according to channel numbers 7 through 0.

4.2.7 PROGRAM CONTROL

Eighteen program instructions are devoted to the control of input/output,
providing positive control and a high degree of sophistication in programming.

4.3 ARITHMETIC

Organization: 18-bit, parallel, one's complement, integer,
                                       binary.

Registers: Two 18-bit, addressable.

Functions: Arithmetic operations including double length
                                       add and subtract. Logical and bit manipulation.

Instruction Execution Times: Add, subtract, logical: 8 usec
Multiply: 38 usec (average); 48 usec (maximum)
Divide: 48 usec (maximum)
Add, double length: 12 usec
Compare/mask compare and branch: 12 usec
Register shifts: right, left, single, double:
$(4 + 0.67n)$ usec
Where n = number of places shifted. Instruction
times represent total execution including instruc-
tion and operand acquisition. Whenever address
modification is desired, add four microseconds.

4.4 CONTROL

Instruction Repertoire: 98 single-address instructions.

Address Modification: Via eight memory contained index registers.

Synchronizing: Either internal or external sync selectable
by control panel switch setting.
Internal sync provides an interrupt every
1/1024 second.
External sync provides capability for variable-
granularity clock or high priority alarm
recognition.

TABLE I-A-1. MEMORY ADDRESS ALLOCATION

Address	Storage Function
000000	Fault Interrupt, Entrance Address
000001	B1, Index Register
000002	B2, Index Register
000003	B3, Index Register
000004	B4, Index Register
000005	B5, Index Register
000006	B6, Index Register
000007	B7, Index Register
000010	B0, Index Register
000011	Memory Word
000012	Memory Word
000013	Memory Word
000014	Memory Word
000015	Memory Word
000016	Synchronizing Interrupt and Real-Time Clock Entrance Address
000017	Scale Factor Shift Count Word
(000020-000037)	External Function Buffer Control (EFCB) Registers

TABLE I-A-1. MEMORY ADDRESS ALLOCATION (CONT.)

Address	Storage Function
000020	EFBC for Channel 0, Terminal Address Word
000021	EFBC for Channel 0, Current Address Word
000022	EFBC for Channel 1, Terminal Address Word
000023	EFBC for Channel 1, Current Address Word
000024	EFBC for Channel 2, Terminal Address Word
000025	EFBC for Channel 2, Current Address Word
000026	EFBC for Channel 3, Terminal Address Word
000027	EFBC for Channel 3, Current Address Word
000030	EFBC for Channel 4, Terminal Address Word
000031	EFBC for Channel 4, Current Address Word
000032	EFBC for Channel 5, Terminal Address Word
000033	EFBC for Channel 5, Current Address Word
000034	EFBC for Channel 6, Terminal Address Word
000035	EFBC for Channel 6, Current Address Word
000036	EFBC for Channel 7, Terminal Address Word
000037	EFBC for Channel 7, Current Address Word
(000040–000057)	Output Buffer Control (OBC) Registers
000040	OBC for Channel 0, Terminal Address Word
000041	OBC for Channel 0, Current Address Word
000042	OBC for Channel 1, Terminal Address Word
000043	OBC for Channel 1, Current Address Word
000044	OBC for Channel 2, Terminal Address Word
000045	OBC for Channel 2, Current Address Word
000046	OBC for Channel 3, Terminal Address Word
000047	OBC for Channel 3, Current Address Word
000050	OBC for Channel 4, Terminal Address Word
000051	OBC for Channel 4, Current Address Word
000052	OBC for Channel 5, Terminal Address Word
000053	OBC for Channel 5, Current Address Word
000054	OBC for Channel 6, Terminal Address Word
000055	OBC for Channel 6, Current Address Word
000056	OBC for Channel 7, Terminal Address Word
000057	OBC for Channel 7, Current Address Word
(000060–000077)	Input Buffer Control (IBC) Registers
000060	IBC for Channel 0, Terminal Address Word
000061	IBC for Channel 0, Current Address Word
000062	IBC for Channel 1, Terminal Address Word
000063	IBC for Channel 1, Current Address Word
000064	IBC for Channel 2, Terminal Address Word
000065	IBC for Channel 2, Current Address Word

TABLE I-A-1. MEMORY ADDRESS ALLOCATION (CONT.)

Address	Storage Function
000066	IBC for Channel 3, Terminal Address Word
000067	IBC for Channel 3, Current Address Word
000070	IBC for Channel 4, Terminal Address Word
000071	IBC for Channel 4, Current Address Word
000072	IBC for Channel 5, Terminal Address Word
000073	IBC for Channel 5, Current Address Word
000074	IBC for Channel 6, Terminal Address Word
000075	IBC for Channel 6, Current Address Word
000076	IBC for Channel 7, Terminal Address Word
000077	IBC for Channel 7, Current Address Word
(000100-000117)	External Interrupt (EI) Registers
000100	EI for Channel 0, Entrance Address
000101	EI for Channel 0, Interrupt Word
000102	EI for Channel 1, Entrance Address
000103	EI for Channel 1, Interrupt Word
000104	EI for Channel 2, Entrance Address
000105	EI for Channel 2, Interrupt Word
000106	EI for Channel 3, Entrance Address
000107	EI for Channel 3, Interrupt Word
000110	EI for Channel 4, Entrance Address
000111	EI for Channel 4, Interrupt Word
000112	EI for Channel 5, Entrance Address
000113	EI for Channel 5, Interrupt Word
000114	EI for Channel 6, Entrance Address
000115	EI for Channel 6, Interrupt Word
000116	EI for Channel 7, Entrance Address
000117	EI for Channel 7, Interrupt Word
(000120-000137)	External Function Monitor Interrupt (EFMI) Registers
000120	EFMI for Channel 0, Entrance Address
000121	Memory Word
000122	EFMI for Channel 1, Entrance Address
000123	Memory Word
000124	EFMI for Channel 2, Entrance Address
000125	Memory Word
000126	EFMI for Channel 3, Entrance Address
000127	Memory Word
000130	EFMI for Channel 4, Entrance Address
000131	Memory Word
000132	EFMI for Channel 5, Entrance Address
000133	Memory Word

TABLE I-A-1. MEMORY ADDRESS ALLOCATION (CONT.)

Address	Storage Function
000134	EFMI for Channel 6, Entrance Address
000135	Memory Word
000136	EFMI for Channel 7, Entrance Address
000137	Memory Word
(000140- 000157)	Output Monitor Interrupt (OMI) Registers
000140	OMI for Channel 0, Entrance Address
000141	Memory Word
000142	OMI for Channel 1, Entrance Address
000143	Memory Word
000144	OMI for Channel 2, Entrance Address
000145	Memory Word
000146	OMI for Channel 3, Entrance Address
000147	Memory Word
000150	OMI for Channel 4, Entrance Address
000151	Memory Word
000152	OMI for Channel 5, Entrance Address
000153	Memory Word
000154	OMI for Channel 6, Entrance Address
000155	Memory Word
000156	OMI for Channel 7, Entrance Address
000157	Memory Word
(000160- 000177)	Input Monitor Interrupt (IMI) Registers
000160	IMI for Channel 0, Entrance Address
000161	Memory Word
000162	IMI for Channel 1, Entrance Address
000163	Memory Word
000164	IMI for Channel 2, Entrance Address
000165	Memory Word
000166	IMI for Channel 3, Entrance Address
000167	Memory Word
000170	IMI for Channel 4, Entrance Address
000171	Memory Word
000172	IMI for Channel 5, Entrance Address
000173	Memory Word
000174	IMI for Channel 6, Entrance Address
000175	Memory Word
000176	IMI for Channel 7, Entrance Address
000177	Memory Word
(00200- 00237)	Nondestructive Readout Memory Allocations

TABLE I-A-1. MEMORY ADDRESS ALLOCATION (CONT.)

Address	Storage Function
00200	Bootstrap Word 1
00201	Bootstrap Word 2
00202	Bootstrap Word 3
00203	Bootstrap Word 4
00204	Bootstrap Word 5
00205	Bootstrap Word 6
00206	Bootstrap Word 7
00207	Bootstrap Word 8
00210	Bootstrap Word 9
00211	Bootstrap Word 10
00212	Bootstrap Word 11
00213	Bootstrap Word 12
00214	Bootstrap Word 13
00215	Bootstrap Word 14
00216	Bootstrap Word 15
00217	Bootstrap Word 16
00220	Bootstrap Word 17
00221	Bootstrap Word 18
00222	Bootstrap Word 19
00223	Bootstrap Word 20
00224	Bootstrap Word 21
00225	Bootstrap Word 22
00226	Bootstrap Word 23
00227	Bootstrap Word 24
00230	Bootstrap Word 25
00231	Bootstrap Word 26
00232	Bootstrap Word 27
00233	Bootstrap Word 28
00234	Bootstrap Word 29
00235	Bootstrap Word 30
00236	Bootstrap Word 31
00237	Bootstrap Word 32
(00240-77777)	Instruction word and data storage organized in 10000_8-word banks with the upper octal character of address specifying the bank address (00240 - 07777 = remainder of bank 0; 10000 - 17777 = bank 1; 20000 - 27777 = bank 2).

SECTION I-B. COMPUTER INSTRUCTIONS

1. GENERAL

The computer has a repertoire of 98 instructions which generally fall into
eight categories: transfer, arithmetic, shift, logical, modifying, jump, skip
and stop, and I/O. Each of these categories is covered under a separate sub-
section of this section I-B. Certain instructions perform functions applicable
to more than one category; therefore, they are listed under both categories.
For example, an instruction which skips on a specific I/O condition is listed
under skip and stop instructions and also under I/O instructions.

The word formats applicable to all instructions and the symbol conventions used
to describe the instructions are defined in the following paragraphs.

2. WORD FORMATS

All instructions conform to one of the two basic word formats.

2.1 FORMAT I

17	12	11	0

$$\underbrace{\qquad\qquad}_{f}\quad\underbrace{\qquad\qquad}_{u}$$
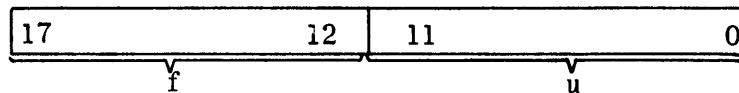
f: function code, six high order bits.
u: twelve low order bits.

Legal function codes for Format I instructions are 02 through 47_8 and 51_8
through 76_8. The definition and usage of u are determined by the function code
utilizing u in two distinct manners:

1) u Used as a Constant.

   In this case, u itself is the operand and requires no further memory
   reference; however, u is extended to 18 bits. (Refer to paragraph 4,
   entitled "Instructions".)

2) u Used as an Address.

   In this case, u is used as the lower order 12 bits of an address refer-
   ring to a memory cell within a 10000_8-word bank. The entire address is
   15 bits, designated as u_P or u_{SR}, and is described below.

u_P is defined as a 15-bit address whose three high order bits consist of the
three higher order bits of P and whose twelve low order bits are u.

u_P =

14	13	12	11	0

$$\underbrace{\qquad}_{P\ 14\text{-}12}\quad\underbrace{\qquad\qquad}_{u}$$
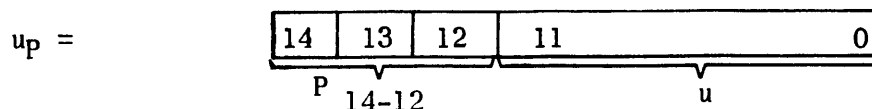
The bits supplied by P specify the bank address.

u_{SR} is defined as a 15-bit address whose three high order bits consists of the three lower order bits and the high order bit of SR and whose twelve low order bits are u. The bits supplied by SR specify the bank address.

u_{SR} =

14	13	12	11		0

SR_{2-0} u
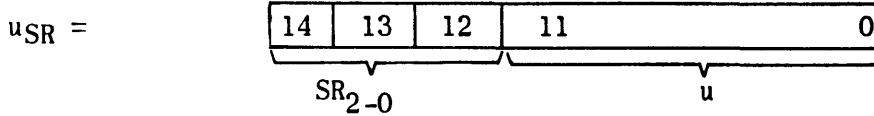
Certain Format I instructions allow the use of either u_p or u_{SR} as the operand address; for these instructions u_{SR} is used if SR is ACTIVE and u_p is used whenever SR is INACTIVE. These instructions are identified specifically in the detailed description of the instructions.

2.2 FORMAT II

17		12	11		6	5		0

f   m   k
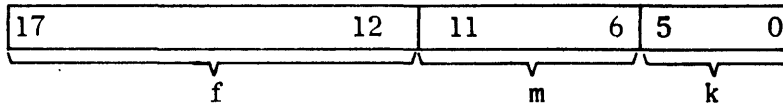
f: function code, six high order bits.
m: minor function code, six bits.
k: operand designator, six bits.

The function code for Format II instructions is always 50_8. The minor function code determines the type of operation to be performed. Format II instructions perform a variety of operations and can be classified in two instruction categories:

1)   No Memory Address Needed.

   In this case, the information existing in the computer's arithmetic or control registers and the operand designator, k, are sufficient to perform the specified operation.

2)   Initiate I/O Buffer.

   In this case, the two memory cells immediately following the instruction are used to contain the buffer control words. The complete instruction must therefore occupy three sequential memory cells (refer to I/O instructions).

3.   SYMBOL CONVENTIONS

The following symbols are used to aid in describing the instructions in the succeeding subsections,

AU	Upper accumulator, 18-bit arithmetic register.
AL	Lower accumulator, 18-bit arithmetic register.
A	AU and AL linked together to form one 36-bit arithmetic register.
B	Contents of the active index register, 18-bit one's complement.
f	Function code, high order six bits of all instruction words.

F	Function register, seven bits.
k	Designator contained in Format II instructions, six bits.
m	Minor function code contained in Format II instructions, six bits.
M	Memory word specified by (y), (y + B), L(y)(AU) or L (y + B)(AU) of compare instruction.
NI	Next instruction.
P	The program address register.
SR	Special register, 4-bit core memory bank designator.
u	The low order 12 bits contained in Format I instruction words.
u_p	u prefaced with core memory bank designator bits of P.
u_{SR}	u prefaced with core memory bank designator bits of SR.
y	u extended or u_p or u_{SR}.
Y	The address or constant formed by y or y + B with or without sign extension.
( )	Contents of the address or register.
( )$_i$	Initial contents of the address or register.
( )$_f$	Final contents of the address or register.
( )$_n$	Designates any single nth bit of the contents of a register.
(Y + 1, Y)	Designates the contents of two consecutive memory locations linked together to form a 36-bit word. Address Y + 1 contains the most significant half of the word while address Y contains the least significant half.
:	The colon in a logical expression indicates comparison.
L( ) ( ) or ( ) ( )	The bit-by-bit or logical product (logical AND) defined by:

$$\begin{array}{c|cc} & 0 & 1 \\ \hline 0 & 0 & 0 \\ 1 & 0 & 1 \end{array}$$

( ) v ( )	Logical sum, or inclusive OR defined by:

$$\begin{array}{c|cc} & 0 & 1 \\ \hline 0 & 0 & 1 \\ 1 & 1 & 1 \end{array}$$

( )$\oplus$( )	Half add, half subtract, or exclusive OR defined by:

$$\begin{array}{c|cc} & 0 & 1 \\ \hline 0 & 0 & 1 \\ 1 & 1 & 0 \end{array}$$

( )' or $\overline{(\ )}$	The one's complement of the contents of the address or register
( )( )	Algebraic product of the contents of two locations
(Y)	When the contents of Y are used as an address, only that lower portion of the word that can be contained in S is transferred
$\longrightarrow$	Transfer the quantity stated at the left of the symbol to the address or register stated at the right of the symbol.
	Console and control panel are used to designate I/O console or the computer control panel.
xY	x preceding some symbol indicates that the sign of the 12-bit constant has been extended to produce an 18-bit word, that is:

xY =

u_{11} ------ u_{11}	11 0
6 bits	12 bits
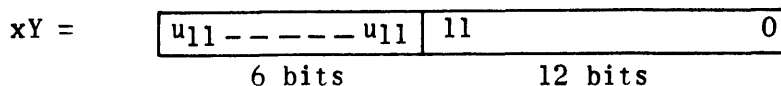
4. INSTRUCTIONS

Table I-B-1 summarizes the complete repertoire of 102 computer instructions in the order of their function codes. The table lists the mnemonic symbol, a brief description of the operation performed, the execution time, and the instruction type. Each instruction is described in detail in subsections I-B-1 through I-B-8 which are organized according to instruction type. Function codes which are not listed in Table I-B-1 are classified as either illegal or not used.

Illegal function codes are: 00, 01, 77, 5000, 5001, and 5077.

If an illegal function code is encountered in a program, program control is transferred to either the fault interrupt entrance address, 000000, or to the first address of the bootstrap program, 00200, depending upon the position of the AUTO RECOVERY switch on the control panel. When the switch is in the up position, program control is transferred to address 00200 and the bootstrap program automatically loads any program previously mounted in the appropriate I/O device. When the switch is in the down position, program control is transferred to address 000000, which may contain a jump to a sequence of instructions designed to identify the fault instruction.

Function codes which are not used are: 5002 through 5010, 5014, 5040, 5064 through 5071, and 5074 through 5076.

If a not-used function code is encountered in a program, a 4-microsecond instruction sequence occurs but no operation is performed. The program then proceeds to the next instruction.

TABLE I-B-1. REPERTOIRE OF INSTRUCTIONS

Function Code	Mnemonic Symbol	Description	Time (usec)	Instruction Type
02	CMAL	Compare Y	8	Logical
03	CMALB	Compare Y + B	12	
04	SLSU	Selective Substitute	8	
05	SLSUB	Selective Substitute Y + B	12	
06	CMSK	Masked Compare Y	8	
07	CMSKB	Masked Compare Y + B	12	
10	ENTAU	Enter AU, Y	8	Transfer
11	ENTAUB	Enter AU, Y + B	12	
12	ENTAL	Enter AL, Y	8	
13	ENTALB	Enter AL, Y + B	12	
14	ADDAL	Add Y, 18 bit	8	Arithmetic
15	ADDALB	Add Y + B, 18 bit	12	
16	SUBAL	Subtract Y, 18 bit	8	
17	SUBALB	Subtract Y + B, 18 bit	12	
20	ADDA	Add Y, 36 bit	12	

TABLE I-B-1.   REPERTOIRE OF INSTRUCTIONS (CONT.)

Function Code	Mnemonic Symbol	Description	Time (usec)	Instruction Type
21	ADDAB	Add Y + B, 36 bit	16	Arithmetic
22	SUBA	Subtract Y, 36 bit	12	
23	SUBAB	Subtract Y + B, 36 bit	16	
24	MULAL	Multiply Y	26-49	
25	MULALB	Multiply Y + B	30-53	
26	DIVA	Divide, Y	48	
27	DIVAB	Divide, Y + B	52	
30	IRJP	Indirect RJP, Y	12	Jump
31	IRJPB	Indirect RJP, Y + B	16	Jump
32	ENTB	Enter B, Y	12	Transfer
33	ENTBB	Enter B, Y + B	16	Transfer
34	JP	Jump, Y	4	Jump
35	JPB	Jump, Y + B	8	Jump
36	ENTBK	Enter, B, U	8	Transfer
37	ENTBKB	Modify, B, U	12	Modifying
40	CL	Store Zero, Y	8	Transfer
41	CLB	Store Zero, Y + B	12	
42	STRB	Store, B, Y	12	
43	STRBB	Store B, Y + B	12	
44	STRAL .	Store AL, Y	8	
45	STRALB	Store AL, Y + B	12	
46	STRAU	Store AU, Y	8	
47	STRAUB	Store AU, Y + B	12	
51	SLSET	Selective Set (IOR), Y	8	Logical
52	SLCL	Selective Clear (AND), Y	8	Logical
53	SLCP	Selective Complement (XOR), Y	8	Logical
54	IJPEI	Indirect Jump (RIL), Y	8	Jump
55	IJP	Indirect Jump, Y	8	Jump
56	BSK	Increment B, Skip, Y	16	Modifying
57	ISK	Decrement Index, Skip, Y	12	Modifying
60	JPAUZ	JP if (AU) = 0, Y	4	Jump
61	JPALZ or JPEQ	JP if (AL) = 0 or (AL) = M, Y	4	
62	JPAUNZ	JP if (AU) $\neq$ 0, Y	4	
63	JPALNZ or JPNOT	JP if (AL) $\neq$ 0 or (AL) $\neq$ M, Y	4	
64	JPAUP	JP if AU_{17} = 0, Y	4	
65	JPALP or JPMLEQ	JP if AL_{17} = 0 or M $\leq$ (AL), Y	4	
66	JPAUNG	JP if AU_{17} = 1, Y	4	
67	JPLNG or JPMGR	JP if AL_{17} = 1 or M $\geq$ (AL), Y	4	

TABLE I-B-1.   REPERTOIRE OF INSTRUCTIONS (CONT.)

Function Code	Mnemonic Symbol	Description	Time (usec)	Instruction Type
70	ENTALK	Enter AL, Y	4.67	Transfer
71	ADDALK	Add U, 12 bits	4.67	Arithmetic
72	STRICR	Store ICR, Y	8	Transfer
73	BJP	Decrement B, Jump, Y	12	Modifying
74	STRADR	Store Address, Y	8	Transfer
75	STRSR	Store SR, Deactivate	8	Transfer
76	RJP	Return Jump, Y	8	Jump
5011	IN	Initiate Input Buff, k	20	I/O
5012	OUT	Initiate Output Buff, k	20	
5013	EXF	External Function	20	
5015	INSTP	Terminate Input, k	4	
5016	OUTSTP	Terminate Output, k	4	
5017	EXFSTP	Terminate External Function, k	4	
5020	SRSM	Set Resume ff (Intercomp)	4	
5021	SKPIIN	Skip Input Inact, k	6	
5022	SKPOIN	Skip Output, Inact, k	6	
5023	SKPFIN	Skip Ext Fnct Inact	6	
5024	WTFI	Wait for Interrupt	4	I/O
5026	OUTOV	Force Output One Word, k	4.67	
5027	EXFOV	Force External Function One Word, k	4.67	
5030	RIL	Remove Interrupt Lockout	4	
5032	RXL	Remove External Interrupt Lockout	4	
5034	SIL	Set Interrupt Lockout	4	
5036	SXL	Set External Interrupt Lockout	4	
5041	RSHAU	Right Shift AU, k	18	Shift
5042	RSHAL	Right Shift AL, k	18	
5043	RSHA	Right Shift A, k	24	
5044	SF	Scale A Left, k, SF	24	
5045	LSHAU	Left Shift AU, k	18	
5046	LSHAL	Left Shift AL, k	18	
5047	LSHA	Left Shift A, k	24	
5050	SKP	Skip Console Key, k	6	Skip
5051	SKPNBO	Skip No Borrow	6	
5052	SKPOV	Skip Overflow	6	
5053	SKPNOV	Skip No Overflow	6	
5054	SKPODD	Skip L(AU, AL) Odd Parity	6	Skip
5055	SKPEVN	Skip L(AU, AL) Even Parity	6	Skip

TABLE I-B-1.   REPERTOIRE OF INSTRUCTIONS (CONT.)

Function Code	Mnemonic Symbol	Description	Time (usec)	Instruction Type
5056	STOP	Stop Console Key, k	4.67	Stop
5057	SKPNR	Skip No Resume ff (Intercomp)	6	Skip
5060	RND	Round AU	5.33	Arithmetic
5061	CPAL	Complement AL	5.33	Logical
5062	CPAU	Complement AU	5.33	↓
5063	CPA	Complement A	5.33	
5072	ENTICR	Enter ICR, k	4	Transfer
5073	ENTSR	Enter, SR, k	4	Transfer

SECTION I-B-1. TRANSFER INSTRUCTIONS

1. GENERAL

Transfer instructions either transfer data from a memory storage location to a register or store the contents of a register in a memory location.

2. INSTRUCTIONS

| 10 | ENTER AU (ENTAU) | $(Y) \longrightarrow AU$ |

Execution time: 8 microseconds

$y = u_P$ or u_{SR}
Clear AU. Then transmit (y) to AU.

| 11 | ENTER AU (ENTAUB) | $(Y) \longrightarrow AU$ |

Execution time: 12 microseconds

$y = u_P$ or u_{SR}
Clear AU. Then transmit (y + B) to AU.

| 12 | ENTER AL (ENTAL) | $(Y) \longrightarrow AL$ |

Execution time: 8 microseconds

$y = u_P$ or u_{SR}
Clear AL. Then transmit (y) to AL.

| 13 | ENTER AL (ENTALB) | $(Y) \longrightarrow AL$ |

Execution time: 12 microseconds

$y = u_P$ or u_{SR}
Clear AL. Then transmit (y + B) to AL.

| 32 | ENTER B (ENTB) | $(Y) \longrightarrow B$ |

Execution time: 12 microseconds

$y = u_P$ or u_{SR}
Transmit (y) to B
The full 18 bits of (y) are transmitted
to the B register (a normally addressable
core cell).

| 33 | ENTER B (ENTBB) | $(Y) \longrightarrow B$ |

Execution time: 16 microseconds

$y = u_P$ or u_{SR}
Transmit (y + B) to B_{ICR}
The full 18 bits (y + B) are transmitted
to the B register (a normally addressable
core cell).

36   ENTER B WITH CONSTANT (ENTBK) $xY \longrightarrow B$

Execution time: 8 microseconds

$y = u$ (sign extended to 18 bits)
Clear B. Then transmit y to B.
NOTE: u is a 12-bit one's complement
        number contained within the in-
        struction; it does not refer
        to an address. Example of
        enter B with constant when
        u = 7701:
        B_i = any value
        B_f = 777701

40   CLEAR Y (STORE ZERO) (CL) $0 \longrightarrow Y$

Execution time: 8 microseconds

$y = u_p$ or u_{SR}
Store an 18-bit word of zeros at
storage address y.

41   CLEAR Y (STORE ZERO) (CLB) $0 \longrightarrow Y$

Execution time: 12 microseconds

$y = u_p$ or u_{SR}
Store an 18-bit word of zeros at
storage address y + B.

42   STORE B (STRB) $B \longrightarrow Y$

Execution time: 12 microseconds

$y = u_p$ or u_{SR}
Store B at storage address y.

43   STORE B (STRBB) $B \longrightarrow Y$

Execution time: 16 microseconds

$y = u_p$ or u_{SR}
Store B at storage address y + B.

44   STORE AL (STRAL) $(AL) \longrightarrow Y$

Execution time: 8 microseconds

$y = u_p$ or u_{SR}
Store (AL) at storage address y.
$(AL)_f = (AL)_i$

45   STORE AL (STRALB) $(AL) \longrightarrow Y$

Execution time: 12 microseconds

$y = u_p$ or u_{SR}
Store (AL) at storage address y + B.
$(AL)_f = (AL)_i$

45   STORE AU (STRAU)   (AU) ⟶ Y

Execution time: 8 microseconds

$y = u_p$ or u_{SR}
Store (AU) at storage address y.
$(AU)_f = (AU)_i$

17   STORE AU (STRAUB)   (AU) ⟶ Y

Execution time: 12 microseconds

$y = u_p$ or u_{SR}
Store (AU) at storage address y + B.
$(AU)_f = (AU)_i$

50 72   ENTER INDEX CONTROL REGISTER (ENTICR) k_{2-0} ⟶ ICR

Execution time: 4 microseconds

Clear the index control register. Then
transmit the three low order bits of k
to the ICR.

50 73   ENTER SPECIAL REGISTER (ENTSR) k_{3-0} ⟶ SR

Execution time: 4 microseconds

Clear the special register. Then
transmit the four low order bits of k
to the SR. ($SR_3 = 1$ activates the SR.)

70   ENTER AL WITH CONSTANT (ENTALK)   xY ⟶ AL

Execution time: 4.67 microseconds

y = u (with sign extended to 18 bits).
Clear AL. Then transmit y to AL.

Example of enter AL with constant when
u = 0001

$(AL)_i$ = any value
$(AL)_f$ = 000001   (+ 1)

Example of enter AL with constant when
u = 7776

$(AL)_i$ = any value
$(AL)_f$ = 777776   (- 1)

NOTE: u is a 12-bit one's complement number
       contained within the instruction; it
       does not refer to an address.

72   STORE INDEX CONTROL REGISTER (STRICR)   (ICR) $\longrightarrow$ Y_{5-0}

Execution time: 8 microseconds

$y = u_p$
Replace the least significant 6 bits of the
(y) with a 6-bit value equal to the memory
address of the index register defined by ICR.
As this instruction effects a 6-bit partial
transfer, the upper 12 bits of (y) remain
unchanged.

NOTE: ICR = 0 produces memory address 10.
       ICR = 1 through 7, memory addresses
       01 through 07 respectively.

74   STORE ADDRESS (STRADR) $(AL)_{11-0}$ $\longrightarrow$ Y_{11-0}

Execution time: 8 microseconds

$y = u_p$
Replace the low order 12 bits of (y) with
the low order 12 bits of (AL). As this
instruction effects a partial transfer,
the higher order 6 bits of (y) remain un-
disturbed.
$(AL)_f = (AL)_i$

Example of a store address instruction:

$(AL)_i$ = 732504
$(y)_i$ = 567777
$(y)_f$ = 562504

75   STORE SPECIAL REGISTER (STRSR)   (SR) $\longrightarrow$ Y_{5-0}

Execution time: 8 microseconds

$y = u_p$
Replace the low order 6-bits of (y) with
a 6-bit value of which the low order 4 bits
are equal to the contents of the special
register with the remaining bits equal to zero;
store the result at y, then clear the special
register. As this instruction effects a 6-bit
partial transfer, the upper 12 bits of (y)
remain undisturbed.

NOTE: This instruction deactivates the special
       register.

1. GENERAL

Arithmetic instructions combine the contents of a specified memory location with the contents of the accumulator. Single length addition and subtraction are performed in an 18-bit parallel mode using one memory location and the lower half of the A register (AL). Double length addition and subtraction are performed in a 36-bit parallel mode using two consecutive memory locations and both halves of the A register. Multiply and divide instructions utilize one memory location and both halves of the A register. After all arithmetic instructions, the result is left in the appropriate portion of the A register.

2. INSTRUCTIONS

| 14 | ADD AL (ADDAL) | $(AL) + (Y) \longrightarrow AL$ |

Execution time: 8 microseconds

$y = u_p$ or u_{SR}
Add (y) to (AL) and leave the result in AL. Set overflow designator if overflow occurs.* $(AL)_f$ are all ones if $(AL)_i$ and (y) are all ones.

| 15 | ADD AL (ADDALB) | $(AL) + (Y) \longrightarrow AL$ |

Execution time: 12 microseconds

$y = u_p$ or u_{SR}
Add (y + B) to (AL) and leave the result in AL. Set overflow designator if overflow occurs.* $(AL)_f$ are all ones if $(AL)_i$ and (y + B) are all ones.

| 16 | SUBTRACT AL (SUBAL) | $(AL) - (Y) \longrightarrow AL$ |

Execution time: 8 microseconds

$y = u_p$ or u_{SR}
Subtract (y) from (AL) and leave the difference in AL. Set overflow designator if overflow occurs.* $(AL)_f$ are all ones if $(AL)_i$ are all ones, and (y) are all zeros.

---

*The overflow designator is cleared only by the execution of instruction skip on overflow (f, m = 50 52) or instruction skip on no overflow (f, m = 50 53).

17   SUBTRACT AL (SUBALB) $(AL) - (Y) \longrightarrow AL$

Execution time: 12 microseconds

$y = u_p$ or u_{SR}
Subtract (y + B) from (AL) and leave the
difference in AL. Set overflow designator
if overflow occurs.* $(AL)_f$ are all ones if
$(AL)_i$ are all ones and (y + B) are all zeros.

20   ADD A (ADDA) $(A) + (Y + 1, Y) \longrightarrow A$

Execution time: 12 microseconds

$y = u_p$ or u_{SR}
Add to (A) the double-length (36-bit) number
contained in storage cells y + 1; y, and
leave the result in A. Set overflow designator
if overflow occurs.* The least significant
half is in cell y, and the most significant
half is in y + 1. The sign of the double
length number is indicated by the most sig-
nificant bit of (y + 1). Address y must
be even; that is, the rightmost octal digit
must be 0, 2, 4, or 6.

NOTE: The instruction is executed in the
        following manner: Clear the borrow
        designator. The AU and AL registers
        are linked to form a continuous 36-bit
        A register. Therefore, any borrow for
        AL comes from AU; and any end around
        borrow for AU is blocked and recorded
        in the borrow designator leaving A
        uncorrected. The skip on no borrow
        instruction (Code 50, 51) is used to
        test for required correction. Only
        add A or subtract A instructions set
        the designator.

Example of a double add with y = 07506

              $(A)_i$ = 201007430145
address 07506 =   351123   (least significant half)
address 07507 = 077430   (most significant half)
              $(A)_f$ = 300440001271   (unadjusted sum)

---

* The overflow designator is cleared only by the execution of instruction skip
  on overflow (f,m = 50 52) or instruction skip on no overflow (f, m = 50 53).

21   ADD A (ADDAB) $(A) + (Y + 1, Y) \longrightarrow A$

      Execution time: 16 microseconds

      $y = u_p$ or u_{SR}
      Add to (A) the double-length (36-bit) number
      contained in storage cells y + B + 1, y + B
      leaving the result in A. Set overflow designator
      if overflow occurs.* The least significant
      half is in cell y + B, and the most significant
      half is in cell y + B + 1. The sign of the
      double-length number is the sign of (y + B + 1).
      Address y + B must be even. (See note of
      instruction 20.)

22   SUBTRACT A (SUBA) $(A) - (Y + 1, Y) \longrightarrow A$

      Execution time: 12 microseconds

      $y = u_p$ or u_{SR}
      Subtract from (A) the double-length (36-bit)
      number contained in storage cells y + 1, y, and
      leave the difference in A. Set overflow desig-
      nator if overflow occurs.* The least significant
      half is in cell y and the most significant
      half is in cell y + 1. The sign of the double-
      length number is the sign of (y + 1). Address y
      must be even. (See note of instruction 20.)

23   SUBTRACT A (SUBAB) $A - (Y + 1, Y) \longrightarrow A$

      Execution time: 16 microseconds

      $y = u_p$ or u_{SR}
      Subtract from (A) the double-length number
      contained in storage cells y + B + 1, y + B,
      and leave the difference in A. Set overflow
      designator if overflow occurs*. The least
      significant half is in cell y + B, and the most
      significant half is in cell y + B + 1. The
      sign of the double length number is the sign
      of (y + B + 1). Address y + B must be even.
      The computer executes subtract A in a manner
      analogous to the add A instruction. (See
      note of instruction 20.)

---

* The overflow designator is cleared only by the execution of instruction skip
 on overflow (f, m = 50 52) or instruction skip on no overflow (f, m = 50 53).

24   MULTIPLY AL (MULALB)   (AL) (Y) ⟶ A

Execution time: 26-49 microseconds

$y = u_p$ or u_{SR}
Multiply (AL) by (y) leaving the double
length product in A. If the factors are
considered integers, the product is an
integer in A. The multiplication process
is executed on the absolute value of the
factors, then corrected for algebraic sign.

25   MULTIPLY AL (MULALB)   (AL) (Y) ⟶ A

Execution time: 30-53 microseconds

$y = u_p$ or u_{SR}
Multiply (AL) by (y + B) leaving the double
length product in A. If the factors are
considered integers, the product is an integer
in A. The multiplication process is executed
on the absolute value of the factors, then
corrected for algebraic sign.

26   DIVIDE A (DIVA)   (A) ÷ (Y) ⟶ AL;REMAINDER ⟶ AU

Execution time: 48 microseconds

$y = u_p$ or u_{SR}
Divide (A) by (y) leaving the quotient in AL
and the remainder in AU. The remainder always
bears the sign of the dividend, A_i, with the
results satisfying the relationship: dividend
= quotient x divisor + remainder. Set overflow
designator if overflow occurs*. If overflow
occurs, (AL) becomes 0.

Examples of the four possible sign combinations
of the dividend/divisor and the results:

Dividend	Divisor	Quotient	Remainder
+5	+4	+1	+1
+5	-4	-1	+1
-5	+4	-1	-1
-5	-4	+1	-1

---

* The overflow designator is cleared only by the execution of instruction skip
on overflow (f, m = 50 52) or instruction skip on no overflow (f, m = 50 53).

27 DIVIDE A (DIVAB) $(A) \div (Y) \longrightarrow AL; REMAINDER \longrightarrow AU$

Execution time: 52 microseconds

$y = u_P$ or u_{SR}
Divide (A) by (y + B) leaving the quotient
in AL and the remainder in AU. The remainder
bears the sign of the dividend, A_i. (See
instruction 26.)

50 51 SKIP ON NO BORROW (SKPNBO)

Execution time: 6 microseconds skip; 4.67 no skip

If the last previous add A or subtract A required
a borrow, take next instruction; otherwise, skip
the next instruction. Ignore k. The skip occurs
if no correction to (A) is needed. This allows
a correcting instruction to be inserted to save
program steps. The correcting instruction will
be subtract A where (Y + 1, Y) = 000000000001.

50 52 SKIP ON OVERFLOW (SKPOV)*

Execution time: 6 microseconds; 4.67 no skip

If an overflow condition occurred on a previous
arithmetic instruction, skip the next instruction;
otherwise, take the next instruction. Ignore k
and clear the overflow designator.

50 53 SKIP ON NO OVERFLOW (SKPNOV)*

Execution time: 6 microseconds skip; 4.67 no skip

If an overflow condition did not occur on a pre-
vious arithmetic instruction, skip the next in-
struction; otherwise, take the next instruction.
Ignore k and clear the overflow designator.

50 60 ROUND AU (RND)

If (AU) pos., $(AU) + AL_{17} \longrightarrow AL$
If (AU) neg., $(AU) - AL_{17} \longrightarrow AL$

Execution time: 5.33 microseconds

If (AU) are positive, add bit position 17 of AL
to (AU); if (AU) are negative subtract the com-
plement of bit position 17 of AL from AU and leave
the resultant rounded (AU) in AL. Ignore k. $(AU)_i$
= $(AU)_f$. An application of this instruction would be:
a double length value in A is normalized as far as
possible to the left; however, only a rounded single
length number is required for the accuracy desired.

---

* The overflow designator is cleared only by the execution of instruction skip
on overflow (f, m = 50 52) or instruction skip on no overflow (f, m = 50 53).

71   ADD CONSTANT TO AL (ADDALK) $(AL) + xY \longrightarrow AL$

Execution time: 4.67 microseconds

$y = u$ (sign extended to 18 bits)
Add y to (AL) and leave the result in
AL. The effect of this instruction is
to increment/decrement (AL) with a
constant contained within the instruction.

Example of add constant to AL when $u = 0002$ (+ 2)

$(AL)_i = 057777$
$(AL)_f = 060001$ (incremented)

Example of add constant to AL when $u = 7775$ (- 2)

$(AL)_i = 067055$
$(AL)_f = 067053$ (decremented)

SECTION I-B-3. SHIFT INSTRUCTIONS

1. GENERAL

Shift instructions shift the contents of a selected register to the right or left a specified number of bit positions. All shift instructions are Format II instructions. The type of shift is specified by the f and m fields and, with one exception, the number of bit position shifts to be executed is specified by the k field. The exception is the scale factor shift instruction (50 44) which uses both the k field and the upper two bits of the A register to determine when the correct number of bit position shifts has been executed.

2. INSTRUCTIONS

50 41 RIGHT SHIFT AU (RSHAU)

Execution time: 4 microseconds ($k = 0$); $5.33 + 2k/3$ microseconds ($k \neq 0$)

Shift (AU) to the right k-bit positions. The higher order bits are replaced with the original sign bit, AU_{17}, as the value is shifted. This is an end-off shift (that is, the low order bits are lost upon completion of the shift).

Example of right shift AU with $k = 2$:

$(AU)_i$ (positive)	= 370000
after first shift	174000
after second shift	076000
$(AU)_i$ (negative)	= 400000
after first shift	600000
after second shift	700000

50 42 RIGHT SHIFT AL (RSHAL)

Execution time: 4 microseconds ($k = 0$). $5.33 + 2k/3$ microseconds ($k \neq 0$)

Shift (AL) to the right k-bit positions. The higher order bits are replaced with the original sign bit, AL_{17}, as the value is shifted. This is an end-off shift (that is, the low order bits are lost upon completion of the shift).

50 43 RIGHT SHIFT A (RSHA)

Execution time: 4 microseconds ($k = 0$); $5.33 + 2k/3$ microseconds ($k \neq 0$)

Shift (A) to the right k-bit positions. The higher order bits are replaced with the original sign bit, A_{35}, as the value is shifted. This is an end-off shift (that is, the low order bits are lost upon completion of the shift).

Example of right shift A with k = 2:

```
(A)ᵢ (positive)           = 370000 000000
after first shift           174000 000000
after second shift          076000 000000

(A)ᵢ (negative)           = 400000 000000
after first shift           600000 000000
after second shift          700000 000000
```

50 44   SCALE FACTOR (SF)

Execution time: 8 microseconds $(k = 0)$; $9.33 + 2k/3$ $(k \neq 0)$

Shift (A) circularly to the left until either $A_{35} \neq A_{34}$
or k minus shift count = 0; then store the
positive quantity k minus shift count at memory
address 00017. The effect of the instruction is to
normalize (A) to the left subject to k. Scale factor
is extremely useful when working with numerical
values in floating point notation.

Example of scale factor with k = 7:

$(A)_i$ = 170000 000000 (positive, not normalized)
after first shift 360000 000000 (positive, normalized).
The computer, sensing (A) now normalized, stores
k minus the shift count (7-1) at address 00017. The
18-bit quantity is 000006.

Example of scale factor with k = 3:

$(A)_i$ = 600000 000000 (negative, not normalized)
after first shift 400000 000001 (negative, normalized).
The computer then stores the quantity 000002 at address 00017.

Example of scale factor with k = 1:

$(A)_i$ = 070000 000000 (positive, not normalized)
after first shift 160000 0000000 (positive, not normalized).
The computer, having exhausted k, stores the quantity
000000 at 00017 leaving (A) only partially normalized.

50 45   LEFT SHIFT AU (LSHAU)

Execution time: 4 microseconds (k = 0); 5.33 + 2k/3 microseconds (k $\neq$ 0)

Shift (AU) circularly to the left k-bit positions. The lower
order bits are replaced with the higher order bits as the
word is shifted.

Example of left shift AU with k = 2:

```
(AU)_i                      = 300000
after first shift             600000
after second shift            400001
```

No bits are lost with the execution of left shift instructions.

50 46   LEFT SHIFT AL (LSHAL)

Execution time: 4 microseconds (k = 0); 5.33 + 2k/3 microseconds (k $\neq$ 0)

Shift (AL) circularly to the left k-bit positions. The lower
order bits are replaced with the higher order bits as the
word is shifted. No bits are lost with the execution of left
shift instructions. (See examples of instruction 50 45).

50 47   LEFT SHIFT A (LSHA)

Execution time: 4 microseconds (k = 0); 5.33 + 2k/3 microseconds (k $\neq$ 0)

Shift (A) circularly to the left k-bit positions. The lower
order bits are replaced with the higher order bits as the
word is shifted. No bits are lost with the execution of left
shift instructions.

Example of left shift A with k = 2:

```
(A)_i                       = 300000 000000
after first shift             600000 000000
after second shift            400000 000001
```

SECTION I-B-4. LOGICAL INSTRUCTIONS

1. GENERAL

Logical instructions perform five basic operations: compare, complement, selective set, selective clear, and selective substitute. The parity skip instructions are also included here since they provide the means for conditional skips based on the results of a logical operation.

2. COMPARE INSTRUCTIONS

Four compare instructions are used to test and record certain conditions in preparation for the execution of an arithmetic conditional jump instruction. The compare instructions set the comparison designator, a 3-stage register, and the conditional jump instruction samples the comparison designator to determine if the jump condition has been satisfied. The comparison designator records the results of compare instructions as follows:

1) The compare stage is set upon the computer's execution of any one of the compare instructions.

2) The less than stage is set if a compare instruction finds (AL) less than the contents of an addressed memory location, or L(AL) (AU) less than the logical product of (AU) and the contents of the addressed memory location (whichever applies).

3) The equals stage is set if a compare instruction finds (AL) equal to the contents of an addressed memory location or finds the logical product of (AL) and (AU) equal to the logical product of (AU) and the contents of the addressed memory location (whichever applies).

The comparison designator is cleared by the execution of any instruction other than the arithmetic conditional jump instructions (function codes 60-67). Therefore, in order to set the compare stages desired, a compare instruction must immediately precede the jump instruction, or immediately precede the first of a consecutive string of jump instructions. Otherwise, these jump instructions are executed without reference to the comparison designator. While the comparison designator is set, all interrupts are locked out. For an explanation of the manner in which the comparison designator is interpreted, refer to the description of the jump instructions.

02   COMPARE AL(CMAL)   (AL) : (Y)

            Execution time: 8 microseconds $(AL)_f = (AL)_i$

            $y = u_p$ or u_{SR}
            Compare algebraically (AL) with (y) and set the
            comparison designator as follows:

            Set the compare stage
            Set the less than stage if (AL) < (y)
            Set the equals stage if (AL) = (y)

03   COMPARE AL (CMALB)   (AL) : (Y)

            Execution time: 12 microseconds $(AL)_f = (AL)_i$

            $y = u_p$ or u_{SR}
            Compare algebraically (AL) with (y + B) and set
            the comparison designator as follows:

            Set the compare stage
            Set the less than stage if (AL) < (y + B)
            Set the equals stage if (AL) = (y + B)

06   COMPARE WITH MASK (CMSK)   L(AU) (AL) : L(AU) (Y)

            Execution time: 8 microseconds $(A)_f = (A)_i$

            $y = u_p$ or u_{SR}
            Compare algebraically the masked bits of (AL)
            with corresponding bits of (y) and set compar-
            ison designator as follows:

            Set the compare stage
            Set the less than stage if L(AL) (AU) < L(y) (AU)
            Set the equals stage if L(AL) (AU) = L(y) (AU)

            The masked bits of (AL) are those bits which corres-
            pond to bits set in (AU).

            Example of compare with mask:

            $(AU)_i$ = 007777   Mask
            (y) = 123451
            $(AL)_i$ = 222351
            Compare 2351 with 3451
            $(AU)_f$ = 007777, $(AL)_f$ = 222351

07   COMPARE WITH MASK (CMSKB)   L(AU) (AL): L(AU) (Y)

Execution time: 12 microseconds $(A)_f = (A)_i$

$y = u_p$ or u_{SR}
Compare algebraically the masked bits of (AL)
with corresponding bits of (y + B) and set the
comparison designator as follows:

Set the compare stage
Set the less than stage if L(AL) (AU) < L(y + B) (AU)
Set the equals stage if L(AL) (AU) = L(y + B) (AU)

The masked bits of (AL) are those bits which correspond
to bits set in (AU).

Example:

$(AU)_i$ = 000377
(y + B) = 674201
$(AL)_i$ = 377601
Compare 201 with 201
$(AU)_f$ = 000377, $(AL)_f$ = 377601

3. COMPLEMENT INSTRUCTIONS

Four complement instructions are provided. Three of these instructions are used
to effect a bit-by-bit complement of the entire contents of the AL, AU, or A
register. The fourth is used to complement selected bits of AL. The bits
complemented are determined by the presence of 1's in corresponding bit positions
of a designated storage location Y.

50 61   COMPLEMENT AL (CPAL) $(AL)' \longrightarrow AL$

Execution time: 5.33 microseconds

Complement (AL), leaving the result in AL.
Ignore k.

NOTE: This instruction effects a bit-by-bit
       complement with the following exception:
       all zeros (positive zero) will remain
       all zeros.

50 62   COMPLEMENT AU (CPAU) $(AU)' \longrightarrow AU$

Execution time: 5.33 microseconds

Complement (AU), leaving the result in AU.
Ignore k. (See note for instruction 50 61).

50 63   COMPLEMENT A (CPA) $(A)' \longrightarrow A$

Execution time: 5.33 microseconds

Complement (A) leaving the result in A.
Ignore k. (See note for instruction 50 61).

53   SELECTIVE COMPLEMENT (SLCP) $L(AL)'\ (Y) \oplus L(AL)\ (Y)' \longrightarrow AL$
                                      or
                                         COMPLEMENT $(AL)_n$ for $(Y)_n = 1$

Execution time: 8 microseconds

$y = u_p$
Complement the individual bits of (AL)
corresponding to ones in (y), leaving the
remaining bits of (AL) unaltered; that is,
complement $(AL)_n$ for $(y)_n = 1$. This is a
bit-by-bit exclusive OR.

Example of selective complement instruction:

$(AL)_i = 123456$
$(Y)\ \ \ = 070007$
$(AL)_f = 153451$

4. SELECTIVE SET INSTRUCTION

The selective set instruction is used to force ones into selected bit positions
of the AL register. The bit positions into which ones are forced are determined
by the presence of ones in corresponding bit positions of a designated storage
location Y.

51   SELECTIVE SET (SLSET)   or $(AL) \vee (Y) \longrightarrow AL$

        Execution time: 8 microseconds   SET $(AL)_n$ for $(Y)_n = 1$

$y = u_p$
Set the individual bits of (AL) to ones corresponding
to ones in (y), leaving the remaining bits of (AL)
unaltered. This is a bit-by-bit inclusive OR.

Example of selective set:

$(AL)_i = 123456$
$(Y)\ \ \ = 000077$
$(AL)_f = 123477$

5. SELECTIVE CLEAR INSTRUCTION

The selective clear instruction is used to force zeros into selected bit posi-
tions of the AL register. The bit positions into which zeros are forced are
determined by the presence of zeros in corresponding bit positions of a desig-
nated storage location Y.

52   SELECTIVE CLEAR (SLCL) $L(AL)\ (Y) \longrightarrow AL$

Execution time: 8 microseconds   or CLEAR $(AL)_n$ for $(Y)_n = 0$

$y = u_p$
Clear the individual bits of (AL) corresponding
to zeros in (y), leaving the remaining bits of (AL)
unaltered. The effect of this instruction is to
compute the bit-by-bit (or logical) product of (AL)
and (y), leaving the result in AL. This is a bit-by-
bit AND.

Example of selective clear:

$(AL)_i = 123456$
$(Y)\ = 707070$
$(AL)_f = 103050$

6. SELECTIVE SUBSTITUTE INSTRUCTIONS

The selective substitute instructions are used to replace bits in selected bit
positions of the AL register with bits from corresponding bit positions of a
designated storage location Y. The bit positions affected by the selective
substitute instructions are determined by the presence of 1's in the correspond-
ing bit positions of the AU register.

04   SELECTIVE SUBSTITUTE (SLSU) $L(AU)'\ (AL) + L(AU)\ (Y) \longrightarrow AL$

Execution time: 8 microseconds   or $(Y)_n \longrightarrow AL_n$ for $(AU)_n = 1$

$y = u_p$ or u_{SR}
Replace the individual bits of (AL) with bits
of (y) corresponding to ones in (AU), leaving
the remaining bits of (AL) unaltered.

$(AU)_f = (AU)_i$

Example of selective substitute:

$(AU)_i = 007777$ Mask
$(Y)\ = 123451$
$(AL)_i = 666666$
$(AL)_f = 663451$

05   SELECTIVE SUBSTITUTE (SLSUB) $L(AU)'\ (AL) + L(AU)\ (Y) \longrightarrow AL$

Execution time: 12 microseconds

$y = u_p$ or u_{SR}
Replace the individual bits of (AL) with bits
of (y + B) corresponding to ones in (AU), leaving
the remaining bits of (AL) unaltered.

$(AU)_f = (AU)_i$

7. PARITY SKIP INSTRUCTIONS

The following two instructions permit programmed conditional skips based on the parity of the bit-by-bit product of the contents of AL and AU. Parity is odd if the number of ones in the resulting product is odd; parity is even if the number of ones in the resulting product is even.

50 54 SKIP ON ODD PARITY (SKPODD)

Execution time: 6 microseconds skip; 4.67 no skip

If the sum of the bits resulting from the bit-by-bit product of (AL) and (AU) is odd, skip the next instruction; otherwise, take the next instruction. Ignore k.
$(AU)_f = (AU)_i$; $(AL)_f = (AL)_i$

Example of skip on odd parity:

```
(AU)                 000077 mask
(AL)                 127723
bit-by-bit product  = 000023
bit sum             = 3
Since the bit sum is odd, the next instruction is skipped.
```

50 55 SKIP ON EVEN PARITY (SKPEVN)

Execution time: 6 microseconds skip; 4.67 no skip

If the sum of the bits resulting from the bit-by-bit product of (AL) and (AU) is even, skip the next instruction; otherwise, take the next instruction. Ignore k.

$(AL)_f = (AL)_i$; $(AU)_f = (AU)_i$

SECTION I-B-5. MODIFYING INSTRUCTIONS

1. GENERAL

Modifying instructions provide a simple method of incrementing or decrementing the current value of a B register or a storage location used as an index.

2. INSTRUCTIONS

37   MODIFY B WITH CONSTANT (ENTBKB) $B_i + xY \longrightarrow B$

                     Execution time: 12 microseconds

                     $y = u$ (sign extended to 18 bits)
                     Add y to B (add a constant to B).
                     The effect of this instruction is to add a constant u to B; however, since u is a 12-bit one's complement number, the instruction can be used to increment or decrement B.

56   B SKIP (BSK)   If B = (Y), Skip NI
                                  If B ≠ (Y), Increment B by 1 and
                                                 Execute NI

                     Execution time: 16 microseconds

                     $y = u_p$
                     Test B and (y) for equality. Skip the next instruction if equal; otherwise, increment B by 1 and execute the next instruction.

57   INDEX SKIP (ISK)   If (Y) = 0, Skip NI
                                     If (Y) ≠ 0, Decrement (Y) by 1 and
                                               Execute NI
                     Execution time: 12 microseconds

                     $y = u_p$
                     If (y) ≠ 0, subtract one from (y) leaving the result in y, and take the next instruction; otherwise skip the next instruction leaving (y) unaltered.

                     If $(y)_i = 777777$, then
                     $(y)_f = 777776$ and there is no skip.

73   BJUMP (BJP)   If $B \neq 0$, $B-1 \longrightarrow B$ and $Y \longrightarrow P$

                                                  If $B = 0$, Execute NI

Execution time: 12 microseconds

$y = u_p$
If $B \neq 0$, subtract 1 from B then jump
to y; otherwise execute the next instruction
leaving B unaltered. (Negative zero $\neq 0$.)

NOTE: As B is a one's complement number and can take
           values less than zero, the B jump will be
           effective only for program loops where B is
           initially positive.

SECTION I-B-6. JUMP INSTRUCTIONS

1. INTRODUCTION

Jump instructions are used to transfer program control to other portions of a
program or to other programs. Jump instructions fall into two general categories:
conditional and unconditional. Conditional jumps transfer program control only
if certain specified conditions exist. Unconditional jumps always transfer
program control.

2. UNCONDITIONAL JUMP INSTRUCTIONS

Seven unconditional jump instructions are provided. The use of each instruction
is dependent upon the purpose for transferring program control. The names and
mnemonics of these instructions convey the suitability of each instruction
for a particular application. The three key words used are defined below.

1) Direct.

The word direct signifies that control is to be transferred directly to
the address specified by the lower 12 bits of the instruction and the
upper 3 bits of the P register. Since the P register must obviously be
set to the bank in which the jump instruction is stored, direct jumps
are normally used to transfer control within a memory bank. However, a
direct jump with B modification can be used to transfer control between
banks.

2) Indirect.

The word indirect signifies that control is to be transferred to an
address contained in the lower 15 bits of the storage location specified
by the lower 12 bits of the instruction and the upper 3 bits of the P
register. Indirect jumps require an additional memory location and an
extra memory access; however, they permit transfer of program control to
any address in memory, regardless of bank designation.

3) Return.

The word return implies that program control is being transferred tem-
porarily and that control may be returned to this point in the program
after a specific task has been performed. Therefore, return jumps store
the address of the next sequential instruction in the program before
transferring program control.

30   INDIRECT RETURN JUMP (IRJP)   (P) + 1 $\longrightarrow$ (Y) ; (Y) + 1 $\longrightarrow$ P

            Execution time: 12 microseconds

            Instruction executed from running program:
            y = up
            Store (P) + 1 at the address given in the low
            order 15 bits of (y), then increment that address
            by one and enter it into the program address register.

I-B-29

Instruction executed from entrance register on interrupt:
y = u
Store (P) at the address which is the low order
15 bits of (y), then increment that address by one
(1) and enter it into the program address register.

Example of an indirect return jump executed from
address 22000:

Address	Initial Contents	Final Contents	Explanation
22000	30 6500	Same	Execute subroutine from main program.
26500	01 7420	Same	Constant defining location of desired subroutine.
17420	37 2164	02 2001	Subroutine exit address.
17421	.. ....	Same	Subroutine entrance address (control is transferred here from indirect return jump).

The effect of the above sequence upon execution
of the indirect return jump at address 22000 is
to transfer control to the subroutine starting at
17421, but at the same time, letting the subroutine
know where to return control.

31   INDIRECT RETURN JUMP (IRJPB)   (P) + 1 ⟶ (Y) ; (Y) + 1 ⟶ P

Execution time: 16 microseconds

Instruction executed from running program:
y = up
Store (P) + 1 at the address given in the low order
15 bits of (y + B), then increment that address by
one and enter it into the program address register.

Instruction executed from entrance register on interrupt:
y = u
Store (P) at the address which is the low order 15
bits of (y + B), then increment that address by one
and enter it into the program address register.

34   DIRECT JUMP (JP) $Y \longrightarrow P$;   NI = (Y)

Execution time: 4 microseconds

$y = u_p$
Unconditionally jump to y. (Reset P = y)

35   DIRECT JUMP (JPB) $Y \longrightarrow P$;   NI = (Y)

Execution time: 8 microseconds

$y = u_p$
Unconditionally jump to y + B.

        NOTE: Because B is an 18-bit one's complement
                   number, care must be taken when using this
                   instruction; in addition, it is possible
                   that address, y + B, may not be relative to
                   the same core bank from which the (35) direct
                   jump was executed. Consider a direct jump
                   with y = 03560 and B = 010000; in this case
                   y + B = 03560 + 010000 = 13560.

54   INDIRECT JUMP AND REMOVE INTERRUPT LOCKOUT (IJPEI)

                                           $(Y) \longrightarrow P$ and RIL

Execution time: 4 microseconds

$y = u_p$ Address = $(y)_{14-0}$
Remove interrupt lockout (enable interrupts).
Then jump to the address which is the low order
15 bits of (y). An application of this instruction
is the termination of a subroutine activated by
an interrupt.

55   INDIRECT JUMP (IJP) $(Y) \longrightarrow P$

Execution time: 8 microseconds

$y = u_p$ Address = $(y)_{14-0}$
Jump to the address which is the low
order 15 bits of (y).

76   DIRECT RETURN JUMP (RJP) $(P) + 1 \longrightarrow Y$; $Y + 1 \longrightarrow P$

Execution time: 8 microseconds

$y = u_p$
Store (P) + 1 at y, then jump to y + 1. This
instruction transfers to y a full 18-bit word,
the lower 15 bits being the address (P) + 1 with
the upper three bits set to zero. When this instruction
is executed from an interrupt entrance register by an
interrupt, store P. Do not initiate the (P) + 1
sequence.

3. CONDITIONAL JUMP INSTRUCTIONS

Eight conditional jump instructions are provided. Since these instructions transfer program control only if the AL and AU registers satisfy certain arithmetic conditions, they are often called arithmetic conditional jump instructions. These instructions (function codes 60-67) may be used with or without an associated compare instruction. When a compare instruction is used, it must immediately precede the conditional jump instruction (or the first of a sequence of conditional jump instructions). If a compare instruction is not used, the jump is executed upon satisfying the condition directly stated by the instruction. If a compare instruction is used with one or more jump instructions, the satisfaction of the jump condition is dependent upon the status of the 3-stage comparison designator. Table I-B-2 provides a summary of the conditional jump instructions when used separately and with a compare instruction. The letter M in the table represents the value that (AL) or L(AL) (AU) are compared to during execution of the compare instruction. This value is actually (y) for an 02 instruction, (y + B) for an 03, (y) masked by (AU) for 06, and (y + B) masked by (AU) for 07. Note that (AU) can be·used as a mask, but never as one of the values compared by a compare instruction. Therefore, when the comparison designator is set (refer to logical instructions), the jump condition is dependent only upon the relative values of (AL) or L(AL) (AU) and M.

TABLE I-B-2. SUMMARY OF CONDITIONAL JUMP INSTRUCTIONS

| Jump Instr Code | Compare Designator Not Set | Compare Designator Set | | | | Results If a Jump Occurs |
| | | Equals Stage | | Less Than Stage | | |
		Set	Not Set	Not Set	Set	
60	JP if (AU) = 0	JP; (AL) = M	No JP	*	*	(AU) = 0 or (AL) = M
61	JP if (AL) = 0	JP; (AL) = M	No JP	*	*	(AL) = 0 or M
62	JP if (AU) ≠ 0	No JP	JP; (AL) ≠ M	*	*	(AU) ≠ 0 or (AL) ≠ M
63	JP if (AL) ≠ 0	No JP	JP; (AL) ≠ M	*	*	(AL) ≠ 0 or M
64	JP if (AU) ≥ 0	*	*	JP; (AL) ≥ M	No JP	(AU) POS. or (AL) ≥ M
65	JP if (AL) ≥ 0	*	*	JP; (AL) ≥ M	No JP	(AL) POS. or (AL) ≥ M
66	JP if (AU) < 0	*	*	No JP	JP; (AL) < M	(AU) NEG. or (AL) < M
67	JP if (AL) < 0	*	*	No JP	JP; (AL) < M	(AL) NEG. or (AL) < M

* Does not apply

60   JUMP AU ZERO (JPAUZ)   Compare stage not set, (AU) = 0, Y ⟶ P
                                        Compare and equals stages set,   Y ⟶ P

Execution time: 4 microseconds

$y = u_p$
Jump to y; that is, Reset P = y, if:

Compare stage of the comparison designator is not
set and (AU) = 0. (negative zero acts as not zero);
or

Compare stage of the comparison designator is set
and the equals stage of the comparison designator is
set.

Otherwise, execute next instruction.

61   JUMP AL ZERO (JPALZ)   Compare stage not set, (AL) = 0, Y ⟶ P
       JUMP IF EQUAL (JPEQ)   Compare and equals stages set,   Y ⟶ P

Execution time: 4 microseconds

$y = u_p$
Jump to y; that is, reset P = y if:

Compare stage of the comparison designator is not
set and (AL) = 0.   (Negative zero acts as not zero); or

Compare stage of comparison designator is set,
and the equals stage of the comparison designator
is set.

Otherwise, execute next instruction.

62   JUMP AU NOT ZERO (JPAUNZ)

                                       Compare stage not set, (AU) ≠ 0, Y ⟶ P
                                       Compare stage set, equals stage
                                       stage not set,   Y ⟶ P

Execution time: 4 microseconds

$y = u_p$
Jump to y; that is, reset P = y, if:

Compare stage of comparison designator is not
set and (AU) ≠ 0; or

Compare stage of comparison designator is set,
and the equals stage of the comparison designator
is not set.

Otherwise, execute next instruction.

63   JUMP AL NOT ZERO (JPALNZ)   Compare stage not set, $(AL) \neq 0$, $Y \longrightarrow P$
JUMP NOT EQUAL (JPNOT)   Compare stage set, equals stage
                       not set, $Y \longrightarrow P$

Execution time: 4 microseconds

$y = u_p$
Jump to y; that is, reset P = y, if:

Compare stage of comparison designator is
not set and $(AL) \neq 0$; or

Compare stage of comparison designator is
set, and the equals stage of comparison
is not set.

Otherwise, execute next instruction.

64   JUMP AU POSITIVE (JPAUP)   Compare stage not set, $(AU) \geq 0$, $Y \longrightarrow P$
                                  Compare stage set, less than
                                  stage not set, $Y \longrightarrow P$

Execution time: 4 microseconds

$y = u_p$
Jump to y; that is, reset P = y, if:

Compare stage of comparison designator is not
set and $(AU) \geq 0$; or

Compare stage of comparison designator is set,
and the less than stage of comparison is not.

Otherwise, execute next instruction.

65   JUMP AL POSITIVE (JPALP)   Compare stage not set, $(AL) \geq 0$, $Y \longrightarrow P$
JUMP M LESS OR EQUAL (JPMLEQ)
                                  Compare stage set, less
                                  than stage not set, $Y \longrightarrow P$

Execution time: 4 microseconds

$y = u_p$
Jump to y; that is, reset P = y, if:

Compare stage of comparison designator is
not set and $(AL) \geq 0$; or

Compare stage of comparison designator is
set, and the less than stage of comparison
designator is not set.

Otherwise, execute next instruction.

66	JUMP AU NEGATIVE (JPAUNG)	Compare stage not set,	$(AU) < 0,$	$Y \longrightarrow P$
		Compare and less than		
		stages set,		$Y \longrightarrow P$

Execution time: 4 microseconds

$y = u_p$
Jump to y; that is, reset P = y, if:

Compare stage of comparison designator is not
set and (AU) < 0; or

Compare stage of comparison designator is set,
and the less than stage of comparison designator
is set.

Otherwise, execute next instruction.

67	JUMP AL NEGATIVE (JPALNG)	Compare stage not set,	$(AL) < 0,$	$Y \longrightarrow P$
	JUMP M GREATER (JPMGR)	Compare and less than		
		stages set,		$Y \longrightarrow P$

Execution time: 4 microseconds

$y = u_p$
Jump to y; that is, reset P = y, if:

Compare stage of comparison designator is
not set and (AL) < 0; or

Compare stage of comparison designator is
set and the less than stage of the comparison
designator is set.

Otherwise, execute next instruction.

73	B JUMP (BJP)	If $B \neq 0,$ $B-1 \longrightarrow B$ and $Y \longrightarrow P$
		If $B = 0,$ execute NI

Execution time: 12 microseconds

$y = u_p$
If $B \neq 0$, subtract one from B then jump to y;
otherwise, execute the next instruction leaving
B unaltered. (Negative zero $\neq 0$.)

NOTE: The B jump instruction is listed here
because it does provide the means to
jump conditionally. Since the jump
condition is based on the contents of
the B register and since the contents of
B are modified each time the condition
is satisfied, the instruction is more
accurately defined as a modifying
instruction.

SECTION I-B-7. SKIP AND STOP INSTRUCTIONS

1. GENERAL

Skip instructions are used to skip one instruction and then continue in sequence.
The stop instruction permits programmed stops in a running program. The in-
structions which follow provide the means for programming both conditional
and unconditional skips and stops.

2. INSTRUCTIONS

50 21   SKIP ON INPUT INACTIVE (SKPIIN)

        Execution time: 6 microseconds skip; 4.67 no skip

        Test for output activity on channel k. If inactive,
        skip the next insturction; otherwise, take the next
        instruction.

50 22   SKIP ON OUTPUT INACTIVE (SKPOIN)

        Execution time: 6 microseconds skip; 4.67 no skip

        Test for output activity on channel k. If inactive,
        skip the next instruction; otherwise, take the next
        instruction.

50 23   SKIP ON EXTERNAL FUNCTION INACTIVE (SKPFIN)

        Execution time: 6 microseconds skip; 4.67 no skip

        Test for external function activity or channel k. If
        inactive, skip the next instruction; otherwise, take
        the next instruction.

50 50   SKIP ON KEY SETTING (SKP)

        Execution time: 6 microseconds skip; 4.67 no skip

        If bit 4, 3, 2, 1, or 0 of k is one and the corresponding
        skip key 4, 3, 2, 1, or 0 is set, or if bit 5 of k is
        a one (unconditional skip), skip the next instruction.

        Otherwise, take the next instruction.

        Examples of skip with:

        k = 01 (bit 0)   skip if skip key 0 is set
        k = 02 (bit 1)   skip if skip key 1 is set
        k = 04 (bit 2)   skip if skip key 2 is set
        k = 10 (bit 3)   skip if skip key 3 is set
        k = 20 (bit 5)   skip if skip key 4 is set
        k = 40 (bit 5)   skip unconditionally
        k = 03 (bits 1, 0)   skip if either key 1 or 0 is set

50 51   SKIP ON NO BORROW (SKPNBO)

Execution time: 6 microseconds skip; 4.67 no skip

If the last previous add A or subtract A required
a borrow, take next instruction; otherwise, skip
the next instruction. Ignore k. The skip occurs
if no correction to (A) is needed. This allows a
correcting instruction to be inserted to save pro-
gram steps. The correcting instruction will be
subtract A where (Y + 1, Y) = 000000000001.

50 52   SKIP ON OVERFLOW (SKPOV)

Execution time: 6 microseconds skip; 4.67 no skip

If an overflow condition occurred on a previous
arithmetic instruction, skip the next instruction;
otherwise, take the next instruction. Ignore k and
clear the overflow designator.

50 53   SKIP ON NO OVERFLOW (SKPNOV)

Execution time: 6 microseconds skip ; 4.67 no skip

If an overflow condition did not occur on a previous
arithmetic instruction, skip the next instruction;
otherwise, take the next instruction. Ignore k and
clear the overflow designator.

50 54   SKIP ON ODD PARITY (SKPODD)

Execution time: 6 microseconds skip ; 4.67 no skip

If the sum of the bits resulting from the bit-by-bit
product of (AL) and (AU) is odd, skip the next instruc-
tion; otherwise, take the next instruction. Ignore k.

$(AU)_f = (AU)_i; \ (AL)_f = (AL)_i$

Example of skip odd parity:

(AU)	000077 mask
(AL)	127723
bit-by-bit product	= 000023
bit sum	= 3

Since the bit sum is odd, the next instruction is skipped.

50 55   SKIP ON EVEN PARITY (SKPEVN)

Execution time: 6 microseconds skip; 4.67 no skip

If the sum of the bits resulting from the bit-by-bit
product of (AL) and (AU) is even, skip the next
instruction; otherwise, take the next instruction. Ignore k.

$(AL)_f = (AL)_i; \ (AU)_f = (AU)_i$

50 56   STOP ON KEY SETTING (STOP)

Execution time: 4.67 microseconds

If bit 4, 3, 2, 1, or 0 of k is one and the corresponding
console stop key 4, 3, 2, 1, or 0 is set, or if bit
5 of k is a one (unconditional stop), stop the computer;
otherwise, take the next instruction.

Examples of stop with:

k = 01 (bit 0)	stop if stop key 0 is set
k = 02 (bit 1)	stop if stop key 1 is set
k = 04 (bit 2)	stop if stop key 2 is set
k = 10 (bit 3)	stop if stop key 3 is set
k = 20 (bit 4)	stop if stop key 4 is set
k = 40 (bit 5)	stop unconditionally
k = 03 (bits 1, 0)	stop if either stop key 1 or 0 is set

50 57   SKIP ON NO RESUME (SKPNR)

Execution time: 6 microseconds skip; 4.67 no skip

If the resume designator on channel k is not set
(indicating unsuccessful transfer of a word to an
output device), skip the next sequential instruction;
otherwise, take the next instruction.

56   B SKIP (BSK)   If B = (Y), SKIP NI
                           If B ≠ (Y), advance B by 1 and execute NI

Execution time: 16 microseconds

y = up
Test B and (y) for equality. Skip next instruction if
equal; otherwise, increment B by 1 and read the next
instruction.

57   INDEX SKIP (ISK)   If (Y) = 0, SKIP NI
                               If (Y) ≠ 0, decrement (Y) by 1 and execute NI

Execution time: 12 microseconds

y = up
If $(y) \neq 0$, subtract one from (y) leaving the result in y,
and take the next instruction; otherwise, skip the next instruc-
tion leaving (y) unaltered.

If $(y)_i$ = 777777, then
$(y)_f$ = 777776 and there is no skip.

SECTION I-B-8. INPUT/OUTPUT INSTRUCTIONS

1. GENERAL

I/O instructions are used to program the transfer of data between the computer and various peripheral devices. All I/O instructions are Format II instructions. The function code, f, is always 50_8; the minor function code, m, defines the operation to be performed; and when applicable, the operand designator, k, specifies the number of the I/O channel on which the operation is to be performed.

2. BUFFER TRANSFER INSTRUCTIONS

Data is transferred between the computer and peripheral devices in a buffer mode. An input buffer is a block of consecutive storage locations into which a peripheral device, connected to an input channel, places data. An output buffer is a block of consecutive storage locations from which a peripheral device, connected to an output channel, receives data. Buffer transfers are normally controlled by assigned memory locations designated as control words. Two control words are assigned to each type of transfer (input, output, and external function) on each I/O channel.

Prior to the beginning of a buffer transfer on a specific channel, the control words for that channel and transfer type must be set to the initial and terminal addresses of the buffer. The channel must then be activated to begin the data transfer. Once the buffer transfer has been started, it is carried out by the I/O section of the computer without further program control. The I/O section transfers one 18-bit buffer word at a time in single channel mode and two 18-bit words at a time in dual channel ESI modes. Before each 18-bit or 36-bit transfer, the buffer control words are checked for equality. If the two control words are not equal, the second control word is incremented (or decremented) and a transfer is made. When the equality check indicates that the two control words are equal, the buffer is automatically terminated.

The input, output, and external function transfer instructions (50 11, 50 12, and 50 13) are used to establish the buffer limits and activate the channel. When coding these instructions, the programmer supplies the buffer terminal address and the buffer initial address in the lower 15 bits of the two instruction locations following the buffer transfer instruction. The format illustrated on the following page defines the contents of the 3-word sequence beginning at address n.

The transfer instruction at address n specifies the type of transfer (input, output, or external function) and the channel on which the transfer is to occur. When the instruction is executed, the contents of addresses n + 1 and n + 2 are stored in the two assigned buffer control registers for the particular channel and type of transfer.

Address n

```
        17              12 11      6 5        0
       ┌────────────────┬──────────┬──────────┐
       │                │          │          │
       └────────────────┴──────────┴──────────┘
                f            m          k
```

Buffer Transfer
Instruction
(50 11, 50 12,
or 50 13)

n + 2

```
       17 16 15 14                             0
       ┌──┬──┬──┬─────────────────────────────┐
       │  │  │  │                             │
       └──┴──┴──┴─────────────────────────────┘
```

Terminal
Address Word

Buffer Direction ──────────┘ │ │   Buffer Terminal Address
Designator

Buffer Monitor ──────────────┘ │
Interrupt Designator

Unassigned ──────────────────────┘

```
       17 16 15 14                             0
       ┌──┬──┬──┬─────────────────────────────┐
       │  │  │  │                             │
       └──┴──┴──┴─────────────────────────────┘
```

Initial Address
Word

Buffer Direction ──────────┘ │ │   Buffer Initial Address
Designator

Buffer Monitor ──────────────┘ │
Interrupt Designator
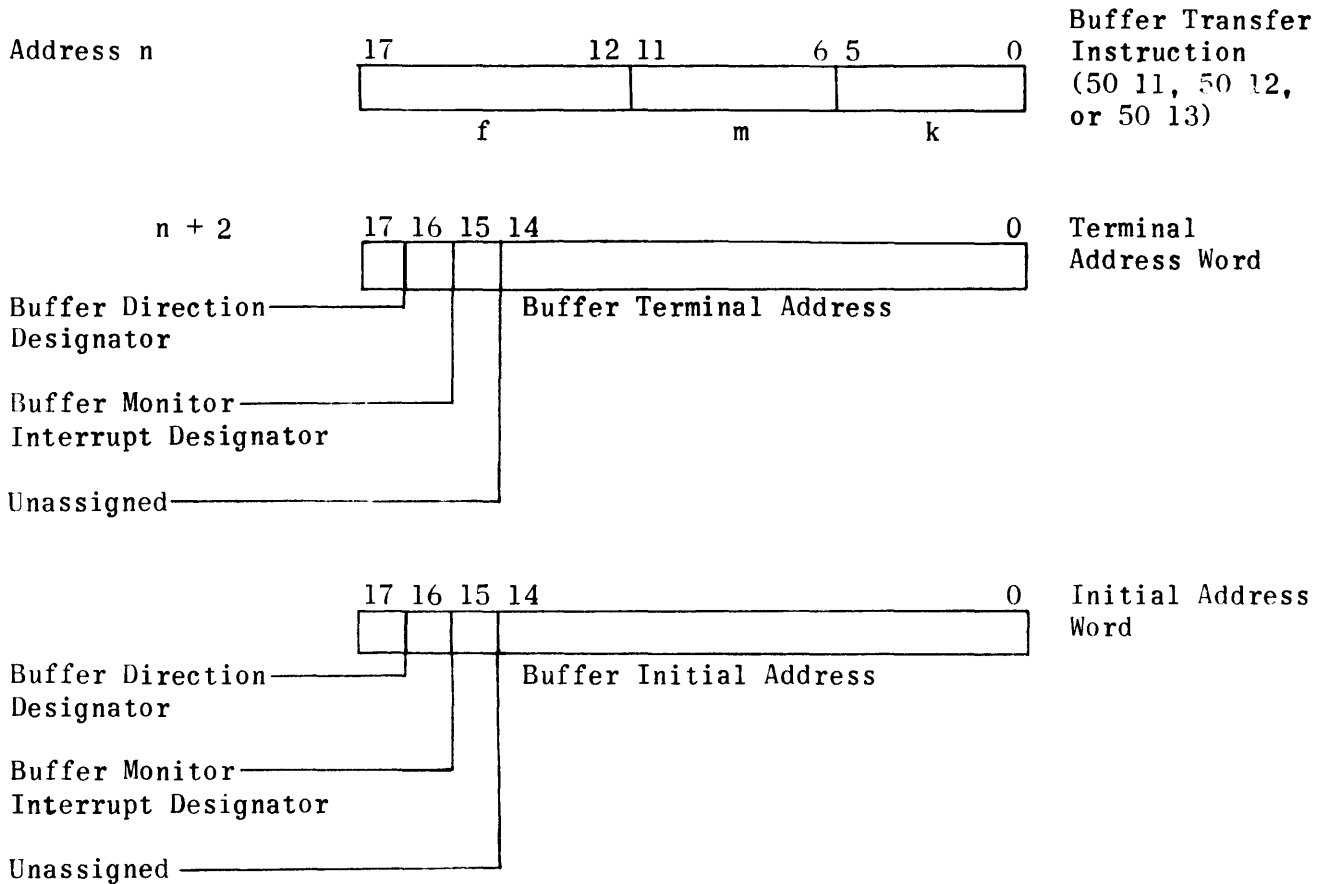
Unassigned ──────────────────────┘

The word at address n + 1 is stored in the first of the two assigned buffer
control registers. This word contains the buffer terminal address in the lower
15 bits, the buffer direction designator, the continuous data in the upper-most
bit, and buffer monitor interrupt designator in bit 16. For input transfers,
the buffer terminal address is the address of the last word to be transferred
during this buffer operation. For output or external function transfers, the
buffer terminal address must be one greater than the address of the last word
to be transferred from an incrementing buffer or one less than the address of
the last word to be transferred from a decrementing buffer.

The word at address n + 2 is stored in the second of the two assigned buffer
control registers. This word contains the initial buffer address in the lower
15 bits and the monitor interrupt designator and buffer direction designator in
bits 16 and 17, respectively. If set to 1, the monitor interrupt designator
causes a monitor interrupt to occur when the buffer is terminated. The buffer
direction designator is used to specify either a forward buffer or a backward
buffer. If this bit is zero, the buffer initial address must be less than the
buffer terminal address and the initial address control word is incremented
after each equality check and word transfer. If this bit is set to 1, the buffer
initial address must be greater than the buffer terminal address and the initial
address control word is decremented after each equality check and word transfer.
Bits 16 and 17 must be set to the same configuration in both words, n + 1 and
n + 2.

I-B-41

It should be noted that, in the input mode, the I/O section performs one last transfer after the equality check indicates that the buffer is to be terminated; therefore, when a one word buffer is to be transferred, both the initial and terminal addresses must be set to the address of the one word to be transferred. In output or external function modes of operation, the terminal address for a one word output or external function buffer must be one greater or one less than the initial address, which is the address of the word to be transferred. The reason for this is that the computer waits for one additional output request or external function request from the peripheral equipment to ensure that the last word of the buffer was received. No transfer results from this additional request.

50 11   INPUT TRANSFER (IN) $(P + 1) \longrightarrow 60 + 2k$

   Execution time: 20 microseconds $(P + 2) \longrightarrow 61 + 2k$

                          SET INPUT ACTIVE ON CHAN. k

   Initiate input transfer on channel k.

   Transfer buffer limit address words (for input buffer)
   from the following two instruction locations to the input
   buffer control registers for the designated channel. Other
   I/O channel and processor activity proceeds normally.

50 12   OUTPUT TRANSFER (OUT) $(P + 1) \longrightarrow 40 + 2k$

   Execution time: 20 microseconds $(P + 2) \longrightarrow 41 + 2k$

                          SET OUTPUT ACTIVE ON CHAN. k

   Initiate output transfer on channel k.

   Transfer buffer limit address words (for output buffer)
   from the following two instruction locations to the out-
   put buffer control registers for the designated channel.
   Other I/O channel and processor activity proceeds normally.

50 13   EXTERNAL FUNCTION (EXF) $(P + 1) \longrightarrow 20 + 2k$

   Execution time: 20 microseconds $(P + 2) \longrightarrow 21 + 2k$

                          SET EXTERNAL FUNCTION ON CHAN k

   Initiate external function mode on channel k.

   Transfer buffer limit addresses (for the function buffer)
   from the following two instruction locations to the EXF
   buffer control registers for the designated channel.

3. BUFFER TERMINATION INSTRUCTIONS

Normally buffer transfers are terminated automatically by the I/O section after the last word of the buffer has been transferred. However, under certain conditions the programmer may wish to terminate a buffer under program control. Three instructions are provided for this purpose. These instructions (50 15, 50 16, and 50 17) immediately terminate the buffer transfer on the specified channel and deactivate the channel.

50 15   TERMINATE INPUT (INSTP)

                                   CLEAR INPUT ACTIVE CHAN. k

          Execution time: 4 microseconds

          Terminate input on channel k.

          No monitor interrupt will occur as a result of
          the execution of this instruction.

50 16   TERMINATE OUTPUT (OUTSTP)

                                   CLEAR OUTPUT ACTIVE CHAN. k

          Execution time: 4 microseconds

          Terminate output on channel k.

          No monitor interrupt will occur as a result of
          the execution of this instruction.

50 17   TERMINATE EXTERNAL FUNCTION (EXFSTP)

                              CLEAR EXTERNAL FUNCTION ACTIVE CHAN. k

          Execution time: 4 microseconds

          Terminate external function mode on channel k.

          No monitor interrupt will occur as a result of
          the execution of this instruction.

4. OVERRIDE INSTRUCTIONS

Certain peripheral equipments accept external functions and output data from
the computer only if the transfer is forced by the computer. The two override
instructions (50 26 and 50 27) force a transfer by simulating a request signal
(output data request or external function request), placing the information on
the data lines, and setting the acknowledge signal. Upon detecting the acknow-
ledge signal, the peripheral device accepts the data just as if the peripheral
device had requested it. It should be noted that the execution of an override
instruction forces only one transmission; it does not initiate an automatic
buffer.

For example, to send a 3-word external function buffer to a magnetic tape unit,
the external function override instruction (50 27) must be executed three times,
and the programmer must provide a delay between executions to allow time for
the tape unit to accept the information.

50 26   OUTPUT OVERRIDE (OUTOV)

          Execution time: 4.67 microseconds

          Wait for the output device to accept the word in the
          C register(s). Then simulate an output request on
          channel k and transfer the word designated by the
          address in the output buffer control register for
          that channel. Ignore the ESI mode if active. This

instruction will transfer a word whether the buffer
is active or not. The transfer takes place under
control of the output buffer control registers.

50 27   EXTERNAL FUNCTION OVERRIDE (EXFOV)

Execution time: 4.67 microseconds

Wait for the output device to accept the word in the
C register(s). Then simulate an external function
request on channel k and transfer the word designated
by the address in the external function buffer control
register for that channel. Ignore the ESI mode if active.
This instruction will transfer a word whether the buffer
is active or not. The transfer takes place under control
of the external function buffer control registers.

5. MISCELLANEOUS I/O INSTRUCTIONS

The instruction repertoire includes the following instructions which are useful
in programming the transfer of information between the computer and peripheral
devices.

50 20   SET RESUME (SRSM)

Execution time: 4 microseconds

Set the resume designator for channel k group to
permit honoring the next requesting output function
on that group. Loss of any information currently
held by that output register(s) for a peripheral
device is allowed by this instruction.

50 21   SKIP ON INPUT INACTIVE (SKPIIN)

Execution time: 6 microseconds skip; 4.67 no skip

Test for input activity on channel k. If inactive, skip
the next instruction; otherwise, take the next instruction.

50 22   SKIP ON OUTPUT INACTIVE (SKPOIN)

Execution time: 6 microseconds; 4.67 no skip

Test for output activity on channel k. If inactive, skip
the next instruction; otherwise, take the next instruction.

50 23   SKIP ON EXTERNAL FUNCTION INACTIVE (SKPFIN)

Execution time: 6 microseconds skip; 4.67 no skip

Test for external function activity on channel k. If
inactive, skip the next instruction; otherwise, take the
next instruction.

50 24   WAIT FOR INTERRUPT (WTFI)
  or
50 25

            Execution time: 4 microseconds

            Stop the computer until any interrupt occurs and
            allow I/O to continue; ignore k, then execute the
            instruction located in the interrupt entrance
            register designated by the interrupt.

50 30   REMOVE INTERRUPT LOCKOUT (RIL)
  or
50 31

            Execution time: 4 microseconds

            Remove the interrupt lockout; enable all external and
            monitor interrupts, all channels. Ignore k. This
            instruction should be used only if interrupt lock-
            out was set with a 50 34 or 50 35 instruction
            (SIL). It will not remove interrupt lockout set
            with a 50 36 or 50 37 instruction (SXL).

50 32   REMOVE EXTERNAL INTERRUPT LOCKOUT (RXL)
  or
50 33

            Execution time: 4 microseconds

            Enable external interrupts, all channels. Ignore k.
            This instruction should be used only if interrupt
            lockout was set with a 50 36 or 50 37 instruction
            (SXL). It will not remove interrupt lockout set
            with a 50 34 or 50 35 instruction (SIL).

50 34   SET INTERRUPT LOCKOUT (SIL)
  or
50 35

            Execution time: 4 microseconds

            Set the interrupt lockout; disable all external and
            monitor interrupts, all channels. Ignore k.

50 36   SET EXTERNAL INTERRUPT LOCKOUT (SXL)
  or
50 37

            Execution time: 4 microseconds

            Disable external interrupts, all channels. Ignore k.

54   INDIRECT JUMP AND REMOVE INTERRUPT LOCKOUT (IJPEI)

$$(Y) \longrightarrow P \text{ and } RIL$$

Execution time: 8 microseconds

$y = u_p$ Address $= (y)_{14-0}$
Remove interrupt lockout (enable interrupts). Then jump
to the address which is the low order 15 bits of (y).
An application of this instruction is the termination of
a subroutine activated by an interrupt.

50 57   SKIP ON NO RESUME (SKPNR)

Execution time: 6 microseconds skip; 4.67 no skip

If the resume designator on channel k is not set (indicat-
ing unsuccessful transfer of a word to an output device),
skip the next sequential instruction; otherwise, take the
next instruction.

SECTION I-C.   INPUT/OUTPUT (I/O) CHARACTERISTICS

1. GENERAL

Communication between the computer and external equipment is accomplished by a computer program and the I/O section of the computer via the I/O channels and their associated control circuits. The modular design of the I/O section provides for two customer selected options:

1) Number of I/O channels.
2) Type of interface.

The I/O section may consist of one or two modules, each of which contains four I/O channels and their associated control circuits. Channel numbering depends upon the I/O configuration selected. The list below gives the channel numbers for the two possible selections.

Number of Channel	Channel Numbers
4	0, 2, 4, and 6
8	0 through 7

Each I/O channel has two cables: one for input and one for output. On the output cable the computer sends command codes (external functions) and output data to external equipment. On the input cable the computer receives status information (external interrupts) and input data from the external equipment. Each cable contains 18 data lines and the control lines necessary to effect data transfer. Data transfer is carried on in a parallel mode; that is, all information bits are transferred at once. In single-channel mode 18 bits are transferred. In dual-channel mode two channels act as one and 36 bits are transferred.

Except for external interrupts, all data is transferred to or from buffers in the computer's main memory. The program need only initiate the transfer and specify the buffer limits (first and last addresses of the buffer area). The I/O section then carries out the buffer transfer without further program control. The transfer is directed by request and acknowledge control signals. The external equipment must request the transfer of each word and the computer must acknowledge the transfer of each word. The rate at which data is transferred is normally determined by the request rate of the external equipment since this is usually slower than the computer's maximum transfer rate.

The maximum data transfer rate of the computer is dependent upon two factors: I/O channel interface design and main memory cycle time. The design of the computer/external equipment interface (I/O channels and associated circuits) limits the maximum data transfer rate on each I/O module (four channel group). Two types of interface are available. One type of interface is designed to transfer data at 0 and -3 volt signal levels and is often called fast interface.

The second type is designed to transfer data at 0 and -15 volt signal levels
and is often called slow interface. The two interface types are optional in
four channel groups, and both types may be used on one computer. For example,
an eight-channel computer may have four -3 volt channels and four -15 volt
channels. Table I-C-1 gives the maximum data transfer rates for both inter-
face types. The single channel rate applies to one channel or a combination
of channels in the same four-channel group. The dual-channel rate applies to
dual-channel mode which combines two channels from different four channel
groups. All rates specified are for one-way communication; the total through-
put rate (simultaneous input and output) is limited to 50,000 36-bit words or
62,500 18-bit words by the maximum main memory rate.

TABLE I-C-1. DATA TRANSFER RATES

Interface Type	Channel Configuration	Maximum Data Transfer Rate (Words per Second)
-3 volt	Single-Channel	41,667 18-bit words
-3 volt	Dual-Channel 1 odd channel and 1 even channel	100,000 18-bit words (50,000 36-bit words)
-3 volt	Multi-Channel 1 or more odd channel(s) and 1 or more even channel(s)	62,500 18-bit words (max main memory rate)
-15 volt	Single-Channel	31,667 18-bit words
-15 volt	Dual-Channel 1 odd channel and 1 even channel	83,334 18-bit words (41,667 36-bit words max -15 volt inter-face rate)
-15 volt	Multi-Channel 1 or more odd channels and 1 or more even channels	41,667 18-bit words

2. INPUT/OUTPUT INTERFACE

2.1 DATA TRANSFERS

All I/O channels are capable of communicating with either a peripheral equip-
ment, which is subordinate to the computer, or with another computer. The option
of peripheral operation or intercomputer operation is selectable by a switch
associated with each channel on the front panel of the computer.

2.1.1 PERIPHERAL OPERATION

When communicating with the computer, a peripheral equipment is subordinate to
the computer. The computer program initiates communication by activating a
channel and defining an input or output buffer. An input buffer is a storage
area in main memory into which data received from an external equipment is
stored. An output buffer is a storage area in main memory from which data is
read and sent to an external equipment. All data transfers, except external
interrupt transfers, are accomplished in a buffer mode. After program initia-
tion, communication is directed by control signals transmitted between the
computer and peripheral equipment over the input and output cables.

Figure I-C-1 shows the interface between the computer and a peripheral equipment.
On the output cable the computer can send either output data to be handled by
the peripheral equipment or external functions, which direct the peripheral
equipment to perform some operation. On the input cable the computer can
receive either input data to be processed by the computer or external interrupts
which provide the computer with status information concerning the peripheral
equipment. The types of transfers are differentiated from one another by the
request/acknowledge signals which control the transfer. All requests are honored
by the computer according to a fixed priority scheme.

The general sequence of events which occurs for each type of transfer is given
below.

1) Sequence for data transfer to the computer from peripheral equipment:

   a) Program control initiates input buffer for given channel.

   b) Peripheral equipment places data word on information lines.

   c) Peripheral equipment sets the input request line to indicate that
      it has data ready for transmission.

   d) Computer detects the input request.

   e) Computer samples the information lines at its own convenience.

   f) Computer sets the input acknowledge line, indicating that it has
      sampled the data.

```
┌──────────┐                                                      ┌──────────┐
│          │      External Function Request (EFR) Line            │          │
│          │◄─────────────────────────────────────────────       │          │
│          │      External Function Acknowledge (EFA) Line        │          │
│          │─────────────────────────────────────────────►       │          │
│          │      Output Data Request  (ODR) Line                 │          │
│          │◄─────────────────────────────────────────────       │          │
│          │      Output Data Acknowledge (ODA) Line              │          │
│          │─────────────────────────────────────────────►       │          │
│          │      Output Data Lines (18)                          │          │
│          │─────────────────────────────────────────────►       │          │
│          │                                                      │          │
│ Computer │                 Output Cable           Peripheral    │          │
│          │                                        Equipment     │          │
│          │                                                      │          │
│          │      External Interrupt Request (EIR) Line           │          │
│          │◄─────────────────────────────────────────────       │          │
│          │      Input Data Request (IDR) Line                   │          │
│          │◄─────────────────────────────────────────────       │          │
│          │      Input Data Acknowledge (IDA) Line               │          │
│          │─────────────────────────────────────────────►       │          │
│          │      Input Data Lines (18)                           │          │
│          │◄─────────────────────────────────────────────       │          │
│          │                                                      │          │
│          │                 Input Cable                          │          │
└──────────┘                                                      └──────────┘
```
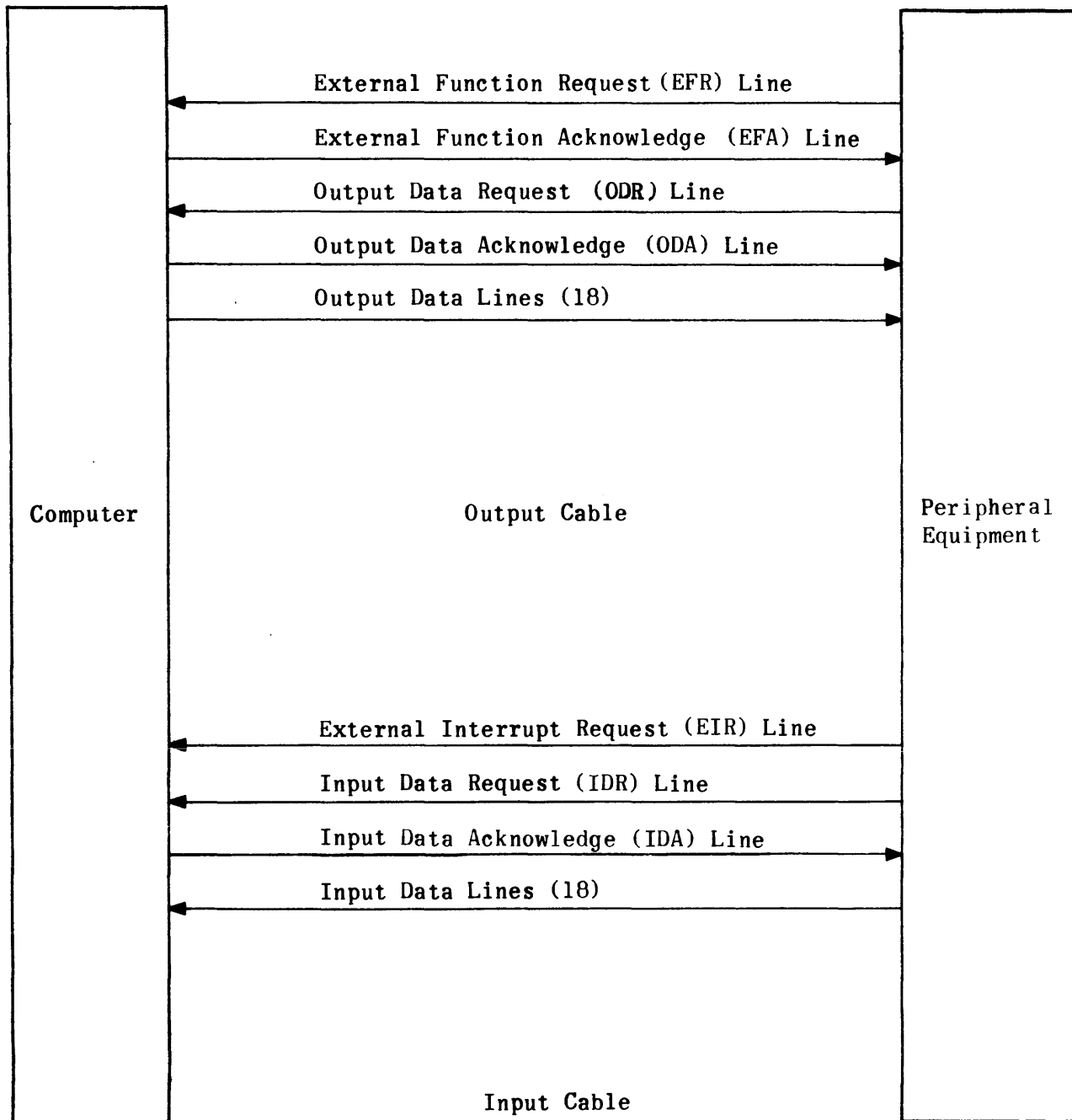
Figure I-C-1. Input/Output Interface

g) Peripheral equipment senses the input acknowledge line.

h) Peripheral equipment drops the input request line.

Steps b) through h) of this sequence are repeated for every data word until the number of words specified in the input buffer have been transferred.

2) Sequence for interrupt code transfer to the computer:

a) Peripheral equipment places the interrupt code on the information lines.

b) Peripheral equipment sets the interrupt line.

c) Computer detects the interrupt.

d) Computer samples the input lines and stores interrupt code in memory location 101 plus twice the channel number.

e) Computer sets the input acknowledge line, indicating that it has sampled the information and when no data requests or interrupt lockout exist it reads its next instruction from memory location 100 plus twice the channel number.

f) Peripheral equipment senses the input acknowledge line.

g) Peripheral equipment drops the interrupt signal.

h) Peripheral equipment may change the data lines anytime after dropping the interrupt signal.

The input acknowledge is the computer response to either an input request or to an interrupt. To eliminate misinterpretation of the input acknowledge signal, peripheral equipment must not interrupt until its last input request has been acknowledged by the computer. Under emergency conditions, when data loss is of secondary importance, a request may be dropped but data lines must remain stable for not less than four microseconds. If, during these four microseconds, an acknowledge is received, the peripheral equipment may assume successful transfer of the last data word. At any time, after the four-microsecond interval, the peripheral equipment may change the data lines and send an interrupt. When these conditions prevail, an input acknowledge signal that occurs after the interrupt is raised will be in answer to the interrupt.

3) Sequence for data transfer from computer to peripheral equipment:

a) Program control initiates output buffer for given channel.

b) Peripheral equipment sets the output request line when it is in a condition to accept data.

c) Computer detects output request.

d) Computer (at its convenience) places data on the output information lines.

e) Computer sets the output acknowledge line, indicating that data is ready for sampling.

f) Peripheral equipment detects the output acknowledge.

g) Peripheral equipment may drop output request any time after detecting output acknowledge.

h) Peripheral equipment samples the data on the output lines.

i) Computer drops output acknowledge.

All steps of this sequence except the first are repeated for every data word until the number of words specified in the output buffer have been transferred. The computer also has the option of forcing any word of an output buffer; that is, it can, under program control, send an output data word regardless of the state of the output request line.

4) Sequence for external function transfer from the computer to peripheral equipment:

a) Program control initiates external function buffer for a given channel.

b) Peripheral equipment sets the external function request line when it is in a condition to accept external functions.

c) Computer detects external function request.

d) Computer (at its convenience) places external function code on the output lines.

e) Computer sets the external function acknowledge line indicating that an external function is ready for sampling.

f) Peripheral equipment detects the external function.

g) Peripheral equipment may drop the external function request any time after detecting the external function.

h) Peripheral equipment samples the external function code on the output lines.

i) Computer drops the external function acknowledge and clears the output lines.

All steps of this sequence except the first are repeated for every external function message until the number of words specified in the external function buffer have been transferred.

The computer also has the option of forcing any word of an external function buffer; that is, it can, under program control, send an external function code regardless of the state of the external function request line for that channel. This option is necessary so that the computer can override whatever function the peripheral equipment is performing in order to re-establish positive control.

2.1.2 INTERCOMPUTER OPERATION

Any I/O channel can be selected as an intercomputer channel by a channel-associated switch on the control panel. The selection of a given channel as an intercomputer channel affects only the logic concerned with the output and external function buffers. A channel which is sending data or external functions to a given peripheral device holds the data in the output registers for a fixed minimum time period, after which any output or external function request on any other channel which is part of the same 4-channel group can cause the data to be changed. However, a channel sending data or external functions to another computer must hold the information in the output register(s) until the receiving computer acknowledges receipt of those words. This acknowledge signal is received on what is known as the output data request line (when not in intercomputer mode). This line, in the intercomputer mode, is known as the resume line.

This resume line is connected to the input acknowledge line of the receiving computer (see Figure I-C-2). Activation of the resume signal on the transmitting computer channel causes the setting of the resume flip-flop for the even or odd group of four channels. It is this flip-flop which, when set, allows the transmitting computer to proceed to the next highest priority output function (the next output data word or external function message). If an output channel is holding data for another computer and no resume is received from that computer, the output registers are tied up indefinitely and no output buffers or external function buffers to other equipment can proceed. To limit the possibility of this hang-up occurring, two instructions are provided by which the computer program can monitor the status of the resume flip-flop. These instructions are: skip on no resume (50 57k) and set resume (50 20k). The former allows examination of the resume flip-flop, and the latter allows the program to correct the situation in which the hang-up exists.

2.1.3 FORCED TRANSFERS (OVERRIDE)

The computer has the ability, under program control, to force the transfer of a word from an external function buffer or output buffer regardless of the state of the request line on that channel. Peripheral devices should have the ability to accept such forced transmissions, realizing that loss of data or even loss of a previous external function word is unimportant under conditions when this option is used. Instructions 50 26 and 50 27 are the override instructions used to accomplish forced transfers.

```
         |
         |
┌──────────┐     ┌─┴──────────────────┐     ┌──────────┐
│          │──Ready (ODA) ┤───Input Data ──────→│          │
│          │     │    Request         │     │          │
│          │──Resume (ODR)┤───Input Data_____→│          │
│          │     │    Acknowledge     │     │          │
│ Computer │  External    │  External          │ Computer │
│  Output  │──Function ───┤──Interrupt ───────→│  Input   │
│ Channel  │  Acknowledge │                    │ Channel  │
│          │  External    │                    │          │
│          │←─Function ───┤                    │          │
│          │  Request     │                    │          │
│          │     │        │                    │          │
│          ├──────18 or 36 Information Lines ──→│          │
└──────────┘     │                    │     └──────────┘
         |
```
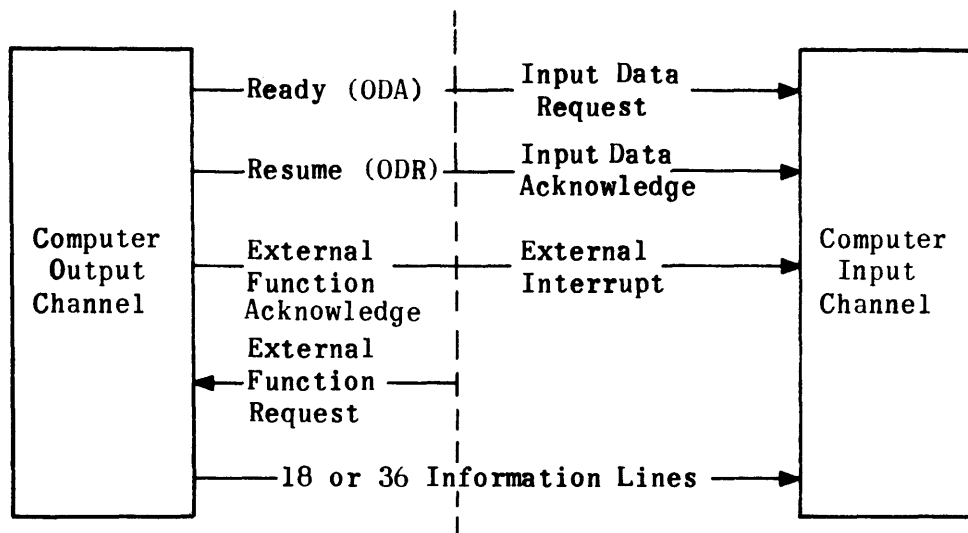
Figure I-C-2.   Intercomputer Communication

An override instruction on an intercomputer channel will not be executed until
a resume (acknowledge) is received from the receiving computer (until a resume
flip-flop is set) unless a set resume instruction precedes the override instruc-
tion in the program. Any delay, therefore, in an acknowledge from the receiving
computer will hold up the program since the program will not proceed until the
override instruction has been executed, which will not occur until the resume
flip-flop has been set.

2.2 INTERRUPTS

The computer design incorporates a means of interrupting program operation when
certain events occur. Events which cause program interruption are called inter-
rupt conditions and the signals generated to cause the interruptions are called
interrupts. When an interrupt condition occurs in the computer, the resulting
interrupt is called an internal interrupt. When an interrupt condition occurs
in an external equipment and is transmitted over an I/O channel, the resulting
interrupt is called an external interrupt.

When an interrupt occurs, a flip-flop associated with that particular interrupt
is set. The flip-flop is checked during the I/O priority scan and the
interrupt is honored according to a fixed priority sequence. When the interrupt
is honored, program control is transferred to a memory location assigned to that
interrupt. Memory locations used for this purpose are called interrupt entrance
addresses and are identified in Table I-A-1. Programmers must anticipate the
occurrence of interrupts and preset the interrupt entrance addresses with in-
structions which will either provide for interrupt processing or return control
to the interrupted program. The instruction normally set in the interrupt en-
trance address is the Indirect Return Jump instruction. This instruction stores
the address of the next instruction in the interrupted program at some memory
location (CAT) and transfers program control to that location plus one (CAT+1).

I-C-8

The location (CAT) is specified by the contents of the address in the lower 12 bits of the instruction. By using this method the programmer may load an interrupt processing routine at CAT+1 and terminate this routine with an indirect jump on CAT to return program control to the interrupted program. Most interrupts are optional in the sense that they can be disabled or locked out. This is accomplished when a set interrupt lockout (SIL) instruction is executed. The SIL instruction locks out all interrupts except fault interrupts. A set external interrupt lockout (SXL) instruction is also provided. This instruction locks out only external interrupts. Two corresponding instructions (RIL and RXL) are provided to remove these interrupt lockout conditions. A lockout condition prevents the transfer of program control; however, it does not inhibit the occurrence of the interrupt or the setting of the interrupt flip-flop. Therefore, after a lockout condition is removed, interrupts which occurred but were not honored during the lockout are honored according to the priority sequence. If two interrupts of the same type occur during a lockout, the second is ignored. Interrupts can be classified as channel interrupts and special interrupts. Channel interrupts are associated with a particular I/O channel. They are either external interrupts received on the input cable of an I/O channel or buffer monitor interrupts generated internally by the I/O section of the computer. Special interrupts are not associated with any particular channel. They provide special-purpose interrupt capability.

2.2.1 CHANNEL INTERRUPTS

1) External Interrupts.

External interrupts originate in equipment outside of the computer and are transmitted to the computer through an I/O channel. Normally an interrupt code is associated with an external interrupt. This code can be used by the computer program to interpret the meaning of the interrupt. The external equipment places the interrupt code on the input data lines and sets the external interrupt request line. During the I/O priority scan, the computer senses the external interrupt request, stores the interrupt code at the external interrupt entrance address plus one, and transfers program control to the external interrupt entrance address.

2) Buffer Monitor Interrupts.

Input data, output data, and external function buffers can be originated with or without monitor. When a buffer is initiated with monitor, an interrupt is generated when the buffer terminates. The interrupt, which is generated internally by the I/O section of the computer, transfers program control to the interrupt entrance address for that channel and buffer type.

2.2.2 SPECIAL INTERRUPTS

1) Fault Interrupts.

Fault interrupts indicate program faults. A fault interrupt is generated when an instruction with an illegal function code is encountered in the program. Illegal function codes are 00, 01, 77, 5000, 5001, and 5077.

2) Synchronizing Interrupt.

The computer is provided with one synchronizing interrupt line. This line may be connected to any external equipment. The external equipment places an interrupt signal on the line when a specific interrupt condition exists. No interrupt code is received at the time of the interrupt. If amplifying data is required, the interrupt processing routine must initiate an input buffer to accept the data. The synchronizing interrupt has higher priority than an external interrupt or any buffer monitor interrupt. The synchronizing interrupt is not locked out by a Set External Interrupt Lockout (SXL) instruction.

3. INPUT/OUTPUT PRIORITY

The computer normally performs an I/O priority scan during each of the major control sequences (I,B,R,W) required in the execution of an instruction. In addition, those instructions which employ a shift sequence (except scale factor shift) also initiate a scan during the last shift. The I/O scan examines all pending I/O operations and establishes priority in two steps.

1) The I/O scan first sets function priority for each pending I/O request and determines which has highest priority according to the function priority list (refer to Table I-C-2). Lower numbered functions have higher priority.

2) I/O scan then examines all channels on which the highest priority func- · tion is requested and sets the channel translator for the highest numbered channel requesting the function.

Priority is therefore first established according to function and then according to channel number. Since the synchronizing interrupt is not channel dependent, no channel priority determination is required for that function. Function priority for all interrupts except the synchronizing interrupt is disregarded if the interrupts are currently locked out.

TABLE I-C-2.   I/O FUNCTION PRIORITY

Priority	Function
1	Synchronizing Interrupt
2	Override Instruction
3	External Function Request
4	Output Data Request
5	Input Data Request
6	External Interrupt
7	External Function Monitor Interrupt
8	Output Monitor Interrupt
9	Input Monitor Interrupt

Priority of output type functions (external function and output data requests) alternates with priority of input type functions (input data and external interrupt) such that after an output type request has been honored, higher priority is granted to input type requests and vice versa.

The priority for fault interrupts is not shown in Table I-C-2 because the fault interrupt is not an I/O function. Priority for a fault interrupt is higher than any I/O function.

4. OPERATING MODES

The computer is capable of operating in a number of I/O modes which provide versatility in communications with external equipment. Three communication modes may be selected by switch setting on the computer control panel. These modes are single-channel, dual-channel, and externally specified indexing (ESI).

4.1 SINGLE CHANNEL MODE

In single channel mode the computer communicates with external equipment over one input/output channel. Normally both the input cable and the output cable are used to form a closed loop transmission path; however, in some cases the external equipment is not capable of two way communications. Except for external interrupt transfers all communications are accomplished in a buffer mode. Buffers are initiated by the computer program and carried to completion by the I/O section of the computer. External interrupt transfers are initiated by the external equipment. In single-channel mode the external equipment may be another computer or a peripheral equipment. Single-channel communication with a peripheral equipment is described in the following paragraphs. Intercomputer communication is carried out in the same manner except for the differences described under paragraph 2.1.2, Intercomputer Operation.

To accomplish communication between the computer and a peripheral equipment, the specific command and data transfers must be programmed in logical order. Prior to initiating an input or output data transfer the computer program must first initiate an external function buffer to prepare the peripheral equipment for the operation to be performed. The program then initiates the data transfer. After the data has been transferred the peripheral equipment may send an external interrupt to inform the computer of the status of the operation. The occurrence of the external interrupt transfer is dependent entirely upon the peripheral equipment. Some peripheral equipment always sends an external interrupt after completing (or attempting) an operation; others never send an external interrupt. In coding a program to direct an input/output operation the programmer must know whether or not an external interrupt will be received and code the program accordingly. For example, if a program is to direct the transfer of data from computer memory to magnetic tape, the program must first send the necessary external functions to command the tape unit to write data on tape in some desired format. The program must then initiate an output data transfer to move the data from memory to the magnetic tape unit. When the data has been transferred, or when an error is encountered, the magnetic tape unit sends an external interrupt indicating the status of the operation. The program may then interpret the status information and proceed based on the status of the last operation.

The following example illustrates the roles of the program and the I/O section
in transferring a 3-word external function buffer to a peripheral equipment.
Assume that the equipment is connected to channel 02 and that the external
function words are stored in memory at addresses 10000, 10001, and 10002.
The program must first activate channel 02 and define the buffer limits. The
coding required is as follows:

Address n-1 - previous instruction.
Address n   - 501302 (I/O instruction).
Address n+1 - 010003 (buffer terminal address).
Address n+2 - 010000 (buffer initial address).
Address n+3 - Next instruction.

When the computer executes the I/O instruction 501302, channel 02 is activated
and the buffer limits are transferred from addresses n+1 and n+2 to the external
function buffer control words for channel 02 (addresses 000024 and 000025).
(Note that the buffer terminal address is specified as one greater than the
address of the last buffer word.) The I/O section then assumes control of the
operation and the program proceeds to the instruction located at address n+3.
From this point on, the transfer is accomplished by request and acknowledge
signals under control of the I/O section. The peripheral equipment sends exter-
nal function requests and the computer sends external function acknowledges after
placing the external function words on the data lines. The number of words
transferred is controlled by the buffer control words. The sequence of actions
for the 3-word buffer is shown below:

I/O Action	State of Buffer Control Words
Honor EF request	(000024) = 010003
Read and compare control words	(000025) = 010000
Increment (000025) by one	010001 ⟶ 000025
Set first word, (010000), on data lines	
Set EF acknowledge line	
Honor EF request	(000024) = 010003
Read and compare control words	(000025) = 010001
Increment (000025) by one	010002 ⟶ 000025
Set second word, (010001), on data lines	
Set EF acknowledge line	
Honor EF request	(000024) = 010003
Read and compare control words	(000025) = 010002
Increment (000025) by one	010003 ⟶ 000025
Set third word, (010002), on data lines	
Set EF acknowledge line	
Honor EF request	(000024) = 010003
Read and compare control words	(000025) = 010003
Clear active flip-flop for channel 02	
Terminate buffer	

Note that the active flip-flop is cleared and the buffer is terminated immed-
iately after the buffer control words are found to be equal. After termination,
the current address control word (000025) is equal to the terminal address con-
trol word (000024). This I/O sequence is used for external function transfers
and for output data transfers. Input data transfers are executed in a slightly

different manner. The differences are illustrated by the example shown below for a 3-word input transfer on channel 02. Assume that the same memory area (addresses 10000, 10001, and 10002) is to be used for buffer storage. The input buffer control word locations for channel 02 are 00064 and 00065. The coding required to initiate the input buffer is as follows:

Address n-1 - previous instruction.
Address n   - 501102 (I/O instruction).
Address n+1 - 010002 (buffer terminal address).
Address n+1 - 010000 (buffer initial address).
Address n+3 - Next instruction.

When the computer executes the I/O instruction 501102, channel 02 is activated and the buffer limits are transferred from addresses n+1 and n+2 to the input buffer control word locations for channel 02 (addresses 00064 and 00065). (Note that in this case the buffer terminal address is specified as the address of the last buffer word.) The I/O section then assumes control of the input operation and the program proceeds to the instruction located at address n+3. The manner in which the I/O section executes the input buffer is illustrated by the sequence of actions shown below.

I/O Action	State of Buffer Control Words
Honor ID request	(000064) = 010002
Read and compare control words	(000065) = 010000
Increment (000065) by one	010001 ⟶ 000065
Set first word, (010000), on data lines	
Set ID acknowledge line	
Honor ID request	(000064) = 010002
Read and compare control words	(000065) = 010001
Increment (000065) by one	010002 ⟶ 000065
Set second word, (010001), on data lines	
Set ID acknowledge line	
Honor ID request	(000064) = 010002
Read and compare control words	(000065) = 010002
Clear active flip-flop for channel 02	
Increment (000065) by one	010003 ⟶ 000065
Set third word, (010002), on data lines	
Set ID acknowledge line	(000064) = 010002
Terminate buffer	(000065) = 010003

Note that the differences are as follows:

1) The terminal address control word must initially be equal to the address of the last buffer word rather than one greater than the address of the last buffer word (one less for decrementing buffer).

2) The active flip-flop is cleared immediately after equality of buffer control words but the sequence continues in order to complete the transfer of the last buffer word.

3) After termination the current address control word is one greater than the terminal address control word (one less for decrementing buffer).

In the examples given above the words at addresses n+1 and n+2 must always contain the buffer limits in the lower 15 bits. However, the uppermost two bits of these words may also contain information used to define two program-selectable options. The bits used and the options designated by them are given below:

1) Bit 17.

This bit is used to designate buffer direction. When this bit is not set, a forward buffer is executed; that is, the first word transferred is taken from (or stored in) the lower of the two buffer limits, and this lower limit is incremented each time a word is transferred. When this bit is set to 1, a backward buffer is executed; that is, the first word transferred is taken from (or stored in) the higher of two buffer limits, and this higher limit is decremented each time a word is transferred. Note that the address in the initial address buffer control word (n+2) must always be the address of the first word to be transferred regardless of buffer direction. Buffer direction is optional for all three types of buffers.

2) Bit 16.

When this bit is set to 1, a buffer monitor interrupt is generated when the buffer terminates. All three types of buffers can be initiated either with or without monitor.

Since the compare circuits compare all 18 bits of the control words, the upper two bits must be set to the same state in both control words.

4.2 DUAL CHANNEL MODE

Dual channel mode combines two adjacent I/O channels (00 and 01 or 02 and 03, and so forth) into one 36-bit I/O channel. Dual channel mode is available only on computers with eight channels. Selection of channels for dual channel mode does not affect the mode of operation on other channels. The channels selected for dual-channel operation must be of the same type; that is, both -3 volt interface and both -15 volt interface.

Dual channel operation combines both the input and output cables of the two channels. Data is transferred in 36-bit parallel mode; however, the I/O section of the computer handles the data as two 18-bit words. The request and acknowledge signals which control the transfer of data are transmitted over the odd numbered channel. Therefore, the odd numbered channel must be used by the programmer to activate the channels and initiate buffer operations. If a pair of channels are in dual channel mode and a request is detected on a request line of an even numbered channel, the computer interprets the request as a desire to communicate in single channel mode. The computer then replies on the even numbered channel. In this case the computer accepts input as one 18-bit word on the even channel and sends output as two identical 18-bit words on the even and odd channels.

In dual channel mode the current buffer control word, used to count the number of words transferred, is incremented or decremented twice for each 36-bit transmission; however, the control words are compared only once for each 36-bit transmission. Since buffer termination occurs only after the addresses in the two control words are found to be equal, the buffer limits must be both odd or both even when the buffer is initiated. During dual channel buffer operations, two buffer locations are filled or emptied as a result of each 36-bit word transfer. Therefore, a buffer of six memory words is sent or received during three 36-bit word transfers.

The example below illustrates the required programmed instructions and the I/O actions which occur when the computer receives a 6-word input buffer in dual channel mode. Assume that the equipment is connected to channels 02 and 03 and that the words are to be stored at locations 003000 through 003005. The coding required to initiate the buffer is as follows:

Address n-1 - previous instruction.
Address n   - 501103 (I/O instruction).
Address n+1 - 003004 (terminal address).
Address n+2 - 003000 (initial address).
Address n+3 - next instruction.

When the computer executes the I/O instruction (501103), channels 02 and 03 are activated and the buffer limits are transferred from addresses n+1 and n+2 to the input buffer control words for channel 03 (addresses 00066 and 00067). The program then proceeds to the instruction at n+3 and the I/O section assumes control of the I/O operation on channel 3.

The sequence of the I/O actions is shown below:

I/O Action	State of Buffer Control Words
Honor ID request	(00066) = 003004
Read and compare control words	(00067) = 003000
Increment (00067) by 1	003001 ⟶ 00067
Accept and store data from channel 02 (18 bits) at 003000	
Increment (00067) by 1	003002 ⟶ 00067
Accept and store data from channel 03 (18 bits) at 003001	
Set ID acknowledge	
Honor ID request	(00066) = 003004
Read and compare control words	(00067) = 003002
Increment (00067) by 1	003003 ⟶ 00067
Accept and store data from channel 02 (18 bits) at 003002	
Increment (00067) by 1	003004 ⟶ 00067
Accept and store data from channel 03 (18 bits) at 003003	
Set ID acknowledge	
Honor ID request	(00066) = 003004
Read and compare control words	(00067) = 003004
Clear active flip-flop for channel 03	

I/O Action	State of Buffer Control Words
Increment (00067) by 1	003005 ⟶ 00067
Accept and store data from channel 02 (18 bits) at 003004	
Increment (00067) by 1	003006 ⟶ 00067
Accept and store data from channel 03 (18 bits) at 003005	
Set ID acknowledge	(00066) = 003004
Terminate buffer	(00067) = 003006

The same two program-selectable options of buffer direction and monitor inter-rupt used in single-channel mode may also be used in dual-channel mode.

In the example above, note that the buffer terminal address as specified in word n+1 is the address of the second to the last word of the buffer. This is required because the last two words of the buffer are transferred after the check indicates that the buffer is to be terminated. If this buffer was to be initiated as a backward buffer, the initial address would be specified as 003005 and the terminal address would be specified as 003001, and addresses 003005 through 003000 would be filled in descending order.

The buffer limits required to initiate an external function or output data buffer in dual-channel mode are the same as those required for single-channel mode; that is, the initial address must specify the address of the first buffer word and the buffer terminal address must be one greater than the address of the last buffer word (one less for decrementing buffer). As in single-channel mode, output type buffers are terminated immediately after equality of buffer control words causes clearing of the active flip-flop. After termination, the buffer control words are equal. For example, to transfer six words, stored in addresses 03000 through 03005, as a forward output data buffer without monitor, words n+1 and n+2 must initially be set to 003006 and 003000 and after termination the control words are both set to 003006. If the same buffer is to be transferred as a backward output data buffer with monitor, words n+1 and n+2 must initially be set to 603005 and 602777 and after termination the control words are both set to 602777.

4.3 EXTERNALLY SPECIFIED INDEXING (ESI) MODE

ESI mode combines adjacent I/O channels into one 36-bit channel. Selection of two channels for ESI mode does not affect the mode of operation on the other channels. The channels selected for ESI mode must be of the same type; that is, both -3 volt interface or both -15 volt interface. ESI mode is available only on computers with eight I/O channels.

The ESI mode allows the external equipment to specify the first address of any pair of control memory locations which are used, instead of the normal buffer control words, to control the I/O operation. The program must establish buffer limits in these control memory locations before initiating an ESI operation. To initiate the operation the program need only activate the odd channel with an I/O buffer instruction (5011XX for input, 5012XX for output.) However, because of the manner in which these instructions are executed, the buffer initiating instruction must be followed by dummy buffer limits. These limits may be zero or any constant value required by other portions of the program. They are

required only because the computer always stores the words from n+1 and n+2 and proceeds to n+3 after executing a buffer initiating instruction at address n. The sequence of events which occurs during an output buffer transfer in ESI mode is as follows:

1) Computer program activates the output on the odd numbered channel and stores the dummy buffer limits from program addresses n+1 and n+2 in the normal output buffer control word locations for the odd numbered channel.

2) External equipment sets a 15-bit even-numbered, index address, I, on odd channel input cable (the address may be anywhere in main memory).

3) External equipment sets the output data request line on the odd channel.

4) When the computer honors the request, the words at addresses I and I+1 are used as buffer control words.

5) After checking control words and incrementing (I+1), the computer sets the output word on both output channels.

6) Computer sets the output acknowledge line on the odd channel.

7) Steps 2) through 6) are repeated for each word of the buffer; when the contents of I and I+1 are equal during the equality check the buffer is terminated in the same manner as a single-channel buffer.

The sequence of events for input is similar. All input control signals are set on the odd channel as is the index address, I. However, the peripheral equipment places the input word (18 bits) on the data lines of the even channel before setting the input request on the odd channel. When it is accepted by the computer, the input word is stored at the address designated by the lower 15 bits of I+1.

After the index address, I, is specified by the external equipment, the two words stored in memory at I and I+1 are treated as normal buffer control words. The three program-selectable options of buffer direction and monitor interrupt, used in single-channel mode, are available in ESI mode. When an ESI buffer with monitor interrupt is terminated, the index address, I, is automatically stored in the appropriate monitor interrupt status word (monitor interrupt entrance address +1 for the odd channel).

If the external equipment raises a request on the even-numbered channel, the computer interprets the request as a desire to communicate in single-channel mode. The computer ignores the index address which may appear on the odd-numbered channel and communicates with the external equipment in single-channel mode over the even-numbered channel. Output data is set on both channels; however, this does not interfere with single-channel communication.

Except for the fact that the index address is stored when a buffer with monitor is terminated, ESI buffers terminate in the same manner as single-channel buffers. Therefore, the same differences between input type buffers and output type buffers described for single-channel mode apply to ESI buffers.

SECTION II. PERIPHERAL EQUIPMENT

The computer may be connected to a variety of military or commercial peripheral equipments. These include:

1) Paper tape reader-punch units.
2) Magnetic tape systems.
3) High speed printer units.
4) Card reader-punch units.
5) Teletype printer units.
6) Display and display interface units.
7) Radars and radar adapter units.
8) Manual entry devices.

To program the communications between the computer and any of these peripheral equipments, the programmer must be familiar with the functional characteristics of the peripheral equipment as well as those of the computer. It is impractical to describe in this document, the functional characteristics of all possible peripheral equipments which could be connected to the computer. Therefore, the following subsections describe only those equipments which are most commonly used with the computer.

Figure II-A-1.   UNIVAC ® 1232A I/O Console

SECTION II-A. UNIVAC 1232 INPUT/OUTPUT CONSOLE

1. BASIC INFORMATION

The UNIVAC® 1232 I/O Console (Figure II-A-1) has a paper tape punch and reader as standard equipment, with a keyboard and printer as an option. Input and output devices communicate with the computer through a single I/O channel. See Figure II-A-2.

1.1 ON-LINE OPERATION

In the on-line operation the I/O console provides means for entering data into the computer by punched tape or an alphanumeric keyboard. It provides means for recording output data from the computer by either punching tape or printing on paper media or both simultaneously.

1.2 OFF-LINE OPERATION

In the off-line operation the I/O console provides means to:

1) Print on paper media by keyboard entry.
2) Perforate tape by keyboard entry.
3) Perforate tape and print on paper media simultaneously by keyboard entry.
4) Print on paper media from a perforated tape.
5) Perforate tape from a perforated tape.
6) Perforate tape and print on paper media simultaneously from a perforated tape.

2. INPUT/OUTPUT CONTROL

The I/O sequences are manually enabled from the control panel or automatically enabled by the computer program.

2.1 COMPUTER CONTROL

The computer controls the I/O console through the external-function word, as specified in Figure II-A-3, as follows:

1) Bits 0, 1, and 2 control the output devices. A one in bit 0 allows the status of the printer (bit 1) and perforator (bit 2) to be controlled by the information in bits 1 and 2. A zero in bit 0 causes bits 1 and 2 to be ignored and the status of the output devices to remain unchanged. With a one in bit 0, a one in bit 1 enables the printer and a zero in bit 1 disables the printer, and a one in bit 2 enables the perforator and a zero in bit 2 disables the perforator.

2) Bits 3, 4, 5, and 6 control the input devices. A one in bit 3 allows the status of the keyboard (bit 4) and reader (bits 5 and 6) to be controlled by the information in bits 4, 5, and 6.

3) A zero in bit 3 causes bits 4, 5, and 6 to be ignored and the status of the input devices remain unchanged. With a one in bit 3; a one in bit 4
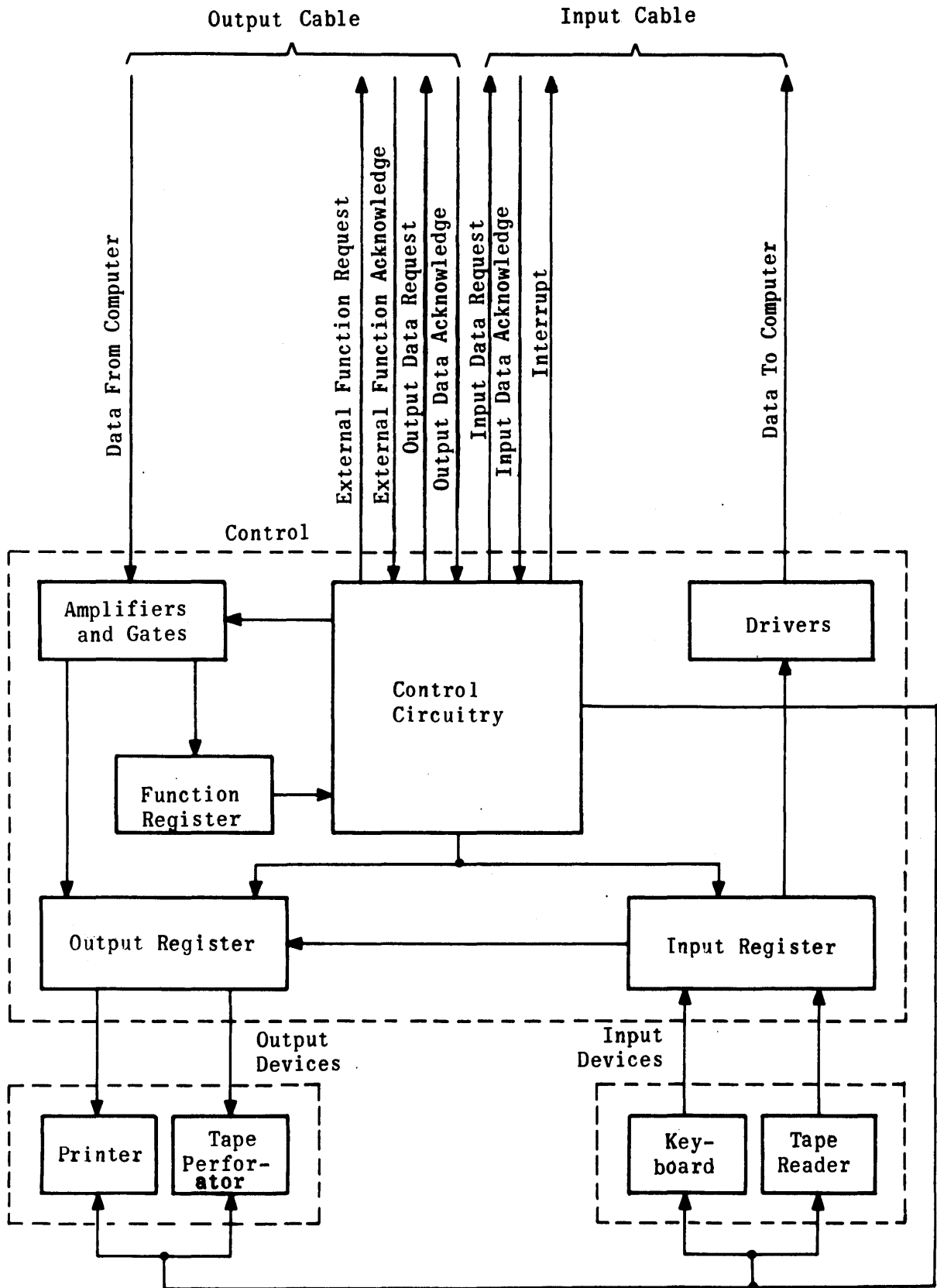
Figure II-A-2. Block Diagram of Console

29	7	6	5	4	3	2	1	0
Not used								

0 = Disable output
1 = Enable output

0 = Disable printer
1 = Enable printer

0 = Disable punch
1 = Enable punch

0 = Disable input
1 = Enable input

0 = Disable keyboard
1 = Enable keyboard

0 = Disable reader
1 = Enable reader

0 = Stop reading operation if $i_5 = 0$
1 = Start reading operation if $i_5 = 1$
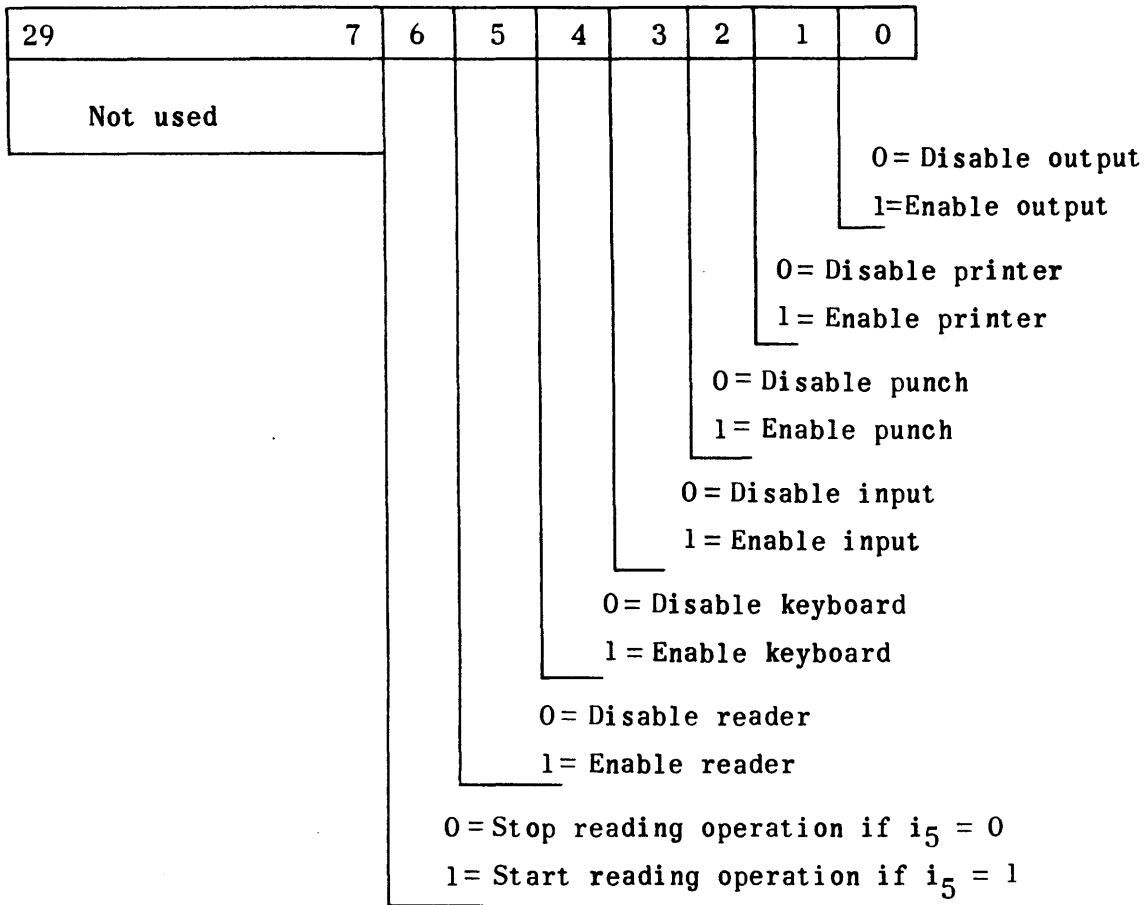
Figure II-A-3.   1232 I/O Console, External Function Word

enables the keyboard and ` zero in bit 4 disables the keyboard, a one in
bit 5 enables the reader and a zero in bit 5 disables the reader, and a
one in bits 5 and 6 starts the reading operation and a zero in bits 5
or 6 stops the reading operation.

The status of the I/O console is determined by the latest external-function word.

2.2 PANEL CONTROL

The computer external-function words are manually duplicated by the operation
of the control panel switches specified in Table II-A-1.

TABLE II-A-1.   MANUAL-AUTOMATIC CONTROLS

Unit(s) Controlled	External-Function Word Bit	Control Panel Switches	
		Set	Clear
Output Devices	0	None	None
Printer	1	Print	Print Clear
Perforator	2	Punch	Punch Clear
Input Devices	3	None	None
Keyboard	4	Keyboard	Keyboard Clear
Reader	5	Read	Read Clear
Reader	6	Start-Read	Start-Read Clear

3. OPERATION OF UNITS

3.1 PERFORATED TAPE READER

The perforated tape reader is adjustable to read chad-type tape with 5, 6, 7,
or 8 channels and widths of 11/16 inch, 7/8 inch, or 1 inch. The reader reads
tape at a rate of 300 frames per second. The tape is transported through the
reader by an electric motor drive with a pinch roller and a brake. The
following sequence is typical:

1) The reader is enabled and the motor attains operating speed.

2) Tape is placed in the reader and the start read indicator-switch is
   operated.

3) If a sprocket hole of the tape is positioned over the sensor, no advance-
   ment of the tape shall occur; if the tape is positioned so that the
   sensor is between sprocket holes, the clutch shall be engaged and the
   tape will be advanced.

4) The next sprocket hole that reaches the sensor actuates the brake and
   the tape stops.

5) The signal caused by the data holes in each frame sets the corresponding
   input lines through the action of the input register.

6) The signal caused by the sprocket hole causes the control circuitry to set the input-data-request line.

7) The computer responds with an input-data acknowledge signal, which indicates that the input-data lines have been sampled. The control circuitry clears the input-data-request line, clears the input-data lines, and engages the clutch to advance the tape.

Steps 4 through 7 are repeated until operation of the reader is stopped.

3.2 TAPE PERFORATOR

The tape perforator perforates chad-type tape. The tape perforator is adjustable to perforate 5, 6, 7, or 8 channels on 11/16 inch, 7/8 inch or 1 inch tape. It perforates 10 frames per inch at a tape speed of 11 inches per second. The tape is transported through the perforator by an electric motor drive. The following sequence is typical.

1) The perforator is enabled.

2) The control circuitry sets the output-data-request line.

3) The computer, in synchronism with internal priorities, detects the output-data-request signal.

4) The computer places data on the output line.

5) The computer sets the output-acknowledge line.

6) The control circuitry detects the output-acknowledge signal, gates the data on the output-data lines to the output register, and clears the output-data-request line.

7) A magnetic head associated with the perforator drive mechanism generates a pulse at the appropriate time to gate the output register content to the tape perforator. This energizes the perforating mechanism while the tape is stopped.

8) The control circuitry generates a pulse that de-energizes the perforating mechanism, clears the output register, and sets the output-data-request line.

Steps 3 through 8 are repeated until perforator operation is stopped.

3.3 PRINTER

The printer prints data, one character at a time, on paper media. The printer prints a character corresponding to the field-data code as specified in Table II-A-2. The printer can print 10 characters per second. The printout has 10 characters per inch horizontally, 72 characters per line, and 6 lines per inch vertically. The following sequence is typical:

1) The printer is enabled.

2) The control circuitry sets the output-data-request line.

3) The computer, in synchronism with internal priorities, detects the output-data-request signal.

4) The computer places data on the output data lines.

5) The computer sets the output-acknowledge line.

6) The control circuitry detects the output-acknowledge signal, gates the data on the output-data lines to the output register, and clears the output-data-request line.

7) The control circuitry causes the printer to perform the print or control function indicated by the data bits in the register.

8) Upon completion of the print function the control circuitry clears the output register and sets the output-data-request line.

Steps 3 through 8 are repeated until operation of the printer is stopped.

3.4 KEYBOARD

The keyboard, Figure II-A-4, generates the data codes in Table II when corresponding labeled keys are operated. Data entered into the keyboard is simultaneously printed by the printer if the printer and the copy mode are enabled. The following sequence is typical:

1) The keyboard is enabled.

2) When a key is operated, the corresponding input-data lines are set through the action of the input register.

3) The control circuitry sets the input-data-request line.

4) When the computer responds with an input-data-acknowledge signal, the control circuitry clears the input-data-request line and the data lines.

Steps 2 through 4 are repeated each time a key is depressed until operation of the keyboard is stopped.

TABLE II-A-2. FIELD DATA CODE

Symbol or Function	Keyboard Symbol	Printed Symbol	Signals on Data Lines						Octal Code
			2^5	2^4	2^3	2^2	2^1	2^0	
Master Space*		↺	0	0	0	0	0	0	00
Upper Case	UC	~	0	0	0	0	0	1	01
Lower Case	LC	‰	0	0	0	0	1	0	02
Line Feed	LF		0	0	0	0	1	1	03
Carriage Return	RETURN		0	0	0	1	0	0	04
Space			0	0	0	1	0	1	05
A	A	A	0	0	0	1	1	0	06
B	B	B	0	0	0	1	1	1	07
C	C	C	0	0	1	0	0	0	10
D	D	D	0	0	1	0	0	1	11
E	E	E	0	0	1	0	1	0	12
F	F	F	0	0	1	0	1	1	13
G	G	G	0	0	1	1	0	0	14
H	H	H	0	0	1	1	0	1	15
I	I	I	0	0	1	1	1	0	16
J	J	J	0	0	1	1	1	1	17
K	K	K	0	1	0	0	0	0	20
L	L	L	0	1	0	0	0	1	21
M	M	M	0	1	0	0	1	0	22
N	N	N	0	1	0	0	1	1	23
O	O	O	0	1	0	1	0	0	24
P	P	P	0	1	0	1	0	1	25
Q	Q	Q	C	1	0	1	1	0	26
R	R	R	0	1	0	1	1	1	27
S	S	S	0	1	1	0	0	0	30
T	T	T	0	1	1	0	0	1	31
U	U	U	0	1	1	0	1	0	32
V	V	V	0	1	1	0	1	1	33
W	W	W	0	1	1	1	0	0	34
X	X	X	0	1	1	1	0	1	35
Y	Y	Y	0	1	1	1	1	0	36

*Master space indicates absence of information.

TABLE II-A-2. FIELD DATA CODE (CONT.)

Symbol or Function	Keyboard Symbol	Printed Symbol	Signals on Data Lines						Octal Code
			2^5	2^4	2^3	2^2	2^1	2^0	
Z	Z	Z	0	1	1	1	1	1	37
)	)	)	1	0	0	0	0	0	40
-	-	-	1	0	0	0	0	1	41
+	+	+	1	0	0	0	1	0	42
<	<	<	1	0	0	0	1	1	43
=	=	=	1	0	0	1	0	0	44
>	>	>	1	0	0	1	0	1	45
—	—	—	1	0	0	1	1	0	46
$	$	$	1	0	0	1	1	1	47
*	*	*	1	0	1	0	0	0	50
(	(	(	1	0	1	0	0	1	51
"	"	"	1	0	1	0	1	0	52
:	:	:	1	0	1	0	1	1	53
?	?	?	1	0	1	1	0	0	54
!	!	!	1	0	1	1	0	1	55
,	,	,	1	0	1	1	1	0	56
Stop	Ⓢ	Ⓢ	1	0	1	1	1	1	57
0	0	Ø	1	1	0	0	0	0	60
1	1	1	1	1	0	0	0	1	61
2	2	2	1	1	0	0	1	0	62
3	3	3	1	1	0	0	1	1	63
4	4	4	1	1	0	1	0	0	64
5	5	5	1	1	0	1	0	1	65
6	6	6	1	1	0	1	1	0	66
7	7	7	1	1	0	1	1	1	67
8	8	8	1	1	1	0	0	0	70
9	9	9	1	1	1	0	0	1	71
'	'	'	1	1	1	0	1	0	72
;	;	;	1	1	1	0	1	1	73
/	/	/	1	1	1	1	0	0	74
.	.	.	1	1	1	1	0	1	75
Special	SPEC	□	1	1	1	1	1	0	76
Idle	↑	↑	1	1	1	1	1	1	77
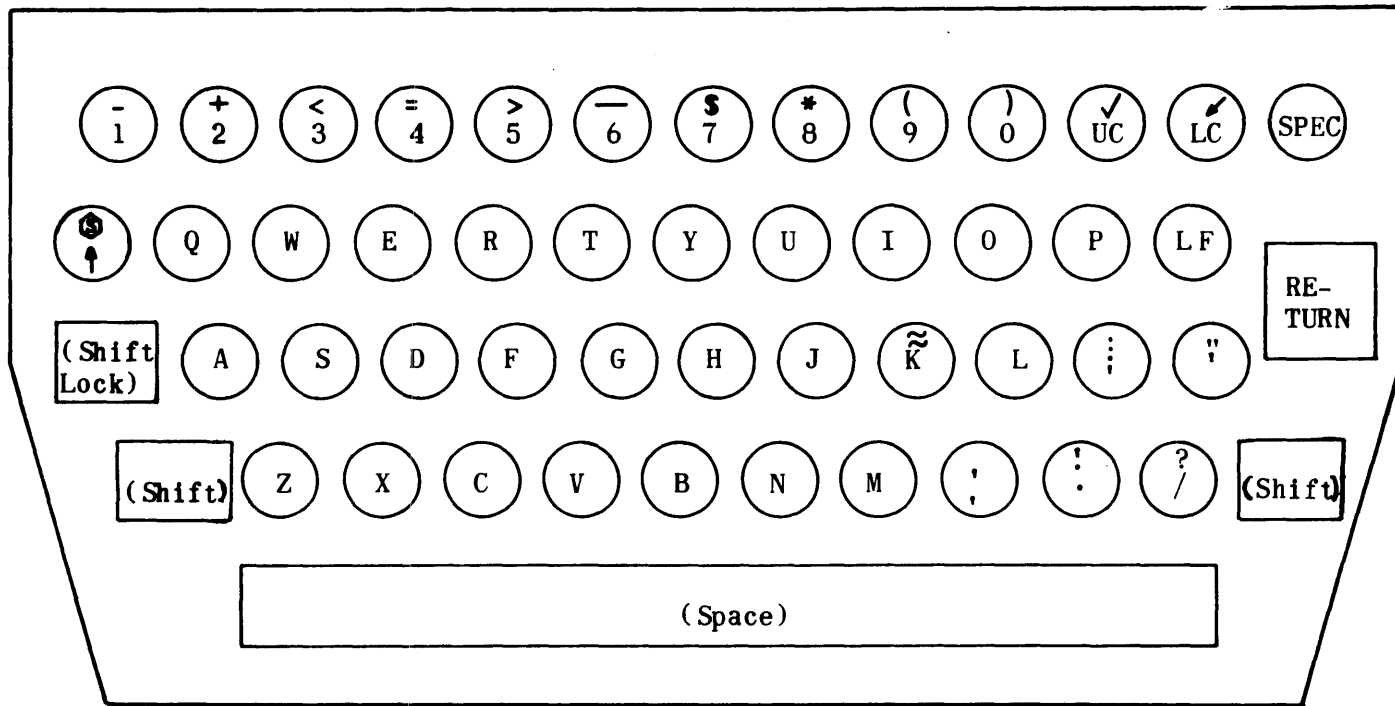
NOTE: Functions shown in parentheses are blank.

Figure II-A-4. Keyboard Layout

3.5 KEYBOARD INTERRUPT

The computer may be interrupted from the keyboard by the following sequence:

1) Keyboard is enabled.

2) Printer and copy mode are enabled if printout of the interrupt code is desired.

3) The interrupt indicator switch on the control panel has been operated.

4) A keyboard key is operated which sets the corresponding field-data code on the input-data lines and generates an interrupt to the computer.

5) When the computer responds with an input acknowledge, the interrupt and the input-data lines will be cleared.

Steps 3 through 5 are repeated for each interrupt code to be sent.

3.6 SWITCHES AND INDICATORS

The switches and indicators of the I/O console operate as follows:

1) Power switch and indicator

    The power switch switches the input power on and off. The power indicator lights whenever the power switch is in the on position.

2) On-line off-line switch

    In the on-line position the I/O console operates as an I/O device for the computer as specified herein. In the off-line position the I/O console operates independently of the computer and performs the off-line functions specified herein.

3) Tape-feed indicator switch

    The tape perforator generates blank tape with only the sprocket holes perforated whenever the tape-feed indicator switch is operated.

4) Tape-levels switch

    The tape-levels switch disables the perforated-tape-reader levels which are not selected.

5) Input-data indicator switches

    The eight input-data indicator switches display the data stored in the input register and enable data to be manually entered into the input register.

6) Output-data indicator switches

The eight output-data indicator switches display the data stored in the output register and enables data to be manually entered into the output register.

7) Master clear switch

The master clear switch stops operation of all units of the I/O console and sets all the logic to an initial state.

8) Interrupt indicator switch

The interrupt indicator switch enables the generation of an interrupt to the computer as specified in Keyboard Interrupt.

9) Read, read-one switch

In the read position the perforated-tape reader will read continuously. In the read-one position the perforated tape reader will read one frame, advance to the next frame, and stop. This switch is for off-line operation only.

10) Start-read indicator switch

The start-read indicator-switch starts the perforated-tape reading operation.

11) Copy indicator switch

The copy switch enables the I/O console to reproduce the data being sent to the computer by any one of the following methods:

a) Print the data on paper media.
b) Perforate the data on tape.
c) Print the data on paper media and perforate the data on tape simultaneously.

12) Copy-clear switch

The copy-clear switch disables the copy mode of operation.

3.7 EXTERNAL FUNCTION MANUAL CONTROLS

The print, punch, keyboard, and read indicator-switches, and the print-clear, punch-clear, keyboard-clear, and read-clear switches enable and disable their respective units as specified in the I/O panel control.

SECTION II-B.   UNIVAC 1532 INPUT/OUTPUT CONSOLE

1. GENERAL DESCRIPTION

The UNIVAC 1532 I/O console is a ruggedized computer I/O and monitoring device designed to perform reliably in applications where extreme environmental conditions exist. The cabinet and packaging of components for the console follow the same design plans found in ruggedized UNIVAC computers and peripherals built for military applications.

The UNIVAC 1532 I/O console consists of a paper tape punch, paper tape reader, page printer*, alphanumeric keyboard*, control and computer interface logic, and power supplies assembled into a compact unit which operates on a single input/output channel. Programs or program modifications on punched paper tape (5 to 8 level) prepared off-line manually or on-line under computer program control by the punch may be loaded by the reader. Alphanumeric entries to the computer may be made at the keyboard with automatic printout. The page printer is also useful as a program monitoring device; it provides a running record of real-time and normal program activities. The 1532 I/O console may be used with all UNIVAC general-purpose military computers.

1.1 OPERATIONAL CHARACTERISTICS

1.1.1 ON-LINE MODE

In the on-line mode of operation, the console with printer option provides the following operations:

1) Read paper tape input to the computer (any code).

2) Punch paper tape (any code) from computer output.

3) Print on paper (7-bit ASCII) from computer output.

4) Keyboard input to the computer (7-bit ASCII) with print on paper.

5) Keyboard input to the computer with print on paper and (under program control) punch paper tape in any code.

6) Read paper tape input to the computer and (under program control) print on paper.

7) Read paper tape input to the computer and (under program control) punch paper tape (any code).

8) Read paper tape input to the computer and (under program control) print on paper and punch paper tape (7-bit ASCII).

9) Computer output to punch paper tape (7-bit ASCII) with print on paper.

---

*The keyboard and printer are optional items.

1.1.2 OFF-LINE MODE

In the off-line mode of operation, the console with printer option provides the following operations:

1) Print on paper by keyboard entry.
2) Punch paper tape (7-bit ASCII) and print on paper by keyboard entry.
3) Print on paper from paper tape entry (7-bit ASCII).
4) Punch paper tape from paper tape entry (any code).
5) Punch paper tape and print on paper from paper tape entry (7-bit ASCII).

2. FUNCTIONAL DESCRIPTION

2.1 GENERAL

The I/O console consists of the mechanical assemblies, control and computer interface logic and power supplies. The control unit provides the required timing and control signals for the reader, punch, keyboard, and printer, and a compatible interface for communication between the computer and the ancillary devices. Figure II-B-1 shows the information flow between the ancillary units, control unit, and computer.

The I/O console may be used as an on-line computer peripheral unit or as an off-line independent unit. As on on-line device, the output units may function as operational status monitors, operational data output units, or monitors of console input information.

In on-line operation, the computer controls the operation of the ancillary units by use of a computer control function instruction. The control unit, upon sensing the function instruction, activates the necessary control logic to switch the console into the proper operating mode. The status of the I/O console is determined by the latest function instruction.

2.2 PUNCHING OUTPUT DATA

When the computer has established the perforator enable function instruction, the output request signal is sent to the computer. The computer detects the output request and in synchronism with internal priorities places data on output lines 2^0 through 2^7. The output acknowledge line is set by the computer. The control circuits in the console detect the output acknowledge signal and gate the data on lines 2^0 through 2^7 to a data storage register. The output request signal is cleared. A magnetic head associated with the reperforator drive mechanism generates a pulse which gates the data in the storage register to the tape perforator. This energizes the perforating mechanism which punches the code on the tape while it is stopped. The control circuits then move the tape one character ahead, de-energize the perforating mechanism, clear the data storage register, and reset the output request signal. The computer continues the output buffer process until all data has been punched on tape, and then de-energizes the perforator by establishing a perforator disable function instruction. This function disables the perforator and clears the output request signal.
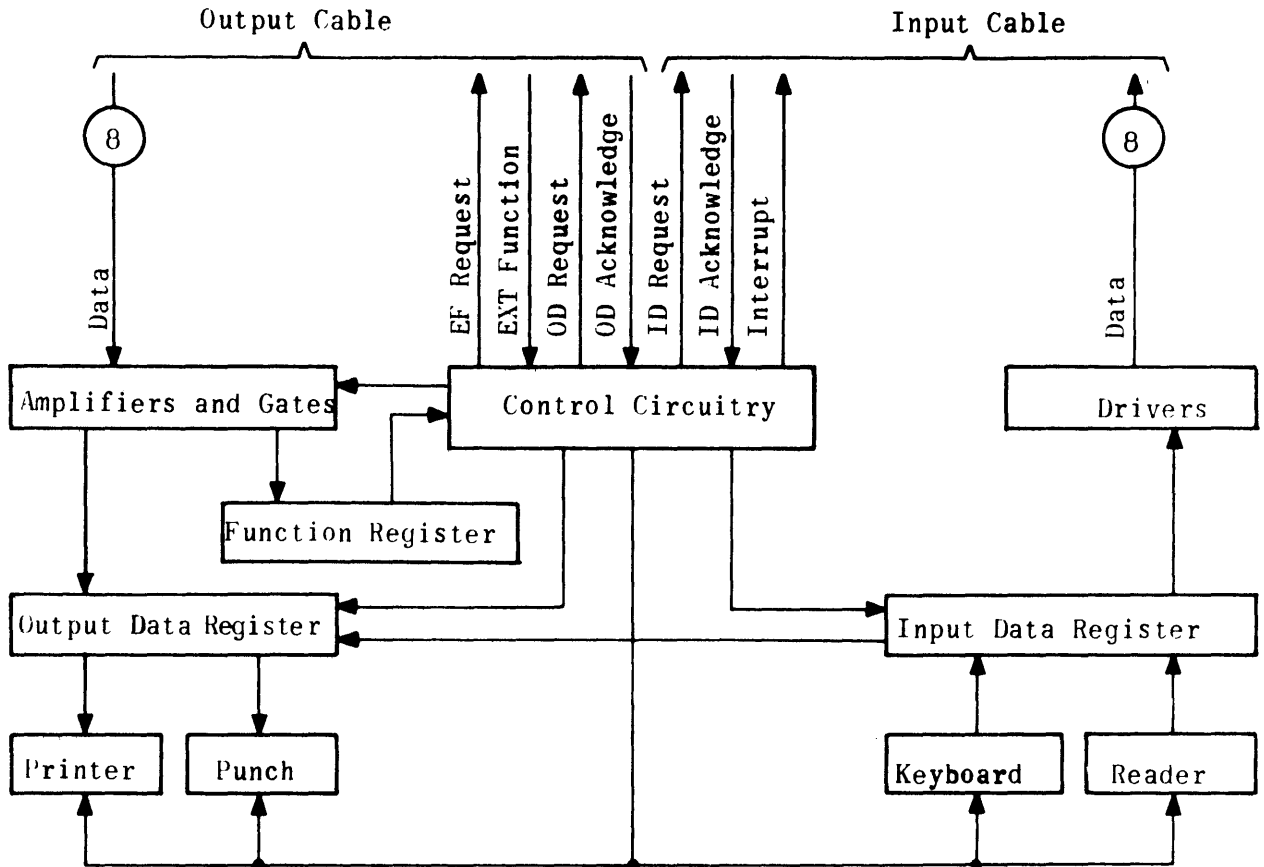
Figure II-B-1. Block Diagram of I/O Console

2.3 PRINTING OUTPUT DATA

When the computer has established the printer enable function instruction, the output request signal is sent to the computer. The computer detects the output request and in synchronism with internal priorities places data on output lines 2^0 through 2^6. The output acknowledge line is set by the computer. The control circuits of the console detect the output acknowledge signal and gate the data on lines 2^0 through 2^6 to a data storage register. The output request signal is cleared. The control circuits cause the printer to perform the print function indicated by the data bits in the storage register. The print function associated with each data code is shown in Table II-B-1. Upon completion of the print function, the control circuits clear the output register and set the output request signal. The computer continues the output buffer process until all data has been printed, and then de-energizes the printer by establishing a printer disable function instruction. This function disables the printer and clears the output request signal.

2.4 READING INPUT DATA

The operator places the tape in the reader and initiates computer operation.

TABLE II-B-1. ASCII CODE FOR THE UNIVAC 1532 KEYBOARD-PRINTER

NOTE: CTRL is a key that deletes the 2^6 bit of seven bit code.

Character Code (Bits 2^6–2^0) Non-Print Codes	Printed Character	Functioning Keys	Character Code (Bits 2^6–2^0) Printed Codes	Printed Character	Functioning Keys
000 0000		CTRL & @	010 1110	.	.
000 0001		CTRL & A	010 1111	/	/
000 0010		CTRL & B	011 0000	0	0
000 0011		CTRL & C	011 0001	1	1
000 0100		CTRL & EOT	011 0010	2	2
000 0101		CTRL & WRU	011 0011	3	3
000 0110		CTRL & RU	011 0100	4	4
000 0111		CTRL & BELL	011 0101	5	5
000 1000		CTRL & H	011 0110	6	6
000 1001		CTRL & I	011 0111	7	7
000 1010		Line Feed	011 1000	8	8
000 1011		CTRL & K	011 1001	9	9
000 1100		CTRL & L	011 1010	:	:
000 1101		Return	011 1011	;	;
000 1110		CTRL & N	011 1100	<	Shift & <
000 1111		CTRL & O	011 1101	=	Shift & =
001 0000		CTRL & P	011 1110	>	Shift & >
001 0001		CTRL & Q	011 1111	?	Shift & ?
001 0010		CTRL & Tape On	100 0000	@	Shift & @
001 0011		CTRL & X Off	100 0001	A	A
001 0100		CTRL & Tape Off	100 0010	B	B
001 0101		CTRL & U	100 0011	C	C
001 0110		CTRL & V	100 0100	D	D
001 0111		CTRL & W	100 0101	E	E
001 1000		CTRL & X	100 0110	F	F
001 1001		CTRL & Y	100 0111	G	G
001 1010		CTRL & Z	100 1000	H	H
001 1011		CTRL, Shift & K	100 1001	I	I
001 1100		CTRL, Shift & L	100 1010	J	J
001 1101		CTRL, Shift & M	100 1011	K	K
001 1110		CTRL, Shift & ↑	100 1100	L	L
001 1111		CTRL, Shift & ←	100 1101	M	M
111 1101		Alt. Mode	100 1110	N	N
111 1111		Rub Out	100 1111	O	O
			101 0000	P	P
Printed Codes			101 0001	Q	Q
			101 0010	R	R
010 0000	(space)	Space Bar	101 0011	S	S
010 0001	!	Shift & !	101 0100	T	T
010 0010	"	Shift & "	101 0101	U	U
010 0011	#	Shift & #	101 0110	V	V
010 0100	$	Shift & $	101 0111	W	W
010 0101	%	Shift & %	101 1000	X	X
010 0110	&	Shift & &	101 1001	Y	Y
010 0111	'(apos.)	Shift & '	101 1010	Z	Z
010 1000	(	Shift & (	101 1011	[	Shift & K
010 1001	)	Shift &)	101 1100	/	Shift & L
010 1010	*	Shift & *	101 1101	]	Shift & M
010 1011	+	Shift & +	101 1110	↑	Shift & ↑
010 1100	,	,	101 1111	←	Shift & ←
010 1101	–	–			

The computer starts the reader motor by establishing the start read function instruction. The tape control positions the tape in the next sprocket hole and activates the brake which stops the tape. The sensor senses the data on the tape and sets the corresponding bits 2^0 through 2^7 on the input data lines. The control circuits set the input request signal. The computer senses the input request and in synchronism with internal priorities stores the data on the input lines in a memory location. The computer sets the input acknowledge signal. The console clears the input lines and input request signal upon sensing the input acknowledge signal from the computer. The control circuits engage the clutch on the reader, advance the tape to the next frame, sense the data bits 2^0 through 2^7, and set the input request signal. The computer continues the input buffer process until all data has been read and de-energizes the reader by establishing a reader disable function instruction. This function disables the reader and clears the input request signal.

2.5 KEYBOARD INPUT

When the computer has established the keyboard enable function instruction, the keyboard-printer is prepared for operation. The operator has the choice of either interrupt or request character transmission.

The operator selects request transmission by depressing the key on the keyboard for the character required for transmission. The layout of the keyboard is shown in Figure II-B-2. When a key is depressed, the code corresponding to that key is set on input lines 2^0 through 2^6 and the corresponding character is printed. The codes corresponding to the various keys on the keyboard are shown in Table II-B-1. The control circuits set the input request signal. The computer detects the input request and in synchronism with internal priorities stores the data on the input lines in a memory location. The computer sets the input acknowledge signal. The console control circuits clear the data lines and input request signal. The operator repeats the processes by setting consecutive keys until the entire message is transmitted.

The operator selects interrupt transmission by depressing the interrupt indicator-switch on the control console and then depresses the key on the keyboard
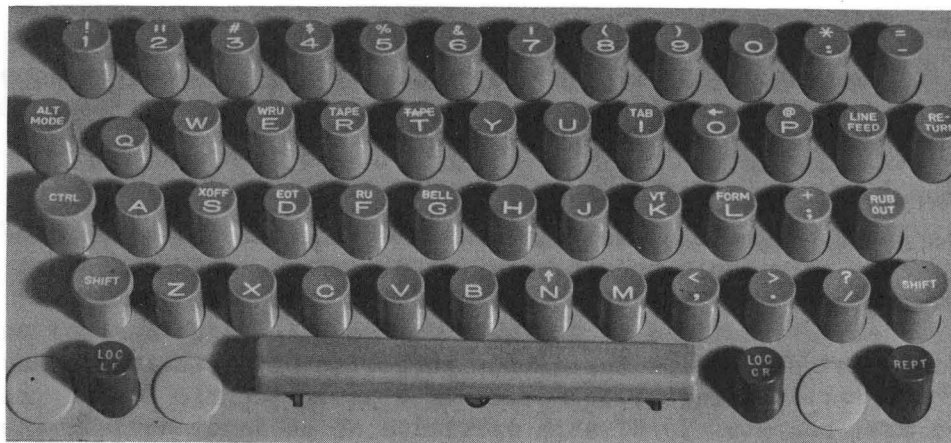
Figure II-B-2. Keyboard Layout

for the character required for transmission. When the key is operated, the code corresponding to the key depressed is set on the input lines 2^0 through 2^6 and the corresponding character is printed. The control circuits set the internal signal. The computer senses the interrupt and in synchronism with internal priorities stores the data on the input lines in a memory location assigned to that interrupt channel. The computer sets the input acknowledge signal. The console control circuits clear the interrupt signal and input data lines. The operator repeats the process by setting the interrupt indicator-switch and a keyboard key until the entire message is transmitted.

2.6 FUNCTION INSTRUCTIONS

The function instructions provide the computer, through the computer program, with a means for maintaining control of the I/O console. The function instruction words are seven-bit, position-encoded words. An illustration of the encoding is shown in Figure II-B-3.
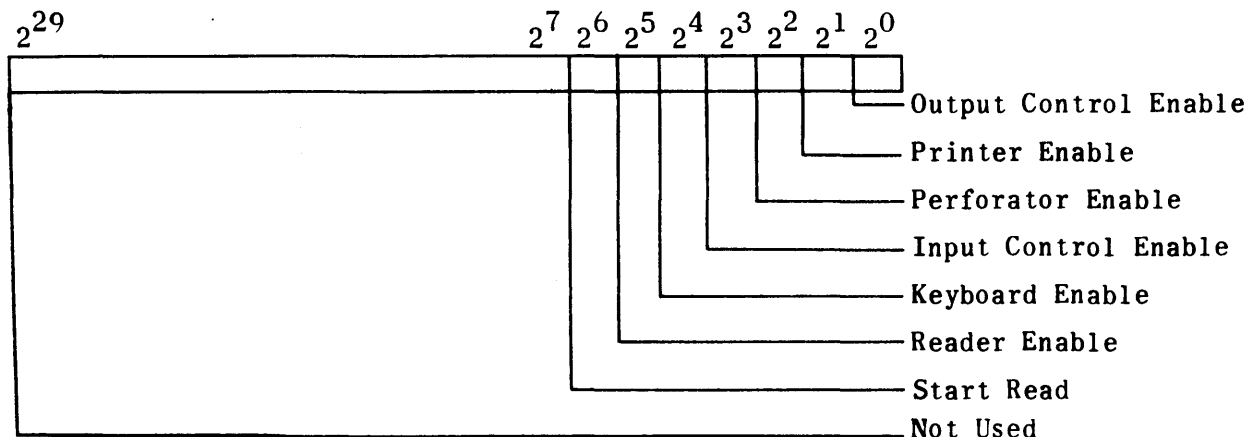
Figure II-B-3. Function Instruction Encoding

3. PROGRAMMING CONSIDERATIONS

3.1 GENERAL

The I/O console is manually placed in the on-line condition when the operator sets the on-line/off-line toggle switch on the console. When it is in the on-line condition, programmed references may begin.

For any operation requested of an output device of the UNIVAC 1532, the output control enable (bit 2^0) must be set along with the device selection bit/bits if it is to perform. To disable operation of an output device, the output control enable bit must be set and the device selection bit cleared. Similarly, for operations requesting functions of an input device, the input control enable (bit 2^3) must be set along with the device selection bit/bits if the device is to perform. To start reading by the paper tape photoelectric reader the start read (bit 2^6) must be set. To disable operation of an input device, the input control enable must be set and the device selection bit cleared.

The console does not respond when it is in the off-line condition. Since no status interrupts are generated by the console when a control function is attempted in the off-line condition, careful consideration should be given to the physical operator functions as well as programming restrictions.

3.2 TAPE READING PROCEDURES

To read a data tape, the operator must position the tape on the reàder. The program procedure for reading a data tape is illustrated in Figure II-B-4. The computer activates the reader by transmitting the start read mode external function code to the I/O console. This function word may be transmitted with force or without force, depending on the type of computer used in the system.
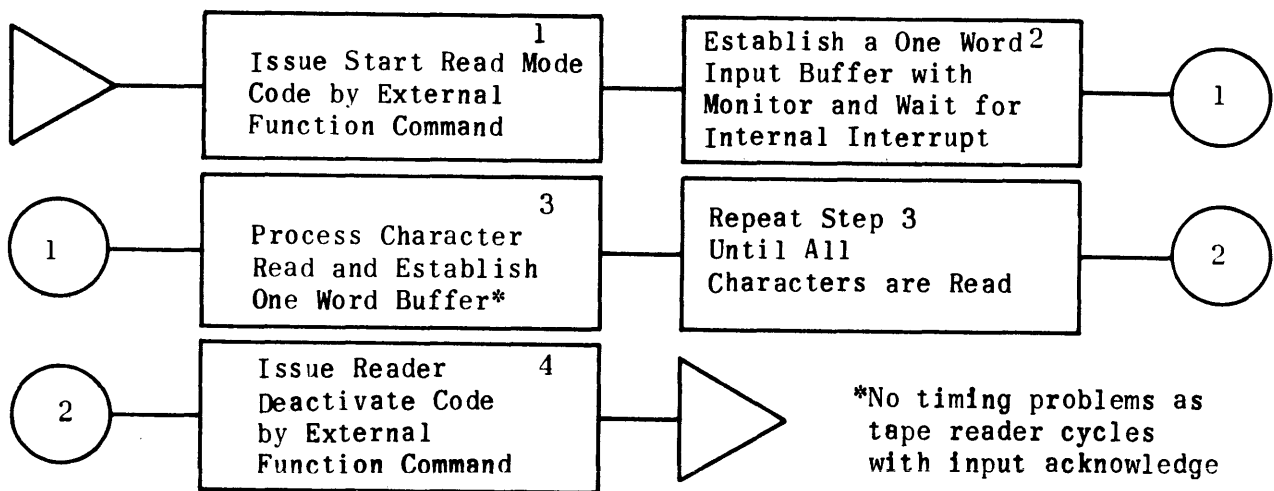
Figure II-B-4. Sequence of Program Operations for Tape Read

After the command has been received by the console, the computer can establish the input buffer mode at its convenience. The reader reads the first frame, sets the input request, and waits for the input acknowledge. The program may transfer one word into the computer, process this word, and re-establish a second buffer without considering time since the console holds each character until the input acknowledge pulse appears.

After the data transmission is complete, the program deactivates the reader by sending a one in bit position 2^3 and zeros in bit positions 2^5 and 2^6 in an external function word.

3.3 KEYBOARD INPUT PROCEDURES

Data or control information may be entered from the keyboard-printer in two different modes.

3.3.1 KEYBOARD-PRINTER ENTRY VIA INTERRUPT

The interrupt mode is enabled by an indicator-switch on the console. When

activated, the next keyed character code is transmitted to the computer with interrupt.

The internal interrupt routine can store the character code, make the proper interpretation and transfer to a desired program that performs the required function or stores the appropriate data word.

3.3.2 KEYBOARD-PRINTER ENTRY VIA COMPUTER COMMANDS

To enter data or control information from the keyboard-printer, the program must previously have activated the keyboard by the keyboard-printer enable mode external function code. The program should establish a one-word input buffer with monitor on the console input channel. As a key is depressed, the buffer completes and the internal interrupt alerts the computer. The keyboard can be disabled by the program by sending a one in bit position 2^3 and a zero in bit position 2^4 in an external function word.

3.4 TAPE PUNCHING PROCEDURES*

To punch a tape, the program must activate the punch by sending the perforator enable mode external function code to the punch. The program then transmits the data to be punched to a buffer storage area and establishes the appropriate length output buffer. The console accepts the data at the convenience of the punch. When the punching operation is complete, the program can deactivate the punch by sending a one in bit position 2^0 and a zero in bit position 2^2 in an external function word.

3.5 PRINTER PROCEDURES*

To operate the printer, the program must activate the printer by sending an input disable code and then the printer enable mode external function code to the printer. The program then transmits the ASCII code data that is to be printed to a buffer storage area and establishes the appropriate length output buffer. The console accepts the data at the printer's convenience. When the printing operation is complete, the program can deactivate the printer by sending a one in bit position 2^0 and a zero in bit position 2^1 in an external function word.

3.6 OFF-LINE OPERATIONS

In off-line operation the console unit has the following capabilities:

1) It prints hard copy, or prints hard copy and punches paper tape from the keyboard.

2) It prints hard copy, punches paper tape, or prints hard copy and punches paper tape from the paper tape reader.

3) It reproduces and permits correcting paper tape, if tape is punched in ASCII code.

---

*No timing considerations need be made in the program since the peripheral unit cycles with the output acknowledge.

SECTION II-C. MAGNETIC TAPE SYSTEM (TYPE 1240A)

1. BASIC INFORMATION

The type 1240A Magnetic Tape System provides a large capacity, medium-speed auxiliary storage area.

The system employs various format selections. They include recording and reading in four moduli, two character types, odd and even parity, and low and high density. In order to provide compatibility with the high-speed printer, the mod 5 format is used with a programmed fixed block length of 128 lines or tape frames to each block of information. One block contains 24 computer words. Mod 6 format is used for compatibility with some non UNIVAC systems. The density selection allows the 1240A tape unit to read or write at 200 frames per inch for low density and at 556 frames per inch for high density. The reading and writing operation is performed at a tape speed of 112.5 inches per second, and the rewind operation is done at a tape speed of 225 inches per second. The block length may vary between 24 computer words and total computer memory. The recording of 1240 tape system is the non-return to zero (change-on-one) technique.

The basic 1240A Magnetic Tape System cabinet (Figure II-C-1) consists of three sections:

1) Magnetic tape control.
2) One tape transport control.
3) Four tape transports.

Auxiliary tape transport controls and tape transports may be added to the system (up to sixteen transports and four tape transport controls, Figure II-C-2).

The magnetic tape control enables the magnetic tape system to communicate with the computer by performing the interface digital-to-digital conversion and performing the logical operations of selecting tape function and tape transport. The tape transport control receives signals from magnetic tape control and performs the logic necessary to control either two or four tape transports. Only one magnetic tape control is necessary in a system and one tape transport control is necessary for each cabinet of transports. Figure II-C-2 shows the 1240A interface of system with sixteen transports, four tape transport controls, and one magnetic tape control.

2. INPUT/OUTPUT SEQUENCE FOR 1240A MAGNETIC TAPE SYSTEM

The I/O sequence for the 1240A tape system and computer begins with the tape system in an idle state and the following events occur:

1) The computer places an address word on the output data lines. (See Figures II-C-3 for configuration of address word.)

2) The computer sets the external function line active.
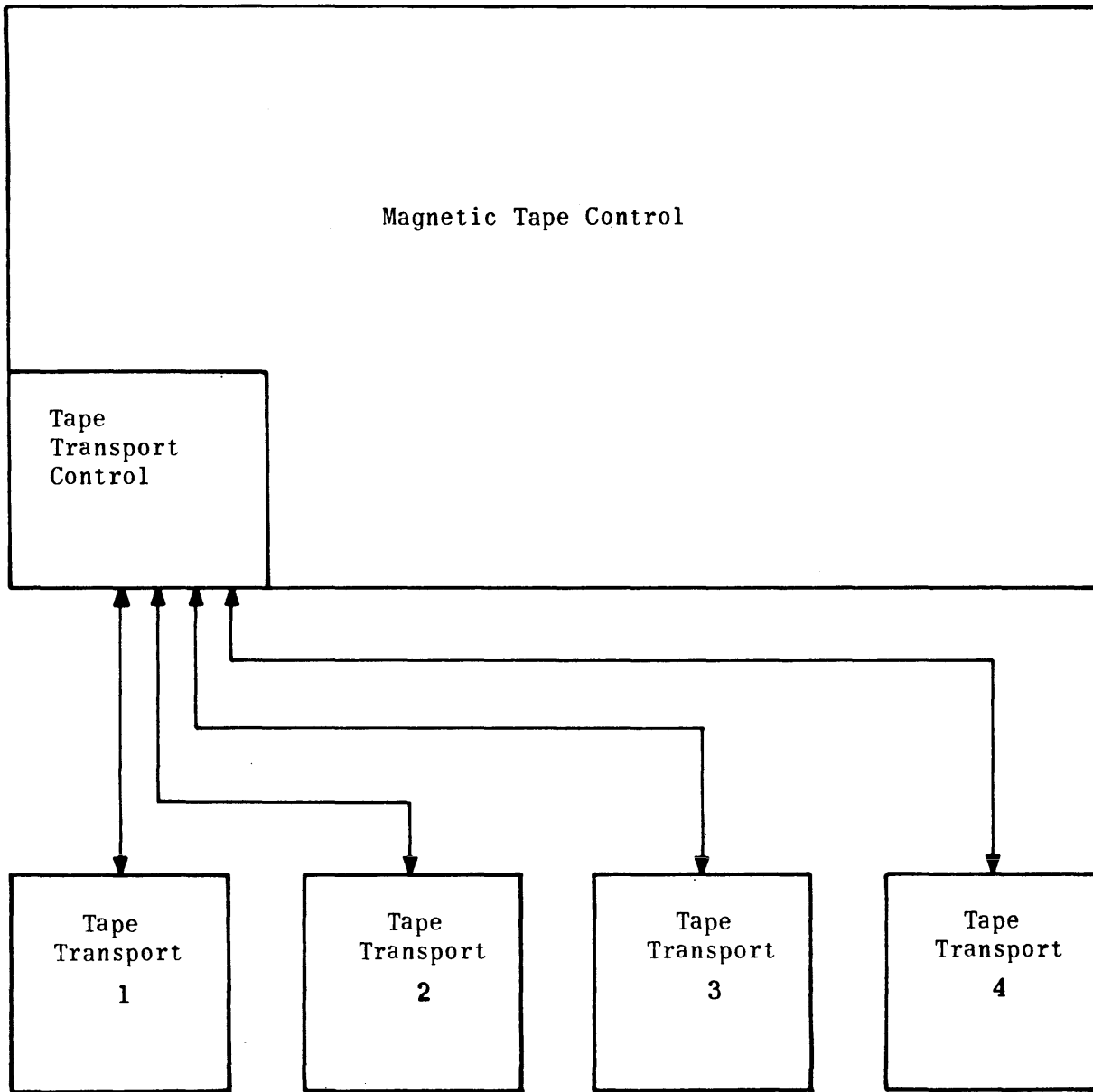
Figure II-C-1. Block Diagram of Magnetic Tape System
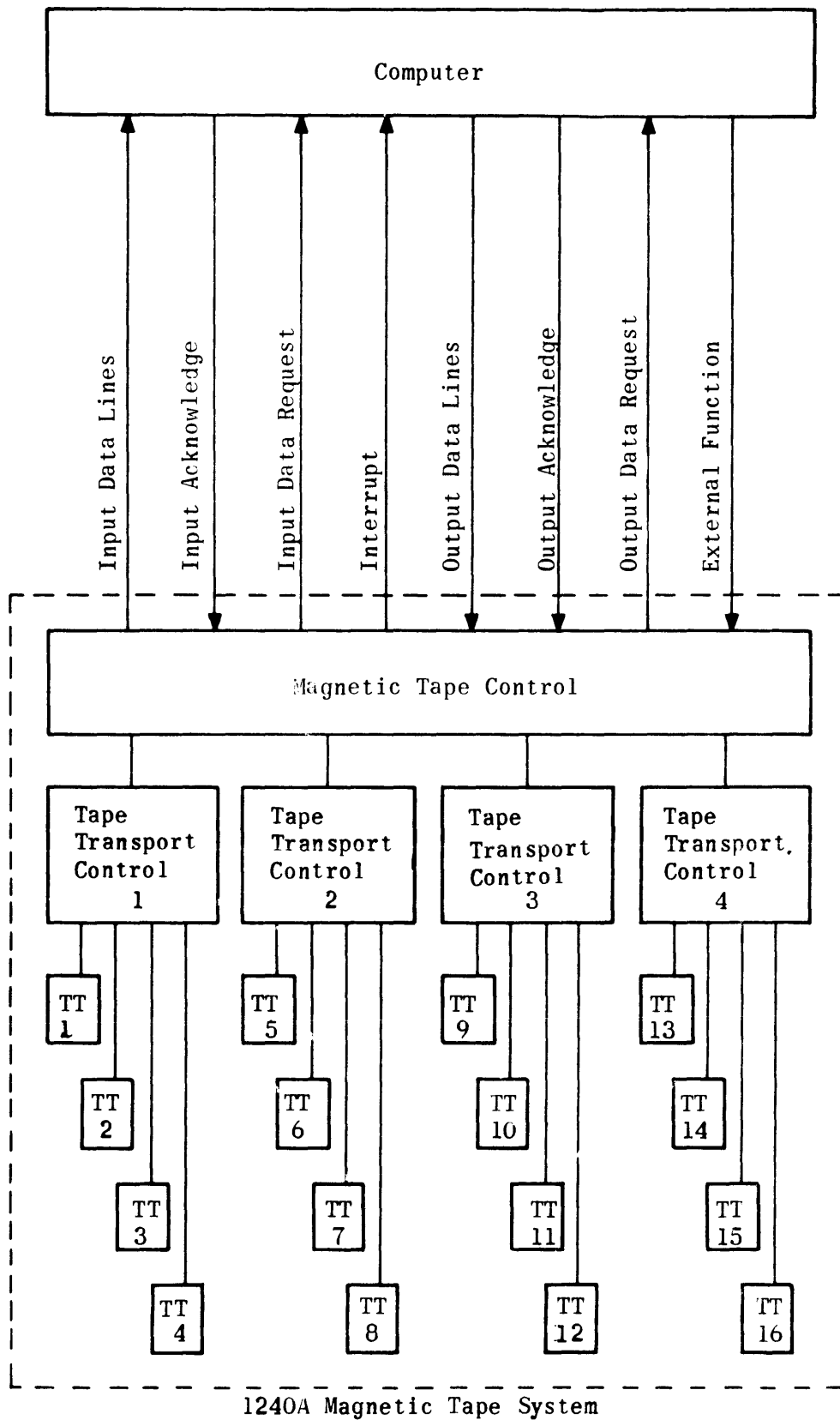
Figure II-C-2. 1240A Interface

3) The magnetic tape control takes the address word and selects the correct cabinet and transport.

4) The computer places an instruction word on the output data lines. (See Figure II-C-4 for configuration of instruction word.)

5) The computer sets the external function line active.

6) The magnetic tape control samples the instruction word, becomes active and performs the operation specified by the instruction word.

7) The magnetic tape control receives a status word from the tape transport control, and places it on the input data lines.

8) The magnetic tape control sets the interrupt line active.

9) The computer accepts the interrupt according to priority.

10) The computer program handles the interrupt and determines the action to be taken using the status word.

11) The magnetic tape system becomes idle.

2.1 ADDRESS WORD

The address word is received by the magnetic tape system via the external command from the computer. Bit 17 is set, and will be as specified in Figure II-C-3. The magnetic tape system operates with the selected cabinet and tape transport for all operations until another address word is received from the computer.

2.2 INSTRUCTION WORD

The instruction word is received by the magnetic tape control via the function command and bit 17 is set to a zero. The instruction word will be as specified in Figure II-C-4.

3. INTERRUPT AND STATUS WORD

A status interrupt is sent to the computer by the magnetic tape control, 222 microseconds following the completion of all functions except the master clear and transport address word operation. Along with the interrupt the magnetic tape control puts a status word on the data lines. This status word is a signal from tape transport control as to the success of an operation performed on a tape transport. The computer acknowledges the interrupt signal and jumps to the interrupt entrance address for that channel. (Address $20 + C_j$). The computer entrance address should contain a RJP (65000 xxxxx) instruction to computer interrupt program. This program determines if the tape operation was successful. (Figure II-C-5 gives status bits that will be set for any errors that may occur.)

30 18	17	16	15 06	05 03	02 00
No Meaning			Not Used	Cabinet Address	Transport Address

- Function Word Designator = 1 (bit 17)
- Master Clear (bit 16)

Cabinet Address:
- 0 = Cabinet 4
- 1 = Cabinet 1
- 2 = Cabinet 2
- 3 = Cabinet 3
- 4 = Cabinet 4
- 5 = Cabinet 1
- 6 = Cabinet 2
- 7 = Cabinet 3

Transport Address:
- 0 = None
- 1 = TT No. 1
- 2 = TT No. 2
- 3 = TT No. 3
- 4 = TT No. 4
- 5 = TT No. 1
- 6 = TT No. 2
- 7 = TT No. 3

Figure II-C-3. Address Word

30 18	17	16	15 11	10 09	08	07	06	05 00
No Meaning			Operation Code Refer to Table II-C-1					Identification Code

- Function Word Designator = 0 (bit 17)
- Master Clear (bit 16)

Identification Code (bits 05–00):
- Selective Read = ID Code
- Write, Tape Mark = 001111
- Write, Tape Mark, -IXRG = 001111

- Density; High = 1/Low = 0 (bit 06)
- Parity; Odd = 1/Even = 0 (bit 07)
- Character; Octal = 1/Bioctal = 0 (bit 08)

Modulus (bits 10–09):
- Mod 3 = 00   Mod 4 = 01
- Mod 5 = 10   Mod 6 = 11

Figure II-C-4. Instruction Word

TABLE II-C-1. OPERATION CODES

Operation Code	Operation
00000	READ
00001	READ, Selective
00010	READ, Ignore Error Halt
00011	Space File
00100	SEARCH Type I
00101	SEARCH Type II
00110	SEARCH-File Type I
00111	SEARCH-File Type II
01000	WRITE
01001	WRITE, XIRG
01010	WRITE, Ignore Error Halt
01011	WRITE XIRG, Ignore Error Halt
01100	WRITE Tape Mark
01101	WRITE Tape, XIRG
10000x*	Backspace
10010	Backspace-Read
10011	Backspace-File
10100	Backsearch Type I
10101	Backsearch Type II
10110	Backsearch File Type I
10111	Backsearch File Type II
110x0*	Rewind
110x1*	Rewind, Clear Write Enable
111x0*	Rewind-Read
111x1*	Rewind-Read, Clear Write Enable

* x may be either "0" or "1"

29	28	27	26	25	24	23	22	21	20	19	18	17	16	15
14	13	12	11	10	09	08	07	06	05	04	03	02	01	00

└Load Point

└Low Tape

└End of Tape

└No Write Enable

└Tape Mark (End of File)

└Last Direction $\begin{matrix} 1 = Backward \\ 0 = Forward \end{matrix}$

└Longitudinal Parity Error

└Lateral Parity Error

└Incorrect Frame Count

└Input Timing Error

└Output Timing Error

└Not Used

└Improper Condition
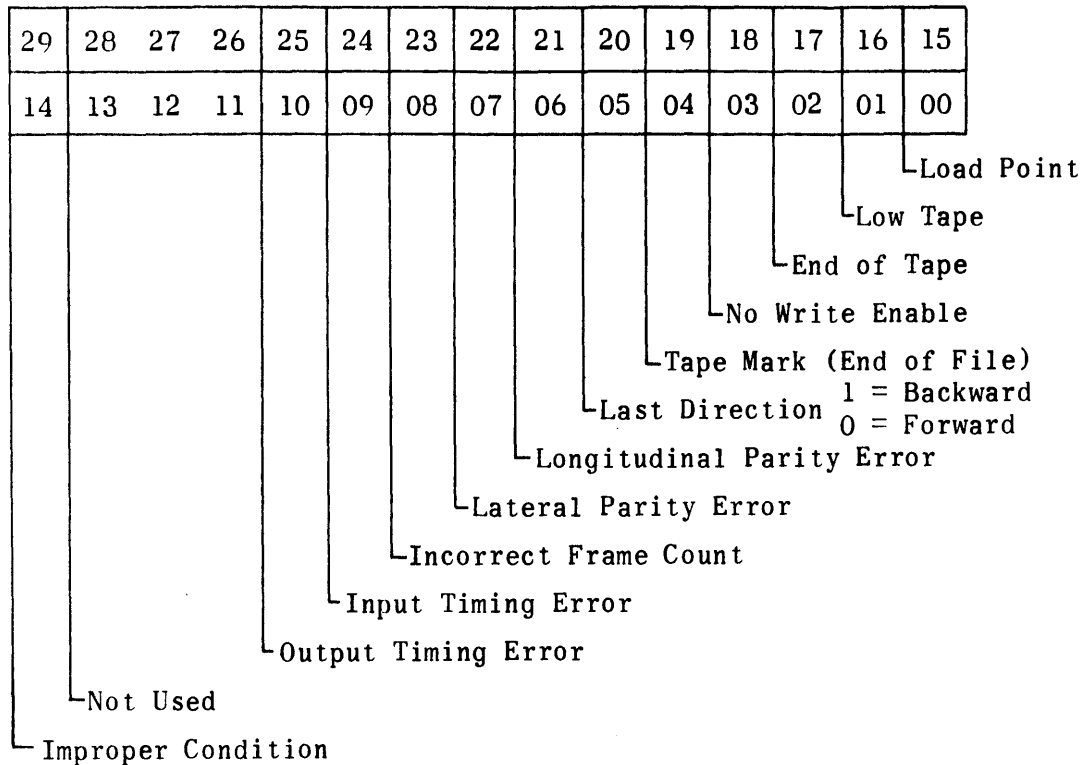
Figure II-C-5. Status Word Format

4. MAGNETIC TAPE OPERATIONS

4.1 MASTER CLEAR (BIT 16)

The magnetic tape control performs a master clear whenever power is applied, when the master clear switch is operated, and whenever the master clear command is received via the external function command from the computer. The master clear performs the following:

1) The master clear is accepted by the magnetic tape control system at any time.

2) The master clear shall not be followed by a status-word interrupt.

3) The master clear stops all tape motion (except a rewind of tape) and places the system in the idle state.

4) The magnetic tape control accepts an external function command anytime after a master clear.

4.2 READ (BITS 11-15)

The read function is supplemented by format, density, and identification code selections. Two types of read operations are performed: normal and selective read.

The read function performs the following:

1) The magnetic tape control, having received the read function, begins passing tape forward over the read head at a speed of 112.5 inches per second.

2) The tape transport control checks the parity of each frame, or seven bits, and passes the information onto magnetic tape control.

3) The magnetic tape control assembles the information into a computer word for transfer to the computer. The number of frames required to make up a computer word will depend on the modulus in which it is written.

4) If the selective read function was selected, the lower six bits of the computer word are compared with the identification code. If a comparison is not correct, the assembled word is disregarded and the next computer word is assembled. Therefore, only words in the record which have the lower six bits equal to the identification code are transferred to the computer. In a normal read all words are transferred to the computer through the input data lines.

5) This process continues until the record has been completely read, assembled, and transferred to the computer.

6) A status word is sent to the computer by magnetic tape control at the completion of the read function, informing the computer of the success of operation.

7) If an error (parity or input timing) is detected during transmission of data, the magnetic tape control ceases to transfer data to the computer for the remainder of that record. At the end of the record a status word is sent to the computer informing it of the nature of failure.

4.3 WRITE (BITS 11-15)

The write function is supplemented by format and density selections. The tape speed for a write function is 112.5 inches per second. The following events will occur:

1) The magnetic tape control takes a word from the computer on the output data lines.

2) The magnetic tape control disassembles the computer word according to modulus selection, generates a parity bit, and transfers the seven-bit frame to the tape transport control for recording on tape.

3) The read-head is activated, causing the tape transport to read back the information recorded, for parity check purposes.

4) If a parity error is detected, the write operation is halted and a status word telling the computer of failure is sent over interrupt lines. The

computer program must then correct the procedure as necessary to per-
form the write function. (Write with extended inter-record gap function
is the suggested correction measure.)

5) If no parity error is detected, the process of disassembling and recording
data continues until the computer no longer acknowledges the output data
request from the magnetic tape control. This means that the complete
buffer has been recorded on tape.

6) When magnetic tape control detects the end of write, tape motion
is stopped after 3/4 inch of tape has passed the write head. This
3/4 inch of tape is for inter-record gap (IRG). Extended inter-record
gap is 3 1/2 inches.

7) A status word is sent to the computer and the tape system becomes idle.

8) In the write ignore halt function, the magnetic tape control does not
stop the write operation if lateral parity error is detected.

4.4 REWIND (BITS 11-15)

The rewind function causes the tape transport selected to rewind tape at a rate
of 225 inches per second. If rewind clear write enable function is selected,
the tape stops at load point and a write function is not accomplished on this
unit. If the tape is located at load point when a rewind function is given, no
improper conditions occur.

4.5 REWIND-READ (BITS 11-15)

The rewind-read function causes the same effect as normal rewind, except when
the tape reaches load point. The first record is read into the computer by nor-
mal read function. The status word is sent after the first record is trans-
ferred to the computer. The rewind-read clear write enable disables the write
enable making further writing on this unit impossible. This function is supple-
mented by format and density selection.

4.6 SPACE FILE FORWARD/BACKWARD (BITS 11-15)

When the magnetic tape control is instructed to space file forward/backward, it
causes the addressed tape transport to move in the specified direction, beyond
the next tape mark. The magnetic tape control notifies the computer via the
status word with a tape mark indication.

4.7 WRITE TAPE MARK (BITS 11-15)

This function causes the magnetic tape control to instruct the tape transport
to write a tape mark, a special record having 17_8 in the first frame, 3 frames
of zeros, and one frame of longitudinal parity.

4.8 BACK SPACE (BITS 11-15)

The back space operation causes the selected tape transport to move one record in the backward direction. The tape is properly positioned in the inter-record gap ready for a read or write function. The parity is checked while the back-space operation is performed and a status word is sent to the computer. The back space function is supplemented by format and density selections.

4.9 SEARCH (BITS 11-15)

The search function combines the normal read with the ability to conduct on the first word of a record (in either the forward or backward direction) and transfer only the find record to the computer. The search comparison is performed on the first word of a record with the identifier (search key) word. The search word is transmitted to the magnetic tape control by the computer in a one-word buffer following the instruction word. The search forward/backward file function performs the same function except the search is limited to a file mark. There are two types of searches which the magnetic tape control can perform in comparing the key word with the first word of the record. Type I is defined as a per bit greater-than-or-equal compare. The following example demonstrates this definition:

        search key: 001101
        find: 011101
        find: 001101
        no find: 010101
        no find: 001100

An identical compare (Type II) is defined as a search comparison with the search key and the first word of the record exactly identical.

5. FORMAT PORTION OF INSTRUCTION WORD

The format portion of the instruction word consists of modulus, character, and parity. A complete format selection must be included in all instruction words which require a record or read operation. The three sections of the format are discussed in the following paragraphs:

5.1 MODULUS

The magnetic tape system is capable of recording and reading in four different moduli. These moduli and the appropriate designator bits (bit 10-09) are:

    1) 00 = Modulus 3.
    2) 01 = Modulus 4.
    3) 10 = Modulus 5.
    4) 11 = Modulus 6.

These are discussed in the section titled Tape System Moduli.

5.2 CHARACTER

There are two types of character recording; octal and bioctal. A one in bit 08 of the instruction word specifies octal. In this type, channels 3, 4, and 5 contain the same information as channels 0, 1, and 2 respectively for each frame, except that when channels 0, 1, and 2 are all zeros, channels 3, 4, and 5 contain all ones. Odd lateral parity is always generated when recording in octal character (see Table II-C-2.). A zero in bit 08 specifies bioctal recording. The octal character allows for redundant recording for added reliability.

TABLE II-C-2. OCTAL RECORDING

Character		Tape Channels
Octal	Binary	6 543 210
0	000	0 111 000
1	001	1 001 001
2	010	1 010 010
3	011	1 011 011
4	100	1 100 100
5	101	1 101 101
6	110	1 110 110
7	111	1 111 111

5.3 PARITY

Two parity modes can be utilized, odd or even, bit 07 is used for parity mode selection. A one in bit 07 specifies odd parity, and in bit 07, a zero specifies even parity.

Data ordinarily is recorded in two formats: binary and binary-coded-decimal. The parity bit is chosen to make the total number of ones (1's) bits in a frame odd in the binary format and even in the binary-coded-decimal format.

5.4 DENSITY

The magnetic tape system is capable of recording data on tape in two different programmable densities. The two densities are low density, at 200 frames per inch, and high density, at 556 frames per inch.

Bit 06 of the instruction word is used for density selection. When bit 06 is a one, high density is selected; when it is a zero, low density is selected.

6. TAPE SYSTEM MODULI

6.1 MODULUS 3: (BITS 10 AND 09 = 00)

Mod 3 is obtained by reducing 18 bits of a computer word to three (bioctal character) frames of data. In reading mod 3, a word is sent to the computer for every three (bioctal character) tape frames. These frames are assembled in the lower 18 bits (17-00) of the data word. The upper bits, if any, of the data word contain zeros.

Recording mod 3, the 18 bits (17-00) of a computer word are recorded in three (bioctal characters) 7-bit frames, consisting of a 6-bit character, plus parity. For octal recording the number of tape frames is doubled. See Figure II-C-6 for bioctal recording and Figure II-C-7 for octal recording of bit arrangements on tape.

6.2 MODULUS 4: (BITS 10 AND 09 = 01)

Mod 4 is obtained by reducing 24 bits of a computer word to four (bioctal characters) frames of data. In reading mod 4, a word is sent to the computer for every four (bioctal character) tape frames. These frames are assembled in the lower 24 bits (23-00) of the data word. The upper bits, if any, of the data word contain zeros.

Recording mod 4, the 24 bits (23-00) of a computer word are recorded in four (bioctal character) 7-bit frames, consisting of a 6-bit character, plus parity. For octal recording the number of tape frames is doubled.

6.3 MODULUS 5: (BITS 10 AND 09 = 10)

Mod 5 is obtained by reducing 30 bits of a computer word to five (bioctal character) frames of data. In reading mod 5, a word is sent to the computer for every five (bioctal character) tape frames. These frames are assembled in the lower 30 bits (29-00) of the data word. The upper bits, if any, of the data word contain zeros.

Recording mod 5, the 30 bits (29-00) of a computer word are recorded in five (bioctal character) 7-bit frames, consisting of a 6-bit character, plus parity. For octal recording the number of tape frames is doubled.

6.4 MODULUS 6: (BITS 10 AND 09 = 11)

Mod 6 is obtained by reducing 36 bits of a computer word to six (bioctal character) frames of data. In reading mod 6, a word is sent to the computer for every six (bioctal character) tape frames. These frames are assembled in the 36 bits (35-00) of the data word.

Recording mod 6, the 36 bits (35-00) of a computer word are recorded in six

Forward Tape
Direction

Oxide Side →

	6	5	4	3	2	1	0	Tape Channel
P	05	04	03	02	01	00		3rd Frame
P	11	10	09	08	07	06		2nd Frame
P	17	16	15	14	13	12		1st Frame

6   5   4   3   2   1   0   Tape Channel

Modulus 3

Oxide Side →

P	05	04	03	02	01	00	4th Frame
P	11	10	09	08	07	06	3rd Frame
P	17	16	15	14	13	12	2nd Frame
P	23	22	21	20	19	18	1st Frame

6   5   4   3   2   1   0   Tape Channel

This Edge
Of Tape
Next To
Transport

Modulus 4

Oxide Side →

P	05	04	03	02	01	00	5th Frame
P	11	10	09	08	07	06	4th Frame
P	17	16	15	14	13	12	3rd Frame
P	23	22	21	20	19	18	2nd Frame
P	29	28	27	26	25	24	1st Frame

6   5   4   3   2   1   0   Tape Channel

Modulus 5

Oxide Side →

P	05	04	03	02	01	00	6th Frame
P	11	10	09	08	07	06	5th Frame
P	17	16	15	14	13	12	4th Frame
P	23	22	21	20	19	18	3rd Frame
P	29	28	27	26	25	24	2nd Frame
P	35	34	33	32	31	30	1st Frame

6   5   4   3   2   1   0   Tape Channel

Modulus 6

Figure II-C-6.   Bioctal Tape Format

Forward Tape
Direction

P	02	01	00	02	01	00	6th Frame
P	05	04	03	05	04	03	5th Frame
P	08	07	06	08	07	06	4th Frame
P	11	10	09	11	10	09	3rd Frame
P	14	13	12	14	13	12	2nd Frame
P	17	16	15	17	16	15	1st Frame
6	5	4	3	2	1	0	Tape Channel

Oxide Side

Modulus 3

P	02	01	00	02	01	00	8th Frame
P	05	04	03	05	04	03	7th Frame
P	08	07	06	08	07	06	6th Frame
P	11	10	09	11	10	09	5th Frame
P	14	13	12	14	13	12	4th Frame
P	17	16	15	17	16	15	3rd Frame
P	20	19	18	20	19	18	2nd Frame
P	23	22	21	23	22	21	1st Frame
6	5	4	3	2	1	0	Tape Channel

Oxide Side

This Edge
Of Tape
Next To
Transport

Modulus 4

P	02	01	00	02	21	00	10th Frame
P	05	04	03	05	04	03	9th Frame
P	08	07	06	08	07	06	8th Frame
P	11	10	09	11	10	09	7th Frame
P	14	13	12	14	13	12	6th Frame
P	17	16	15	17	16	15	5th Frame
P	20	19	18	20	19	18	4th Frame
P	23	22	21	23	22	21	3rd Frame
P	26	25	24	26	25	24	2nd Frame
P	29	28	27	29	28	27	1st Frame
6	5	4	3	2	1	0	Tape Channel

Oxide Side

Modulus 5

P	02	01	00	02	01	00	12th Frame
P	05	04	03	05	04	03	11th Frame
P	08	07	06	08	07	06	10th Frame
P	11	10	09	11	10	09	9th Frame
P	14	13	12	14	13	12	8th Frame
P	17	16	15	17	16	15	7th Frame
P	20	19	18	20	19	18	6th Frame
P	23	22	21	23	22	21	5th Frame
P	26	25	24	26	25	24	4th Frame
P	29	28	27	29	28	27	3rd Frame
P	32	31	30	32	31	30	2nd Frame
P	35	34	33	35	34	33	1st Frame
6	5	4	3	2	1	0	Tape Channel

Oxide Side

Modulus 6

Figure II-C-7. Octal Tape Format

(bioctal character) 7-bit frames, consisting of a 6-bit character, plus parity.
For octal recording the number of tape frames is doubled.

7. STATUS WORD

A status interrupt is sent to the computer 222 microseconds following the
completion of every function except master clear and transport address selection.
A status word is placed on the data lines of the input cable. The bit
structure of the status word enables the computer to determine whether or
not the previous function was successfully completed.

The computer program must recognize that after issuing an external function
instruction to the magnetic tape system, no subsequent external function
command (except addressing and master clear) will be recognized until receipt
of the acknowledge to the status interrupt, signifying the end of the first
instruction.

Figure II-C-5 shows bit assignments in the status word. These conditions
are described below.

7.1 IMPROPER CONDITION (BITS 29 AND 14)

A one in bits 29 and 14 may imply that operator intervention is necessary to
overcome the difficulty. An improper condition will occur whenever:

1) Reference tape transport is not in automatic condition.

2) No tape transport is selected when one is required.

3) A forward command is sent to a tape transport whose tape is positioned
   at end of tape.

4) A reverse command is sent to a tape transport whose tape is positioned
   at load point (except a rewind operation).

5) A write instruction is issued to a tape transport that has no write
   enable.

When the computer has been notified of an improper condition, the computer
program may then refrain from issuing further external function commands to
the tape system to allow visual inspection of the trouble, or it may issue
another external function command. An incoming external function command to the
tape system causes the improper condition indicator to extinguish.

A tape transport not in automatic condition implies one of the following
situations:

1) Tape transport was manually removed from automatic.

2) Tape transport not in ready condition for one of the following reasons:

   a) Power off.

b) Tape broken.
c) Lamp burnout.
d) Tape load was not accomplished when tape was mounted.

7.2   OUTPUT TIMING ERROR (BITS 25 AND 10)

A one in bits 25 and 10 indicates that the computer did not acknowledge the
first output data request, or the computer acknowledged the output data
request too late (however, it did acknowledge the output data request for the
data word to be written in its proper place). The acknowledge time is related
to format and density.

Also an output timing error can occur during search operations if the magnetic
tape system does not receive the search key before the start of reading the
record. This time requirement may be as short as two milliseconds.

7.3   INPUT TIMING ERROR (BITS 24 AND 09)

A one in bits 24 and 09 indicates that magnetic tape control information on
the input cable was not accepted by the computer before the subsequent word
was to be placed on the input cable. This condition indicates that the
computer lost one or more words of the last record. If an input timing
error occurs, data transmission to the computer ceases for the remainder of
the record.

7.4   INCORRECT FRAME COUNT (BITS 23 AND 08)

A one in bits 23 and 08 indicates either some frames were lost, or improper
modulus specified (that is; there were not enough frames in the record to
complete an integral number of computer words). This situation may result from
one or more of the following:

1) One or more characters were not properly read or recorded.
2) Bad spots on the tape caused characters to be lost.
3) Reading a record with the wrong format (for example, reading mod 4
    with a tape record in mod 5).

A longitudinal parity error usually occurs in conjunction with an incorrect
frame count if frames were lost.

7.5   LATERAL PARITY ERROR (BITS 22 AND 07)

A one in bits 22 and 07 informs the computer that the lateral parity of one or
more frames read did not agree with that specified in the format.

7.6   LAST TAPE MOTION (BITS 20 AND 05)

A one in bits 20 and 05 indicates that the last tape motion was backward.
A zero indicates that the last tape motion was forward.

7.7 LONGITUDINAL PARITY ERROR (BITS 21 AND 06)

When recording, longitudinal parity is generated by magnetic tape control for each channel and recorded after the last frame of the record. When reading (read, back space, post-write check) the longitudinal parity of a record is checked by magnetic tape control, and if in error, noted in the status word (bits 21 and 06).

7.8 TAPE MARK (BITS 19 AND 04)

A one in bits 19 and 04 indicates that the magnetic tape control has sensed a tape mark during a read, write, (before a search comparison is made during a search file instruction) or back space function.

7.9 NO WRITE ENABLE (BITS 18 AND 03)

A one in bits 18 and 03 informs the computer that the referenced tape transport has no write enable when a write operation is attempted or that the write enable ring is not inserted in the tape reel.

7.10 END OF TAPE (BITS 17 AND 02)

To prevent reading or writing off the end of the tape, an end of tape reflective marker is placed a minimum of 14 feet from the physical end of the tape.

When the end of tape mark is sensed, a 1/2 second time-out begins. When this time period is completed, no further forward movement of the tape will be possible. However, the tape may be moved in the reverse direction past the reflective marker and then moved forward. When the marker is again sensed, the time-out is initiated again, and the forward tape motion will halt after 1/2 second.

7.11 LOW TAPE (BITS 16 AND 01)

A one in bits 16 and 01 informs the computer that the tape is positioned less than 100 feet from the end of tape.

7.12 LOAD POINT (BITS 15 AND 00)

Since the first several feet of tape undergo excessive wear and are required to load the transport, no recording is done on this protion of the tape. Recording begins at load point and this point is recognized by the magnetic tape system by means of a reflective marker placed at least ten feet from the physical beginning of tape. The write, load point, delay allows infor- mation to be written on the tape approximately 3/4 inch beyond the load point marker with the tape moving in the forward direction.

8. TAPE MARKERS

The load point and end of tape markers are pressure-sensitive, adhesive-coated strips of aluminum 1 by 3/16 inch. The markers are detected by reflective photo-sensing means. Both markers are placed on the base (uncoated) side

of the tape with the 1-inch dimension parallel to the tape. The load point
marker is placed 1/32 inch from track 0, or the front edge of the tape.
The end of tape marker is placed 1/32 inch from track 6 or inside edge of
the tape.

9. LOGICAL SELECTION OF TAPE TRANSPORTS

Selector switches to change the logical address of each tape transport are
provided. Any physical tape transport may be switched to any logical address.
If logical address selections are duplicated, the lowest order physical tape
transport has priority; however, no two cabinets may have the same logical
address. This will be the responsibility of the operator.

10. 1240A HIGH-SPEED PRINTER OFF-LINE COMPATIBILITY

The magnetic tape subsystem is capable of communicating directly with the
high-speed printer subsystem for off-line operation. The interface between
the magnetic tape unit and printer is shown in Figure II-C-8.
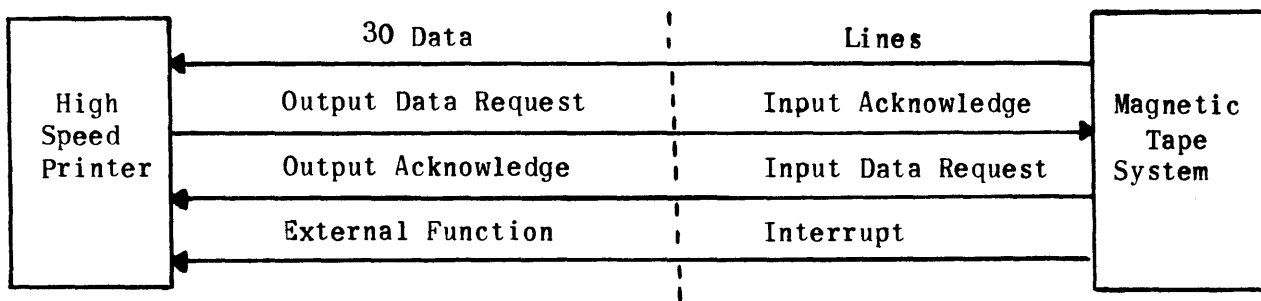
Figure II-C-8. Magnetic Tape - High-Speed Printer Interface

1) The printer output is connected to the magnetic tape system's input
    (input and output as used here are in reference to the computer).

2) The data to be printed off-line must be recorded in records on tape in
    the following format: Record length of 120 Field data characters.

3) The magnetic tape system reads each record of data from tape in the
    following format: Modulus 5.

4) At the magnetic tape unit function register, the operator manually
    selects the character, parity, and density.

5) Each record of 120 characters forms 24 30-bit data words which is
    printed as one line by the high-speed printer.

6) A tape mark is recognized by the high-speed printer as a top of form
    command. This positions the paper to the top of the next page.

7) A record of less than 24 words (preferably one computer word, five
    characters) causes the high-speed printer to stop the printing operation.
    This record is not printed if the characters are space codes (05).

8) With the magnetic tape system switched to the printer mode, the desired tape transport selected and the tape positioned at load point, the high-speed printer initiates operation when it is placed in the on-line position.

The normal sequence of events for transfer of data from the magnetic tape system to the printer is as follows: (See Figure II-C-9).

1) The printer sets its output data request (e).

2) The magnetic tape system, in the idle state (a) recognizes this first output data request from the printer as an external function and starts the read operation.

3) The magnetic tape system places the information on the data lines and sets its input data request (c).

4) The printer recognizes the magnetic tape system input data request as an output acknowledge (f).

5) The printer samples the data lines and clears its output data request.

6) The magnetic tape system recognizes the clearing of the printer output data request as an input acknowledge (b).

Steps 3, 4, 5, and 6 are repeated for each word of the record.

The normal sequence for sending an external function command top of form from the magnetic tape system to the printer is the same as reading a record except that when the magnetic tape system detects the tape mark, it will set bit 4 in the status word, and when the interrupt (d) is set, the printer will recognize this as an external function command top of form (g) (Figure II-C-9).

11. PROGRAMMING CONSIDERATIONS

11.1 GENERAL

The magnetic tape system is manually placed in an operational condition. The operator functions include mounting tapes on transports, turning power on, and initially positioning tapes. With the magnetic tape system operational, programmed references may begin. Generally, all programming of the tape system must be done with force and must conform to a standard sequence of reference. (Figure II-C-10 illustrates this sequence.)

Once the tape system receives and starts to execute the operations in an instruction word, further external function commands, other than master clear, are ignored. The programmer must remember when an external function command may be logically issued. After issuing an external function command other than address word and master clear, the computer may not logically issue another function command until the computer acknowledges the receipt of an interrupt from the tape system.

Step 1 of Figure II-C-10 is required only on the initial reference of a tape transport or when a reference to another transport is desired.
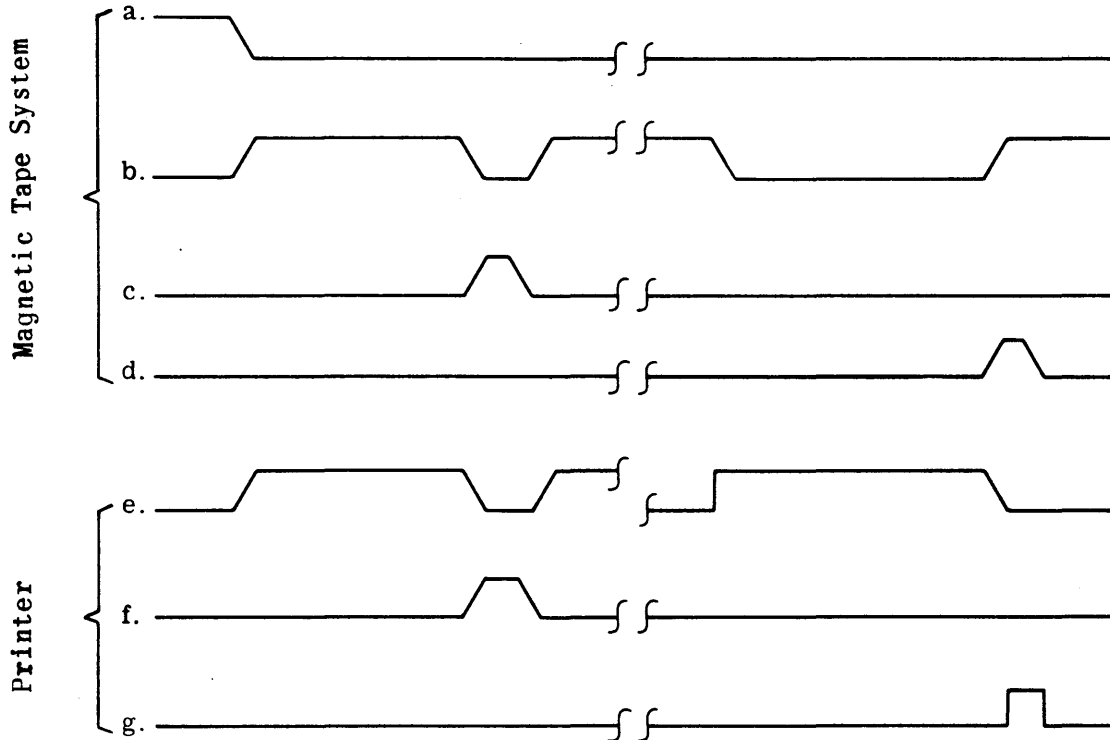
Figure II-C-9.   Sequence of Events in Tape-Printer Operation

11.2 WRITE PROCEDURES

To write, the instruction word must include complete format selection (modulus, character, and parity), and density. Use of the procedure outlined in Figure II-C-10 will result in a record of words boing written on tape. Length of record is determined when the output buffer is initiated.

After the status interrupt is received, signifying end of a write operation, the program must check the following four conditions to determine successful completion of the write operation:

   1) No improper condition in the status word.
   2) No output timing error in the status word.
   3) No lateral parity error in the status word.
   4) Output buffer is terminated.

```
                               ▽


            ┌─────────────────────────┐
            │ Issue Address Word      │
  Step 1    │ By The External         │
            │ Function Command        │
            └─────────────────────────┘


            ┌─────────────────────────┐
            │ Form Instruction        │
            │ Word, Include Operation  │
  Step 2    │ Code, Format, And       │
            │ Density As Applicable   │
            └─────────────────────────┘


            ┌─────────────────────────┐
            │ Initiate Required       │
  Step 3    │ Output And/Or Input     │
            │ Buffers                 │
            └─────────────────────────┘


            ┌─────────────────────────┐
            │ Issue Instruction       │
            │ Word By The External    │
  Step 4    │ Function Command.       │
            │ Indicate Tape           │
            │ System Busy             │
            └─────────────────────────┘


            ┌─────────────────────────┐
            │ When Tape System        │
            │ Interrupt Occurs,       │
            │ Execute Interrupt       │
            │ Subroutine, Which       │
  Step 5    │ Takes  The Status       │
            │ Word Off Input          │
            │ Cable.  Indicate        │
            │ Tape System Is          │
            │ Not Busy                │
            └─────────────────────────┘


                               ▽
```

Figure II-C-10. Sequence of Programming References - Magnetic Tape System

If the status word indicates an output timing error, the computer did not acknowledge the first output data request, or the computer acknowledged the output data request too late (however, it did acknowledge the request) for the data word to be written in its proper place. The acknowledge time is related to format and density.

It is possible for an output timing error to occur that will not be shown in the status word. Such a condition results if the computer did not acknowledge the output data request (other than the first output data request from the tape system). The system recognizes this situation as end of record and, consequently, indicates no error. However, if words are left unwritten in the output buffer, this constitutes an output timing error condition.

If a lateral parity error is indicated in the status word, the write operation was terminated when the post write check detected an incorrect frame. It is the responsibility of the program to decide and provide the recovery procedures.

11.3 READ PROCEDURES

To read, the instruction word must include complete format selection, identification code (if read selective) and density. Use of the procedure outlined in Figure II-C-10 will result in a record being read from tape. The length of the input buffer must be long enough to cover the record to be read.

An input timing or parity error will terminate the data input to the computer unless the read operation is with the ignore error halt option. The status interrupt will still be sent by the magnetic tape system at the end of that record.

11.4 SEARCH PROCEDURES

To search, the instruction word must include complete format selection and density. The identifier word will be rec ived by the magnetic tape system via a one-word output buffer. Tape motion is started upon the receipt of the instruction word and an output timing error will occur if the magnetic tape system does not receive the identifier within the time from start of tape motion and when the compare is made. This time requirement may be as short as two milliseconds. One record will then have been passed. The search is terminated by an output timing error and a parity error.

When searching backwards, the 30-bit identifier word sent by the computer must be reversed; characterwise, its configuration is dependent upon format. Examples are given below:

	Bioctal	Octal
Original Computer Word	3456745321	3456745321
Mod 5 Identifier	2153745634	1235476543
Mod 4 Identifier	0053745634	0035476543
Mod 3 Identifier	0000745634	0000476543
Mod 6 Identifier	5374563400	3547654300

11.5 RECORD LENGTH

There are no limits on record length within physical tape capacity. When reading tapes of unknown record length, the input buffers must be made sufficiently large to insure reading the entire record. Another method is to initiate an input buffer with monitor and make provision for the initiation of additional buffers to read the complete record.

11.6 END OF FILE

The normal end-of-file inter-record gap is approximately 3/4 inch long followed by a tape mark (001 111) and its associate check character. The end of file is always recorded with even parity.

11.7 EDITING OF TAPE

By suitable programming, an inter-record gap of any length may be written to precede any record. Records may be rewritten for tape updating, and they may be lengthened, provided suitable inter-record gap was used on a previous recording. A record may be inserted for a previously used extended inter-record gap.

11.8 BAD TAPE

When writing a record, if a tape bad spot is encountered where recording is marginal or impossible, the tape may be back spaced to the beginning of the record and rewritten with an extended inter-record gap. The long inter-record gap will probably be sufficient to move the bad spot past the recording head. Successive extended inter-record gaps may be written if the bad spot still appears.

SECTION II-D. MAGNETIC TAPE SYSTEM (TYPE 1540/1541)

1. GENERAL INFORMATION

The militarized UNIVAC® 1540/1541 Magnetic Tape Units provide large capacity auxiliary storage devices for computing systems operating under severe environmental conditions.

The UNIVAC 1540 Magnetic Tape Unit employs a pinch-roller type of tape transport and the 1541 Unit is equipped with a single capstan vacuum loop transport. The units are functionally identical; however, the 1541 has a higher data transfer rate. The UNIVAC 1540/1541 Magnetic Tape Units may be operated on-line under complete computer program control as an I/O storage device or with a high-speed printer for off-line printing of tape recorded information. A flexible format allows recording and reading of four moduli (18-, 24-, 30-, or 36-bit computer words) and three densities, and provides recording of magnetic tapes which is compatible in all respects with industry-accepted tape systems. Either even or odd frame parity may be utilized and for added reliability, the redundant octal format is provided. A read-after-write feature checks each frame for parity immediately after recording. Longitudinal parity recording and checking are automatic. A duplexing capability is provided so that two computers communicating with the same magnetic tape unit may share its facilities under program control. In this way, data or programs stored on one tape are available to both computers. Further savings in the facilities are enhanced by the ability of the 1540/1541 Magnetic Tape Units to read information in either the forward or backward motion of the tape.

Records of data may be of variable lengths and are separated by 3/4-inch interrecord gaps (IRG) unless otherwise extended by suitable programming. Records may be lengthened if suitable interrecord gaps were provided in previous recordings.

Either the UNIVAC 1540 or the 1541 Magnetic Tape Unit is compatible with all UNIVAC military computers. Compatibility both in tape format and computer programs with the UNIVAC® 1240 Magnetic Tape Unit is provided through a manual switch selection on the 1540 or 1541 basic unit. This feature gives the functional characteristics that are identical with those of the 1240 subsystem and permits the use of software packages designed for earlier systems.

2. PERFORMANCE OF FUNCTION

Either the 1540 or the 1541 Magnetic Tape Unit communicates with the computer in the request-acknowledge mode (see Figure II-D-1). The computer issues commands to the magnetic tape unit by means of the external function signal and function words. When the magnetic tape control inspects the function word, it selects the specified tape transport and performs the specified operation. Each tape transport cabinet contains a 16-position address switch for each tape transport so that each can be assigned a logical number (1 through 16). UNIVAC 1540/1541 programs use positions 1 through 8 only. The additional positions (9 through 16) are used to provide compatibility with UNIVAC 1240 programs. The address selection bits of the instruction word are interpreted by the

magnetic tape control according to the physical position of these switches. No
two transports may be assigned the same logical number (that is; to allow iden-
tical function on both units). If duplication does exist, priority is allocated
to the transport at the most remote position to the left of the basic unit (the
top transport having priority over the bottom unit in the 1540) regressing
toward the basic unit, then those connected to the right side beginning at the
remote position. The operator must determine the logical addresses required by
each program and set the switches accordingly. (See Figures II-D-2 and II-D-3
for order of priority scheme.)

The instruction word contains the code for one of six basic operations -
duplex selection, read, search, write, space file, and rewind - or a combination
of two basic operations. To accept an external function command, the magnetic
tape unit must be in the ready state (that is; operable but not performing
a specific operation). The completion of an operation or a master clear
places the magnetic tape unit in the ready state.

The general sequence of events for on-line operation with a computer (the
magnetic tape control in automatic mode and in the ready state) is as follows:

1) Computer issues an instruction word via the external function command.

2) Magnetic tape control samples the instruction word and becomes busy.

3) Magnetic tape control selects the addressed tape transport.

4) Operations stated in the instruction word are initiated and carried
   to completion.

5) Magnetic tape control sets a status word on the input lines as described
   in subsequent paragraphs.

6) Magnetic tape control interrupts the computer with external interrupt
   signal after completion of the operation.

7) Magnetic tape control issues stop command to transport.

8) Computer samples status word and acknowledges interrupt whereby the
   magnetic tape unit becomes idle.

Steps 7 and 8 may be interchanged, or may occur simultaneously.

Status words and input data are transferred on the input lines with identifying
signals on the external interrupt line and the input request line, respectively.
The computer acknowledges receipt of these transfers via the input acknowledge
line.

Function words and output data are transferred on the output lines with iden-
tifying signals on the external function line and the output acknowledge lines,
respectively. The output request line notifies the computer of the ability
of the magnetic tape unit to accept output data.
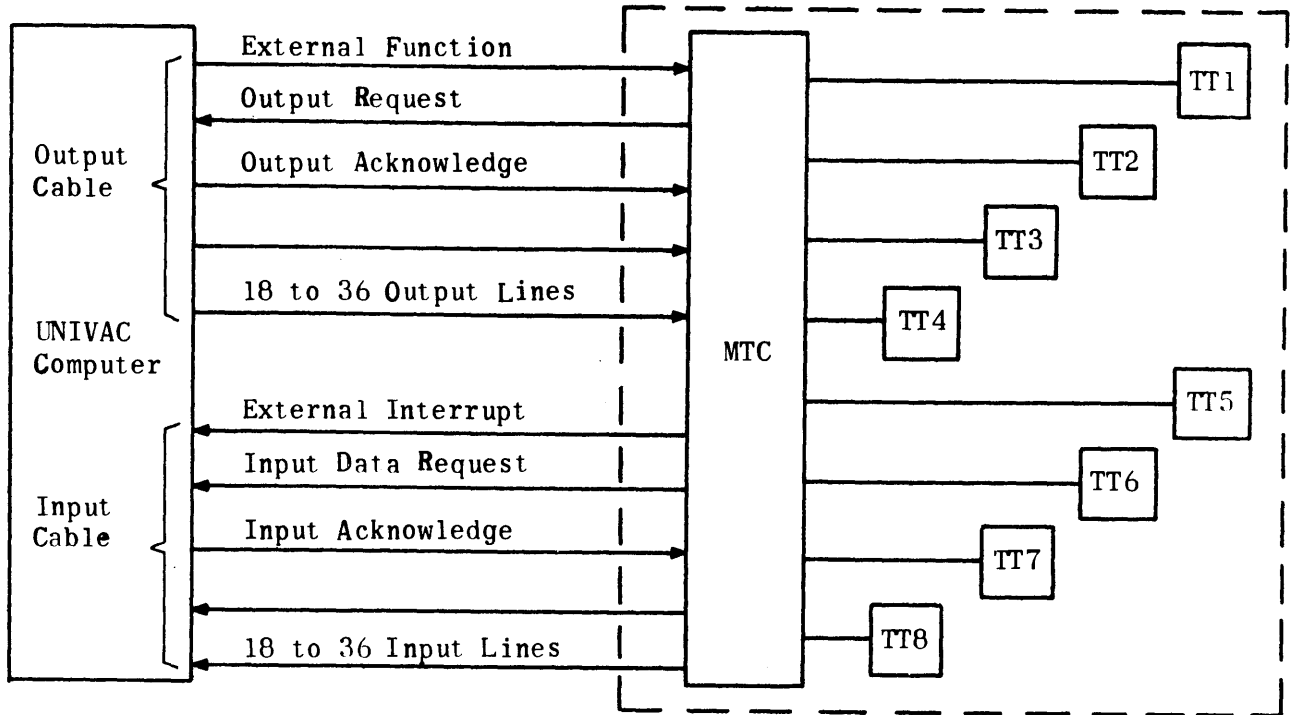
Figure II-D-1. Magnetic Tape Unit - Computer Interface

3. DUPLEXING

Either of two computers with compatible interface can exercise control over
the magnetic tape unit when the duplexing capability is utilized. The fol-
lowing duplex control functions are provided via the instruction word:

1) Demand duplex control (master clear).
2) Request duplex control.
3) Release duplex control.

To complete the communication, the status word with interrupt provides duplex
control status to the computers as follows:

1) Not in duplex control.
2) In duplex control.

Through these messages, each computer has control of the switching functions of
the duplexer and each is informed of the operational status of the magnetic
tape units it shares with the other.

4. TAPE MARKERS

The load point and end of tape markers are adhesive-coated strips of aluminum
one inch by 3/16 inch, placed on the base (uncoated) side of the tape with
the one-inch dimension parallel to the tape edge; see Figure II-D-4 for

Basic Magnetic
Tape Unit

Add-On Unit

Order of
Priority →

Add-On Unit	Basic Magnetic Tape Unit		Add-On Unit	Add-On Unit
1	3		7	5
2	4		8	6
2 Tape Handlers	2 Tape Handlers	Magnetic Tape Control	2 Tape Handlers	2 Tape Handlers
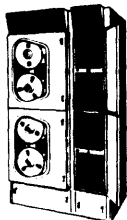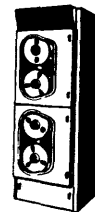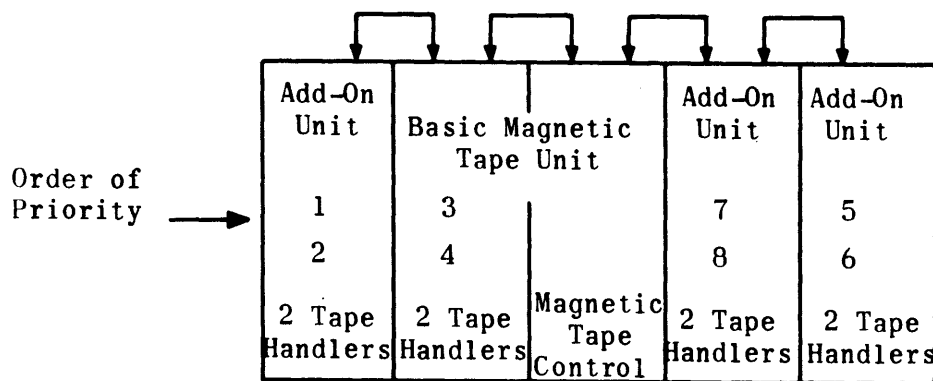
Figure II-D-2. Type 1540 Magnetic Tape System
(Maximum Configuration)

Basic Magnetic
Tape Unit

Add-On Unit

Add-On Unit	Add-On Unit	Add-On Unit	Basic Magnetic Tape Unit		Add-On Unit	Add-On Unit	Add-On Unit	Add-On Unit
1 Tape Handler	1 Tape Handler	1 Tape Handler	1 Tape Handler	Magnetic Tape Control	1 Tape Handler	1 Tape Handler	1 Tape Handler	1 Tape Handler
1	2	3	4 ←	Order of → 8 Priority	7	6	5	
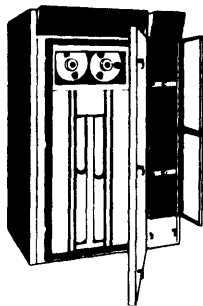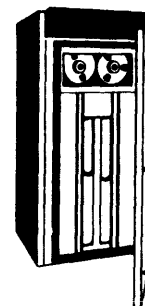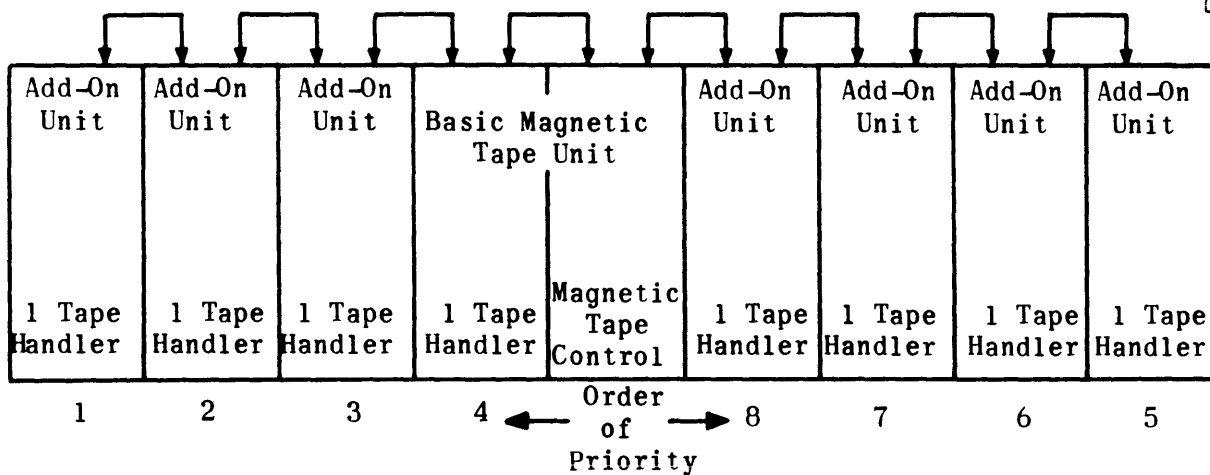
Figure II-D-3. Type 1541 Magnetic Tape System
(Maximum Configuration)

tape format. The load point marker is placed 1/32 inch from track 0 or out-
side edge of the tape and at least ten feet from the beginning of the tape.
The end of tape marker is placed 1/32 inch from track 6 or inside of the tape
and at least 15 feet from the end of the tape. The markers are detected by
reflective photoelectric sensors.

5. STATUS WORD AND INTERRUPT (STATUS INTERRUPT)

The computer program is interrupted after completion of every operation per-
formed by the magnetic tape control, except master clear and transport address
selection. The magnetic tape control places a status word on the channel
input lines and a signal on the channel external interrupt line. The bit
structure of the status word (see Figure II-D-5) enables the computer program
to determine the status of the magnetic tape unit and whether or not the
requested operation was completed successfully. Errors encountered during a
requested operation, as well as the physical status of the magnetic tape unit,
are indicated in the status word. The term, status interrupt, is used to express
this philosophy since the computer program is interrupted and the status of the
magnetic tape unit and the encountered errors are designated in the status word.
Any such interrupt sent to the computer must be acknowledged by the computer
before another external function with an instruction word is recognized by the
magnetic tape control. Successful completion of an operation contains no
error indications, but other indications of tape status may be present.

The status word requires a word of at least 15 bits. If the computer accepts
words larger than 15 bits, the information in the next higher order bits
beginning at bit 15 is not interpreted. A detailed explanation of each bit
of the status word follows.

Z = Longitudinal parity
P = Lateral parity
X = Data bit

Load point marker
3/16" x 1" on non-
oxide side of tape

Forward direction oxide down

Figure II-D-4. Tape Format

35 ← Not Interpreted → 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

1 = Improper Condition

1 = No
0 = Yes } - Duplex Control

1 = Transport Ready

1 = XIRG Detected

1 = Output Timing Error

1 = Input Timing Error

1 = Incorrect Frame Count

1 = Lateral Parity Error

1 = Longitudinal Parity Error

1 = Backward
0 = Forward } - Last motion of tape

1 = Tape Mark (End of File)

1 = No Write Enable

1 = End of Tape

1 = Low Tape

1 = Load Point

Figure II-D-5. Magnetic Tape Unit - Status Word Format

5.1 IMPROPER CONDITION (BIT 14 = 1)

An improper condition occurs whenever:

1) Selected tape transport is not in automatic condition. A tape transport not in automatic condition implies one of the following situations:

   a) Tape transport was manually removed from automatic.
   b) Tape transport not in ready condition for one of the following reasons:

      1. Power off.
      2. Tape broken.
      3. Lamp burnout.
      4. Tape load was not accomplished when tape was mounted.

      (This situation also causes the transport ready bit in the status word to be cleared.)

2) No tape transport is selected when one is required.

3) A forward command is sent to a tape transport whose tape is positioned at end of tape.

4) A reverse command other than a rewind operation is sent to a tape transport whose tape is positioned at load point.

5) A write instruction is issued to a tape transport that has no write enable. (This situation also causes the no write enable bit in the status word to be set.) After the computer is notified of an improper condition, the computer program may then refrain from issuing further external function commands to the tape system to allow visual inspection of the trouble and operator intervention to overcome the difficulty, or it may issue another external function command. An incoming external function command to the tape system clears the improper condition indication.

5.2 DUPLEX CONTROL (BIT 13; 0 = IN CONTROL; 1 = NOT IN CONTROL)

1) The status of the duplexer is indicated by bit 13 of the status word and is sent to one of the two computers, depending on the action initiated.

2) The condition not in control (bit 13 = 1) is sent to the issuing computer when a word is transmitted by that computer while the duplexer is not in the proper position.

3) The condition not in control (bit 13 = 1) is sent to the nonissuing computer when that computer loses control as a result of a demand duplex control issued by the other computer.

4) The condition in control (bit 13 = 1) is sent to the issuing computer when a request duplex control is issued and the duplexer is transferred to the control of that computer.

5.3 TRANSPORT READY (BIT 12 = 1)

The transport ready bit indicates that the last-addressed tape transport is in
a ready condition as follows:

1) Power is on.
2) Magnetic tape reel is mounted and tape is properly loaded.
3) Tape marker detector lamp is operating.

5.4 XIRG DETECTED (BIT 11 = 1)

The XIRG detected bit indicates that an extended interrecord gap (3 1/2 inches
between records) was sensed during tape read. Tape movement continues until
the next record is read.

5.5 OUTPUT TIMING ERROR (BIT 10 = 1)

If the computer issues a write instruction to the magnetic tape control and
does not transfer the first output data word, or transfers a requested data
word too late to be written in its proper place and before the interrupt is
sent to the computer following end of record, an output timing error occurs.
This word transfer time is related to format and density as shown in
Table II-D-1. An output timing error can occur during search or selective
read operations if the magnetic tape control does not receive a search key
or selective read code before assembling the first word. The time require-
ment may be as short as 2 1/2 milliseconds from the time the instruction
word is received by the magnetic tape control until the search key, selective
read code, or the first data word must be received.

5.6 INPUT TIMING ERROR (BIT 09 = 1)

If the computer issues a read instruction and fails to accept a word placed
on the input cable by the magnetic tape control before the next word is to
be placed on the input cable, an input timing error occurs. This error
indicates that the computer lost one or more words of the last record since
data transmission to the computer ceases for the remainder of the record.
The tape continues to move to the end of record, at which time the magnetic
tape control sends the status word indicating the error with an interrupt
to the computer.

5.7 INCORRECT FRAME COUNT (BIT 08 = 1)

An improper modulus specified or some frames lost causes an incorrect frame
count error. This may be caused by one or more of the following:

1) There were not enough frames in the record to complete an integral
   number of computer words.

2) One or more characters were not properly read or recorded.

TABLE II-D-1. WORD ASSEMBLY TIME (MICROSECONDS)

Format		UNIVAC 1540			UNIVAC 1541		
Modulus	Character	200 fpi	556 fpi	800 fpi*	200 fpi	556 fpi	800 fpi*
3	Bioctal	125	45	31.2	100	36	25
4	Bioctal	167	60	41.6	133.3	48	33.3
5	Bioctal	208	75	52.0	166.6	59.9	41.6
6	Bioctal	250	90	62.4	200	72	50
3	Octal	250	90	62.4	200	72	50
4	Octal	334	120	83.2	266.6	96	66.6
5	Octal	416	150	104.0	333.2	119.8	83.2
6	Octal	500	180	124.8	400	144	100

*Refer to individual computer technical description brochures for interface compatibility definition.

   3) Bad spots on the tape caused characters to be lost.

   4) Reading the record with the wrong format (for example, reading mod 4 with a tape record in mod 5).

Longitudinal parity error can be expected with incorrect frame count error except during reading with the wrong modulus.

5.8 LATERAL PARITY ERROR (BIT 07 = 1)

During a writing process, a parity bit is added to each six-bit character according to a format specified and the seven bits are recorded as one frame. If the magnetic tape control detects a frame whose lateral parity does not agree with that specified by the format, during any read type operation or during the post-write check of the recording operation, a lateral parity error occurs.

5.9 LONGITUDINAL PARITY ERROR (BIT 06 = 1)

During a writing process, a longitudinal even parity bit is generated by the magnetic tape control for each tape channel and recorded after the last frame of the record. If the magnetic tape control detects an error in this parity during any read type operation or during the post-write check of the recording operation, a longitudinal parity error occurs. If a frame count error ever occurs, the longitudinal parity error usually occurs. Both would be indicated in the status word.

5.10 LAST TAPE MOTION (BIT 05; 1 = BACKWARD, 0 = FORWARD)

Any status word with interrupt sent to the computer at the completion of an
operation indicates the direction of the last tape motion. The program can
determine whether the tape is positioned at the beginning or the end of
the record.

5.11 TAPE MARK (BIT 04 = 1)

A recorded tape mark (refer to write tape mark) separates files of information
on the tape. Any read, space file, search file or back read operation that is
limited to a file and the post-write check of the write tape mark operation
indicates a tape mark in the status word.

5.12 NO WRITE ENABLE (BIT 03 = 1)

When a write operation is attempted on a selected transport that has its
write enable cleared or the write enable ring is not inserted in the tape
reel, the no write enable is indicated in the status word.

5.13 END OF TAPE (BIT 02 = 1)

When the end of tape reflective marker is sensed by the magnetic tape unit,
a 1/2 second time-out begins, after which no forward movement of tape is
possible. Reverse direction tape motion past the tape marker is possible.
The end of tape indication appears in the status word. If forward tape motion
is reinitiated, the marker is sensed again and after the 1/2 second time-out,
the forward tape motion is stopped.

5.14 LOW TAPE (BIT 01 = 1)

A tape supply detector senses less than 100 feet of tape remaining on the
selected transport reel. The magnetic tape control indicates a low tape any
time a status word is sent to the computer with the tape positioned within
100 feet of end of tape.

5.15 LOAD POINT (BIT 00 = 1)

Recording on a tape begins at load point (a reflective tape marker placed at
least ten feet from the physical beginning of the tape). The write-load
point delay allows for a gap of at least 1/2 inch beyond the load point marker
(in the forward direction) before the first record may be written. The mag-
netic tape control indicates load point in the status word whenever an opera-
tion requesting backward motion of tape is attempted with the selected
tape positioned at load point.

6. EXTERNAL FUNCTION COMMANDS - FUNCTION WORDS

Operations and tape selections are requested by function words being sent to
the magnetic tape unit with an external function from the computer. A master
clear of the magnetic tape unit is performed when a demand duplex control

command is sensed by the magnetic tape control. It differs from the other operations in these three respects:

1) It may be performed at any time, even when magnetic tape unit is busy.

2) It has priority over all other operations in the instruction word (see Figure II-D-6).

3) It does not result in a status interrupt to the issuing computer.

The master clear stops all tape motion (except a rewinding tape) and sets the magnetic tape unit in the ready state. At any time after a master clear, the magnetic tape control accepts another external function. Since this function is not considered a normal operation, its use should be restricted to times when the magnetic tape unit is believed to be in an illogical state or when its state cannot be determined. The master clear does not clear the write enable which is set manually. To clear the write enable, a form of clear write enable instruction must be used.

Individual operations are performed by the magnetic tape unit under direction of an instruction word. When the computer output word is transmitted with an external function signal, it is sensed at the magnetic tape control as a command. The operation to be performed, format and density, if required, and the transport selection address or reading bits are defined in the instruction word. The format for the instruction word is shown in Figure II-D-6. The individual tape transport is selected by bits 05 through 00 of the instruction word. The magnetic tape unit consists of a maximum of eight tape transports, each of which must be assigned a logical number by the operator on the transport selection switch provided for each tape handler.

6.1 FORMAT (BITS 10 - 7)

The format portion of the instruction word contains modulus, character, and parity designators. A complete format selection must be included in all instruction words which request a reading or recording operation with the exception that modulus may be ignored in the write tape mark instruction. The modulus designator and the character designator direct the magnetic tape control in the assembly and disassembly of computer words from or to tape frames.

6.2 CHARACTER DESIGNATOR (BIT 8); 1 SELECTS OCTAL, 0 SELECTS BIOCTAL

Bioctal or octal (redundant) format is specified in operations requiring reading or writing. The bioctal format disassembles 18-, 24-, 30-, or 36-bit computer words into 3, 4, 5, or 6 six-bit-plus-parity tape frames, respectively during recording (vice versa for reading). (See Figure II-D-7.) The octal format disassembles 18-, 24-, 30-, or 36-bit computer words into 6, 8, 10, or 12 tape frames, respectively, during recording (vice versa for reading). Tape channels 3, 4, and 5 contain the same information as channels 0, 1, and 2, respectively, in each frame, except when channels 0, 1, and 2 contain zeros, channels 3, 4, and 5 contain ones. Odd parity is selected by the magnetic tape control when writing or reading octal characters. The redundant recording in octal format adds to the reliability (see Figure II-D-8). For compatible tapes, data must be recorded in bioctal format.

Transmit Extra

| 35 ———————— 18 | 17 16 | 15 14 13 12 11 | 10 9 | 8 | 7 | 6 5 | 4 3 2 1 0 |

Not Used   **Operation Code**   Modulus   Character   Parity   **Addressing**   **Switch* Position**

	bit 6
Duplex Control Code	
Nonduplex	0 0 0
Release Control	0 1 0
Release Request	1 0 0
Demand Control (Master Clear)	1 1 0
Transmit Extra Code	1 0 1

	Modulus
Mod 3	0 0
Mod 4	0 1
Mod 5	1 0
Mod 6	1 1

	Character
Bioctal	0
Octal	1

	Parity
Even	0
Odd	1

200 Frames per Inch	0 0
556 Frames per Inch	1 0
800 Frames per Inch	0 1
Same as Last Instruction	1 1

Addressing **   Switch* Position

4	3	2	1	0	Position
0	1	⊕	0	1	1
0	1	⊕	1	0	2
0	1	⊕	1	1	3
0	1	1	0	0	4
1	0	⊕	0	1	5
1	0	⊕	1	0	6
1	0	⊕	1	1	7
1	0	1	0	0	8
1	1	⊕	0	1	9
1	1	⊕	1	0	10
1	1	⊕	1	1	11
1	1	1	0	0	12
0	0	⊕	0	1	13
0	0	⊕	1	0	14
0	0	⊕	1	1	15
0	0	1	0	0	16

Bias

0	0	0	0	0	Normal Bias
0	1	0	0	0	Force Bias
1	0	0	0	0	Force High Bias
1	1	0	0	0	Force Low Bias

* Indicates position of address switch on each tape transport. Allowance is made for 16 tape transports to retain program conpatibility with UNIVAC 1240 Magnetic Tape Subsystems. Programs written for use on the UNIVAC 1540/1541 Magnetic Tape Subsystems address only tape transport positions 1 through 8.

** ⊕ = zero or one.

Figure II-D-6. External Function Word Format

Forward Tape
Direction

	P	05	04	03	02	01	00	3rd Frame
Oxide | P | 11 | 10 | 09 | 08 | 07 | 06 | 2nd Frame
Side | P | 17 | 16 | 15 | 14 | 13 | 12 | 1st Frame

| 6 | 5 | 4 | 3 | 2 | 1 | 0 | Tape Channel |

Modulus 3

	P	05	04	03	02	01	00	4th Frame
	P	11	10	09	08	07	06	3rd Frame
Oxide | P | 17 | 16 | 15 | 14 | 13 | 12 | 2nd Frame
Side | P | 23 | 22 | 21 | 20 | 19 | 18 | 1st Frame

| 6 | 5 | 4 | 3 | 2 | 1 | 0 | Tape Channel |

Modulus 4

	P	05	04	03	02	01	00	5th Frame
	P	11	10	09	08	07	06	4th Frame
	P	17	16	15	14	13	12	3rd Frame
Oxide | P | 23 | 22 | 21 | 20 | 19 | 18 | 2nd Frame
Side | P | 29 | 28 | 27 | 26 | 25 | 24 | 1st Frame

| 6 | 5 | 4 | 3 | 2 | 1 | 0 | Tape Channel |

Modulus 5

This edge
of tape
next to
transport

	P	05	04	03	02	01	00	6th Frame
	P	11	10	09	08	07	06	5th Frame
	P	17	16	15	14	13	12	4th Frame
	P	23	22	21	20	19	18	3rd Frame
Oxide | P | 29 | 28 | 27 | 26 | 25 | 24 | 2nd Frame
Side | P | 35 | 34 | 33 | 32 | 31 | 30 | 1st Frame

| 6 | 5 | 4 | 3 | 2 | 1 | 0 | Tape Channel |

Modulus 6

Figure II-D-7. Bioctal Tape Format

Forward Tape
Direction

	P	02	01	00	02	01	00	6th Frame
Oxide	P	05	04	03	05	04	03	5th Frame
Side	P	08	07	06	08	07	06	4th Frame
	P	11	10	09	11	10	09	3rd Frame
	P	14	13	12	14	13	12	2nd Frame
	P	17	16	15	17	16	15	1st Frame

| | 6 | 5 | 4 | 3 | 2 | 1 | 0 | Tape Channel |

Modulus 3

	P	02	01	00	02	01	00	8th Frame
	P	05	04	03	05	04	03	7th Frame
	P	08	07	06	08	07	06	6th Frame
	P	11	10	09	11	10	09	5th Frame
	P	14	13	12	14	13	12	4th Frame
Oxide	P	17	16	15	17	16	15	3rd Frame
Side	P	20	19	18	20	19	18	2nd Frame
	P	23	22	21	23	22	21	1st Frame

| | 6 | 5 | 4 | 3 | 2 | 1 | 0 | Tape Channel |

Modulus 4

This edge
of tape
next to
transport

	P	02	01	00	02	01	00	10th Frame
	P	05	04	03	05	04	03	9th Frame
	P	08	07	06	08	07.	06	8th Frame
	P	11	10	09	11	10	09	7th Frame
	P	14	13	12	14	13	12	6th Frame
	P	17	16	15	17	16	15	5th Frame
	P	20	19	18	20	19	18	4th Frame
Oxide	P	23	22	21	23	22	21	3rd Frame
Side	P	26	25	24	26	25	24	2nd Frame
	P	29	28	27	29	28	27	1st Frame

| | 6 | 5 | 4 | 3 | 2 | 1 | 0 | Tape Channel |

Modulus 5

Forward Tape
Direction

	P	02	01	00	02	01	00	12th Frame
	P	05	04	03	05	04	03	11th Frame
	P	08	07	06	08	07	06	10th Frame
	P	11	10	09	11	10	09	9th Frame
	P	14	13	12	14	13	12	8th Frame
	P	17	16	15	17	16	15	7th Frame
	P	20	19	18	20	19	18	6th Frame
Oxide	P	23	22	21	23	22	21	5th Frame
Side	P	26	25	24	26	25	24	4th Frame
	P	29	28	27	29	28	27	3rd Frame
	P	32	31	30	32	31	30	2nd Frame
	P	35	34	33	35	34	33	1st Frame

| | 6 | 5 | 4 | 3 | 2 | 1 | 0 | Tape Channel |

Modulus 6

Figure II-D-8. Octal Tape Format

6.3 MODULUS

The modulus specifies the length of the computer word to be recorded on tape or read from the tape (Refer to Table II-D-2).

6.3.1 MODULUS 3 (DESIGNATOR BITS 10 AND 09 = 00)

An 18-bit computer word is disassembled and recorded as three tape frames of bioctal character format or six tape frames of octal character format. If a computer delivers a word larger than 18 bits for recording, the magnetic tape control records the lower order 18 bits of the word on the tape and discards the remaining high order bits. During mod 3 reading operations, three tape frames are assembled as an 18-bit computer word for bioctal character format, or six tape frames are assembled as an 18-bit computer word in octal character format. If the computer word size is larger than 18 bits, the frames are assembled in the lower order 18 bits and zeros are placed in remaining high order bits (see Figures II-D-7 and II-D-8).

6.3.2 MODULUS 4 (DESIGNATOR BITS 10 AND 09 = 01)

A 24-bit computer word is disassembled and recorded as four tape frames of bioctal character format or eight tape frames of octal character format. If a computer delivers a word larger than 24 bits for recording, the magnetic tape control records the lower order 24 bits of the word on the tape and discards the remaining high order bits. During mod 4 reading operations, four tape frames are assembled as a 24-bit computer word for bioctal character format or eight tape frames are assembled as a 24-bit computer word for octal character format. If the computer word size is larger than 24 bits, the frames are assembled in the lower order 24 bits and zeros are placed in remaining high order bits (see Figures II-D-7 and II-D-8).

6.3.3 MODULUS 5 (DESIGNATOR BITS 10 and 09 = 10)

A 30-bit computer word is disassembled and recorded as five tape frames of bioctal character format or ten tape frames of octal character format. If a computer delivers a word larger than 30 bits for recording, the magnetic tape control records the lower-order 30 bits of the word on the tape and discards the remaining high order bits. During mod 5 reading operations, five tape tape frames are assembled as a 30-bit computer word for bioctal character format or ten tape frames are assembled as a 30-bit computer word for octal character format. If the computer word size is larger than 30 bits, the frames are assembled in the lower order 30 bits and zeros are placed in remaining high order bits (see Figures II-D-7 and II-D-8).

6.3.4 MODULUS 6 (DESIGNATOR BITS 10 AND 09 = 11)

A 36-bit computer word is disassembled and recorded as six tape frames of bioctal character format or twelve tape frames of octal character format. During mod 6 reading operations, six tape frames are assembled as a 36-bit computer word for bioctal character format or twelve tape frames are assembled as a 36-bit computer word for octal character format (see Figures II-D-7 and II-D-8).

TABLE II-D-2. CHART SHOWING THE EFFECTS OF VARIOUS UNIVAC COMPUTERS
OPERATING WITH THE UNIVAC 1540 OR 1541 MAGNETIC TAPE SUBSYSTEMS

NOTE: LSB = least significant bits; MSB = most significant bits

Computer	Mod 3 BCD Write	Mod 4 BCD Write	Mod 5 BCD Write	Mod 6 BCD Write
UNIVAC 1218 and 1219 single channel (one 18-bit word is output/request)	For each 18-bit word received, 1540 or 1541 writes 3 frames on tape. Mod 3 recommended for single channel operation	For each 18-bit word received, 1540 or 1541 writes 4 frames on tape: 1 frame of zeros followed by 3 data frames	For each 18-bit word received, 1540 or 1541 writes 5 frames on tape: 2 frames of zeros followed by 3 data frames	For each 18-bit word received, 1540 or 1541 writes 6 frames on tape: 3 frames of zeros followed by 3 data frames
UNIVAC 1218 and 1219 dual channel (one 36-bit word is output/request)	For each 36-bit word received, 1540 or 1541 writes 3 frames on tape. Since only 18 LSB are written, 18 MSB are lost	For each 36-bit word received, 1540 or 1541 writes 4 frames on tape. Since only 24 LSB are written, 12 MSB are lost	For each 36-bit word received, 1540 or 1541 writes 5 frames on tape. Since only 30 LSB are written, 6 MSB are lost. Mod 5 commonly used when preparing tapes for 30-bit computers (CP-642B, UNIVAC 1206 or 1230)	For each 36-bit word received, 1540 or 1541 writes 6 frames on tape. Mod 6 recommended for dual channel operation
UNIVAC 1206, 1212 CP-642B, or 1230 (one 30-bit word is output/request)	For each 30-bit word received, 1540 or 1541 writes 3 frames on tape. Since only 18 LSB are written, 12 MSB are lost	For each 30-bit word received, 1540 or 1541 writes 4 frames on tape. Since only 24 LSB are written, 6 MSB are lost	For each 30-bit word received, 1540 or 1541 writes 5 frames on tape. Mod 5 recommended for operation with CP-642B, 1206, or 1230 computers	For each 30-bit word received, 1540 or 1541 writes 6 frames on tape. 1 frame of zeros followed by 5 data frames. Mod 6 is commonly used when preparing tapes for 36-bit computers

6.4 PARITY DESIGNATOR (BIT 7), 1 SELECTS ODD; 0 SELECTS EVEN

Either odd (the total number of ones in a frame is odd) or even (the total number of ones in a frame is even) lateral parity may be specified in the instruction word for bioctal character writing and reading operations; however, odd parity is selected by the magnetic tape control for the octal character writing and reading operations. For compatible tapes, odd parity is chosen for binary data codes and even parity is chosen for binary coded decimal (BCD) data.

6.5 DENSITY DESIGNATOR (BITS 6 AND 5)

00 selects 200 fpi; 10 selects 556 fpi; 01 selects 800 fpi; and 11 selects same density as last instruction.

At low density, data is recorded at 200 frames per inch, at medium density 555.5 frames per inch, and at high density 800 frames per inch. Density must be specified in instruction words requesting reading or writing operations. Refer to Table II-D-1 for word assembly and disassembly time.

6.6 OPERATION CODE

The operation code is located in bits 15 through 11 of the instruction word. Legal operation codes exist for the six basic operations and for combinations of these operations. The six basic operations are duplex selection, read, search, write, space file, and rewind. Operation codes using any basic operation (except rewind) must be supplemented by format and density codes placed in bits 10 through 07 and bits 06 and 05 respectively, of the instruction word. Table II-D-3 is a listing of the operation codes and Figure II-D-6 shows the structure of the entire instruction word.

6.6.1 READ OPERATIONS

The selected transport moves tape at 120 inches per second (by the 1540) or 150 inches per second (by the 1541) in either direction and transfers seven bit frames (read from tape) to the magnetic tape control. Parity, even or odd, as specified in the format is checked for each frame of the record. The six data bits are assembled into 18-, 24-, 30-, or 36-bit computer words according to the modulus and character designator of the format. The assembled computer word is placed on the data lines of the computer input cable and the input request (IR) line is set. The computer samples the data lines at its convenience and sets the input acknowledge line to the magnetic tape control. The tape continues to move and new words are being assembled until the end of record (interrecord gap) is reached. The computer must sample the input lines and acknowledge each IR within a specified time (governed by density, character, and modulus - Table II-D-1) to prevent loss of one or more words in the record. If the computer fails to sample the input lines and acknowledge the IR within the allotted time during any type of read operation, an input timing error occurs and the magnetic tape control ceases to transfer data to the computer for the remainder of the record. Following the detection of the end of record, the magnetic tape control sets an input timing error status word on the input lines and interrupts the computer program by

TABLE II-D-3. OPERATION CODES

Operation Code	Operation
00000	Read (read forward)
00001	Read: selective (selective read-forward)
00010	Read: modified stop
00011	Space file
00100	Search type I
00101	Search type II
00110	Search file type I
00111	Search file type II
01000	Write
01001	Write XIRG
01010	Write ignore error halt
01011	Write XIRG; ignore error halt
01100	Write modified stop
01101	Write edit
01110	Write tape mark
01111	Write tape mark
10000	Backread (read backward)
10001	Backread selective (selective read-backward)
10010	Backread modified stop
10011	Backspace file
10100	Backsearch type I
10101	Backsearch type II
10110	Backsearch file type I
10111	Backsearch file type II
11000	Rewind
11001	Rewind, clear write enable
11010	Rewind
11011	Rewind, clear write enable
11100	Rewind-read
11101	Rewind-read, clear write enable
11110	Rewind-read
11111	Request transport-status

Bits 17, 16, and 6 = 1, 0, and 1, respectively, transmit extra

setting the external interrupt (EI) line on that channel. When the computer acknowledges the interrupt, the magnetic tape unit becomes ready.

In all types of read operations, format and density selections must be made in each instruction.

6.6.1.1 READ-FORWARD

The selected transport reads one record according to the format stated and checks each frame for parity. If a parity error is detected, the magnetic tape control continues to transfer data to the computer for the remainder of that record. After sensing the end of record, the magnetic tape control sends a status word to the computer with a signal on the external interrupt line. This status word contains magnetic tape unit status and any or all error indications encountered during the reading of the record.

6.6.1.2 READ-BACKWARD

The selected transport reads one record backward to the next interrecord gap (back one record) according to format and density stated in the instruction word. Lateral and longitudinal parity are checked while reading. If an error is detected during backward motion, the reading operation continues and the status word, upon detection of interrecord gap, contains the magnetic tape unit status and the error indication. Characters are assembled in each computer word in the same position as in a forward read. Computer words, however, are transmitted in reverse order.

6.6.1.3 READ-MODIFIED STOP

The selected transport reads one record according to the format stated and checks each frame for parity. At the completion of the read, the magnetic tape is stopped farther in the IRG. The status word sent to the computer after detection of end of record contains magnetic tape unit status and any or all error indications including parity error.

6.6.1.4 SELECTIVE READ-FORWARD/BACKWARD

The selected transport reads one record according to the format stated and checks each frame for parity. Words are read and assembled as in a read-forward/ backward. A selective read code contained in the least significant six bits (05 through 00) of a computer word, is sent by a single word output buffer to the tape unit before it can read and assemble one word. Should this word be formed before the selective read code is transmitted, an output timing error is detected. The magnetic tape control compares the least significant six bits (05 through 00) of each assembled computer word with the selective read code. If the comparison is negative, the word is discarded and reading continues. A positive comparison causes the word to be transmitted to the computer. However, if a parity error is detected in either case, further transfer of data ceases for the remainder of the record. The status word sent to the computer after detection of end of record contains magnetic tape unit status and any or all error indications.

6.6.2 WRITE OPERATION - GENERAL INFORMATION

When the magnetic tape control senses a write function, the selected transport moves the tape forward and records the interrecord gap. A signal is placed on the computer output request (OR) line. The computer, at its convenience, responds with a word on the data lines and places a signal on the channel output acknowledge (OA) line. The magnetic tape control recognizes the OA, samples the data lines and removes the output request. The word is transferred to the disassembly register and another output request is issued. The magnetic tape control dissembles each word according to the modulus selected in the write function word, generates frame parity, and transfers the seven bits to the transport for recording on tape according to the density selected. As the recording frame passes over the read head, it is checked for parity. If a parity error is detected, the magnetic tape control stops the write operation and the tape motion. A status word, indicating an error in recording, is placed on the channel input lines with a signal on the external interrupt line. If no error occurs during recording, the process continues until the computer no longer acknowledges the output request within the time allotted for another word to be disassembled and written. This time is dependent on format and density (refer to Table II-D-1). When the computer does not respond within the allotted tine, the end of write is assumed by the magnetic tape control. Longitudinal parity is written and the recording process is terminated. Tape motion is stopped after a portion of the interrecord gap is written on the tape. The magnetic tape control removes the output request and places a status word, indicating successful conpletion of the write, on the input lines and sets the external interrupt line. When the computer acknowledges the interrupt, the magnetic tape subsystem becomes idle. If the computer acknowledges the output request after the allotted time but before the interrupt is sent, the magnetic tape control interprets the action as an output timing error and notifies the computer in the status word.

NOTE: The normal interrecord gap is approximately 3/4 inch in length and the extended interrecord gap is approximately 3 1/2 inches in length.

6.6.2.1 WRITE

The selected transport writes on the tape according to the format and density stated in the function word. If no recording error is detected, the normal operation continues until the computer no longer transfers data, at which time longitudinal parity is written and the status word with interrupt is sent to the computer.

6.6.2.2 WRITE-IGNORE ERROR HALT

The selected transport writes on the tape according to the format and density stated in the function word, but the magnetic tape control does not stop the writing process if lateral parity errors are detected as the recorded frames pass over the read head. The status word sent to the computer with interrupt after completion contains magnetic tape unit status and any or all errors encountered.

6.6.2.3 WRITE-EXTENDED INTERRECORD GAP (XIRG)

The selected transport records an extended interrecord gap of 3 1/2 inches instead of the normal 3/4 inch IRG preceding a normal write portion of the operation. If no data is transferred from the computer for recording, the extended interrecord gap is present on the tape and an output timing error occurs. The status word sent to the computer after completion contains magnetic tape unit status and any or all error indications detected as in a normal write operation.

NOTE: There is a special length interrecord gap. Under program control, interrecord gaps other than the fixed 3/4 inch and 3 1/2 inch lengths may be written. Successive 3/4 or 3 1/2 inch gaps may be written by issuing the appropriate write functions without initiating output buffers at the computer. The program must be prepared to handle the output timing error that is indicated in the interrupt status word following each write operation performed in this manner.

6.6.2.4 WRITE TAPE MARK

The selected transport writes a fixed format tape mark. The tape mark is a special record having ones in only the 0-, 1-, 2-, and 3-bit positions of the first frame, followed by three frames of zeros and one frame of longitudinal parity. The entire record is written by the magnetic tape control upon receiving the instruction word. Format selection can be ignored. To be compatible with other tape systems, the tape mark must be exactly as specified above. A status word with interrupt is sent to the computer after completion of the write tape mark operation.

6.6.3 SPACE FILE-FORWARD/BACKWARD

The selected transport moves the tape in the selected direction to the IRG beyond the next tape mark and indicates tape mark in the status word. The tape is positioned in the IRG for reading or writing. (See positions A and B in Figure II-D-9). Space file forward positions the tape at A; space file backward positions the tape at B. Format and density must be stated in the instruction word since parity is checked during the tape motion. Any error detected and magnetic tape unit status is indicated in the status word sent to the computer with interrupt after completion. If the tape is at load point at the time the back space or at end of tape at the time the space file instruction is given, an improper condition exists and is noted in the status word.

6.6.4 REWIND

The selected transport rewinds the tape backward to the load point at rewind speed. The status word with interrupt is sent to the computer after the magnetic tape control initiates the rewind and not at the completion of the rewind. If the tape is at load point, when the instruction is received, no

tape notion or improper condition results, but the status word indicates load point. This provides a method of testing for completion of the rewind operation.

6.6.5 MULTIFUNCTION OPERATIONS (GENERAL INFORMATION)

Multifunction operations consist of combinations of basic operations of the magnetic tape unit, and can be performed in response to the one instruction word from the computer. Examples are the search operations which combine the features of a read with the ability to do a search on the first word of records, compare these words against an identifier (search key) word, and read on a find. Other multifunction operations combine a read with a rewind operation. Combinations of functions such as these save on computer instructions, and provide some capabilities that cannot be achieved by using the basic operations one at a time.

6.6.5.1 SEARCH (TYPE I AND TYPE II - FORWARD/BACKWARD)

The search operation combines the features of read forward/backward and a search. The selected tape transport reads/records from the tape either forward or backward and compares the first word* of each tape record with a search key (identifier word) which is transmitted from the computer to the magnetic tape unit by an output buffer of one word. When a compare is affirmative, that find record is transmitted to the computer as in a read forward/backward.

Tape motion is started upon receipt of the instruction word. If the magnetic tape unit does not receive the **search** key from the computer before it starts reading the record, an output timing error occurs. This reading start time may be as short as two and one-half milliseconds. The search operation is terminated by the magnetic tape unit when this timing error is detected by the magnetic tape control. The status word containing magnetic tape unit status and any or all error indications is sent to the computer upon detecting the end of the record in which the error occurred. When the tape motion is stopped due to an error, the tape will be positioned in the interrecord gap before the record in which the error occurred if the motion is backward, and after the record if the motion is forward. A parity error can result from a faulty parity check on any frame of the tape being searched.

The ones (type I) compare is a bit-by-bit greater-than-or-equal compare. If the first word of the record is greater than or equal to the search key identifier word, a find is made. A six-bit example is shown below.

Search key or identifier word	001101
Find, if first word is	011101
Find, if first word is	001101
No find, if first word is	010101
No find, if first word is	001100

---

* In a forward search, the first word encountered in each record is the first word of the record. In a backward search, the first word encountered in each record is the last word of the record.

The identical (type II) compare is an exact equal compare. The first word of the record must be exactly equal to the search key identifier word to define the find record.

6.6.5.2 SEARCH FILE * FORWARD/BACKWARD

The magnetic tape control performs a search forward/backward type I or type II as directed by operation code, on the selected tape transport, until it detects a find or a tape mark.** If a tape mark is detected before a find, the search file operation is terminated and the tape mark status code is present in the status word sent to the computer after detecting end of record.

6.6.5.3 REWIND-READ

The selected transport rewinds the tape to the load point at rewind speed and then performs a normal read of the first record according to the format and density stated in the instruction word. A status word containing magnetic tape unit status and any or all errors is sent to the computer with interrupt after detecting the end of record.

6.6.5.4 REWIND-CLEAR WRITE ENABLE

The selected transport performs a normal rewind of the tape to load point and clears the write enable. This selected transport no longer performs a write function without manual intervention. The status interrupt is presented upon initiation of the rewind and not upon completion.

6.6.5.5 REWIND-READ-CLEAR WRITE ENABLE

The selected transport performs a rewind-clear write enable and then a normal read of the first record in the forward direction according to the format stated in the instruction word. A status word containing magnetic tape unit status and any or all errors is sent to the computer with interrupt after detecting the end of record.

6.6.6 REQUEST TRANSPORT STATUS

No tape operation is performed. The magnetic tape unit sends a status word reflecting the status of the selected tape transport to the computer with interrupt. However, when the not ready indication is obtained, the remainder of the status word may not be valid as it may have been if derived from other handlers.

---

* A file is defined as one or more records separated by tape marks (see Figure II-D-9).

** A tape mark is a special record on a tape placed there by the operation, write tape mark (see Figure II-D-4).

Tape Forward Direction ⟶
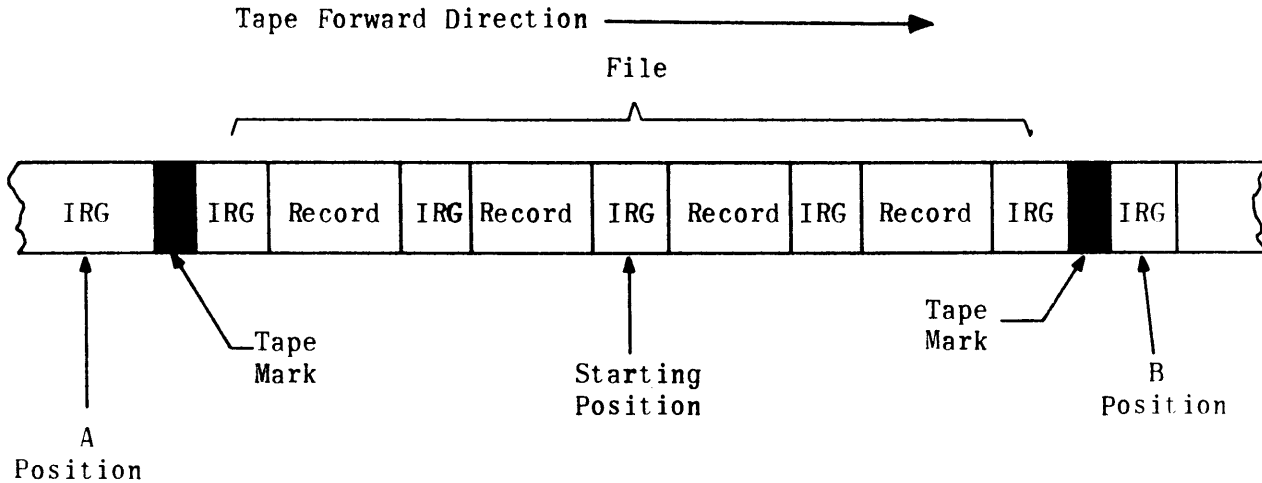
File

Figure II-D-9. Magnetic Tape Unit - Tape File

6.6.7 TRANSMIT EXTRA (BITS 17, 16, AND 6 = 1, 0, AND 1, RESPECTIVELY)

The transmit extra instruction word is sent under program control in response to an interrupt indicating a frame count error at the end of a read operation. All bits other than 17, 16, and 6 of the instruction word are ignored.

The transmit extra provides data recovery capabilities by the transmission of a single data word containing the extra characters of an incomplete computer word (6 bits per character) and denoting those character positions void of data. The extra characters appear in the most significant character positions consistent with the specified modulus. Each bit of the least significant six refers to a character position in the word originating in the magnetic tape control assembly register. All bits beyond the modulus limits contain 0's. Whenever a bit in the least significant character is 1, the corresponding character in the assembly register is invalid; when a bit is 0, the corresponding character is valid (see Figure II-D-10).

In the redundant format, the lack of one frame (character) would not allow the last frame to enter the assembly register and the frame would be lost. However, this would be a valid improper frame count and would be accompanied by a longitudinal parity error.

7. MAGNETIC TAPE UNIT - HIGH-SPEED PRINTER OFF-LINE CAPABILITY

Either magnetic tape unit is capable of communicating directly with the high-speed printer* for off-line operation. The magnetic tape unit communicates with the high-speed printer unit in the request-acknowledge moae. (The magnetic tape-printer interface is shown in Figure II-D-11).

Using terms based on computer-magnetic tape unit communication and computer-high-speed printer communication in this discussion, the output to the high-speed printer interface is connected to the input from the magnetic tape unit

*Does not apply to the 9200/9300 Subsystem.

Extra Frame Indicator

Bioctal Assembled Word   Binary

5 4 3 2 1 | 0   5 4 3 2 1 0

U V W X Y 00*   0 0 0 0 0 0

V W X Y Z 01   0 0 0 0 0 1

W X Y Z - 03   0 0 0 0 1 1

X Y Z - - 07   0 0 0 1 1 1   Mod 6

Y Z - - - 17   0 0 1 1 1 1

Z - - - - 37   0 1 1 1 1 1

- V W X Y 00*   0 0 0 0 0 0

- W X Y Z 01   0 0 0 0 0 1

- X Y Z - 03   0 0 0 0 1 1   Mod 5

- Y Z - - 07   0 0 0 1 1 1

- Z - - - 17   0 0 1 1 1 1
                                               Mod 5 Limits

- - W X Y 00*   0 0 0 0 0 0

- - X Y Z 01   0 0 0 0 0 1

- - Y Z - 03   0 0 0 0 1 1   Mod 4

- - Z - - 07   0 0 0 1 1 1
                                               Mod 4 Limits

- - - X Y 00*   0 0 0 0 0 0

- - - Y Z 01   0 0 0 0 0 1   Mod 3

- - - Z - 03   0 0 0 0 1 1
                                               Mod 3 Limits
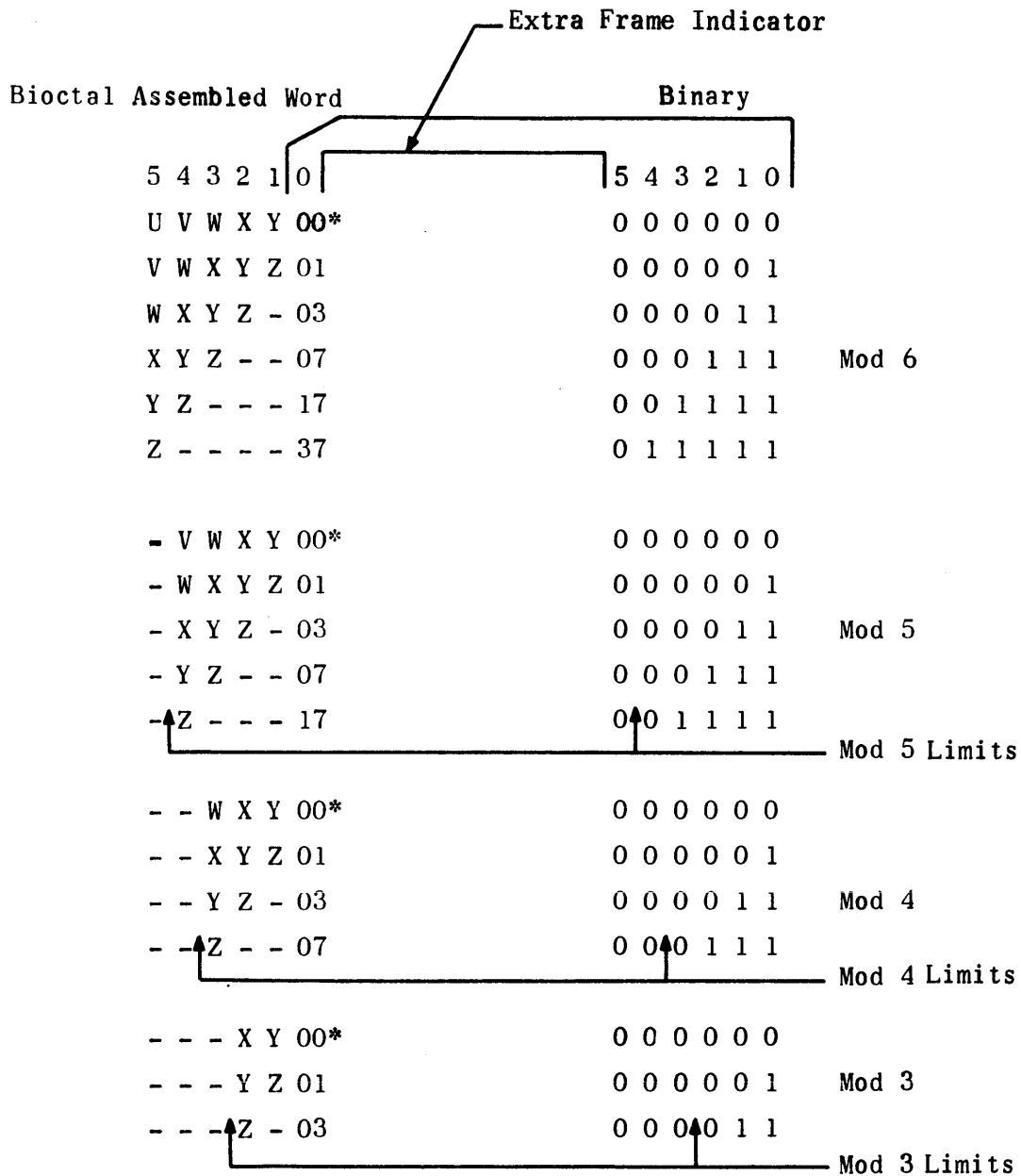
NOTE: Z denotes last frame. Y denotes second last frame, and so forth. A 0 in the extra frames indicator denotes a valid frame of data in the corresponding portion of the computer word within the modulus limits.

* The word is identical to the last (normally transmitted) word except that Z_0 has been destroyed, no extra frames are present, and the transmit extra command should not have been used.

Figure II-D-10. Transmit-Extra Computer Word Format

interface; that is, the high-speed printer output request line is connected to the magnetic tape unit input acknowedge line; the magnetic tape unit input request line is connected to the high-speed printer output acknowledge line; the magnetic tape unit interrupt line is connected to the high-speed printer external function line; the magnetic tape unit data lines are connected to the high-speed printer data lines.

The high-speed printer exercises control of the off-line system after the magnetic tape unit is switched to printer mode, the desired tape transport is selected, and the tape is positioned at load point. The high-speed printer initiates the operation when it is placed in off-line position.

The data on magnetic tape to be printed off-line must be recorded in 120 Field data character record lengths (120 characters per line on high-speed printer). As each record is read from the tape and transmitted to the high-speed printer, the 120 characters are printed as one line and the paper is advanced to the next line position. Each 30-bit word delivered to the high-speed printer must contain five Field data code characters (refer to Table II-D-4). These are in turn disassembled into six-bit characters and stored in the character core memory of the high-speed printer control unit. One word can be stored each 54 microseconds. (Refer to Table II-D-1 for recording density limitations.) When the core memory character counter indicates 120 characters, the print cycle cycle is initiated and the line is printed. A record of less than 24 30-bit words indicates to the high-speed printer to stop the print operation. A record of five space codes (05) stops the print operation without printing a line.
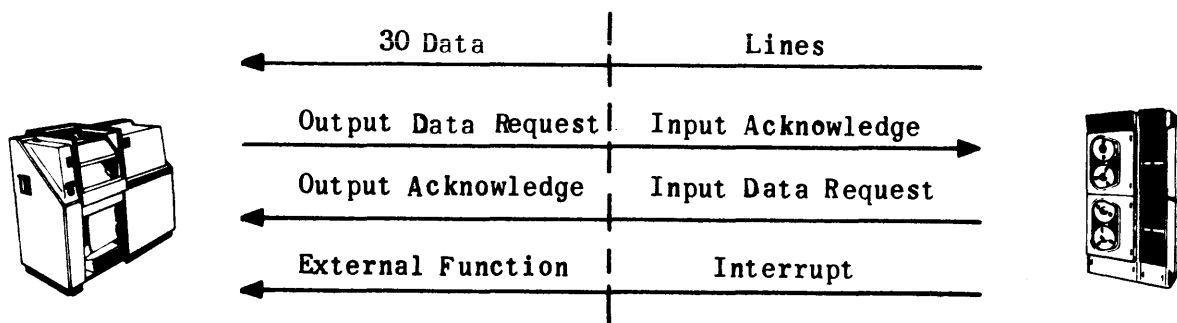
Figure II-D-11. Magnetic Tape — Printer Interface

8. OPERATING INSTRUCTIONS

To prepare the off-line magnetic tape unit-high-speed printer system for operation, the operator must select:

1) Character, parity, and density of the recorded tape at the magnetic tape unit cabinet.

2) Switch the magnetic tape unit to printer mode.

3) Select the desired tape transport.

4) Load and position the tape at load point.

5) Place the high-speed printer in off-line position.

9. SEQUENCE OF EVENTS

The normal sequence of events for transfer of data to the high speed printer is as follows:

1) High-speed printer sets its output data request.

2) Magnetic tape subsystem, in the ready state, recognizes the first output data request.

3) The magnetic tape unit places a word on the data lines and sets its input data request.

4) High-speed printer recognizes this input data request as an output acknowledge.

5) High-speed printer samples the data lines and clears its output data request.

6) Magnetic tape unit recognizes the clearing of the output data request as an input acknowledge.

Steps 3 through 6 are repeated until the complete record is transferred, at which time the line is printed, the paper is advanced, and the cycle is re-initiated. The process continues until the end of file tape mark is read. The high-speed printer recognizes the tape mark as a command to position the paper at top of form on the next page. The interrupt line of the magnetic tape unit being connected to the external function line of the high-speed printer permits the end of record and the tape mark codes to be sent to the high-speed printer with commands to move paper one line space or top of form, respectively.

II-D-27

TABLE II-D-4. TYPE SYMBOLS AND CODES

Octal	Binary Code	Character	Octal Code	Binary Code	Character
00	000 000	| Absolute value	27	010 111	R
01	000 001	↑ Arrow (up)	30	011 000	S
02	000 010	8 Subscript eight	31	011 001	T
03	000 011	[Bracket (open)	32	011 010	U
04	000 100	] Bracket (close)	33	011 011	V
05	000 101	Space (undercut)	34	011 100	W
06	000 110	A	35	011 101	X
07	000 111	B	36	011 100	Y
10	001 000	C	37	011 111	Z
11	001 001	D	40	100 000	)
12	001 010	E	41	100 001	−
13	001 011	F	42	100 010	+
14	001 100	G	43	100 011	<
15	001 101	H	44	100 100	=
16	001 110	I	45	100 101	>
17	001 111	J	46	100 110	≤ Equal to or less than
20	010 000	K	47	100 111	{ Left-hand brace
21	010 001	L	50	101 000	*Star
22	010 010	M	51	101 000	(
23	010 011	N	52	101 010	≥ Equal to or greater than
24	010 100	O	53	101 011	,
25	010 101	P	54	101 100	} Right-hand brace
26	010 110	Q			

TABLE II-D-4. TYPE SYMBOLS AND CODES (CONT.)

Octal	Binary Code	Character		Octal	Binary Code	Character
55	101 101	∨ Or		67	110 111	7
56	101 110	,		70	111 000	8
57	101 111	≠		71	111 001	9
60	110 000	0		72	111 010	∧ And
61	110 001	1		73	111 011	;
62	110 010	2		74	111 100	/
63	110 011	3		75	111 101	.
64	110 100	4		76	111 110	→ Arrow right
65	110 101	5		77	111 111	x Multiply sign
66	110 110	6				

SECTION II-E. UNIVAC HIGH-SPEED PRINTER (MODEL 1469)

(This section has been intentionally omitted.)

SECTION II-F. UNIVAC 1004 CARD PROCESSOR

1. BASIC INFORMATION

The UNIVAC(R) 1004 Card Processor is a character-oriented data processing computer. It may be adapted to be used as a peripheral equipment to a larger computer. The UNIVAC 1004 Card Processor has the ability to perform arithmetic functions, transfers, and compare operations using XS-3 coded characters (refer to Table II-F-1). It contains a magnetic core memory consisting of 961 character locations. It also contains a built-in card reader and high speed printer. A card punch as well as other peripherals may be included in the 1004 System.

The operation of the 1004 Processor is under the control of a series of instructions wired on a plugboard. This series of instructions is referred to as the 1004 program and each individual instruction is called a step. Each step indicates that data is to be manipulated in some manner. When the machine has completed all of the program steps, it is said to have completed its internal cycle.

The plugboard, housed in a recess at the right of the 1004 processor, can be removed for wiring or for storage when other plugboards are in use.

The discussion of 1004 operation presented here is with respect to the standard 1004 plugboard. This is a universal UNIVAC pre-wired plugboard for use with a 1004 equipped with the universal computer/1004 interface adapter. It is identified as: Part No. 4010507 B; Plugboard Assembly, Wired. No attempt is made to describe the full capabilities of the 1004 processing section. For a detailed description of programming a 1004 plugboard, refer to the reference manual for the UNIVAC 1004 Card Processor, 80-Column.

The format of the data interface between the computer and the 1004 System is a computer word. The adapter disassembles the words from the computer into 6-bit XS-3 characters for the 1004. Similarly, the adapter accepts the XS-3 characters from the 1004 and assembles them into the word length of the associated computer. Because of the operation of the interface adapter, all buffers to or from the 1004 must be packed XS-3 characters.

Figure II-F-1 shows the interface signals, both control and data, transmitted between equipments during operation. Note that there are no external function or external interrupt lines.

Five types of messages are utilized in communication between the computer and the 1004:

1)	Command message	(computer to 1004).
2)	Reply message	(1004 to computer).
3)	Read data message	(1004 to computer).
4)	Punch data message	(computer to 1004).
5)	Print data message	(computer to 1004).

TABLE II-F-1. 80-COLUMN CODE

80-Column Card Code	Printable Characters	XS-3 Code	80-Column Card Code	Printable Characters	XS-3 Code
12-1	A	01 0100	7	7	00 1010
12-2	B	01 0101	8	8	00 1011
12-3	C	01 0110	9	9	00 1100
12-4	D	01 0111	12	&	01 0000
12-5	E	01 1000	11	– (minus)	00 0010
12-6	F	01 1001	12-8	?	01 0011
12-7	G	01 1010	11-8	! (exclam.)	10 0011
12-8	H	01 1011	0-1	/	11 0100
12-9	I	01 1100	2-8	+	11 0011
11-1	J	10 0100	3-8	#	01 1101
11-2	K	10 0101	4-8	@	10 1110
11-3	L	10 0110	5-8	: (colon)	01 0001
11-4	M	10 0111	6-8	>	11 1110
11-5	N	10 1000	7-8	' (apos.)	10 0000
11-6	O	10 1001	12-3-8	. (period)	01 0010
11-7	P	10 1010	12-4-8	□	11 1101
11-8	Q	10 1011	12-5-8	[	00 1111
11-9	R	10 1100	12-6-8	<	01 1110
0-2	S	11 0101	12-7-8	=	01 1111
0-3	T	11 0110	11-3-8	$	10 0010
0-4	U	11 0111	11-4-8	*	10 0001
0-5	V	11 1000	11-5-8	]	00 0001
0-6	W	11 1001	11-6-8	; (semi-col.)	00 1110
0-7	X	11 1010	11-7-8	Δ	10 1111
0-8	Y	11 1011	0-2-8	≠	11 0000
0-9	Z	11 1100	0-3-8	, (comma)	11 0010
0	0	00 0011	0-4-8	%	11 0001
1	1	00 0100	0-5-8	(	10 1101
2	2	00 0101	0-6-8	\	00 1101
3	3	00 0110	0-7-8	)	11 1111
4	4	00 0111			
5	5	00 1000	Blank	No Space N.B.	00 0000
6	6	00 1001			

Computer Input Cable

Input Data Lines

Input Request

Input Acknowledge

Computer

A
D
A
P
T
E
R

1004 Card
Processor

Output Data Lines

Output Acknowledge

Output Request

Computer Output Cables
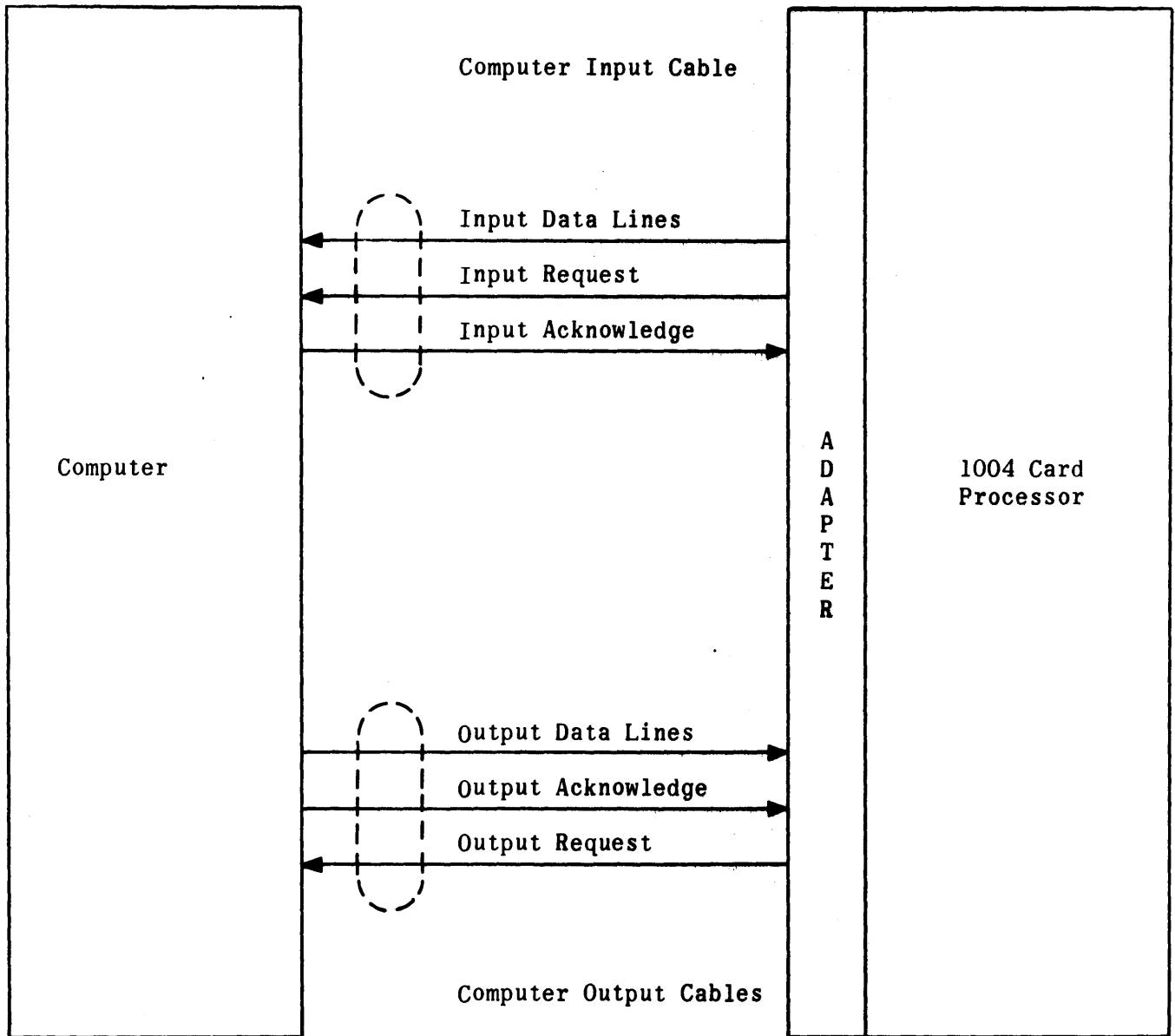
Figure II-F-1. Computer/1004 Card Processor Interface

When the 1004 is turned on and started, the only message it will respond to is
the command. When a computer chooses to utilize the 1004 Subsystem, it must
first initiate an output buffer to send a 30-character command message to the
1004. The first word of the command message is coded to signify what 1004
function is to be performed. When the 1004 receives a legitimate command
message it will immediately respond by sending a 30-character reply message
to the computer. To receive the reply message, the computer must initiate
an input buffer after sending the command. The content of the reply message
is not significant since its only purpose is to inform the computer that the
1004 has received a complete command message. The reply, however, will always
be identical in content to the last command.

After the command has been sent and the reply has been received, the next step
depends upon the type of function to be performed. In general, the function
will require the transfer of data, and the computer is required to initiate
the appropriate input or output data buffer. If an output data message is sent
to the 1004, the 1004 will respond with a reply message in exactly the same
manner as it does for a command. If an input data message is received from
the 1004, no reply message will follow.

The standard plugboard (#4010507 B) provides for the following major functions
to be performed by the 1004 subsystem: card reading, card punching, printing,
and data transmission to and from the computer. Each function is initiated by
the computer by transmission of a command message to the 1004. It is per-
missible to specify more than one function in a single command message. When
this is done, a fixed internal priority scheme determines the order in which
the functions are performed. The computer cannot send a new command until all
functions specified by the previous command have been completed.

Each of the 1004 functions is discussed briefly below:

1) Transmit Read Data

    This function transfers a 90-character read data message from the 1004
    to the computer. The first 80 characters are from the 1004 read storage;
    the last 10 characters are not significant. The computer must initiate
    an input buffer to receive the data. The data transferred is from the
    last card that was read into read storage.

2) Read

    This function causes the card reader to read the next card. The 80-column
    card code is automatically translated into 80 XS-3 characters and stored
    in the 1004 read storage where it will destroy any data previously stored
    there. After reading a card, a transmit read data command must be used
    to transfer the data into the computer. The computer must initiate these
    two commands for each card to be read.

3) Receive Print Data

    This function transfers a 150-character print data message from the com-
    puter to the 1004. The first 132 characters are stored in the 1004 print

storage. The last 18 characters are not significant. The computer must initiate an output buffer to send the data.

4) Print

This function causes the 1004 printer to print one line consisting of the 132 characters currently in the print storage. After printing the line, the 1004 will automatically space to the next line. When 64 lines of a 66-line page are printed, the 1004 will home paper to a position defined by a pre-punched paper tape control loop (usually top of next page).

5) Receive Punch Data

This function transfers a 90-character punch data message from the computer to the 1004. The first 80 characters are stored in the 1004 punch storage. The last 10 characters are not significant. The computer must initiate an output buffer to send the data.

6) Punch

This function causes the 1004 card punch to punch one card containing the 80 characters currently in the punch storage. The 1004 automatically translates from the XS-3 code to 80-column card code during this process.

7) Receive Special Data

This function is used for test purposes only. It transfers 90 characters from the computer to a special storage area in the 1004 memory. This area is filled in reverse (ascending) order.

8) Transmit Special Data

This function is used for test purposes only. It transfers 90 characters from the 1004 special storage area to the computer. This area is emptied in reverse (ascending) order.

9) Space One Line

This function causes the 1004 printer to space one line without printing.

10) Home Paper

This function causes the paper form in the printer to be advanced to a position defined by a pre-punched paper tape control loop (usually top of next page).

11) Punch Blank Card

This function clears 1004 punch storage and causes one blank card to be punched.

12) Stop

This function causes the 1004 processor to stop. The unit must then be manually restarted before any more functions can be performed.

2. MESSAGE AND WORD FORMATS

A 30-character command message must first be sent to the 1004. The purpose of this message is to allow the computer to signal the 1004 to perform one or more of its functions (read, punch, and so forth). The format of the command message is shown in Figure II-F-2. Note that only the first word contains significant command information. The command message is master bit encoded and it is possible to set more than one command bit in a single message. Several instances in which it may be convenient to set multiple command bits are listed below.

1) If bits 4 and 5 of character 2 are both set, the command instructs the 1004 to read a card and transmit data to the computer in one operation. It should be noted, however, that due to the priority of the plugboard logic, the transmit read data function is performed first and then the read function. Thus, each successive operation actually transfers the data from the card that was read in the previous operation. If this order of events is not satisfactory for a particular user application, it will be necessary to program the two commands separately.

2) If bits 2 and 3 of character 2 are both set, the command instructs the 1004 to receive a print data message and to print that data in the same operation.

3) If bits 0 and 1 of character 2 are both set, the command instructs the 1004 to receive a punch data message and to punch that data in the same operation.

The computer cannot send a new command message to the 1004 until the previous operation has been completed. After receiving a command message the 1004 will always respond with a reply message. Only the commands, receive punch data and receive print data, may be followed by a data output buffer. The first will be followed by 80 characters of data plus ten non-significant characters. The second will be followed by 132 characters of data plus 18 non-significant characters. These characters must be in XS-3 code packed into data words. Figure II-F-3 shows the format of 30-bit and 18-bit data words. After receiving an output data message the 1004 always responds with a reply message.

There are two possible buffer lengths that will be transmitted by the 1004: a 30-character buffer (reply message) or 90 characters of data read from a card.

The 1004 transmits a reply message to the computer to signal the end of an output operation. This message is sent by the 1004 either after receipt of a command message or after receipt of output data (print data message or punch data message).

The reply message plays a significant role in the communication between the computer and 1004 by signifying the end of an output buffer.
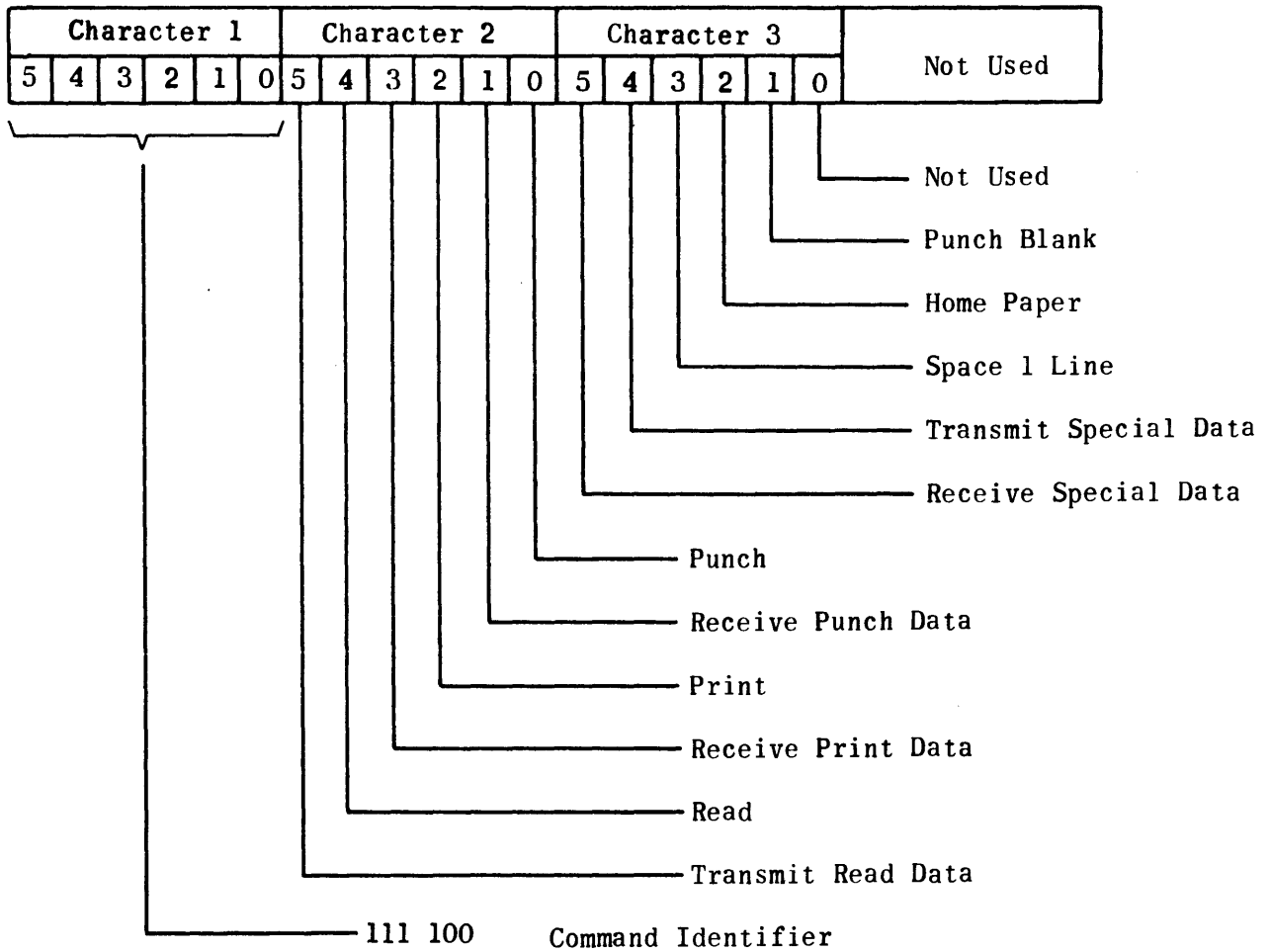
Figure II-F-2. Command Code Format (First Word Only)
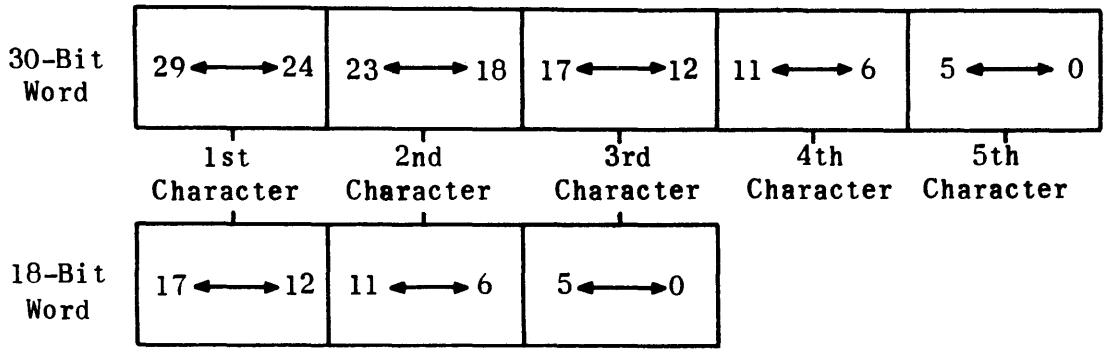
Figure II-F-3. Data Words

Receipt of a reply message guarantees that the 1004 has accepted all of the output data and permits the computer to initiate its next input or output buffer. The reply message is 30 characters. While the content of the reply is not significant, it might be noted that it will be identical to the last command message received by the 1004.

Only the command transmit read data may be followed by a data input buffer (after receipt of reply). This buffer will be 80 characters of data plus 10 non-significant characters (a total of 90 characters). The data word format is identical to that of output data and is shown in Figure II-F-3.

Tables II-F-2 and II-F-3 summarize the buffer sizes required for computer/1004 communication and the commands used to initiate communications.

TABLE II-F-2. BUFFER SIZES FOR COMPUTER/1004 COMMUNICATIONS

Message Type	30-Bit Words	18-Bit Words	Number Of 6-Bit Characters
Command Message	6	10	30
Reply Message	6	10	30
Print Data Message	30	50	150
Punch Data Message	18	30	90
Read Data Message	18	30	90

TABLE II-F-3. SUMMARY OF COMMAND CODES

Command Type	First Command Word*	
	30 Bit	18 Bit
Read	7420000000	742000
Read and Transfer	7460000000	746000
Punch and Transfer	7403000000	740300
Print and Transfer	7414000000	741400
Home Paper	7400040000	740004
Space One Line	7400100000	740010

* Only the first word is significant; however, six 30-bit words or ten
18-bit words must be sent.

Sample routines which may be used to program communications between a computer
and a 1004 Card Processor are given below. The routines are coded in
TRIM III source language and are intended for use with the computer in the
1219 I/O buffering mode and a 1004 with a standard plugboard. The routines
assume that the buffer areas referenced contain the proper number of words
and that the first word of each command buffer contains the proper command
code.

   1) Read-a-Card Routine

```
OUT*CH1004
O*READCOM+11 (+12 for 1218 modes)
O*READCOM
IN*CH1004
O*WAIT+11
O*WAIT
SKPIIN*CH1004
JP*LOK-1
OUT*CH1004
O*TRANS+11 (+12 for 1218 modes)
O*TRANS
IN*CH1004
```

1) Read-a-Card Routine (Continued)

    O*WAIT+11
    O*WAIT
    SKPIIN*CH1004
    JP*LOK-1
    IN*CH1004
    O*INBUFF+35
    O*INBUFF
    SKPIIN*CH1004
    JP*LOK-1

2) Print/Punch Routine

    OUT*CH1004
    O*PRINTFCT+11 (+12 for 1218 modes)
    O*PRINTFCT
    IN*CH1004
    O*WAIT+11
    O*WAIT
    SKPIIN*CH1004
    JP*LOK-1
    OUT*CH1004
    O*PRINTBUFF+61 (+62 for 1218 modes, +35 or 36 for punch)
    O*PRINTBUFF
    IN*CH1004
    O*WAIT+11
    O*WAIT
    SKPIIN*CH1004
    JP*LOK-1

3) Home Paper Routine

    OUT*CH1004
    O*HOMPAP+11 (+12 for 1218 modes)
    O*HOMPAP
    IN*CH1004
    O*WAIT+11
    O*WAIT
    SKPIIN*CH1004
    JP*LOK-1

3. MANUAL OPERATING PROCEDURES

Prior to utilizing the UNIVAC 1004 Card Processor as peripheral equipment, it is necessary to place all switches and controls in the normal operating position and initialize the particular input/output device to be exercised. The manual procedures which must be performed at the 1004 control panel for each input/output device are given on the next page.

3.1 CARD READER

The input card deck to be read must be mounted in the card reader input hopper. To initialize the card reader, depress the following 1004 switches in order:

   1) CLEAR
   2) START
   3) FEED
   4) RUN

3.2 CARD PUNCH

Prior to performing a card punch operation, the operator must ensure that there are sufficient blank cards in the punch to perform the operation. To initialize the card punch, depress the following 1004 punch switches in order:

   1) OFF
   2) ON
   3) START

After a card punch operation is performed, the last card punched remains in the card punch. To remove this card, depress the following 1004 punch switches in order:

   1) OFF
   2) ON
   3) START

3.3 HIGH-SPEED PRINTER

Prior to performing a printing operation, the operator must ensure that there is sufficient paper in the printer to perform the operation. To initialize the high-speed printer, depress the following 1004 switches in order:

   1) CLEAR
   2) START
   3) RUN

SECTION II-G. UNIVAC 9200/9300 SUBSYSTEM

1. GENERAL INFORMATION

The UNIVAC Ⓡ 9200/9300 Computer is a character-oriented data processing computer. It may be adapted as a peripheral to a larger computer through the use of the UNIVAC Intercomputer Control Unit (ICCU). The 9200/9300 Computer is internally programmed and is available with a complete line of software. It contains plated-wire memory divided into 8-bit bytes with a minimum of 8K (8191) bytes. It also contains a built-in high speed printer, card reader and card punch. Other peripherals are also available with the 9200/9300 System.

The operation of the 9200/9300 Military Computer Subsystem is accomplished by the ICCU. The ICCU provides for the conversion and transmission of words to bytes and bytes to words between the 9200/9300 Subsystem and the military computer. See Figure II-G-1. To control the flow of data, the ICCU uses external functions and interrupts.

Univac standard software (handlers resident in both computers) work together in order to transfer data. The military computer program initiates all data transfers and, as such, acts as the master. The 9200/9300 Program acts as a slave, accepting data or control commands, printing lines, and punching or reading cards as directed by the master.

The discussion of the 9200/9300 military computer interface was prepared using the Definition of Interface: UNIVAC 18, 30, or 36-Bit Computer with UNIVAC Intercomputer Control Unit, PX 5458, and the 9200/9300 Resident Handler Program created by the System Programming Department of UNIVAC Federal Systems Division.

No attempt is made to describe the complete operation or programming of the 9200/9300 Computer. For detailed information on the 9200/9300 System refer to the documentation describing that system; especially, Preliminary Operation Instructions for the UNIVAC 9200 Systems, UP-7537.

2. MILITARY COMPUTER/ICCU INTERFACE

2.1 INTRODUCTION

To avoid duplication, one basic interface will be used for the 18, 30, and 36-bit word lengths with a resident handler, for each unique machine, written from the basic interface.

The basic interface between a military computer and the ICCU shall be defined as follows.

2.2 DATA FORMATS
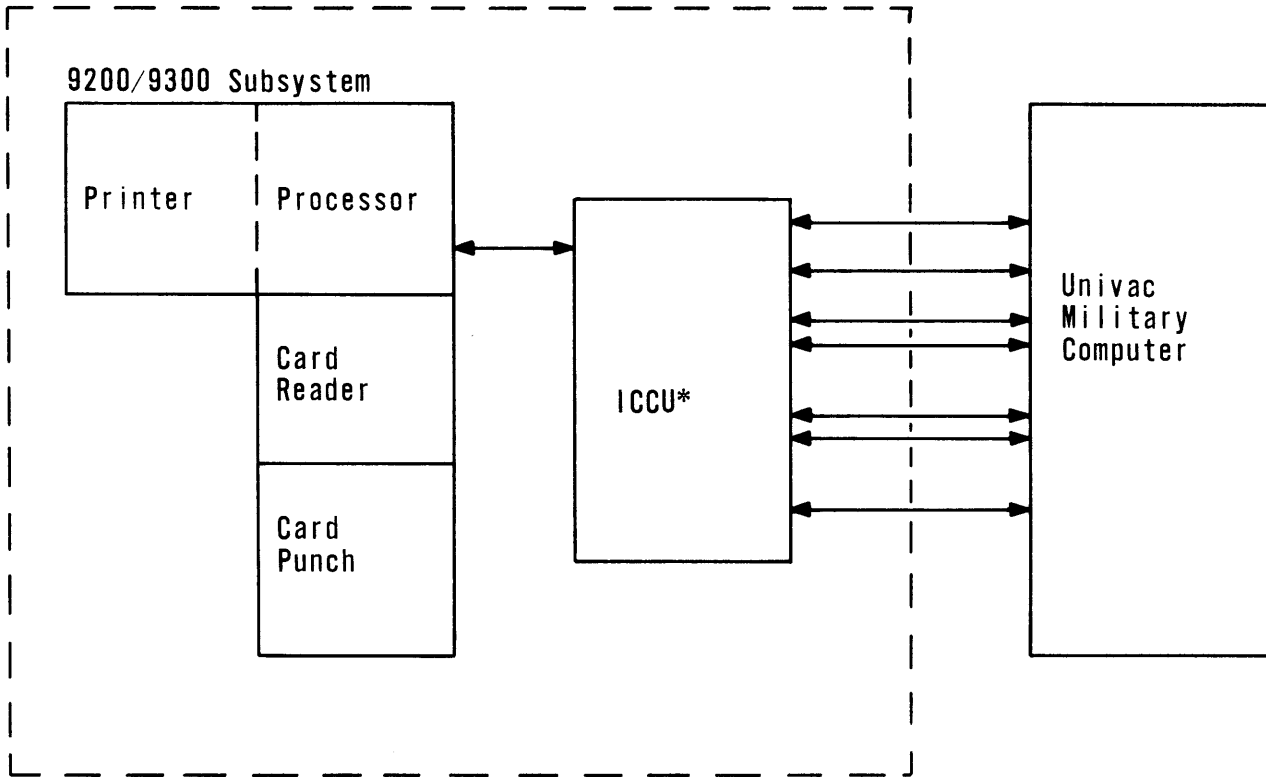
* ICCU contained in the 9200/9300 Processor Cabinet

Figure II-G-1. ICCU Communication and Interface

2.2.1 ICCU DATA TRANSFER FORMATS

The ICCU can transfer data for a specific word length (18, 30, or 36-bits) in three data format modes. Presently, the handler programs are written in data format mode B only.

Figure II-G-2 illustrates data format modes for an 18-bit interface; Figure II-G-3, a 30-bit interface and Figure II-G-4, a 36-bit interface.

2.2.2 HEADER FORMATS

Header blocks transfer functional information from the master to the slave. The blocks, in fixed word length format, are transformed to 8-bit bytes for the slave.

Figure II-G-5 illustrates fixed-word length format for an 18-bit interface; Figure II-G-6, a 30-bit interface and Figure II-G-7, a 36-bit interface.

2.3 HEADER INFORMATION

Message headers and control block headers contain information that is used by the slave program to perform the desired operation. The message header is sent once to each device to establish the format for the desired operation. Any subsequent calls, using the special function (see Subsection 2.6.3), will result in the same operation as specified in the last header sent to that device. The message header can precede the output buffer and be sent with each data transfer that takes place.

2.3.1 MESSAGE HEADER FORMAT

The message header is 6-bytes as shown in the word format in Figure II-G-5, II-G-6, or II-G-7. The byte format is:

Bit Position

7 6 5 4			0	Byte
XX	S	Function		1
XX	Device Number			2
XX	Translation Code			3
XX	Control Field			4
XX	Character			5
XX	Count			6

Format A
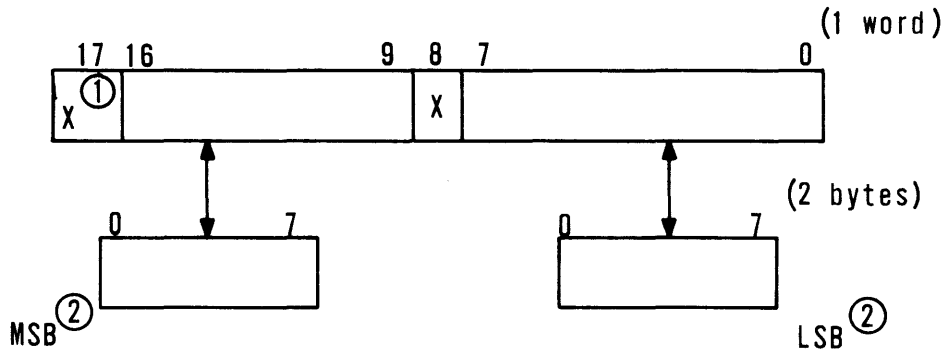
Format B
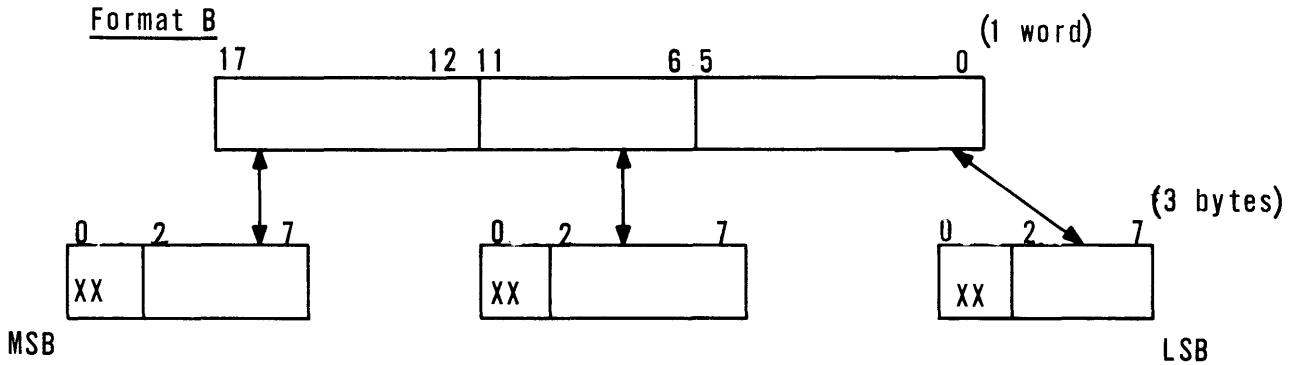
Format C
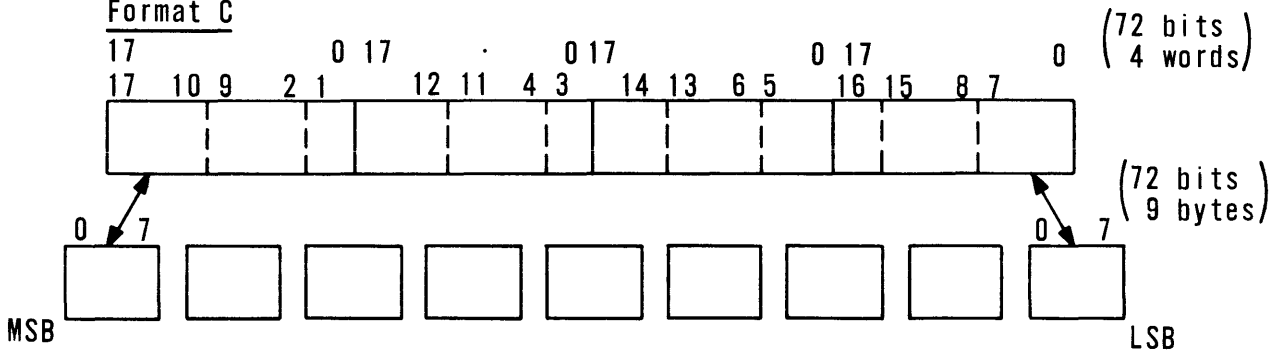
① X - Bits ignored on output from register, zero filled on input to register
② MSB - Most Significant Bit
   LSB - Least Significant Bit

Figure II-G-2. Data Formats, 18-bit Interface

Format A

(1 word)

29   23 22   15 14   8 7   0

X X X X X X X ① | | X X X X X X X X |

(2 bytes)

0   7   0   7

MSB ②   LSB ②

Format B

(1 word)

29   24 23   18 17   12 11   6 5   0

(5 bytes)

0 2 7   0 2 7   0 2 7   0 2 7   C 2 7

XX | XX | XX | XX | XX |

MSB   LSB

(4 words)
(120 bits)

Format C

29   0 29   0 29   0 29   0
  23 22 15 14 7 6 29 27 20 19 12 11 4 3 26 25 18 17 10 9 2 1 24 23 16 15 8 7 0

120 bits
15
bytes

0 7   0 7

MSB   LSB

① X - Bits ignored on output from register, zero filled on input to register
② MSB - Most Significant Bit
   LSB - Least Significant Bit

Figure II-G-3. Data Formats, 30-bit Interface

Format A
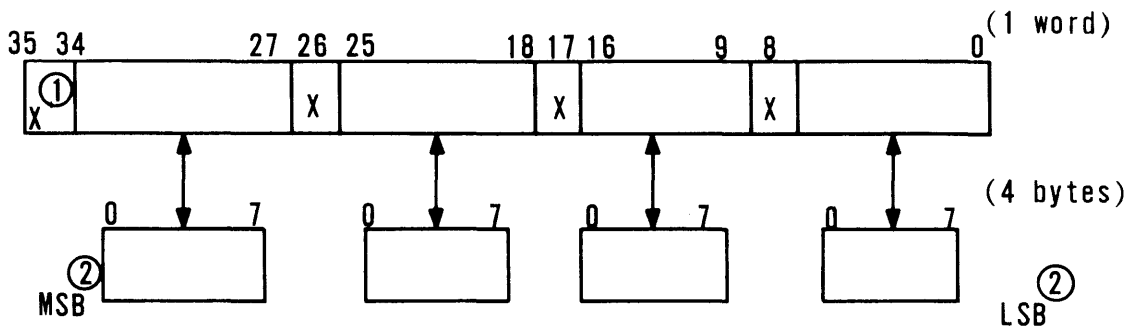
Format B
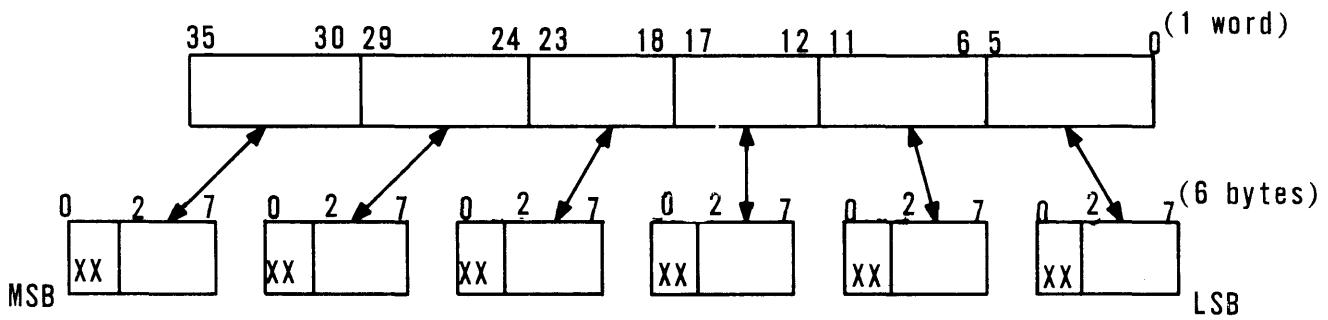
Format C
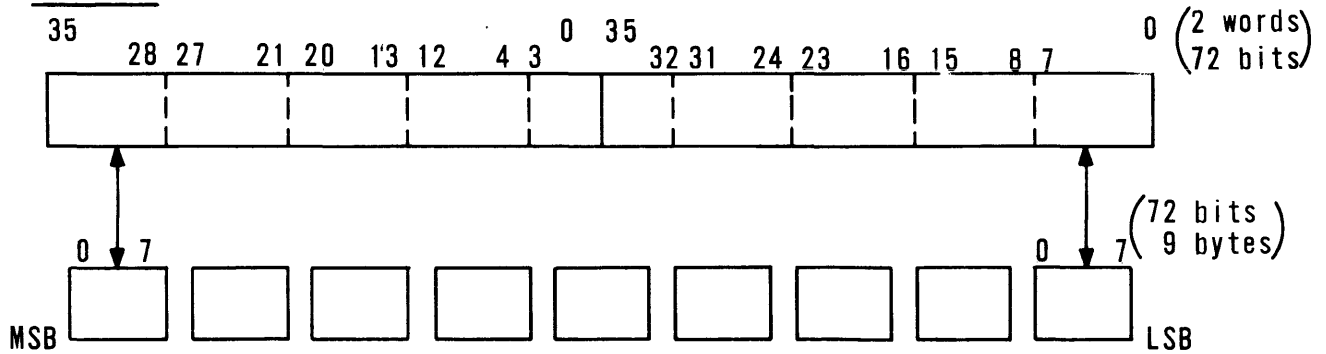
① X - Bits ignored on output from register, zero filled on input to register

② MSB - Most Significant Bit

   LSB - Least Significant Bit

Figure II-G-4. Data Formats, 36-bit Interface

Format A

	17 16	9 8 7	0
X	1	X	2
X	3	X	4
X	5	X	6

Format B

17 12	11 6	5 0
1	2	3
4	5	6

Format C

17 0	17 0	17 0	17 0
1 2	3 4 5	6 X	X X

Figure II-G-5. Message Header Format, 18-Bits

Format A

29	22	15	7	0
	1		2	
	3		4	
	5		6	

Format B

29 24	23 18	17 12	11 6	5 0
1	2	3	4	5
6	X	X	X	X

Format C

29 0	29 0	29 0	29 0
1 2 3 4	5 6 X X	X X X	X X X X

Figure II-G-6. Message Header Format, 30 Bits

Format A

35 34	27	25	18	16	9	7	0
X	1	X	2	X	3	X	4
	5		6		X		X

Format B

35	30 29	24 23	18 17	12 11	6 5	0
1	2	3	4	5	6	

Format C

35					0 35			0
1	2	3	4	5	6	X	X	X
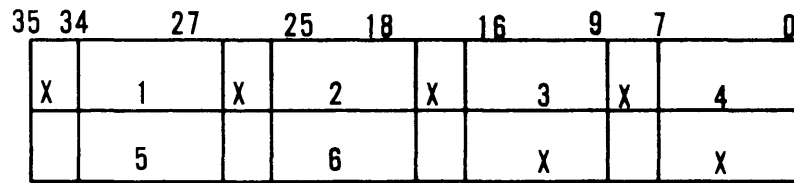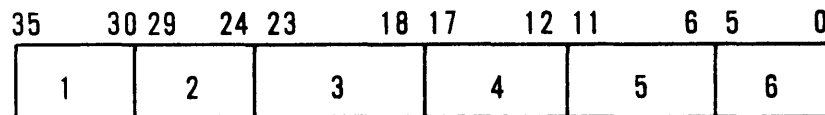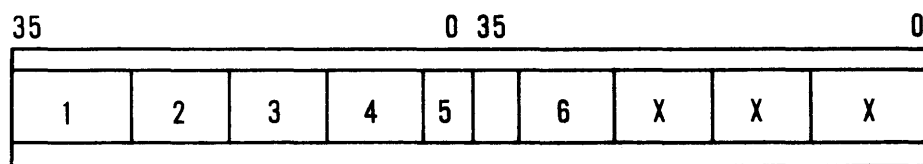
Figure II-G-7. Message Header Formats, 36 Bits

Where:

XX is not used

S – Bit 5 of Byte 1. If S = 0, setup function performed by this header, no data transferred. If S = 1, header immediately precedes output data or an input function.

Function – Bits 0-4 of Byte 1. Describes the action to take place as given in the following codes.

Code	Operation
0	Illegal
1	Write
2	Read Forward
3	Read Backward
4	Not Used
5	Maintenance Turnaround

Device Number – Byte 2. Specifies the device to or from which the data is to be transferred according to the following codes.

Code	Device
1	Card Reader
2	Serial Read/Punch
3	Printer
4	Row Read/Punch
5	1001 Card Controller

Translation Code – Byte 3. Specifies type of translation desired by the master. The data will be translated to the desired code on input and translated from the desired code on output.

Code	Translation
0	No Translation
1	XS-3
2	Field Data
3	ASCII
4	BCD
5	Binary

Control Field – Byte 4. The interpretation of this field depends upon the device specified in byte 1 according to the following:

Card Reader – The contents of this field are ignored.

Serial Read/Punch - Contains one half the number of columns to be punched on a card. If equal to zero, a full 80 columns are punched.

Printer - The upper two bits (4 and 5) of this field contain a print line length code, where: 00 denotes a 132 character print line, 01 denotes a 120 character print line. Bits 0 to 3 contain a spacing code of 0, 1, or 2. The printer will space this number of lines automatically after each print function.

Character Count - 12 bits. The number of data bytes to be sent to or from master on input or output. The count is right justified with the least significant bits in byte 5 and contains the number of characters in the input/output block.

2.3.2 CONTROL BLOCK

Special functions of the slave peripherals are signaled by the use of a control block.

The control block has the following format.

Bit Position

7		0	Byte
XX	Function Code		1
XX	Device Number		2
XX	N1		3
XX	N2		4
XX	N3		5
XX	N4		6

Where:

XX is not used

Function code - Byte 1. Describes the action to take place.

Code (Octal)	Operation
20	Space N1 lines on printer
21	Skip printer carriage to channel N1.
22	Select alternate punch stacker.

II-G-11

Device Number - Byte 2. Specifies the device on which the control function is to be performed. The codes are the same as those used for the message header.

N1, N2, N3, N4 - Bytes 3-6. Contain parameters necessary for the specific function according to the following codes.

Function Code	Parameters
20	N1 = Space count N2, N3, N4 not used.
21	N1 = Channel to skip to on forms control tape. Channel 7 is top of form. N2, N3, N4 not used.

2.4 CONTROL WORD FORMATS

2.4.1 MASTER EXTERNAL FUNCTION WORD

The Master External Function Word (MEFW) is sent by the master to the ICCU to initiate all data transfers, special functions and control operations. The format of the MEFW is:

Master
Bit Position

```
17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
┌───┬─────────────────────────────┬───┬───┐
│ X │              D              │ F │ C │
└───┴─────────────────────────────┴───┴───┘
```
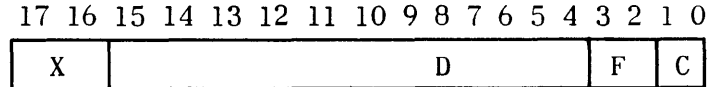
Where:

XX - Bits 17 and 16. Ignored (not transferred to slave)

D - Bits 15-4. Input data transfers: message header byte count; $6 \leq (D) \leq 7777_8$. Output data transfers: byte count for message header and data; $6 \leq (D) \leq 7777_8$ (Number of words sent to slave times number of bytes per master computer word.) Special functions: device number; (1 = card reader, 2 = card punch, 3 = printer etc.); $1 \leq (D) \leq 5$. Control functions: 0

F - Bits 3 and 2. Data Format Selection: F = 00 select format A, F = 01 select format B, F = 10 select format C, F = 11 not defined.

C - Bits 1 and 0. Control Field: C = 00 set "ATTENTION" interrupt to slave, C = 01 input data to master, C = 10 output data from master, C = 11 invalid.

2.4.2 EXTERNAL INTERRUPT STATUS WORD

When the slave sends an external interrupt to the master, this interrupt is accompanied by an External Interrupt Status Word (EISW) made up of a command byte from the slave and sense byte (1) from the ICCU. The EISW accompanying an interrupt terminating a data transfer will have a command byte of zero. The format of the EISW is:

Master
Bit Position   17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

X	R	S	X	C
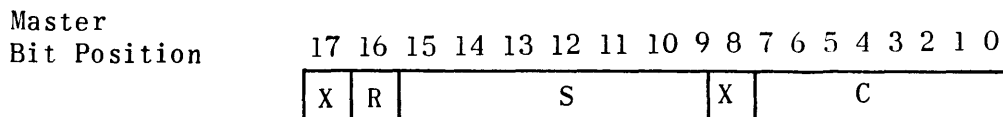
Where:

X – Bits 17 and 8. Always zero.

R – Bits 16. Selective Reset – One if a preceding operation has been terminated by the slave with a selective reset. (Error detected by slave channel.)

S – Bits 15-9. Error Status – Presented to both the master and the slave by the ICCU. (See Subsection 2.4.2.1.)

C – Bits 7-0. The command byte from the slave. (See Subsection 2.4.2.2.)

2.4.2.1 ERROR STATUS

15 14 13 12   9

R	X	B	X
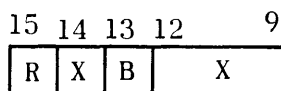
Where:

R – Bit 15. Command Reject – Unspecified command issued to ICCU.

X – Bits 14 and 12-9. Not used.

B – Bit 13. Bus Out Check – Even parity on bus out during transmission of data or command to ICCU from slave channel.

2.4.2.2 COMMAND BYTE

7 6 5 4 3 2 1 0

E	I	C
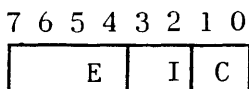
Where:

E – Bits 7-4. Error Code – An error has been detected by the slave and has the following interpretation.

Code	Reason
1	An illegal format detected by slave.
2	An illegal device number referenced.

Code	Reason
3	A punch check has occurred and recovery has failed.
4	An illegal function code received by the slave.
5	No initiation sequence performed to initiate the master/slave interface.

I - Bits 3, 2. Interpretation code.   01 : Termination of output functions.   10 : Hardware or software error

C - Bits 1, 0. Always 11 identifying the external interrupt command.

2.5 INITIATION SEQUENCE

Prior to the execution of any data transfers or control functions, the master computer must perform an initiation sequence (handshake).

The handshake is performed as follows.

1) Send MEFW with control field, (bits 1, 0), set to zero.

2) Receive external interrupt from slave. The command byte of the accompanying EISW will have a S field of 0, I field of 01 and a C code of 11.

2.6 DATA TRANSFER SEQUENCES

All data transfers sequences shall originate in the master computer.   The interface for output, input, and special functions is defined below.

2.6.1 OUTPUT DATA TRANSFER

Any data sent to the slave for output must be preceded by a message header. The message header may be sent to the slave without any accompanying data to set up the slave handler for a specific format and the subsequent use of the special function. Multiple card images may be sent depending on the memory size of the slave. The output data transfer sequence is:

1) Send the MEFW to the ICCU via an external function command.

2) Initiate an output buffer with a length equal to the word length of message header plus the word length of the output data buffer plus one word (length in 18, 30, or 36 bit words, not bytes).

3) Accept an external interrupt and accompanying EISW upon completion of data transfer (channel end interrupt). The command byte of the EISW will be zero upon termination of data transfer.

4) Accept an external interrupt and accompanying EISW signaling completion of the output function (device end interrupt).

2.6.2   INPUT DATA TRANSFER

To input data, a message header is sent to the slave to request input of data or to set up the slave handler for a specific input format and the subsequent use of the special function. Multiple cards may be read depending on the memory size of the slave. The input data transfer sequence is:

1) Send the MEFW to the ICCU via an external function command.

2) Initiate an output buffer with a length equal to the word length of the message header plus one word (number of words not bytes).

3) Accept an external interrupt and accompanying EISW upon completion of data transfer (channel end interrupt). The command byte of the EISW will be zero upon termination of data transfer.

4) Initiate an input buffer with a length equal to the number of words being sent multiplied by the number of characters per word plus one.

5) Send the MEFW to the ICCU via an external function command.

6) Accept input data and upon completion of data transfer accept an external interrupt and EISW (device end interrupt). The command byte of the EISW will be zero.

Due to the timing characteristics of the slave handler and the ICCU, the master should issue the MEFW for input immediately upon receipt of the first interrupt.

2.6.3   SPECIAL FUNCTIONS

For convenience in performing I/O operations, special functions are provided as specified in the last message header sent to each device. The handshake sequence sets up the slave handler to perform as follows.

1) Read one card (80 column), translate to XS-3

2) Punch one card (80 column), translate from XS-3

3) Print one line (120 characters), translate from XS-3

If an I/O function is to be performed other than specified above, a message header must be sent for that device to reset the slave handler to perform in the manner desired.

A special function code must be given in the D field of the MEFW in place of the byte count to use the special function.

Special function codes are:

Code	Device	MEFW Control Field
1	Reader	01
2	Serial Punch	10
3	Printer	10
4	Row Punch	10
5	Card Controller	10

The data transfer format for special functions must be format B.

2.6.3.1 OUTPUT DATA TRANSFER

The output of data using the special function is:

1) Send the MEFW to the ICCU via an external function command.

2) Initiate an output buffer with a length equal to the length of output data buffer plus one word (18, 30, or 36-bit words not bytes).

3) Accept an external interrupt and accompanying EISW upon completion of data transfer (channel end interrupt). The command byte of the EISW will be zero on termination of data transfer.

4) Accept an external interrupt and accompanying EISW signaling completion of the output function (device end interrupt).

After the first interrupt is received, output data may be sent to another device in the slave. Output to the same device must not be sent until the second interrupt (device end) is received for that device.

2.6.3.2 INPUT DATA TRANSFER

The input of data using the special function is:

1) Initiate an input buffer with a length equal to the number of bytes per word being sent multiplied by the number of words and rounded to the next largest word if a fractional word exists.

2) Send the MEFW to the ICCU via an external function command.

3) Accept input data and upon completion of data transfer accept an external interrupt and EISW (device end interrupt).

The same timing note for the output data transfer also applies to input data transfer.

2.6.4 MAINTENANCE DATA TURNAROUND

The maintenance function provides a means for checking the linkage between
the master and the slave. Performance of this function does not involve any
slave peripheral device.

When using the maintenance turnaround, full cycles must be used; that is,
send data to slave and retrieve data from slave. Data will be returned to
the master in the same code as sent to slave. The maintenance turnaround
sequence is:

   1) Send a message header and output buffer containing turnaround data to
      the slave. The MEFW should have a D field of 0. The Message header
      has a function code of 5.

   2) Upon completion of output send a message header to the slave for input
      of turnaround data. The D field of the MEFW = 0 and the function code
      in the message header = 5.

   3) Send a MEFW for input to return data to the master.

The output and input data transfer rules are found in Subsection 2.6.1 and
2.6.2.

2.7 ERROR NOTIFICATION

Peripheral or format errors and interface errors can occur with the slave
system. Subsection 2.4.2.2 describes the format errors. The sense byte of
the EISW presents interface errors to the master. If the slave is unable
to get the proper response from the ICCU, it comes to a stop and displays
'F3'. This indicates that the slave attempted recovery and failed. The
system should be cleared and restarted.

3. 9200/9300 / ICCU INTERFACE

3.1 INTRODUCTION

To avoid duplication one basic interface will be used for the 9200/9300 Com-
puter with the ICCU 18, 30, and 36-bit word lengths. The ratio between the
number of characters per computer word and the length of the message header,
creates a unique program for the 30-bit computer family. The only modification
required between various data formats is in the number of bytes accepted in
the header that is sent from the master computer.

Since the 9200/9300 / ICCU interface is dependent upon the military computer/
ICCU interface frequent reference will be made to Section 2.

The basic interface between a 9200/9300 Computer and the ICCU shall be de-
fined as follows.

3.2 DATA FORMATS

3.2.1 ICCU DATA TRANSFER FORMATS

The transfer of data between the ICCU and the 9200/9300 Computer is always in the form of one 8-bit byte per character (regardless of the number of bits used per character). Thus, even though only data format B (Figures II-G-2, II-G-3, II-G-4) is being discussed here, the number of characters per master computer words (transferred multiple) is the only factor that requires change for a different data format.

3.2.2 HEADER FORMATS

The header formats are the same as described in Subsection 2.2.2 except for the additional consideration given to the number of characters per master computer words which changes with the change in data format.

3.3 HEADER INFORMATION

The contents of the message header (Subsection 2.3) controls the performance of the slave. A message header should be sent to each slave peripheral device to enable it to perform the required operations. For convenience, a predetermined message header (Subsection 2.6.3) is given to each peripheral handler whenever a handshake function is performed. Upon receiving the message header, the slave sets up its peripheral handlers to perform in the specified manner. The control header causes the slave to perform the required action requested by the master. The execution of the control function is done immediately upon receipt of the header.

3.4 SLAVE COMMAND WORDS

The slave command format controls all interface between the slave and the ICCU. The command is sent to the ICCU via the least significant byte of the XIOF instruction. A status byte is presented to the slave when a TIO instruction is executed. The ICCU adds sense bytes to the MEFW and presents them to the slave when the appropriate command is issued.

3.4.1 SLAVE COMMAND BYTE

Slave command bytes are:

Command	Function
XX00 0000	Test I/O – No operation, supply status to the slave.
XX01 0000	Set Inhibit Status – This command is processed as a Test I/O. (If accepted,

Command	Function
	it does not generate new status.) The status byte is presented to the channel and Inhibit Status In (bit 6 of sense byte) is set.
XX10 0000	Reset Inhibit Status - Same action as set inhibit status, except bit 6 of sense byte is reset.
XXXX 0100	Sense - Transmit 4 bytes of sense data to the slave. The second and third bytes are taken from the word register (bits 0-7 and 8-15). This provides a means of transmitting 16 bits of the MEFW (Subsection 2.4.1) from the master to the slave. The first byte is detail for the unit check status. The fourth byte has the same format as bits 16-9 of the EISW. (Subsection 2.4.2)
DDDD DD10 ①	Input - Transmit data to the slave.
DDDD DD01 ①	Output - Call for data from the slave.
DDDD DD11 ①	Set External Interrupt Request - No data transfer. Send external interrupt request to the master, with the 8-bit command code on master input data lines 0-7.

2.4.2 SLAVE STATUS BYTE

Slave status bytes are as follows.

Bit	Designation	Interpretion
0	Attention	The master has issued a control command to the ICCU, or the master has issued the first command and a slave command is required to initiate data transfer.
		When the attention interrupt has been generated, the slave must issue a sense command in order to examine the MEFW. Hardware restrictions will cause all subsequent commands to be rejected if the sense command is not issued.

① X and D bits are not interrupted by the ICCU. D bits are passed on to the master via an EISW (Subsection 2.4.2).

Bit	Designation	Interpretation
1	Status Modifier	Used only on control unit busy sequence.
2	Not used	
3	Busy	Indicates the control unit cannot accept a command because: 1) It is executing a previously initiated I/O operation; status modifier also is set. The control unit is defined as busy from the time a command from the channel is loaded into the command register until both Channel End and Device End are set. 2) The control unit is holding pending status conditions detected subsequent to completion of the last data transfer command. (Not applicable to Test I/O or Set Reset Inhibit Status.)
4	Channel End	Always occurs with Device End.
5	Device End	A data transfer was terminated or a control command was accepted. (Control commands are control immediate types.)
6	Unit Check	Set simultaneously with bits 0-5 in sense byte 1.
7	Unit Exception	Set whenever an external function is received from the master during a data transfer; that is, master termination.

3.4.3 SENSE BYTE FORMATS

Four sense bytes are presented to the slave channel. Sense bytes 2 and 3 contain the MEFW from the master (Subsection 2.4.1).

3.4.3.1 SENSE BYTE 1

Sense byte 1 contains the following.

Bit	Designation	Interpretation
0	Command Reject	Unspecified command issued to ICCU. This indication (to slave) is suppressed if command byte has incorrect parity.
1	Not used	Transmitted as zero.

Bit	Designation	Interpretation
2	Bus Out Check	Even parity on bus out during transfer of data or command to ICCU from slave channel.
3	Not used	Transmitted as zero.
4	Not used	Transmitted as zero.
5	Not used	Transmitted as zero.
6	Inhibit Status in FF state	Number 1 whenever the FF is set.
7	Not used	Transmitted as zero.

3.4.3.2   SENSE BYTE 4

Sense byte 4 contains the following.

Bit	Designation	Interpretation
0	Selective Reset	An operation has been terminated by the slave with selective reset (error detected by slave channel).
1	Master Termination	An operation has been terminated by the master before normal completion. (Will always be a 0 to a sense command, status FF 7 is set by same condition.)
2	Not used	Always transmitted as zero.
3	Bus Parity Error	A parity error has been detected on the slave output bus. (Will always be a 0 to a sense command, bit 2 of first sense byte is set by the same condition.)
4	Format Register FF 2	Same as bit 3 of last MEFW. If the master is off-line, then same as bit 0 of slave sense command byte (Subsection 3.4.1).
5	Format Register FF 1	Same as bit 2 of last MEFW. If the master is off-line, then same as bit 1 of slave sense command byte (Subsection 3.4.1).
6	Master Function Register FF 2	Same as bit 1 of last MEFW.
7	Master Function Register FF 1	Same as bit 0 of last MEFW.

3.5 INITIATION SEQUENCE

The slave will not accept any data transfers until a handshake function has been issued by the master. (Subsection 2.5) The handshake establishes the interface and allows the slave to set up its peripherals to operate in a pre-determined format (Subsection 2.6.3).

The response to the MEFW is performed as follows.

1) Accept interrupt from ICCU looking for attention bit in status word.

2) Load ICCU Buffer Control Word (BCW) to accept four sense bytes from ICCU.

3) Send sense command to ICCU.

4) Accept interrupt upon termination of sense byte input.

5) Perform handshake set up function.

6) Send an external interrupt request to the master with a D code of 0.

3.6 DATA TRANSFER SEQUENCES

All data transfer termination sequences shall originate in the slave upon termination of the slave BCW. The interface for I/O and special functions is defined below.

3.6.1 INPUT DATA TRANSFER

Any input data to the slave is output data from the master (Subsection 2.6.1) and will contain a message header followed by the data, if so selected, in the message header. The input data transfer sequence is.

1) Accept interrupt from ICCU looking for attention bit in status word.

2) Load ICCU BCW to accept four sense bytes from ICCU.

3) Send sense command to ICCU.

4) Accept interrupt upon termination of sense byte input.

5) Load ICCU BCW to accept number of bytes in D field of MEFW.

6) Send XIOF instruction in order to input data to the slave.

7) Accept an external interrupt upon completion of transfer.

8) Set up peripheral according to message header and do the desired function if data present.

9) Send an external interrupt request to master with a D code equal to the device address of the peripheral used.

3.6.2 OUTPUT DATA TRANSFER

Any output data from the slave is input data to the master (Subsection 2.6.2) and will contain a message header. The message header is transferred into the slave which performs the necessary functions and sends the output to the master. The output data transfer sequence is:

1) Accept interrupt from ICCU looking for attention bit in status word.

2) Load ICCU BCW to accept four sense bytes from ICCU.

3) Send sense command to ICCU.

4) Accept interrupt upon termination of sense byte input.

5) Load ICCU BCW to accept number of bytes in D field of MEFW.

6) Send XIOF instruction in order to input data to the slave.

7) Accept on external interrupt upon completion of transfer.

8) Set up peripheral according to message header and

    a) if data bit set, perform the desired function.
    b) if data bit not set, send an external interrupt request with a D code equal to the device address of set up peripheral. This completes this transfer sequence.

9) Accept interrupt from ICCU looking for attention bit in status word.

10) Load ICCU BCW to accept four sense bytes from ICCU.

11) Send sense command to ICCU.

12) Accept interrupt upon termination of sense byte input.

13) Load ICCU BCW in order to output data from the slave.

14) Send XIOF in order to output data from the slave.

15) Accept an external interrupt upon completion of transfer from the ICCU.

3.6.3 SPECIAL FUNCTIONS

Special functions streamline the data transfer operations by avoiding message headers. The device address of the peripheral to be used is sent in the D field of the MEFW. (Subsection 2.6.3) The slave performs as before, sending or receiving data.

3.6.3.1 INPUT DATA TRANSFER

The input of data to the slave using the special function is:

1) Accept interrupt from ICCU looking for attention bit in status word.

2) Load ICCU BCW to accept four sense bytes from ICCU.

3) Send sense command to ICCU.

4) Accept interrupt upon termination of sense byte input.

5) Load ICCU BCW with pre-determined buffer length for input to the slave.

6) Send XIOF instruction which calls for input data.

7) Accept external interrupt which signals the end of data transfer.

8) Perform the desired output function.

9) Send an external interrupt request with address of device used in D field.

3.6.3.2 OUTPUT DATA TRANSFER

The output of data from the slave using the special function is:

1) Accept interrupt from ICCU looking for attention bit in status word.

2) Load ICCU BCW to accept four sense bytes from ICCU.

3) Send sense command to ICCU.

4) Accept interrupt upon termination of sense byte input.

5) Perform the desired input from the slave peripheral.

6) Load the ICCU BCW with pre-determined buffer length in order to output data from the slave to the master.

7) Send XIOF instruction in order to output data from the slave

8) Accept an external interrupt upon completion of data transfer.

3.6.4 MAINTENANCE DATA TURNAROUND

Maintenance data turnaround uses the same sequences as I/O data transfers. However, the slave peripheral reference is omitted. (Subsections 3.6.1 and 3.6.2)

3.7 ERROR NOTIFICATION

The slave will notify the master via an external interrupt request of errors it detects according to the error codes found in Subsection 2.4.2.1. Error stops will also be displayed on the slave console according to the standard 9200/9300 peripheral error codes listed in Subsection 4.

Recovery procedures given in Subsection 4 should be followed when these error stops occur. The error stop and display of 'F3' means an unrecoverable ICCU error has occured and a master clear, restart procedure must be performed.

4. 9200/9300 OPERATING PROCEDURES

The operating instructions and program haulting information are presented for programmer reference. Complete information on the operation of the 9200/9300 Computer can be found in the <u>UNIVAC 9200 Systems, Preliminary Operating Instructions,</u> UP-7537, Revision 1. Figure II-G-8 shows the 9200/9300 Control Console. The bulk of the operating procedures discussed in this section are performed at this console.

4.1 NEXT INSTRUCTION/HALT DISPLAY (White indicators)

Figure II-G-8 shows the NEXT INSTRUCTION/HALT DISPLAY Indicators. When a halt or stop occurs, a code related to the particular halt or stop is displayed on the four groups of indicators. The various halt and stop codes are given in a following section together with the action to be taken by the operator in each case.

The indicators function individually as follows:

1) Lit - Represents a binary 1 in the related bit position.

2) Unlit - Represents a binary 0 in the related bit position.

The four bit positions (8, 4, 2, and 1) are represented by the four indicators, from left to right, within each group. The least significant group of indicators is at the right.

The significance of this display and the interpretation of the indicators at any one time depends upon the role they are called upon to play at that time.

For the operator, the primary concern is the use of these indicators to display the code related to a particular halt or stop in the processing. These codes are displayed in hexadecimal form.

4.2 INITIALIZING PROCEDURES

The first step in an operating procedure is to prepare the peripheral units to be used in the run for an operating (ready) condition.
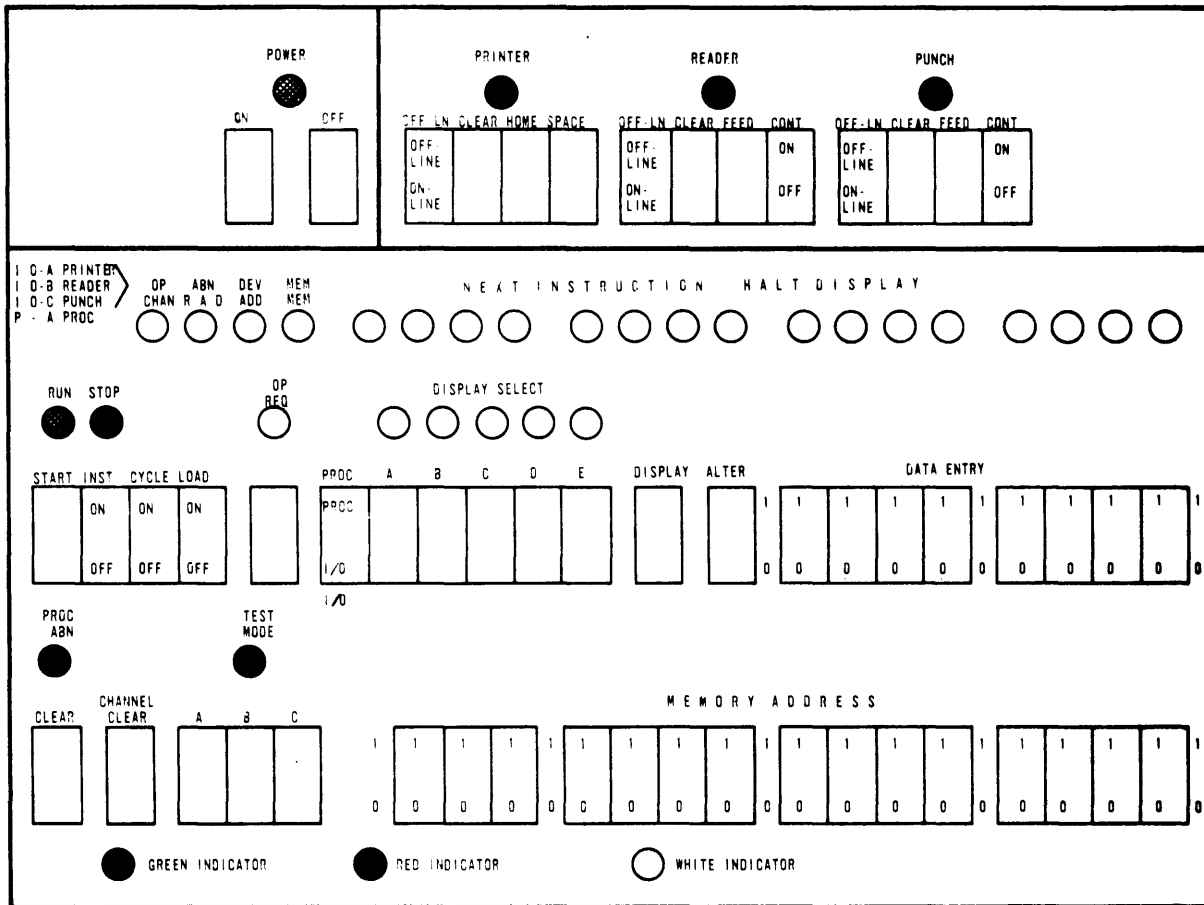
Figure II-G-8. 9200/9300 Control Console

4.2.1 POWER

Turn on the system operating power.

4.2.2 PRINTER

With no continuous form paper in the printer, the following procedure is followed.

1) Install the proper form control tape loop.

    Check the setting of the optional Line Spacing switch, if provided.

2) Check the type bar. Change the bar, if necessary.

    Be certain that the Character Font switch setting is in agreement with the kind of type bar (48 or 63) to be used.

3) Install the proper continuous form paper.

    If the proper paper is already installed, follow the appropriate elements of this procedure.

4) Set the PRINTER ON-LINE switch.

5) Depress the PRINTER CLEAR switch.

4.2.3 CARD READER

With the input magazine empty of cards, the following procedure is followed.

1) Set the READER ON-LINE switch.

2) Press the READER FEED switch to clear the ready station of any card that may have remained from a previous run.

3) Place the program cards for the run in the input magazine.

    The first reader data cards can be placed on top of the program cards if the capacity of the magazine will permit.

4) Press the READER CLEAR switch.

5) Press the READER FEED once to introduce the first program card into the ready station.

4.2.4 CARD PUNCH

With the input magazine empty the following procedure is followed.

1) Set the PUNCH ON-LINE switch.

2) Press the PUNCH FEED twice to clear the wait and ready stations of any cards that may have remained from a previous run.

The red PUNCH Abnormal indicator will light following each of these FEED switch depressions. In each case, press the PUNCH CLEAR switch.

3) Place the initial supply of cards in the input magazine.

NOTE: If card reading is to be performed, a blank card is placed before the first card to be read.

4) Press the PUNCH FEED twice to feed cards to the wait and ready stations. The PUNCH CLEAR is pressed between the two depressions of the FEED switch to clear the PUNCH Abnormal indicator.

NOTE: If the punch is left loaded with cards when the system operating power is turned off, feed at least one card after turning the power on. A card that has been left in the wait station can be mispunched when operation is resumed.

4.3 PROGRAM LOADING

With the desired peripheral units in a ready condition, the following procedure is followed to load the program.

1) Set the device address in the DATA ENTRY switches. (Standard reader device address is 0000 0001).

2) Depress the CLEAR (Processor Clear). While holding the CLEAR switch depressed, the operator makes the lamp test of the indicators on the control console.

NOTE: This is the only time that the Processor CLEAR switch need be used until the run is completed. If it is used at any other time, except under abnormal conditions, the continuity of the program will be disturbed.

3) Set the processor in the load mode.

4) Press the START switch once to cause the reading of the first program card.

5) Set the processor in the run mode.

6) Press the START switch to initiate the reading of the balance of the program cards.

The reading of the program cards may be interrupted by one or more halt and proceed instructions. There is a possibility that there may be some action required of the operator at this time before the program loading is resumed by again pressing the START switch (See Subsection 4.5).

At the end of program loading, the system can go directly into the run automatically if the input magazine or magazines are already supplied with cards and unless a halt and proceed instruction intervenes between the program loading of the run.

NEXT INSTRUCTION/HALT DISPLAY indication of "OOEE" (0000 0000 1110 1110) means the program was loaded correctly. If a start does not occur, press START to execute the program.

4.4   RUNNING AND STOPPING

4.4.1   MANUAL STOPPING

During operation, the operator can stop the processing at any time by pressing down the upper portion of the INST switch on the control console; the RUN indicator will be on, the STOP indicator will turn on.

This is the only proper means of manually stopping the operation. With the operation stopped in this manner, it is resumed by pressing the START.

4.4.2   AUTOMATIC STOPPING

If the processing stops automatically during the run while the system operating power remains on, it is usually due to one of the following:

1)   A programmed halt. The RUN indicator will be OFF, the STOP indicator will be ON (See Subsection 4.5).

2)   An abnormal stop. The RUN indicator may be either ON or OFF, the STOP indicator will be ON (See Subsection 4.6).

4.4.3   POWER

If there is no immediate need for the equipment, press the POWER OFF switch.

If the system is not to be used for an extended period of time, such as overnight, press the POWER OFF switch. When the POWER indicator turns off, throw the main circuit breaker to OFF.

4.5   PROGRAMMED HALTING

During the loading of a program or during the running of data, the operation may stop automatically as the result of a halt and proceed instruction included in the program. At that time, a code related to the halt will be displayed on the NEXT INSTRUCTION/HALT DISPLAY indicators.

A programmed halt calls for some action to be taken by the operator to make the run proceed. Until the operator is fully acquainted with the programs, the reason for a halt and the action to be taken are furnished by a run guide.

After performing the action required, the operation is usually resumed simply by pressing the START switch.

4.6 ABNORMAL STOPPING

If at the start of a run, the operation cannot get under way or, during a run, the operation stops automatically for some reason other than a halt and proceed instruction, the reason for the abnormal stop is indicated on the control console.

4.6.1 ABNORMAL STOP INDICATIONS

Two types of abnormal stop indications are provided:

1) NEXT INSTRUCTION/HALT DISPLAY - These indicators are used to display the nature of an abnormal stop in code form. These codes are generated to be displayed when required by various routines (software) such as the Report Program Generator (RPG) and the Input Output Control System (IOCS).

2) DISPLAY SELECT - The source of the abnormal stop indications displayed here is directly from the equipment (hardware).

NOTE: It is quite possible for two or more abnormal conditions to occur at the same time resulting in the display of a peculiar code in the NEXT INSTRUCTION/HALT DISPLAY. This would be a display of the combination of the codes, for example:

Code (6304) - HALT DISPLAY (0110 0011 0000 0100)
Code (6320) - HALT DISPLAY (0110 0011 0010 0000)
Code (6340) - HALT DISPLAY (0110 0011 0100 0000)
      Combined Display 0110 0011 0110 0100

The 1-bit indications in the right-hand groups can give the clue to the individual abnormal indications. Use of the DISPLAY SELECT to analyze the abnormalties may also be helpful at this time.

In addition to these two types of indications, the operator should be aware of the functioning of the following.

1) The green POWER indicator to show that main power is being supplied to the system.

2) The green and red RUN and STOP indicators to show that the system is running or how it was stopped.

3) The read abnormal indicators; PRINTER, READER, PUNCH, PROC ABN (Processor Abnormal), and TEST MODE to show whether the related element of the system is functional.

4.6.2 ABNORMAL CONDITIONS

The following subsection is a listing of the various abnormal conditions which includes:

1) Description of the abnormal condition.

2) The software code applied to the condition. For the peripheral units, the first two characters signify the particular unit as follows:

    61 (0110 0001) – Card Reader
    62 (0110 0010) – Card Punch
    63 (0110 0011) – Printer

3) The indications displayed by both the NEXT INSTRUCTION/HALT DISPLAY and the DISPLAY SELECT. Some abnormal conditions are signalled by one type and not the other.

4) The reason for the abnormal condition.

5) The action to be taken to resume operation.

4.6.2.1 PRINTER

Description – Out-of-forms

Code – 6301

Indications – HALT DISPLAY (0110 0011 0000 0001)
              DISPLAY SELECT -- I/O A – OP

Reason – The forms supply is either exhausted or torn.

Action – Install a new supply of forms.
         Press the PRINTER CLEAR.
         Press the START.

Description – Form Overflow Set

Code – 6302

Indications – HALT DISPLAY (0110 0011 0000 0010)
              DISPLAY SELECT -- I/O A – DEV

Reason – This is not an error condition. The form overflow code 1 (001) has been sensed in the form control tape and, before the form overflow operation is completed, some abnormal condition, usually Out-of-Forms, takes place. Thus, this HALT DISPLAY will appear in conjunction with the display of the abnormal condition.

Action - When the abnormal condition causing the stop is corrected, the operation is resumed according to the nature of the abnormal condition.

Description - Interrupt Pending

Code - 6304

Indications - HALT DISPLAY (0110 0011 0000 0100)
          DISPLAY SELECT -- None

Reason - A program interrupt related to the printer has been set pending action by the program. This is not an error condition. Some abnormal condition has occurred at this time to cause the operation to stop. Thus, this HALT DISPLAY will appear in conjunction with the display of the abnormal condition.

Action - When the abnormal condition causing the stop is corrected and operation resumed, the interrupt should be satisfied quickly by the program without further operator intervention.

Description - Wrong Character Font Switch Setting

Code - 6308

Indications - HALT DISPLAY (0110 0011 0000 1000)
          DISPLAY SELECT -- I/O A - OP

Reason - The setting of the Character Font switch does not agree with the kind of type bar installed.

Action - Correct the Character Font switch setting or install the proper bar. Because this indication occurs at the time of the first printing operation, it may be necessary to restart the run.

Description - Memory Overload

Code - 6320

Indications - HALT DISPLAY (0110 0011 0010 0000)
          DISPLAY SELECT -- I/O A - MEM

Reason - At the time the printer is ready to print a line, the data for that line has been entered into the print storage area of memory but the printer has not obtained sufficient access time to perform the operation.

Action - This should occur very rarely with proper programming. If it does:
          Press the PRINTER CLEAR.
          Press the START.

Description - Paper Runaway

Code - 6340

Indications - HALT DISPLAY (0110 0011 0100 0000)
          DISPLAY SELECT -- I/O A - DEV

Reason - This can include:
        At the start of a run when the HOME switch is pressed, the tape may
        not be perforated with the home position code.
        During the run, the program may have called for a skip code not per-
        forated in the tape.
        The form control tape lamp may have burned out.

Action - Correct the cause of the abnormal condition in either the tape or the
        program. Install a new lamp if necessary. Reposition the form to
        the terminal line related to the skip code causing the runaway.
        Press the PRINTER CLEAR.
        Press the START.

Description - Not Ready

Code - 6380

Indications - HALT DISPLAY (0110 0011 1000 0000)
          DISPLAY SELECT -- I/O A - ABN

Reason - The printer is not ready for operation. This can include:
        The shield at the left end of the type bar path is open.
        The cover over the right end of the type bar path is open.
        The internal power to the printer may be off.

Action - Correct the abnormal condition.
        Press the PRINTER CLEAR.
        Press the START.

Description - Off-Line

Indications - DISPLAY SELECT -- A/O A -- OP

Reason - The program has called for the use of the printer while the PRINTER
        switch is set OFF-LINE.

Action - Set the PRINTER ON-LINE.
        Press the PRINTER CLEAR.
        Press the START.

Description - Control or Data Parity Error

Indication - DISPLAY SELECT -- I/O A - MEM

Reason - A parity error has occurred in either a control byte governing the operation of the printer or in a data byte.

Action - Consult the supervisor. It will first be necessary to determine if the parity error was in a control or a data byte. It will also be necessary to determine whether this failure is only momentary or requires field engineering service.

4.6.2.2 CARD READER

Description - Off-Line

Code - 6100

Indications - HALT DISPLAY (0110 0001 0000 0000)
              DISPLAY SELECT -- None

Reason - The program has called for the use of the reader while the READER switch is set OFF-LINE.

Action - Set the READER ON-LINE.
         Press the READER CLEAR.
         Press the START.

Description - Interrupt Pending

Code - 6104

Indications - HALT DISPLAY (0110 0001 0000 0100)
              DISPLAY SELECT -- None

Reasons - A program interrupt related to the reader has been set pending action by the program. This is not an error condition. Some abnormal condition has occurred to cause the operation to stop. Thus, this HALT DISPLAY will appear in conjunction with the display of the abnormal condition.

Action - When the abnormal condition causing the stop is corrected and operation resumed, the interrupt should be satisfied quickly by the program without further operator intervention.

Description - Empty Magazine or Full Stacker

Code - 6140

Indications - HALT DISPLAY (0110 0001 0100 0000)
              DISPLAY SELECT -- I/O B - OP

Reason - This can include:
         The input magazine is empty.
         The output stacker is full.

Action - If the magazine is empty, refill it. If the stacker is full, empty it.
         Press the READER CLEAR.
         Press the START.

Description - Magazine Jam or Interlock Open

Code - 6140

Indications - HALT DISPLAY (0110 0001 0100 0000)
              DISPLAY SELECT -- I/O B - ABN

Reason - This can include:
         A card in the magazine has jammed or otherwise failed to feed to the
         ready station.
         The top cover of the reader is open or the internal power to the
         reader is shut off.

Action - If a card has jammed in the magazine; remove the cards, examine
         the card at the bottom of the stack, repair or remake it if necessary,
         and return it to its position at the bottom of the stack.

         Open the top cover to be sure that there is no card in the ready sta-
         tion. If there is, be sure that the card is not damaged. A damaged
         card here should be repaired or remade and returned to its position at
         the bottom of the stack.

         Place the stack of cards in the magazine.
         Press the READER CLEAR.
         Press the READER FEED.
         Press the START.

         If the top cover is open, close it.

         Press the READER CLEAR.
         Press the START.

         If there is no card jam and the top cover is closed, the internal power
         to the reader may be off. Consult the supervisor.

Description - Read Station or Card Stacker Jam

Code - 6180

Indications - HALT DISPLAY (0110 0001 1000 0000)
              DISPLAY SELECT -- I/O B - DEV

Reason - This can include:
         Failure of a card to be fed properly to or through the read station.
         Failure of a card to be fed properly to the card stacker.

Action – If a card has jammed at the read station, clear the jam and repair
or remake the card. Remove the card from the ready station. Take
the cards from the magazine, place the two cards removed from inside
the reader in their proper sequence at the bottom of the stack.

Place the stack of cards in the magazine.
Press the READER CLEAR.
Press READER FEED.
Press the START.

If a card has jammed in the card transport, clear the jam and repair
or remake the card. Place that card in its proper sequence at the
rear of the stacker.

Press the READER CLEAR.
Press the START.

If there is no card jam, the last card fed from the ready station and
now in the card stacker may not have been read. This can indicate
a failure in the reading mechanism. Consult the supervisor.

Such a reading failure may be momentary or may require field engineering
service. After clearing all cards from the reader, make a second
attempt at reading that card in order to determine if the failure was
only mementary.

Description – Control Parity Error

Indication – DISPLAY SELECT -- I/O B – MEM

Reason – A parity error has occurred in a control byte governing the operation
of the card reader.

Action – Consult the supervisor. It will be necessary to determine if this
failure is only momentary or requires field engineering service.

4.6.2.3 CARD PUNCH

Description – Off-Line

Code – 6200

Indications – HALT DISPLAY (0110 0010 0000 0000)
DISPLAY SELECT -- I/O C – OP

Reason – The program has called for the use of the punch while the PUNCH switch
is set OFF-LINE.

Action – Set the PUNCH ON-LINE.
Press the PUNCH CLEAR.
Press the START.

Description - Empty Magazine or Full Stacker

Code - 6202

Indications - HALT DISPLAY (0110 0010 0000 0010)
            DISPLAY SELECT -- I/O C - OP

Reason - This can include:
            The input magazine is empty.
            A stacker is full.

Action - If the magazine is empty, refill it. If a stacker is full, empty it.
            Press the PUNCH CLEAR.
            Press the START.

Description - Interrupt Pending

Code - 6204

Indications - HALT DISPLAY (0110 0010 0000 0100)
            DISPLAY SELECT -- None

Reason - A program interrupt related to the punch has been set pending action
            by the program. This is not an error condition. Some abnormal
            condition has occurred to cause the operation to stop. Thus, this
            HALT DISPLAY will appear in conjunction with the display of the
            abnormal condition.

Action - When the abnormal condition causing the stop is corrected and operation
            resumed, the interrupt should be satisfied quickly by the program
            without further operator intervention.

Description - Punch Check Error

Code - 6220

Indications - HALT DISPLAY (0110 0010 0010 0000)
            PROGRAM SELECT -- I/O C - DEV

Reason - The last card fed to the error stacker has not passed the punch check.

Action - Because the setup for the last card punched (error card) is still
            available, another card punched from that same setup can be obtained
            by pressing the PUNCH CLEAR and the START. This card will also be
            subject to the punch check.

            The punching may pass the check on the second attempt. If it does,
            the run will continue.

If another punch check error occurs, it might be advisable to examine the contents of the punch storage area if the punching error is not immediately apparent. Consult the supervisor. There is the possibility of failure in the punching mechanism requiring field engineering service.

Description - Magazine or Card Transport Jam, Interlock Open

Code - 6260

Indications - HALT DISPLAY (0110 0010 1000 0000)
              DISPLAY SELECT -- I/O C - ABN

Reason - This can include:
    A card has failed to feed from the magazine to the ready station usually because of a jam in the throat of the magazine.
    A card has failed to feed from the ready station to the wait station usually because of a jam at the read station.
    A card has failed to feed properly through the punch station or to the card stackers.
    The top-front cover of the punch is open or the internal power to the punch is shut off.

Action - When card reading is involved in the punch operation, it is best to have the advice of the supervisor when handling card jams. The proper procedure may depend upon the nature of the program.

When the application involves punching only into blank cards, the following is suggested. It is based on the premise that, unless all of a card has passed into the stacker area beyond the punch station, the punching setup for that card remains.

Thus, recovery from a jam in the magazine, ready station, read station, wait station, or punch station can be made by using this procedure.

Remove the blank cards from the magazine. Discard any damaged cards at the bottom of the stack.
Clear all cards from the card transport; damaged or undamaged.
Place a supply of cards in the magazine.
Press the PUNCH CLEAR and then the PUNCH FEED to advance the first card to the ready station.
Again press the PUNCH CLEAR and the PUNCH FEED to advance the first card to the wait station.
Press the START.
A jammed card in a card stacker is remade and placed in its proper sequence in the file of punched cards.

Description - Control or Data Parity Error

Indication - DISPLAY SELECT -- I/O C MEM

Reason - A parity error has occurred in either a control byte governing the operation of the punch or in a data byte.

Action - Consult the supervisor. It will be necessary to determine if the parity error was in a control or data byte. It will also be necessary to determine whether this failure is only momentary or requires field engineering service.

4.6.2.4   PROCESSOR

Description - General Purpose Channel Error

Indication - DISPLAY SELECT -- PROC A - CHAN

Reason - An abnormal condition exists in a peripheral unit connected to the processor through the general purpose channel.

Action - Consult the supervisor to locate the source of the error.

Description - Address Error

Indication - DISPLAY SELECT -- PROC A - ADD

Reason - An attempt has been made to enter a restricted address location or to address a location beyond the capacity of the memory of the system being used.

Action - Consult the supervisor to locate the source of the error.

Description - Parity Error or Divide Error (Hardware)

Indication - DISPLAY SELECT -- PROC A - MEM

Reason - A parity error has occurred in the processor or, if the system is equipped with the optional multiply, divide, and edit feature, an error has occurred while performing division.

Action - Consult the supervisor to determine the nature of the error and whether it is momentary or requires field engineering service.

SECTION III. ASSEMBLY SYSTEMS

The TRIM family consists of three operational assemblers. The user can select the assembler which best fits his equipment configuration, thus getting the maximum use of the computer.

1. TRIM I

TRIM I is a simple assembler which operates with a minimum of equipment, requiring only a computer with a paper tape reader-punch unit. The assembler translates monocode (one-to-one) mnemonic operations into machine code instructions with appropriate address allocation.

In operation, TRIM I makes two passes on the source program tapes. The first pass stores the labels from the source program to allow forward references. Those labels and indicators giving the relative position in the program are stored and retained for the second pass. The second pass makes the actual assembly of machine instructions and allocates the addresses. The source program size is limited only by the number of labels used. TRIM I requires a minimum of 4000_{10} words of core memory.

2. TRIM II

TRIM II is an assembler which operates on a computer with a paper tape reader-punch unit and a console typewriter. In addition to the monocode (one-to-one) mnemonics of TRIM I, it also accepts polycode (one-to-many) mnemonic operations in the source program. The source language also has debugging aids which cause dumps of registers and memory contents wherever desired by the programmer. The assembler can be instructed to ignore debugging operations if desired. TRIM II requires a minimum of 8000_{10} words of core memory.

3. TRIM III

TRIM III is an assembler which operates on a computer with a magnetic tape unit, paper tape reader-punch unit, a console typewriter, a card processor, and a high speed printer.

This assember has a source language librarian for aiding the programmer in selecting subroutines for incorporation into the program during the assembly process. The programmer uses call operations in his source program to implement retrieval from the source library.

The source programming language includes the operations of TRIM I and TRIM II. Operations which aid debugging in this language cause generations that present diagnostic information to the programmer during a run. This works with the TRIM debugging package (DEBUG) discussed later. In addition TRIM III may be controlled via control operations.

In operation, TRIM III makes only one pass on the source program input. Subroutines are retrieved from the magnetic tape source library and added to the end of the source program. Assembled programs can then be written on magnetic tape, cards, or paper tape. Diagnostic errors are typed on the console type-

writer. These features cut TRIM III assembly time to a minimum.

The assembler possesses source language correction capability in conjunction with an assembly run.

TRIM III requires a minimum of 16000_{10} words of core memory.

SECTION III-A.   TRIM I ASSEMBLY SYSTEM

1.   BASIC INFORMATION

TRIM 1 is a simple assembly system which converts a source program tape written with symbolic addressing into an object program tape with absolute addressing suitable for loading into the computers via the utility packages.

2.   SYMBOLIC ADDRESSING

2.1   LABELS

In an absolute-addressed program every word is assigned an absolute address during the coding process. In the TRIM I system only those words which are referred to by instructions require an address, although the address is symbolic rather than absolute. The term label is used rather than address since it more accurately describes the function of the symbolic address. A label may never be incremented or decremented. Words which are not referred to need not be labeled. Unlabeled words following one another on the source program tape are ultimately assigned to consecutive memory addresses. For any given assembly run each label used must be unique.

2.2   TAGS

Instructions are written with mnemonic or octal notation for the function and/ or subfunction codes. The u or k portion of an instruction word may be either a constant, an octal notation, or a symbolic notation (alphanumeric) referring to a constant (either an absolute address or an item of data).

Whenever any instruction refers to a label, the u portion of that instruction is called a tag. The tag must be identical with the label to which it refers but may be followed by a + or - and an octal integer to provide for increments or decrements. Thus an instruction may refer to an unlabeled instruction in terms of its sequential position preceding or following a labeled instruction.

Symbolic addressing simplifies the task of coding. In addition it permits the transfer of programs to any region of memory since the symbolic addressed program is independent of memory locations.

3. INPUT LANGUAGE FORMAT

The programmer uses a uniform set of symbols as separators in all coding (refer to Table III-A-1).

TABLE III-A-1. TRIM I CODING SYMBOLS

Symbol	Coding Significance
⟶	Delimits the statement. Must always precede the statement operator. Must precede notes; omit if notes are not given.
↵	Signifies the end of an operation. Must precede header operations. Must precede end-of-tape double period symbols.
•	Separates statement components.
+	Specifies an integer increment to follow.
-	Specifies an integer decrement to follow.
..	Specifies end-of tape read-in. Must terminate every input tape.

The input language as prepared by the programmer consists of a list of operations which perform the step-by-step processing of a problem. An operation has the following configuration:

$$\underset{\text{[label]}}{L} \longrightarrow \underset{\text{[statement]}}{S} \longrightarrow \underset{\text{[notes]}↵}{N}$$
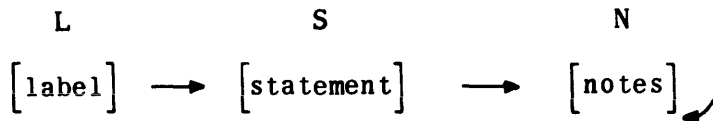
L   -   The label is a name that uniquely identifies the operation. It consists of up to six alphanumeric characters but never starts with O, a number, or LOK. The first operation of each program or subroutine must have a label. Otherwise only an operation referred to by a tag in the statement of another operation requires a label. Tags have the same notation restrictions as labels except that tags may be incremented by ± an octal integer. Each label of an assembly run must be unique. Conversely, any number of tags may refer to a given label.

S   -   The statement defines the operation and is always required.

N   -   Descriptive notes may follow the statement; they are for the pro-
grammer's use and in no way affect the meaning of the operation.
Notes should not exceed 40 characters.

→   The straight arrow is a major separator which delimits the statement.

↲   The curved arrow designates the end of each operation and signals the
start of the next one. It **must** precede the first operation of any
program.

The operation statement may be in one of the following formats:

3.1 FORMAT A

$$\overset{\textstyle W}{\longrightarrow \left[\text{operator}\right]} \quad \bullet \quad \overset{\textstyle V_0}{\left[\text{operand}\right] \longrightarrow}$$

W   -   The operator is the f or fm portion of the operation statement and is
the mnemonic representation of the desired function code of the com-
puter instruction repertoire.

V_0 -   Represents the u or k portion of the statement and may be a tag, a tag
$\pm$ an integer, or an integer only. Integers must be in octal repre-
sentation. Incrementing or decrementing of integers is not permitted.
If V_0 is absent, TRIM I generates zeros for the operand without any
error indication.

Examples:

→ ENTAL•CAT ↲

→ STRADR•CAT +1 ↲

→ CMAL•CAT-4 ↲

→ ENTBK•27 ↲

→ ENTALK•7776 →   Minus 1 to AL ↲

→ STOP•DOG →   DOG defined by an EQUALS opn ↲

→ SKPOIN•7 ↲   Results in 502207

→ CPAU ↲   Results in 506200

→ JP•LOK-10 →   LOK signifies THIS address ↲

→ OUT•6 →   Output transfer channel 6 ↲

→ 0•CHEESE →   Buffer terminal address ↲

——→ 0•CHEESE ——→   Buffer initial address

——→ ENTAL   Results in 120000

——→ JP•   Results in 340000

3.2 FORMAT B

TRIM I also accepts programs coded with absolute function codes and absolute or
symbolic addressing. Normal instructions are represented by a 2-digit function
code followed by a point separator and the desired u or mk. However, absolute
instructions may also be represented by six consecutive digits without a point
separator.

Examples:

——→ 12•3505 ——→

——→ 63•CAT+6

——→ 50•1306 ——→

——→ 50•6200 ——→

——→ 506200 ——→

3.3 FORMAT C

Constants may be represented in a number of ways.

——→ 7 ——→   Results in 000007

——→ 7•0 ——→   Results in 700000

——→ 77 ——→   Results in 000077

——→ 77•0 ——→   Results in 770000

——→ 777 ——→   Results in 000777

——→ 77070 ——→   Results in 077070

——→ 777•7 ——→   Illegal*

——→ 777• ——→   Results in 000777*

——→ 123456 ——→   Results in 123456

---

*Whenever there is an expressed value following the point separator, only one
or two digits are permitted in the operator position.

4. SPECIAL OPERATORS

Each program to be assembled by TRIM 1 requires an initial header operation for identification purposes. A header operation consists of the program name in the L coding position, a header operator in the W coding position, and two identifying operands, V_0 and V_1, in which the programmer specifies his name or initials and the date of program preparation respectively.

```
      L              W    V₀    V₁           N
  ↙ POKER   ──►   PROG•SMITH•JAN68        ──►
```

A carriage return must always precede a header operation.

In addition to the special PROG header operator, TRIM I provides for five additional special operators useful to the programmer.

1) The SETADR operation informs TRIM I that the next instruction is to be assigned to the address specified by the V operand. Addressing will follow sequentially until interrupted by a subsequent SETADR operation.

```
      L                W                  V                   N

  ┌optional┐ ──► SETADR  •  ┌absolute address┐ ──►
  └label   ┘               └in octal        ┘
```

2) The REMARK operation is for the programmer's use to amplify normal explanatory notes and in no way affects the generated programs. The operation may not exceed one line in length.

```
      L                W                  V

  ┌optional┐ ──► REMARK  •  ┌desired statement up to┐ ↙
  └label   ┘               └one line in length      ┘
```

3) The EQUALS operation permits the programmer to assign absolute values to symbolic tags in his program which do not have corresponding labels. The EQUALS operation may never be used like a SETADR operation, but pertains only to constants or addresses outside the program, such as fixed memory locations, key settings, channel designations, or references to addresses of program labels other than those of the program currently being assembled.

```
      L              W          V           N

  LABEL   ──►   EQUALS  •  ┌absolute value┐ ──►
                          └in octal       ┘
```

Examples:

CAT ⟶ EQUALS • 4 ⟶

DOG ⟶ EQUALS • 36 ⟶

COW ⟶ EQUALS • 7776 ⟶

Examples of use of the above:

⟶ STOP • CAT

⟶ LSHAL • CAT

⟶ OUT • CAT

⟶ ENTAL • DOG

⟶ STRADR • DOG

⟶ ENTALK • COW

⟶ ENTBK • COW

4) The DBLSET operation frees the programmer from the responsibility for insuring that the Y of a double add or subtract instruction be located at an even address. The DBLSET operation is normally followed by a Y constant to which it refers. TRIM I examines the address to which the constant (or instruction) would normally be assigned. If the address is odd, a word of zeros is first generated to insure that the constant (or instruction) will be assigned to an even address. If the address is even, no generation results.

L	W	N
[optional label]	⟶ DBLSET	⟶
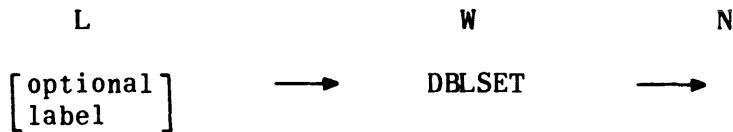
5) The RESERV operation causes the desired number of sequential words to be reserved within a program. The operation generates as many zero words as specified by V. The V operand may never be zero.

L	W	V	N
[optional label]	⟶ RESERV •	[no. of words to be reserved in octal]	⟶
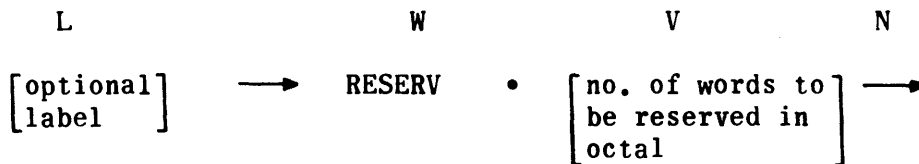
5. THE LOK TAG

If the programmer wishes to reference unlabeled instructions in his program, he may do so in terms of a specific instruction by means of the LOK tag plus or minus an octal value. LOK always refers to the instruction in which it appears. For example if the instruction JP·LOK-3 appears at address 04503, the resulting generation will be 34 4500. Thus, the instruction falling at address 04500 need not have been labeled. No valid program label may begin with the letters LOK. Reasonable care should be taken in the use of the LOK tag since corrections to the original program may affect the LOK references.

6. INPUT TAPE FORMAT

TRIM I is available in two versions; one version accepts a source program prepared in field data code; the other version accepts a source program prepared in ASCII code (Refer to Appendix A, Tables A-1, A-2, and A-3).

The term source code refers to the code in which the input tapes are prepared.

Input to TRIM I is via punched paper tape. The source program tape must begin with a carriage return and terminate with a carriage return and two periods.

7. TRIM I OUTPUTS

TRIM I provides four optional punched paper tape outputs of the assembled program. All the outputs are loadable via the utility packages.

The available outputs are:

No. 2 - Absolute assembled program, sequential line identifier, source program, and assembly error alarms when applicable. This is a side-by-side listing in source code preceded by a program summary consisting of the number of memory locations used and inclusive addresses.

No. 3 - Absolute assembled program in source code, consisting of carriage return, 88, carriage return, addresses and instructions, a carriage return, double period, and checksum.

No. 4 - Absolute assembled program in bioctal format, consisting of a 76 code, inclusive area addresses followed by the instructions only, with a checksum.

No. 5 - Relocatable assembled program in bioctal format starts with a 75 code followed by the assembled program relative to base 00000, and terminates with a checksum. The output tape may be loaded starting at any desired memory location. One restriction is placed on No. 5 outputs; that is, if the program contains double-length add or subtract instructions and was assembled at an even address, it must

always be loaded at an even address. If the same program was assembled at an odd address, it must always be loaded at an odd address.

8. GROUND RULES

1) No TRIM I label may exceed six characters. The label must not begin with a number, the letter O, or LOK.

2) Each input program tape prepared for the TRIM I assembler must begin with a carriage return and terminate with a carriage return and two periods.

3) Each break in sequence of addressing constitutes a program area. A total of 24 such areas is permitted.

4) The SETADR operation automatically creates a new program area.

5) A RESERV operation of zero words is illegal.

6) The maximum size program which TRIM I can assemble is limited only by the number of program labels including the labels of EQUALS operations. The maximum number of labels allowed is approximately 1100_{10} for each 8000_{10} words of memory.

7) All numbers must be octal integers. Decimal numbers will cause an error.

8) TRIM I provides a limited amount of error detection and console display. All other errors will be indicated by multiples of 100 following the notes of the instruction concerned on the No. 2 output. Thus 100 indicates one error, 200 indicates two errors in this instruction, and so forth. This error display mechanism assumes that no on-line typewriter is available.

9) Keyboard correction methods for TRIM assembler: Typing-error correction procedures have been incorporated in all versions of the TRIM I, TRIM II, and TRIM III assemblers, and the TRIM corrector for deleting immediate keyboard errors that might be made in the preparation of input tapes for these same programs on the I/O console.

Since it is impossible to back up the paper tape in preparation and punch code-delete codes over the erroneous frame or frames, any typing errors must be identified by a special code or codes following the erroneous data. On the UNIVAC ® 1232 I/O console, the backspace code of the keyboard is designated as a reject code. The backspace is identified by the upward pointing arrow (↟) below the stop code (Ⓢ) on the same key on the left hand side of the keyboard. In lower case, this key punches a 77 and types the same arrow (↟) on the printer. On the UNIVAC ® 1532 I/O console the RUB OUT key is used to generate a reject code. This key punches a 177 code on tape. No symbol is printed on paper.

One reject code – a single reject code (77 or 177) anywhere on an input tape to the TRIM assemblers or corrector informs that routine that the legal code that immediately preceded the 77 or 177 should be rejected.

Example:

       ☐ ER ↑ NTE ↑ AL/ ↑ 'CAG ↑ T   (carriage return – line feed)

This statement appearing on the console printer and punched as one of the statements on a program tape will be interpreted as:

☐ ENTAL'CAT   (carriage return)
by that assembler or corrector.

Three consecutive reject codes – three consecutive rejects (77-77-77 or 177-177-177) on an assembler or corrector input tape inform that routine that the entire statement being formed should be rejected, and that processing of a new statement should not begin until a carriage return is found.

Example 1:

      ☐ ENTAL'CAT (carriage return – line feed)

MOOSE ☐ STRAL'D ↑ ↑ ↑ JP'TIGER (carriage return – line feed)

      ☐ ADDALK'63 (carriage return – line feed)

MICE   ☐ STRAL'DOG (carriage return – line feed)

These statements appearing in a program to be assembled by a TRIM assembler will be interpreted by that assembler as:

      ☐ ENTAL'CAT (carriage return)

      ☐ ADDALK'63 (carriage return)

MICE ☐ STRAL'DOG (carriage return)

Example 2:

    143'1 ↑ 2 (carriage return – line feed)

    GOOF ☐ SLSUB'JUNK ↑ ↑ ↑ (carriage return – line feed)

    RIGHT ☐ JK ↑ P'HON1 ↑ OR (carriage return – line feed)

These statements on a correction tape will be interpreted by the TRIM corrector as:

143'2   (carriage return)

RIGHT   JP'HONOR (carriage return)

NOTE: All examples are for correcting inputs to the field data version on the 1232 I/O Console.

9.   LOADING AND OPERATING PROCEDURES

Prior to loading the TRIM I assembler the computer and paper tape reader-punch unit must be placed in the operational state with all switches in the normal operating position. Since the assembler is loaded by a utility package, the utility package must be already loaded into memory at addresses which will not be occupied by the assembler.

9.1   LOADING THE ASSEMBLER

1)   Master clear the computer and the reader-punch unit.

2)   Mount the TRIM I assembler tape in the reader.

3)   Set the P register to the utility package starting address for paper tape load.

4)   Start the computer.

9.2   USING THE ASSEMBLER

1)   Pass 1.

   a)   Master clear the computer and reader-punch unit.

   b)   Set the P register to 1400.

   c)   Set PROGRAM SKIP keys 1 and 2.

   d)   If no error displays are desired during assembly, set PROGRAM SKIP key 3.

   e)   Mount a source program tape in the reader.

   f)   Start the computer.

   g)   The computer will stop after the tape has been read. Repeat steps e) and f) for each source program tape.

2) Pass 2.

    a) Release PROGRAM SKIP key 1.

    b) Set the AL register to the number of the desired output (2, 3, 4, or 5; refer to paragraph 7, TRIM I OUTPUTS).

    c) Mount a source program tape in the reader (program tapes must be loaded in exactly the same order as for pass 1).

    d) Start the computer.

    e) The assembler will assemble the input tape and punch it. Repeat steps c) and d) for each source program tape.

    f) Release PROGRAM SKIP key 2.

    g) Start the computer. The assembler will finalize the output tape with a checksum and trailer.

    h) Set PROGRAM SKIP key 2 and start with step b) to obtain additional outputs.

9.3 ERROR DETECTION AND DISPLAY

TRIM I contains limited error detection capabilities. The majority of programmer errors can be handled internally. However, since PROGRAM SKIP key settings are essential to the assembly process, assembly will always stop when these keys are improperly set. These error stops are indicated by a 1 or a 2 in the AL register.

  1) (AL) = 1   Set PROGRAM SKIP key 1.
  2) (AL) = 2   Set PROGRAM SKIP key 2.

Set the appropriate PROGRAM SKIP key and start the computer to continue assembly.

The programmer has the option of requesting that the assembler stop and display pertinent information in the A registers for basic programmer errors or requesting that the assembler handle these errors internally, thereby forcing an assembly.

If PROGRAM SKIP key 3 is set, the assembler will force the assembly. If PROGRAM SKIP key 3 is not set, the assembler will stop for the following errors:

  1) (AL) = 3; (AU) = sequential line identifier. This error stop occurs during pass 1 and means that no starting address has been given for the program via the SETADR operation. To correct:

    a) Clear AL.

    b) Set $AL_{(14-0)}$ to the desired address.

c) Start the computer.

d) If the stop and display option has not been selected, assembly will not stop and TRIM I will arbitrarily assign the program to the base address 01200.

2) (AL) = 4; (AU) = sequential line identifier. This error stop occurs during pass 1 and means that the source program contains too many program segments (24 are permitted). This error is non-recoverable. Assembly may be continued by starting the computer after each such stop; however, all addressing will be sequential from the point of overflow. The effect of this error is the same whether or not the stop option is selected.

3) (AL) = 5; (AU) = sequential line identifier. This error stop occurs during pass 2 and indicates an unallocated tag. The recovery procedure is:

a) Start the computer. The computer will stop again with the unallocated tag displayed in AU and AL in TRIM internal code (see Appendix A, Table A-4). The codes are left justified in AU with overflow to AL.

b) Start the computer. The computer will stop with AL cleared.

c) Set $AL_{(14-0)}$ to the desired address*

d) Start the computer.

4) (AL) = 6; (AU) = sequential line identifier. This error stop may occur during pass 1 or pass 2 and indicates that the source program contains too many labels. The error is non-recoverable. Assembly may be continued by starting the computer, but all subsequent unallocated labels and tags will be assigned to address 07777. The effect of this error is the same whether or not the stop option is selected.

5) (AL) = 7. This error stop occurs during pass 2 and indicates that no output number or an illegal output number was selected. The recovery procedure is:

a) Start the computer. The computer will stop again with AL cleared.

b) Enter the desired output number in bits 2-0 of AL.

---

*If the unallocated tag refers to an instruction within the program, set AL_{17} to 1. If the tag is a constant or refers to a fixed address outside the program, AL_{17} should be zero.

c) Start the computer.

6) (AL) = 10. This error stop indicates an illegal character preceding an operator. Reposition the tape and start over. If error persists the source program tape should be examined and corrected.

SECTION III-B. TRIM II ASSEMBLY SYSTEM

1. INTRODUCTION

The TRIM II assembly system provides programming assistance through the use of its symbolic shorthand. This simplified system converts a source program written with symbolic addressing into an object program with absolute or relocatable addressing. TRIM II produces the assembled object program on punched paper tape suitable for loading into the computer via the utility packages.

‎. DESCRIPTION

TRIM II is a 2-pass assembler designed for a minimum equipment configuration of a computer with at least 8,192 (decimal) words of core memory and an I/O console containing a paper tape reader, paper tape punch, and console typewriter. The assembler accepts a source program expressed symbolically, absolutely, or in combination thereof and converts it into an ordered set of machine instructions suitable for loading via the utility packages.

The term 2-pass means the source program tapes must be loaded into the computer twice. The first such loading, constituting pass 1, assimilates and stores information needed for pass 2 (refer to Figure III-B-1). At the completion of pass 1 the source program tapes must again be loaded, and the desired output must be selected. Using the information accumulated during pass 1, pass 2 reads, assembles, and punches on paper tape each source program instruction, statement by statement. This second loading constitutes pass 2 (refer to Figure III-B-2). Subsequent outputs are achieved by repeating pass 2.

2.1 SOURCE LANGUAGE

A TRIM source program as prepared by the programmer is composed of a list of operations which perform the step-by-step processing of a problem. An operation has the following general format:

$$\left[\text{label}\right] \longrightarrow \left[\text{statement}\right] \longrightarrow \left[\text{notes}\right]$$

The general format may be further subdivided into:

$$\left[\text{label}\right]^{L} \longrightarrow \left[\text{operator}\right]^{W} \bullet \left[\text{operand(s)}\right]^{V_n} \longrightarrow \left[\text{notes}\right]^{N}$$
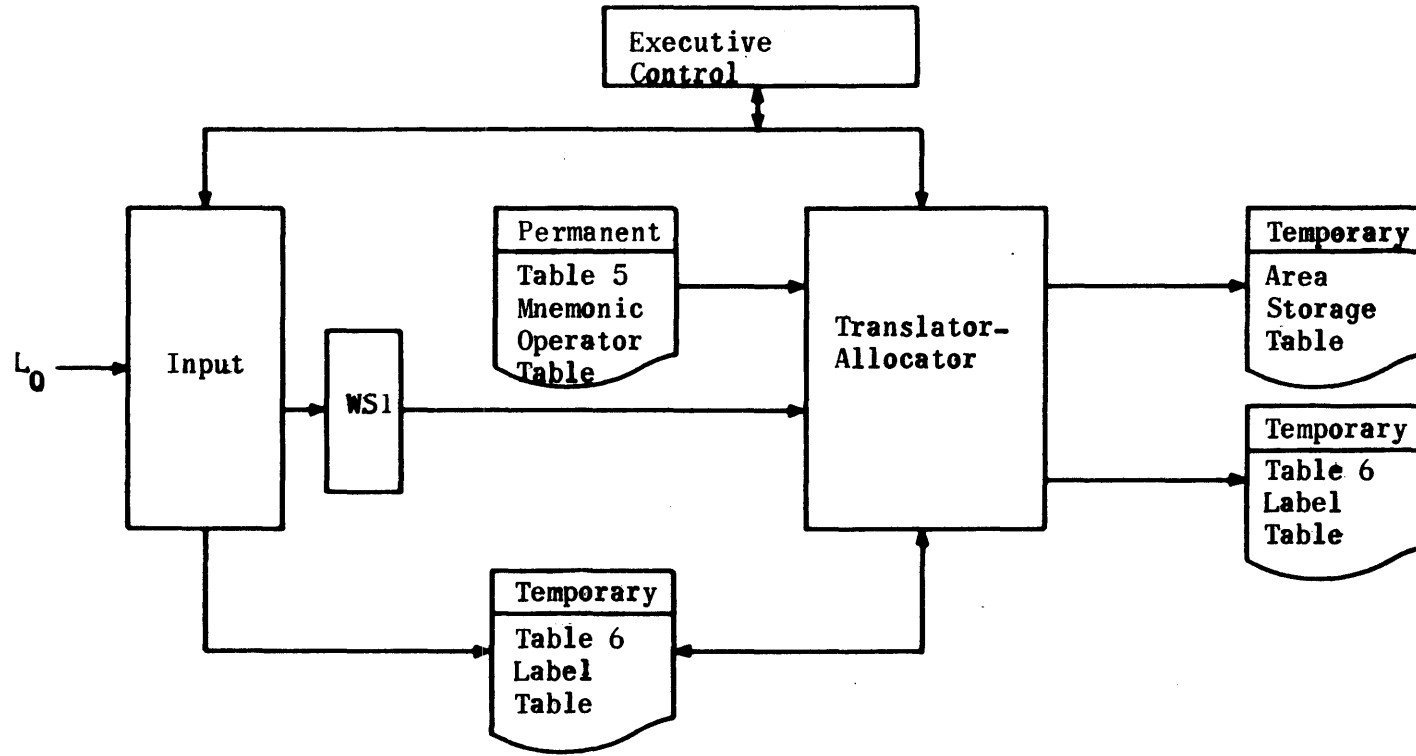
INPUT: a. Reads one item into WS1
             b. Adds allocations to Table 6
             c. Checks for debug status

TRAN/ALLOC: a. Adds labels to Table 6
             b. Performs pseudo generation
             c. Forms the area storage table for output 4

EXECUTIVE: a. Monitors key settings
             b. Performs initializations
             c. Executes secondary subroutines

Figure III-B-1.   Block Chart for TRIM II - Pass 1

Figure III-B-2. Block Chart for TRIM II – Pass 2
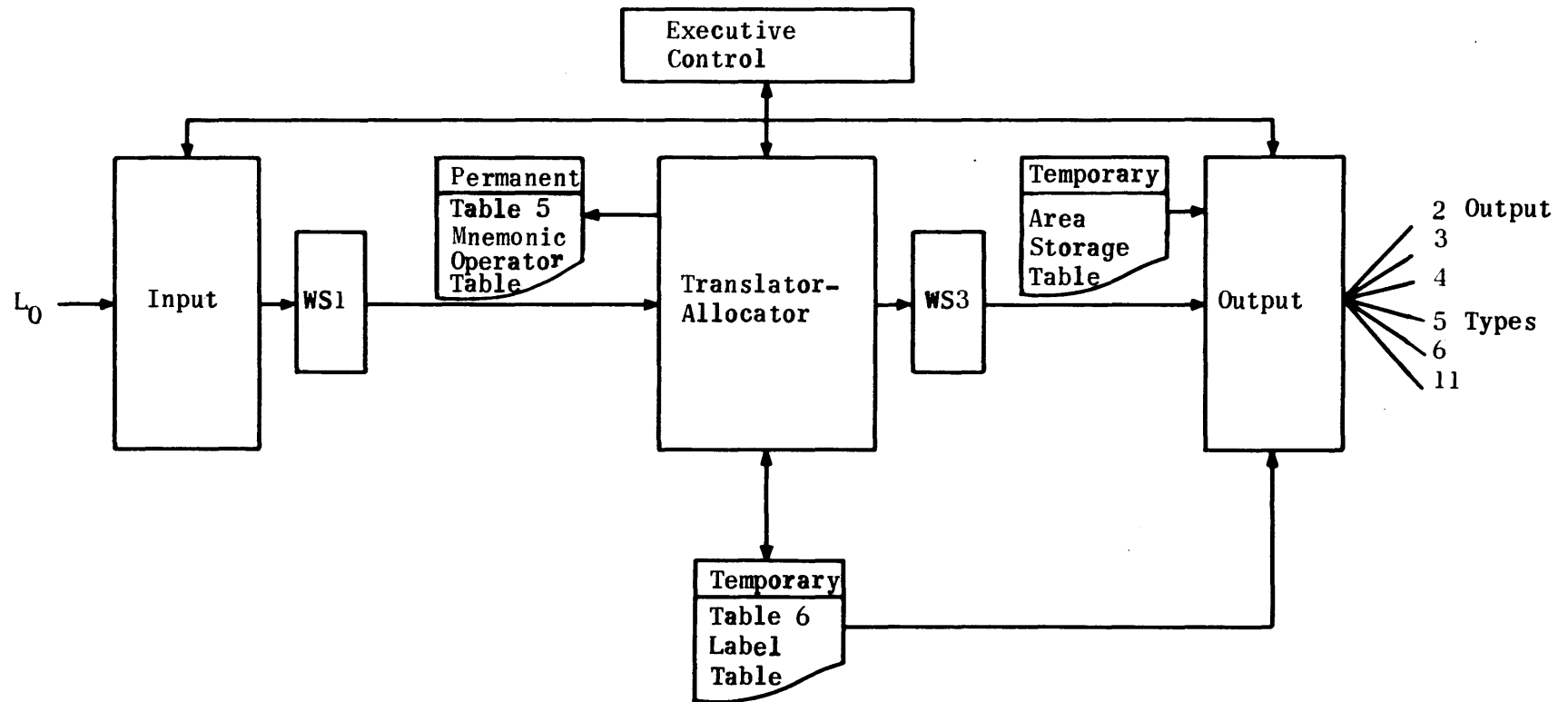
EXECUTIVE:	a.	Monitors key settings
	b.	Performs initializations
	c.	Executes secondary routines
INPUT:	a.	Reads one item into WS1
TRAN/ALLOC:	a.	Performs generation into WS3 one item at a time
	b.	Adds unallocated tags to Table 6
OUTPUT:	a.	Punches one WS3 item at a time in accordance with the numbered output request (for output 2 also punches the WS1 item with the first WS3 item)
	b.	Output 6 will be produced all at once; and does not require that the source tape be reloaded as for other outputs

2.1.1 LABEL

The label identifies this particular statement. A label is not required for every statement. In an absolute-addressed program every word is assigned an absolute address during the coding process. The assembling process of the TRIM II system equates the label to the machine address assigned to the instruction generated by the statement. Only those statements which are referred to by other statements require a label or symbolic address. Where more than one instruction is generated by a statement, the label refers to the address of the first instruction generated. The term label is used rather than address since it more accurately describes the function of the symbolic address. A label may never be incremented or decremented. The instructions or words generated from unlabeled statements following one another on the source program tape are ultimately assigned to consecutive memory addresses. Each label of an assembly run must be unique.

A label may consist of not more than six alphanumeric characters, but it never begins with the letter O or a number; it never consists of the letters LOK alone. The first instruction of each program or subroutine must have a label.

An operand which refers to another operation label is called a tag. The tag must be identical with the label it refers to but may be followed by a ± octal or decimal integer to facilitate reference to unlabeled operations. Whenever a decimal integer is used, it must be followed by the letter D. A tag coincides with the u or k portion of the instruction word. Tags have the same notation restrictions as labels except they may be incremented. Any number of tags may refer to a given label.

If the programmer wishes to reference unlabeled instructions in his program, he may do so in terms of a specific instruction by means of the LOK tag plus or minus an integer. LOK always refers to the instruction in which it appears. For example, if the instruction JP•LOK-3 appears as address 04503, the resulting generation will be 34 4500. Thus the instruction falling at address 04500 need not have been labeled. No valid program label may consist only of the letters LOK. Reasonable care should be taken in the use of the LOK tag since corrections to the original program may affect the LOK references.

2.1.2 STATEMENT

The statement of an operation is made up of an operator and operand(s). The statement defines the operation.

2.1.2.1 OPERATOR

The operator may be a symbolic shorthand or octal notation which identifies the basic function to be performed. The operator must be present. It may cause the assembler to generate one machine instruction or a group of machine instructions. The operator coincides with the function code, f, and/or subfunction code, m, of the instruction word.

2.1.2.2 OPERAND(S)

One or a series of operands associated with the basic operator are referred to as V_0, V_1 ... V_n. These may take several forms depending upon the basic operator. They define, modify, or complete the function.

The operand(s) coincides with the u or k portion of an instruction word and may be either a constant in octal notation or a symbolic alphanumeric notation referring to a constant (either an absolute address or an item of data).

2.1.3 NOTES

Descriptive notes may follow the statement; they are for the programmer's use and in no way affect the instructions generated from the statement. Notes may not exceed 40 (decimal) characters.

2.1.4 SYMBOLS

The programmer uses a uniform set of symbols as separators in all coding. These symbols are defined in Table III-B-1.

TABLE III-B-1. TRIM II CODING SYMBOLS

Symbol	Coding Significance
➞ (tab)	Major separator delimiting the statement. Must always precede the statement operator. Must precede notes; omitted if notes are not given.
↲ (CR)	Specifies the end of an operation. Must precede end-of-tape double period.
, (comma)	Separates certain subsets of statement components.
• (point)	Separates statement components.
+	Specifies an integer increment to follow.
−	Specifies an integer decrement to follow.
Δ (delta)	Specifies space.
| (vertical line)	Special control character.
.. (double periods)	Specifies end-of-tape read-in. Must terminate every input tape.

2.2 HEADER AND DECLARATIVE OPERATIONS

TRIM II recognizes two types of header operations.

L	W	V_0	V_1	N
POKER → ALLOC	• JONES	• 10 MAY1963		→
POKER → PROG	• JONES	• 10 MAY1963		→

2.2.1 ALLOCATION HEADER (ALLOC)

The ALLOC header informs TRIM II that the operations following constitute assignments of absolute values to labels and/or tags. Any number of ALLOC tapes may be loaded, but all must be loaded prior to the loading of program tapes. An allocation tape must always begin with a carriage return. When the allocations are on a separate tape, the tape must terminate with a carriage return and two periods. An ALLOC tape has the following format:

L	W	V_0	V_1
↙ [label]	⟶ [ALLOC]	• [name]	• [date] ↙
[label]	⟶ [assigned value] ↙		
[label]	⟶ [assigned value] ↙		
[label]	⟶ [assigned value] ↙		
[label]	⟶ [assigned value] ↙		
	etc.		

1) L – The label of the ALLOC header operation itself is optional. However, each assignment operation following must have a label.

2) W – The operator of this header operation is always ALLOC, and must be present. For the subsequent assignment operations, W must be an absolute numeric value expressed either in octal or decimal. When expressed decimally, the number must be followed by the letter D; for example:

CAT ⟶ 01000 ↙

DOG ⟶ 512D ↙

CHIPS ⟶ 12 ↙

CHOPS ⟶ 10D ↙

3) V – The V operands of this header operation take the form name and date as illustrated. These operands are omitted for subsequent assignment operations.

2.2.2 PROGRAM HEADER (PROG)

The PROG header informs TRIM II that the operations to follow are program operations as distinguished from allocation operations. The PROG header must precede the first statement of a program. The PROG header operation must always be preceded by a carriage return. A program header has the following format:

$$\qquad L \qquad\qquad\qquad W \qquad\qquad V_0 \qquad\qquad\qquad V_1$$

$$\overset{\curvearrowleft}{}\; \begin{bmatrix} \text{program} \\ \text{name} \end{bmatrix} \longrightarrow \begin{bmatrix} \text{PROG} \end{bmatrix} \quad \bullet \quad \begin{bmatrix} \text{name} \end{bmatrix} \quad \bullet \quad \begin{bmatrix} \text{date} \end{bmatrix} \longrightarrow$$

1) L – The label of the PROG header operation is optional; however, when present, it is considered to be the name of the program.

2) W – The operator of this header operation is always PROG and must be present.

3) V – The V operands of this header operation normally take the form name and date as illustrated. The operands are optional and completely flexible in number and length within the maximum line length.

2.2.3 DEBUG DECLARATIVE

TRIM II accepts the declarative operation:

$$\qquad L \qquad\qquad\qquad W \qquad\qquad\qquad N$$

$$\begin{bmatrix} \text{label} \end{bmatrix} \quad \longrightarrow \quad \text{DEBUG} \longrightarrow$$

The DEBUG operator informs TRIM II that generation is to be performed for debugging operations contained in the source program. If the DEBUG operator is absent, no generation will occur for such debugging operations. The DEBUG operation when used must be loaded prior to the first PROG header. It may be loaded separately or as the last operation on an ALLOC tape; for example:

$$\qquad L \qquad\qquad\qquad W \qquad\qquad V_0 \qquad\qquad\qquad V_1$$

$$\overset{\curvearrowleft}{}\; \text{POKER} \longrightarrow \text{ALLOC} \bullet \text{JONES} \bullet \text{10MAY1963}\; \overset{\curvearrowleft}{}$$

$$\qquad \text{CHIPS} \longrightarrow 01234\; \overset{\curvearrowleft}{}$$

$$\qquad \text{CHOPS} \longrightarrow 1245\; \overset{\curvearrowleft}{}$$

$$\qquad \vdots$$

$$\qquad \text{CHAPS} \longrightarrow 1388D\; \overset{\curvearrowleft}{}$$

$$\qquad\qquad\qquad \longrightarrow \text{DEBUG}\; \overset{\curvearrowleft}{}$$

2.3 MONO-OPERATIONS

Mono or one-to-one operations consist of mnemonic function codes (refer to computer instructions) and symbolic addresses, absolute machine codes, or constants.

Mono-operation statements may be in one of the following formats.

2.3.1 FORMAT A

$$\longrightarrow \overset{W}{[\text{operator}]} \quad \bullet \quad \overset{V_0}{[\text{operand}]} \longrightarrow$$

W – The operator is the f, or fm portion of the operation statement and is the mnemonic representation of the desired function code of the computer instruction repertoire.

V_0 – Represents the u or k portion of the statement and may be a tag, $\pm$ an integer, or an integer only. Integers may be in octal or decimal representation. When decimal representation is used, the integer must be followed by the letter D. Incrementing or decrementing of integers is not permitted. If V_0 is absent, TRIM II generates zeros for the operand without any error indication.

Examples:

→ ENTAL•CAT

→ STRADR•CAT+1

→ CMAL•CAT-8D

→ ENTBK•28D

→ ENTALK•7776 →   Minus 1 to AL

→ STOP•DOG →   DOG defined by an ALLOC opn

→ SKPOIN•7   Results in 502207

→ CPAU   Results in 506200

→ JP•LOK-10 →   LOK signifies this address

→ OUT•6 →   Output transfer channel 6

→ 0•CHEESE+1 →   Buffer terminal address

      ⟶ 0 • CHEESE ⟶   Buffer initial address

        ⟶ ENTAL   Results in 120000

         ⟶ JP   Results in 340000

2.3.2 FORMAT B

TRIM II also accepts programs coded with absolute function codes and absolute
or symbolic addressing. Normal instructions are represented by a 2-digit func-
tion code followed by a point separator and the desired u or mk operand. How-
ever, absolute instructions may also be represented by six consecutive digits
without a point separator.

    Examples:

        ⟶ 12 • 3505 ⟶

        ⟶ 63 • CAT+6 ⟶

        ⟶ 50 • 1306 ⟶

        ⟶ 50 • 6200 ⟶

        ⟶ 506200 ⟶

2.3.3 FORMAT C

Constants may be represented in a number of ways:

        ⟶ 7 ⟶   Results in 000007

        ⟶ 7 • 0 ⟶   Results in 700000

        ⟶ 77 ⟶   Results in 000077

        ⟶ 77 • 0 ⟶   Results in 770000

        ⟶ 777 ⟶   Results in 000777

        ⟶ 77070 ⟶   Results in 077070

        ⟶ 777 • 7 ⟶   Illegal*

        ⟶ 777• ⟶   Results in 000777*

        ⟶ 123456 ⟶   Results in 123456

---

*Whenever there is an expressed value following the point separator, only 1 or
2 digits are permitted in the operator position.

Two special mono-operations are available for the programmer's use: STOP and SKP. If either of these operators is used without a k operand, TRIM II will automatically generate an unconditional instruction of 50 56 40 or 50 50 40 respectively.

2.4 POLY-OPERATIONS

Frequently groups of instructions which perform a specific function appear iteratively in a program. A single poly-operation generates a unique sequence of instructions designed to perform some such specified function. This is the one-to-many relationship between instructions herein termed poly-coding; the parent instruction being termed a poly-operation. TRIM II provides 14 such poly-operations. In some cases TRIM II generates only a single instruction or, as in the case of the REMARK operation, no instructions. It is permissible when coding a routine to intermix mono- and poly-operations in any desired order.

The CLEAR and MOVE poly-operations use the currently active B register, and the MOVE poly-operation also uses AU in the generated coding. If the programmer does not wish the data in these registers to be destroyed, he must store and restore the data around a MOVE or CLEAR operation. The MOVE and CLEAR operations store and restore the programmer's special register setting.

Since poly-operations may generate more than one machine instruction, the tag LOK $\pm$ an integer must not be used in poly-operation coding.

2.4.1 RESERVE OPERATION (RESERV)

$$\begin{array}{ccc} L & W & V_0 \end{array}$$

$$\begin{bmatrix} label \end{bmatrix} \longrightarrow RESERV \quad \bullet \quad \begin{bmatrix} number\ of\ words \end{bmatrix} \longrightarrow$$

The RESERV operation causes the desired number of sequential words to be reserved within a program. The operation generates the number of zero words* specified by the V_0 operand.

1) L - The label for this operation is optional.

2) W - RESERV must always be present.

3) V_0- Specifies by an octal or decimal integer the number of zero words to be generated. V_0 may never be left blank or specified as zero.

---

*TRIM II output No. 2, used primarily for hard-copy debugging and documentation reflects only the first generated zero word of each RESERV operation. TRIM II outputs 3, 4, and 5 contain the requested number of zero words.

Examples:

Assume CAT = 1000 and DOG = 2000

CAT ⟶ RESERV • 12 ⟶ Generates zeros at addresses 1000-1011

DOG ⟶ RESERV • 10D ⟶ Generates zeros at addresses 2000-2011

2.4.2 CLEAR OPERATION

L	W	V_0	V_1
$\left[\text{label}\right]$ ⟶	CLEAR •	$\left[\begin{array}{l}\text{number}\\ \text{of words}\end{array}\right]$ •	$\left[\text{starting address}\right]$ ⟶

The CLEAR operation clears to zero those memory addresses specified in the operation.

1) L - The label for this operation is optional.

2) W - CLEAR must always be present.

3) V_0- Specifies by an octal or decimal integer the number of consecutive memory locations to clear. V_0 may never exceed 4000 octal or 2048D. A V_0 of zero is not permitted.

4) V_1- Specifies the first address of the area to be cleared. The address may be expressed as an absolute octal number or as a symbolic tag plus or minus an octal or decimal integer; that is, CAT-12D or CAT-14. All the words to be cleared must be wholly contained within one memory bank.

Examples of CLEAR operations and the absolute coding generated by the assembler are given below.

Examples:

Assume EXAM1 = 1000, EXAM2 = 1006, and CAT = 10123.

Input Operation	Generated Coding
EXAM1 ⟶ CLEAR • 70 • 7000 ⟶	36 0067
	75 1005
	50 7300
	41 7000
	73 1003
	50 7300

Input Operation	Generated Coding

Input Operation	Generated Coding
EXAM2 ⟶ CLEAR • 21D • CAT ⟶	36 0024
	75 1013
	50 7311
	41 0123
	73 1011
	50 7300

A symbolic representation of the instructions generated is given below.

ENTBK • [No. of locations - 1] ⟶ Set B for No. of locations

STRSR • LOK+4 ⟶ Store current SR

ENTSR • [Bank No. of Clear area] ⟶ Set SR to Clear area

CLB • [First location] ⟶ Clear word at first location + B

BJP • LOK-1 ⟶ Decrement B and repeat loop

ENTSR • 0 ⟶ Return to current bank when B is zero

2.4.3 MOVE OPERATION

L	W	V_0	V_1	V_2
[label] ⟶ MOVE •		[number of words] •	[from address] •	[to address] ⟶

1) L - The label for this operation is optional.

2) W - MOVE must always be present.

3) V_0- Specifies by an octal or decimal integer the number of sequential words to be moved. V_0 may never exceed 4000 octal or 2048D. A V_0 of zero is not permitted.

4) V_1- Specifies the first address of a block of data to be moved. It may be expressed as an absolute address in octal or as a symbolic tag plus or minus an octal or decimal integer. All the words to be moved must be wholly contained within one blank.

5) V_2- Specifies the first address to which the block of data is to be moved. It is expressed the same as the V_1 operand. All the destination addresses must also be wholly contained within one bank.

Examples of move operations and the absolute coding generated by the assembler are listed on the following page.

Examples:

Assume EXAM3 = 1014, EXAM4 = 1024, and CAT = 1056

Input Operation	Generated Coding
EXAM3 —➤ MOVE • 10 • CAT • 7000 —➤	36 0007
	75 1023
	50 7300
	11 1056
	50 7300
	47 7000
	73 1016
	50 7300
EXAM4 —➤ MOVE • 100 • 1200 • CAT-100 —➤	36 0077
	75 1033
	50 7311
	11 2000
	50 7310
	47 0756
	73 1026
	50 7300

A symbolic representation of the instructions generated is given below.

ENTBK • [No of locations - 1] —➤ Set B for No. of words

STRSR • LOK+6   —➤ Store current SR

ENTSR • [Bank No. of from address] —➤ Set SR to origin bank

ENTAUB • [from address] —➤ Get word at from address + B

ENTSR • [Bank No. of to address] —➤ Set SR to destination bank

STRAUB • [to address] —➤ Store word at to address + B

BJP • LOK-4 —➤ Decrement B and repeat loop

ENTSR • 0 —➤ Return to current bank when B is zero

2.4.4 I/O OPERATIONS

L	W	V_0	V_1	V_2	V_3
[label] →	EXFCT •	[channel number] •	[AD / MAD / BK / MBK] •	[number of buffer words] •	[buffer starting address] →
[label] →	BUFIN •	[channel number] •	[AD / MAD / BK / MBK] •	[number of buffer words] •	[buffer starting address] →
[label] →	BUFOUT •	[channel number] •	[AD / MAD / BK / MBK] •	[number of buffer words] •	[buffer starting address] →

1) L - Label for these operations is optional.

2) W - The operator must always be present.

3) V_0- Specifies the channel number expressed as an integer or symbolic tag.

4) V_1- Specifies the buffer mode and must be present:

      AD - Advance without monitor.

      MAD - Advance with monitor.

      BK - Back without monitor.

      MBK - Back with monitor.

5) V_2- Specifies as an octal or decimal integer the number of buffer words involved. Maximum of five digits.

6) V_3- Specifies the address in memory at which buffering is to begin. V_3 may be expressed absolutely or as a symbolic tag plus or minus an octal or decimal integer.

Examples:

Assume CAT = 1000 and CHAN = 05

[label] → EXFCT • 7 • AD • 1 • CAT ⟶ Generates 501307
001001
001000

[label] → BUFIN • 6 • MAD • 10 • CAT ⟶ Generates 501106
201007
201000

[label] → BUFOUT • 0 • BK • 10 • CAT+7 ⟶ Generates 501200
400777
401007

[label] → BUFOUT • 1 • MBK • 100D • CAT+100D ⟶ Generates 501201
601000
601144

[label] → BUFIN • CHAN • AD • 77 • 25000 ⟶ Generates 501105
025076
025000

NOTE: The examples above illustrate the fact that,
for output and external function buffers, the
inclusive buffer limits define a number of words
which is one greater than the actual number of
words to be transferred. These buffers terminate
before transferring the word located at the ter-
minal address.

2.4.5 REMARK OPERATION

$$L \qquad\qquad W \qquad\qquad V_0$$

[label] ⟶ REMARK • [desired statement] ⟶

The REMARK operation causes no object program generation. It is simply an aid
to the programmer in expanding normal program notes. REMARK operations may not
exceed one line in length.

2.4.6 DATA OPERATION

$$L \qquad\qquad W \qquad\qquad\qquad V_0$$

[label] ⟶ DATA • [integer, binary point specification] ⟶

The DATA operation allows the programmer to specify a positive or negative data integer and its binary point position. The bits are numbered from right to left 0-17D. The binary point specification must be separated from its associated integer by a comma. The absence of a minus sign implies a positive integer. The label is optional.

Examples:

$$[\text{label}] \longrightarrow \text{DATA} \cdot 24D,9D \longrightarrow \text{Generates } 030000$$

or

$$[\text{label}] \longrightarrow \text{DATA} \cdot 30,11 \longrightarrow \text{Generates } 030000$$

The binary representation is:

17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0

and

$$[\text{label}] \longrightarrow \text{DATA} \cdot 123,4 \longrightarrow \text{Generates } 002460$$

The binary representation is:

17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	1	0	1	0	0	1	1	0	0	0	0

2.4.7 PUNCH CONTENTS OPERATION (PCHC)

$$\begin{array}{ccc} L & W & V_0 \\ [\text{label}] \longrightarrow \text{PCHC} \cdot & & \left[\begin{array}{l}\text{information to be punched} \\ \text{and/or typewriter commands}\end{array}\right] \longrightarrow \end{array}$$

The PCHC operation produces generated coding which, when run on the computer with the PCHC* subroutine, causes the octal contents of A, AU, AL, B, or any memory location to be punched on the high-speed paper tape punch. The words to be punched may be interspersed with the following typewriter control symbols to provide subsequent listing in the desired format.

operand	performance
• |CR| •	carriage return, line feed
• Δ • or • |SP| •	space

The vertical bars indicate the information enclosed is a special symbol directing the typewriter. Each CR and SP must begin and end with the vertical bar. Controls are separated from other operands by point separators.

---

* See paragraph 3.2 8).

V_O specifies the operands in the order in which they are to be punched. Except for the typewriter commands, all operands imply their contents are to be punched. Such operands may be A, AU, AL, active B, a tag or a tag $\pm$ an absolute value, or an absolute address.

Examples:

CAT →PCHC • A • Δ • 7070 • Δ • DOG-11D • |CR| →

→PCHC • DOG • Δ • B • |SP| • AU • |CR| • AL →

→PCHC • |CR| • |CR| • TA →

→PCHC • DOG+10 • |SP| • CAT →

2.4.8 PUNCH TEXT OPERATION (PCHT)

L   W V_O

[label] →PCHT • [text and/or typewriter commands]

The PCHT operation produces coding which, when run on the computer with the PCHT* subroutine, causes the text and/or typewriter commands in the V_O operand position to be punched by the high-speed paper tape punch. The text may be interspersed with the following typewriter control symbols as desired, each CR and SP must be set off between two vertical bars.

operand	performance
|CR|	carriage return, line feed
Δ or |SP|	space

NOTE: Point separators are not required within V_O; they will be punched if present.

V_O is the text to be punched interspersed with typewriter command desired by the programmer. If the text is too long for one PCHT operation, the programmer may write successive operations.

Examples:

CAT → PCHT • PROFIT Δ AND Δ LOSS Δ FOR |SP| →

→ PCHT • JULY Δ 10, Δ 1967 |CR| →

2.4.9 TYPE TEXT OPERATION (TYPT)

L   W V_O

[label] → TYPT • [text and/or typewriter commands] →

*See paragraph 3.2 8).

The TYPT operation results in generated coding which, when run on the computer with the TYPT* subroutine, causes the text and/or commands in the V_0 operand position to be typed by the typewriter. The text may be interspersed with the following typewriter commands:

operand	performance
|CR|	carriage return, line feed
Δ or |SP|	space (may be used for formatting)

The vertical bars indicate the information enclosed is a special symbol directing the typewriter. Each CR or SP must begin and end with a vertical bar.

V_0 is the text to be typed interspersed with typewriter commands. If the text is too long for one TYPT operation, the programmer may use successive operations to complete the text.

Examples:

CAT —→ TYPT • PROFIT \|SP\| AND Δ LOSS Δ FOR —→

—→ TYPT • JULY Δ 10, Δ 1967 \|CR\| —→

NOTE: Point separators are not required within V_0; they will be typed if present.

2.4.10 TYPE CONTENTS OPERATION (TYPC)

L	W	V_0
[label]	—→ TYPC •	[information to be typed and/or typewriter commands] —→

The TYPC operation results in generated coding which, when run on the computer with the TYPC* subroutine, causes the octal contents of A, AU, AL, current B, or any memory location to be typed on the typewriter. The words to be typed may be interspersed with the following typewriter commands:

operand	performance
• |CR| •	carriage return, line feed
• Δ • or • |SP| •	space (may be used for formatting)

The vertical bars indicate the information enclosed is a special symbol directing the typewriter. Each CR or SP must begin and end with a vertical bar.

---

*See paragraph 3.2 8).

V_0 specifies the operands in the order in which they are to be typed. Except for the typewriter commands, all operands imply their contents are to be typed. Such operands may be A, AU, AL, active B, a tag or tag $\pm$ an absolute value or an absolute address.

Examples:

CAT ⟶ TYPC • A • Δ • Δ • 7070 • Δ • Δ • DOG-11D • |CR| ⟶

⟶ TYPC • AU • Δ • AL • |SP| • B • HORSE ⟶

2.4.11 DOUBLE SET OPERATION (DBLSET)

      L   W

[label] ⟶ DBLSET ⟶

The **DBLSET** operation insures that the Y of a double add or subtract instruction is located at an even address. The DBLSET operation is normally followed by a Y constant. TRIM II examines the address to which the constant (or instruction) would normally be assigned. If the address is odd, a word of zeros is first generated to insure that the constant (or instruction) will be assigned to an even address. If the address is even, no generation results.

2.4.12 SETSR OPERATION

      L   W V_0

[label] ⟶ SETSR • [alphanumeric tag] ⟶

The SETSR operation enables the programmer to place responsibility for setting k of an ENTSR instruction upon TRIM II. Based upon an ALLOC operation for the assembled address of the referenced tag, TRIM II will generate an ENTSR instruction (5073 k) with the proper k value for each SETSR operation.

1) L - Label is optional.

2) W - SETSR must be present.

3) V_0- Must be an alphanumeric name corresponding to a program label or an allocated value. V_0 may not be incremented or decremented.

Examples:

Assume CAT is a label at 36421 and DOG is a label at 70460 and COW is allocated to 01000, then:

          SETSR ⟶ CAT ⟶ Generates 507313

          SETSR ⟶ DOG ⟶ Generates 507317

          SETSR ⟶ COW ⟶ Generates 507310

2.5 DEBUGGING OPERATIONS

TRIM II provides two debugging operations for punching a paper tape output of either the contents of registers AU, AL, and current B, or the contents of specified sequential memory locations. These operations are recognized by TRIM II only if a DEBUG declarative operation is present in the program prior to the first PROG header operation. If this condition is satisfied, these operations generate a set of three or five instructions in the object program which, when run on the computer with the DEBUG* subroutine, produce the desired punched output. Each set of instructions is assigned a sequential identifying number which appears with each punched output, thereby enabling programmer recognition of repeated times through given coding paths. The debugging operations take the following form:

```
     L              W              N

  [label]    ──▶  DUMPR  ──▶


     L              W              V₀                    V₁              N

  [label]    ──▶  DUMPM  •  ⌈number of  ⌉ • ⌈address of first⌉  ──▶
                            ⌊words to dump⌋   ⌊word to dump     ⌋
```

1) L - Label is optional.

2) W - DUMPR or DUMPM must always be present.

3) V_0- Applicable to the DUMPM operation only. Specifies the total number of memory locations to be dumped. The number may be expressed in octal or in decimal followed by the letter D.

4) V_1- Applicable to the DUMPM operation only. Expresses the address of the first word to be dumped. It may be expressed as an integer or a tag plus or minus an integer.

Examples:

```
  [label]    ──▶  DUMPR  ──▶

  [label]    ──▶  DUMPM  •  12   •  10000  ──▶

  [label]    ──▶  DUMPM  •  10D  •  110000 ──▶

  [label]    ──▶  DUMPM  •  10D  •  CAT+28D ──▶

  [label]    ──▶  DUMPM  •  12   •  CAT-15  ──▶

  [label]    ──▶  DUMPM  •  64D  •  CAT    ──▶
```

---

*See paragraph 3.2 8).

Examples of the DUMPR and DUMPM operation and the coding generated by the assembler are given below.

Examples:

Assume EXAM5 = 1000, EXAM6 = 1050, and DEBUG = 30000

Input Operation	Generated Coding
EXAM5 ➞ DUMPR ➞	301001 030000 000001
EXAM6 ➞ DUMPM • 5 • 10000 ➞	301051 030000 400002 000005 010000

A symbolic representation of the instructions generated is given below. The first three instructions apply to both DUMPR and DUMPM. The last two instructions apply to DUMPM only.

IRJP • DEBUG ➞ Indirect return jump to DEBUG

0 • DEBUG ➞ Address of DEBUG

X • [Y] ➞ X = 0 for DUMPR, 4 for DUMPM
Y = No. of DUMPR or DUMPM operation in this program

[No. of words] ➞ No. of words to be dumped

[First address] ➞ Address of first word to be dumped

Both DUMPR and DUMPM operations preserve existing values in AU, AL, and the current B register.

2.6 TRIM II OUTPUTS

TRIM II provides six optional punched paper tape outputs of the assembled program. All the outputs except outputs No. 6 and 11 are loadable via the utility package.

The available outputs are:

No. 2 - Absolute assembled program, sequential line identifier, source program, and assembly error alarms when applicable. This is a side-by-side listing in source code preceded by a program summary consisting of the number of memory locations used and inclusive addresses.

No. 3 - Absolute assembled program in source code, consisting of a carriage return, 88, carriage return, addresses and instructions, carriage return, double period and checksum.

No. 4 - Absolute assembled program in bioctal format, consisting of 76 code, inclusive area addresses followed by the instructions only, and a checksum.

No. 5 - Relocatable assembled program in bioctal format consisting of a 75 code followed by the assembled program relative to base 00000 and a checksum. The output tape may be loaded starting at any desired memory location (if the program is to be loaded in different memory banks, SR register manipulation must be handled by the program).

No. 6 - Allocation output in source code consisting of an ALLOC header followed by all program tags and labels and addresses in allocation format. To insure a complete allocation tape, output 6 should not be the first requested output of an assembly run.

No. 11- Source program on paper tape in source code.

3. PROGRAMMING PROCEDURES

3.1 INPUT TAPE FORMAT

Source program tapes must be punched in source code, and the resulting punched paper tape serves as input to TRIM II.

3.2 GROUND RULES

1) TRIM II is available in two versions; one version accepts a source program prepared in field data code, the other version accepts a source program in ASCII code. Refer to Appendix A, Tables A-2 and A-3. Input to TRIM II is via punched paper tape. Each source tape must begin with a carriage return and terminate with a carriage return and two periods. The term source code used herein refers to code in which the input tapes are prepared.

2) No TRIM II label may exceed six characters. The label must not begin with a number, the letter O, or consist only of the letters LOK. The first instruction of every program must have an assigned label.

3) Each break in sequence of addressing constitutes a program area. A total of 27 such areas are permitted.

4) The maximum size program which TRIM II can assemble is limited only by the number of program and allocation labels. The maximum number of labels allowed is approximately 1100_{10} for each 8000_{10} words of memory.

5) TRIM II provides a limited amount of error detection with error typeouts (see TRIM II assembler operating procedure). All other errors are indicated by multiples of 100 following the notes of the instruction concerned on the No. 2 output. Thus 100 indicates one error; 200 indicates two errors in this instruction, and so forth. Typical errors are unconvertible numbers, illegal operators, no label first instruction, duplicate label, and so forth.

6) TRIM I operators SETADR and EQUALS are ignored by TRIM II. The ALLOC operation replaces these two functions.

7) When specifying a decimal integer, the letter D occupies one digit position; therefore, the maximum decimal integer that can be expressed is 99999D.

8) Due to space considerations the TYPT, TYPC, PCHT, PCHC, and DEBUG subroutines are supplied on tape separate from the TRIM II package in both source language and object language formats. Therefore, when using these operations, it is necessary either to assemble the subroutine(s) with the running program or load them independently with the running program. If the programmer assembles any of these subroutines with his program he may allocate them or let TRIM II allocate them sequentially following the end of his program. In either case the programmer must allocate the tag CHAN used by these source language subroutines to reference the paper tape I/O channel. If the programmer does not assemble any of these subroutines with his program but uses poly-operations calling on them, he must allocate the subroutines to a desired address. If he does not, TRIM II will arbitrarily allocate them as follows:

TYPT	17000
TYPC	17160
PCHT	16400
PCHC	16560
DEBUG	17470

9) Keyboard correction methods; Typing error correction procedures have been incorporated in TRIM I, TRIM II, and TRIM III assemblers, and the TRIM corrector for deleting immediate keyboard errors that might be made in the preparation of input tapes for these same programs on the UNIVAC 1232 and 1532 I/O consoles. These procedures are described under TRIM I, paragraph 8.

4. LOADING AND OPERATING PROCEDURES

Prior to loading the TRIM II assembler, the computer, paper tape reader-punch unit, and console typewriter must be placed in the operational state with all switches in the normal operating position. Since the assembler is loaded by a utility package, the utility package must be already loaded into memory at addresses which will not be occupied by the assembler.

4.1 LOADING THE ASSEMBLER

To load TRIM II, perform the following steps:

1) Master clear the computer and the I/O console.

2) Mount the TRIM II assembler tape in the reader.

3) Set the P register to the utility package starting address for paper tape load.

4) Start the computer.

4.2 USING THE TRIM II ASSEMBLER

1) Pass 1

a) Master clear the computer.

b) Set the P register to 1400.

c) Set PROGRAM SKIP keys 1 and 2.

d) Set PROGRAM SKIP key 3 for error typeout suppression during assembly (see paragraph 4.3).

e) Mount a source program tape in the reader.

f) Start the computer.

g) The computer will stop after the source program tape has been read. Repeat steps e) and f) until all tapes have been read in.

2) Pass 2

a) Release PROGRAM SKIP key 1.

b) Set the AL register to desired output number (2, 3, 4, 5, 6, or 11; refer to paragraph 2.6).

c) Mount a source program tape in the reader (program tapes must be loaded in exactly the same order as for pass 1).

d) Start the computer.

e) The Assembler will assemble the input tape and punch the output tape. Repeat steps c) and d) until all tapes have been read in and punched.

f) Release PROGRAM SKIP key 2.

g) Start the computer.

h) The assembler will finalize the output tape with a checksum and trailer.

i) Set PROGRAM SKIP key 2 and repeat from step b) to obtain additional outputs.

4.3 ERROR DETECTION AND DISPLAY

TRIM II contains certain error detection capabilities. The majority of programmer errors can be handled internally. However, since PROGRAM SKIP key settings are essential to the assembly process, assembly will always stop when PROGRAM SKIP keys 1 and 2 are not set. The proper action to take is indicated by the following typeout: SET KEYS 1 AND 2. Set both PROGRAM SKIP keys 1 and 2 and start the computer to continue assembly.

The programmer has the option of requesting that the assembler stop and type out pertinent information for basic programmer errors, or requesting that the assembler handle these errors internally, thereby forcing an assembly.

If PROGRAM SKIP key 3 is set, the assembler will force the assembly. If PROGRAM SKIP key 3 is not set, the assembler will stop after typing the following identifiers.

4.3.1 'SET BASE ADDRESS IN AL'

This typeout and error stop occurs during pass 1 and means that no starting address has been given. To correct:

1) Clear the AL register.

2) Set $AL_{(15-0)}$ to the desired address in octal.

3) Start the computer. The assigned base address will then be typed out and assembly will continue.

If typeout suppression has been selected (PROGRAM SKIP key 3 set), assembly will not stop and TRIM II will arbitrarily assign the program to the base address 01200.

4.3.2 'ILLEGAL OUTPUT RESELECT IN AL'

This typeout and error stop indicates an illegal output has been selected at the start of pass 2. To recover, start the computer. When the computer stops again, reselect the output in AL, and start the computer.

> NOTE: If poly-operation generation results in a memory bank overflow, output 2 is the only legal output that may be requested. If a legal output has been selected and the above typeout occurs, bank overflow is the cause.

4.3.3 'UNALLOC TAGS'

This typeout and error stop occurs during pass 2 and indicates an unallocated tag. The first such typeout is followed by a typeout of the sequential line identifier and the tag name. After recovery, the address is also typed.

Thereafter only the sequential line identifier, the tag name, and the address to which the tag was equated during recovery are typed.

The recovery procedure is:

1) Set the AL register to the desired value.

2) If the tag refers to an instruction contained within the program being assembled, set AL_{17} to 1. If the tag is a constant or refers to a fixed address outside the program, AL_{17} must be 0.

3) If the user wishes all later unallocated tags allocated to the same address, set the AU register to any nonzero value.

4) Start the computer. TRIM II will type the manual allocation and use it to continue assembly.

4.3.4 'DUP LBL'

If during generation a duplicate label is discovered, TRIM II types the sequential line identifier, DUP LBL, and the label name. The assembly will continue without a computer stop.

SECTION III-C. TRIM III ASSEMBLY SYSTEM

1. INTRODUCTION

The TRIM III assembly system provides programming assistance through the use of its symbolic shorthand. As illustrated in Figure III-C-1, this assembly system converts a source program written with symbolic addressing into an object program with absolute or relocatable addressing. TRIM III produces the assembled object program on punched paper tape, punched cards, or magnetic tape.
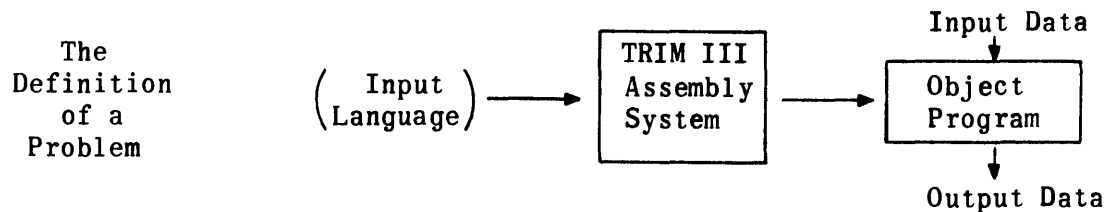
Figure III-C-1. TRIM III Solution of a Problem

TRIM III has an easy-to-use but effective library retrieval capability. The library of subroutines is stored on the assembler magnetic tape. The user simply calls by name those subroutines he wishes to include with his assembled program. TRIM III honors the calls by automatically adding them to the end of the source program during assembly. A companion program to TRIM III called the library builder routine provides easy library building, insertion, replacement, deletion, and listing capabilities.

TRIM III possesses source language level correction capability in combination with an assembly run. Although this feature is primarily designed for use with paper tape input, it may be used with any combination of input modes.

2. DESCRIPTION

TRIM III is basically designed for a minimum equipment configuration of a computer with at least 16,384 words of core memory, a magnetic tape system with two or more tape transports, and an I/O console consisting of a punched tape reader, tape punch, keyboard, and console typewriter. Optional equipment is an on-line card processor system with card reader, card punch, and high-speed printer.

The TRIM III assembler is stored on magnetic tape in functional segments. During an assembly run the segments are read into computer memory and executed in the proper sequence by the assembler controlling routine. See Figures III-C-2 and III-C-3. TRIM III is a single external pass assembler. It accepts a source program, converts it to TRIM code, and stores it on magnetic tape for subsequent processing. If the user has included calls for library subroutines in his source program, TRIM III selects them from the library and adds them to the end of the source program before proceeding with assembly. TRIM III also has source language correction capability in conjunction with an assembly run.
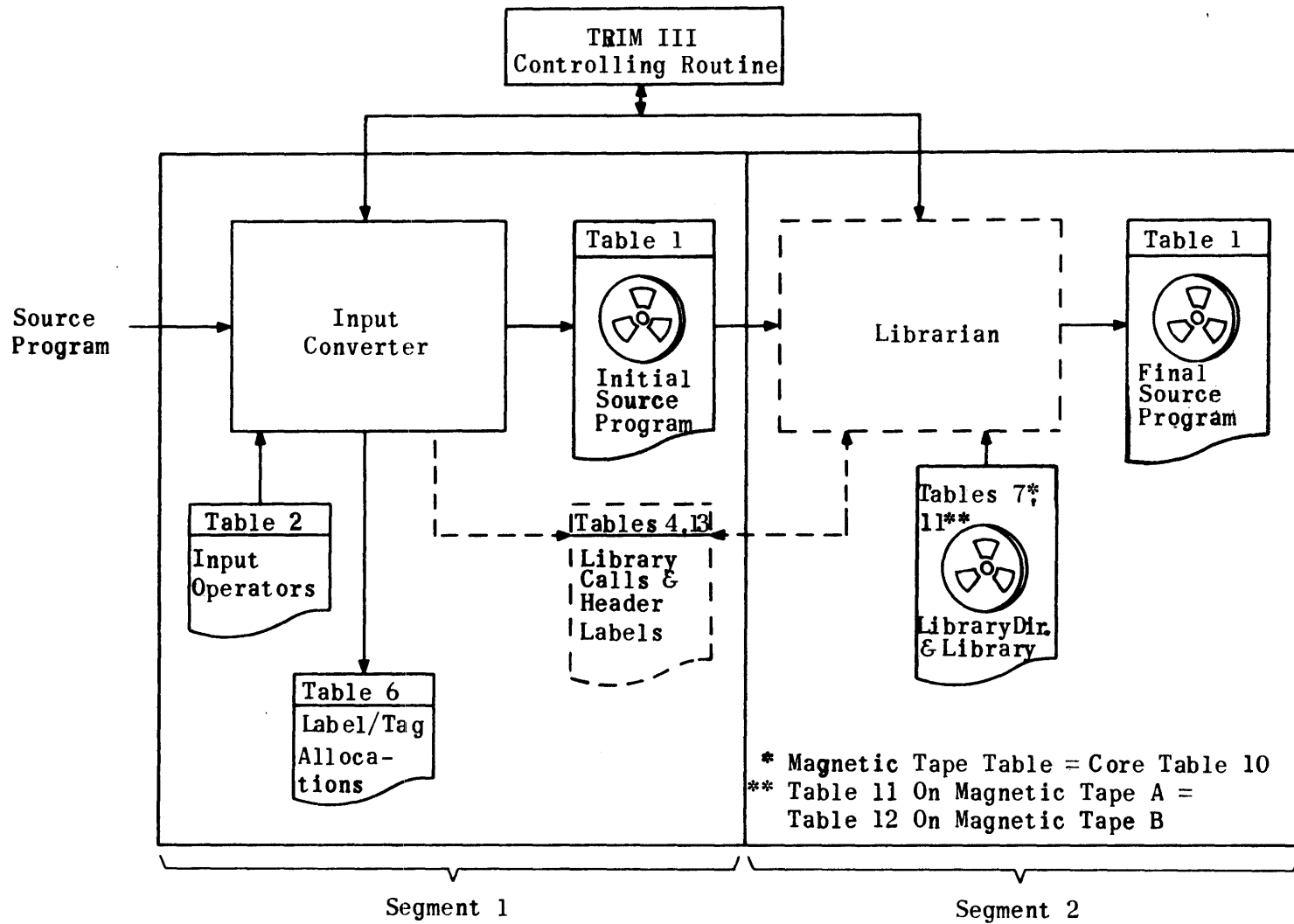
Figure III C-2. TRIM III Segments 1 and 2
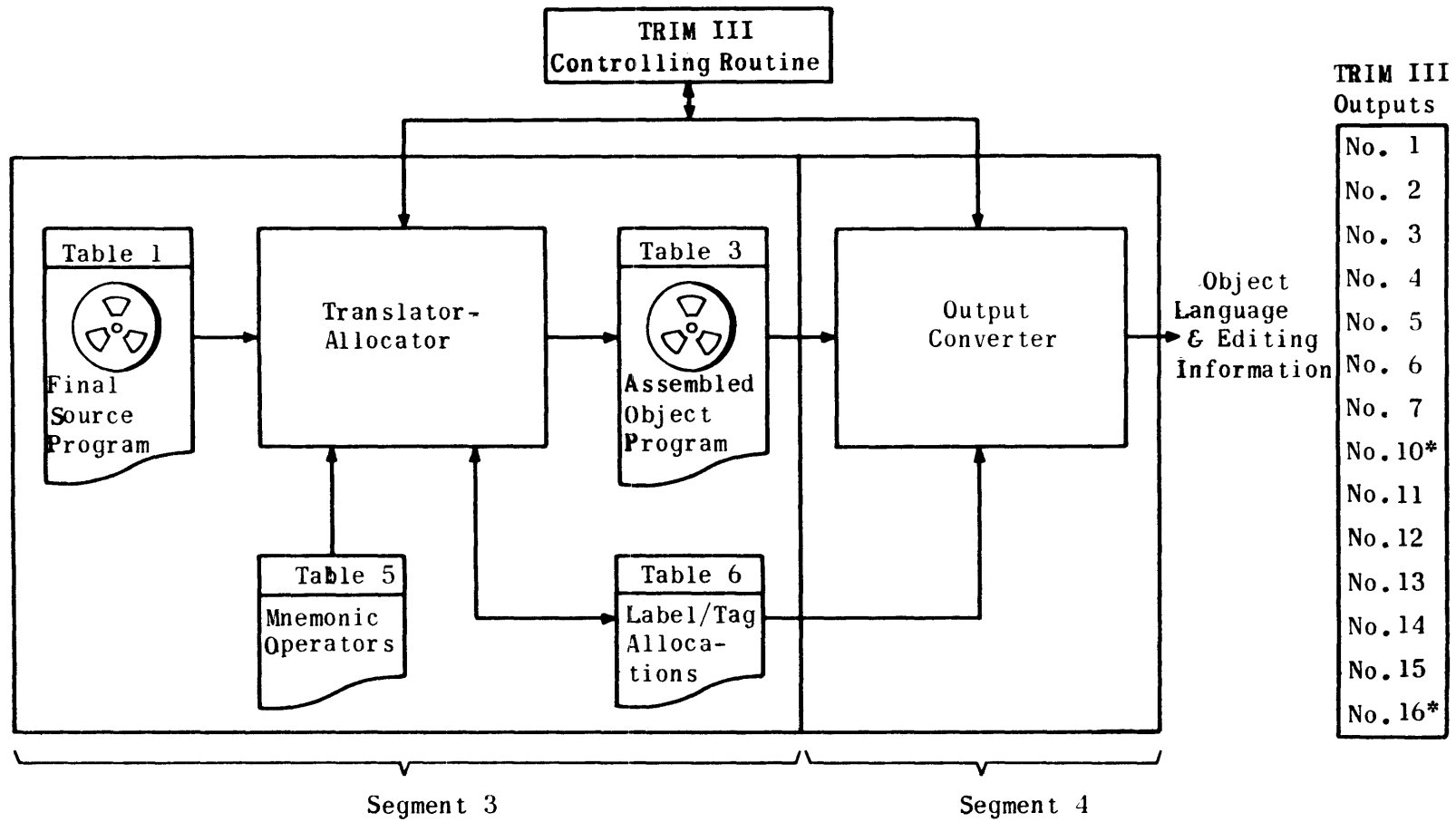
Figure III-C-3. TRIM III Segments 3 and 4

2.1 SOURCE LANGUAGE

A TRIM program as prepared by the programmer is composed of a list of operations
which perform the step-by-step processing of a problem. An operation has the
following general format:

$$[\text{label}] \longrightarrow [\text{statement}] \longrightarrow [\text{notes}]$$

The general format may be further subdivided into:

$$
\begin{array}{cccc}
L & W & V_n & N
\end{array}
$$

$$[\text{label}] \rightarrow [\text{operator}] \bullet [\text{operand(s)}] \longrightarrow [\text{notes}]$$

2.1.1 LABEL

The label identifies this particular statement. A label is not required for
every statement. In an absolute-addressed program every word is assigned an
absolute address during the coding process. The assembling process of the
TRIM III system equates the label to the machine address assigned to the in-
struction generated by the statement. Only those statements which are referred
to by other statements require a label or symbolic address. Where more than
one instruction is generated by a statement, the label refers to the address of
the first instruction generated. The term label is used rather than address
since it more accurately describes the function of the symbolic address. A
label may never be incremented or decremented. The instructions or words
generated from unlabeled statements following one another on the source program
tape are ultimately assigned to consecutive memory addresses. Each label of
an assembly run must be unique.

A label may consist of not more than six alphanumeric characters; it never
begins with the letter O or a number, and never consists of the leters LOK
alone. The first instruction of each program or subroutine must have a label.

An operand which refers to another operation label is called a tag. The tag
must be identical with the label it refers to except that it may be followed
by a $\pm$ octal or decimal integer to facilitate reference to unlabeled operations.
Whenever a decimal integer is used, it must be followed by the letter D. A
tag coincides with the u or k portion of the instruction word. Tags have the
same notation restrictions as labels except they may be incremented. Any
number of tags may refer to a given label.

If the programmer wishes to reference unlabeled instructions in his program
in another manner, he may do so in terms of a specific instruction by means
of the LOK tag plus or minus an integer. LOK always refers to the instruction
in which it appears. For example, if the instruction JP•LOK-3 appears at
address 04503, the resulting generation is a jump to address 04500. Thus the
instruction falling at address 04500 need not have been labeled. No valid
program label may consist only of the letters LOK. Reasonable care should be
exercised in the use of the LOK tag since corrections to the original·program
may affect the LOK references.

2.1.2 STATEMENT

The statement of an operation is made up of an operator and operand(s). The statement defines the operation.

2.1.2.1 OPERATOR

The operator may be a symbolic shorthand or octal notation which identifies the basic function to be performed. The operator must always be present. It may cause the assembler to generate one machine instruction or a group of machine instructions. The operator coincides with the function code f, and/or sub-function code m, of the instruction word.

2.1.2.2 OPERAND(S)

One or a series of operands associated with the basic operator are referred to as V_0, V_1 ... V_n. These may take several forms depending upon the basic operator. They define, modify, or complete the function.

The operand(s) coincides with the u or k portion of an instruction word and may be either a constant in octal notation or a symbolic alphanumeric notation referring to a constant (either an absolute address or an item of data).

2.1.3 NOTES

Descriptive notes may follow the statement; they are for the programmer's use and in no way affect the instructions generated from the statement. Notes must be restricted in length such that the entire source statement does not exceed one line or one card.

2.1.4 SYMBOLS

The program uses a uniform set of symbols as separators in all coding. These symbols are depicted in Table III-C-1 below.

TABLE III-C-1. TRIM III CODING SYMBOLS

Symbol		Coding Significance
�	(tab)	Major separator delimiting the statement. Must always precede the statement operator. Must precede notes; omitted if notes are not given.
⤶	(CR)	Specifies the end of an operation. Must precede end-of-tape double period.
,	(comma)	Separates certain subsets of statement components.
•	(point)	Separates statement components.

TABLE III-C-1. TRIM III CODING SYMBOLS (CONT.)

Symbol		Coding Significance
+		Specifies an integer increment to follow.
-		Specifies an integer decrement to follow.
Δ	(delta)	Specifies space.
|	(vertical line)	Special control character.
..	(double period)	Specifies end-of-tape read-in. Must terminate every input tape.

2.2 HEADER AND DECLARATIVE OPERATIONS

TRIM III recognizes four types of header operations:

L	W	V_0	V_1	N
POKER ⟶ CONTR	•	JONES	•	10 DEC1964 ⟶
POKER ⟶ ALLOC	•	JONES	•	10 DEC1964 ⟶
POKER ⟶ PROG	•	JONES	•	10 DEC1964 ⟶
POKER ⟶ CORREC	•	JONES	•	10 DEC1964 ⟶

2.2.1 CONTROL HEADER (CONTR)

The CONTR header operation is a convenience for the user. It enables him to group all of his assembler declarative operations following one CONTR header. A label and identifying operands may be used with the CONTR header, but TRIM III does not require them.

$$L \quad\quad W \quad\quad V_0 \quad\quad V_1$$
$$[label] \longrightarrow [CONTR] \quad\bullet\quad [name] \quad\bullet\quad [date] \longrightarrow$$

Operations which may follow a CONTR header are ALLOC, DEBUG, OUTPUT, REMARK, DECKID and CALL. CALL operations may also follow a PROG header. Figure III-C-4 shows typical coding for a CONTR header and the declarative operations used with it.

TITLE _____

PAGE _____ of _____

LABEL	OPERATOR	OPERANDS AND NOTES
POKER	→ CONTR HEADER TYPE	• JONES • 16NOVEMBER1963
POKER	→ ALLOC	• JONES • 16NOVEMBER
POKER	→ 05000	•
CHIP	→ 05500	•
DEBUG	→ 13000	•
TYPT	→ 12700	•
	→ OUTPUT	• 1•6•2•5•6•11
	→ DEBUG	•
	→ CALL	• SINE TODEC TYPT

	→ REMARK	• CONTR TAPE FOR DATAX REVISION 3
	→ DECKID	• SINE
••	→	•
	→	•
	→	•
	→	•
	→	•
	→	•

	→	•
	→	•
	→	•
	→	•
	→	•
	→	•
	→	•
	→	•

Figure III-C-4.   Sample CONTR Header And Delcarative Operations

2.2.2 ALLOCATION HEADER (ALLOC)

The ALLOC header follows a CONTR operation and informs TRIM III that the operations which follow constitute assignments of absolute values to labels and/ or tags. Any number of ALLOC tapes or cards may be loaded. An allocation tape must always be preceded by a carriage return (see paragraph 3). When the allocations are on a separate tape, the tape must terminate with a carriage return and two periods. ALLOC operations have the following format:

$$
\begin{array}{cccc}
L & W & V_0 & V_1 \\
[\text{label}] \rightarrow [\text{ALLOC}] & \bullet & [\text{name}] & \bullet & [\text{date}] \\
[\text{label}] - [\text{assigned value}] \\
[\text{label}] \quad [\text{assigned value}] \\
[\text{label}] \quad [\text{assigned value}] \\
\qquad\qquad \text{etc.}
\end{array}
$$

1) L – The label of the ALLOC header operation itself is optional. However, each assignment operation following must have a label.

2) W – The operator of this header operation is always ALLOC, and must be present. For the subsequent assignment operation, W must be an absolute numeric value expressed either in octal or decimal. When expressed decimally, the number must be followed by the letter D, for example:

CAT $\longrightarrow$ 0100

DOG $\longrightarrow$ 512D

CHIPS $\longrightarrow$ 12

CHOPS $\longrightarrow$ 10D

3) V – The V operands of this header operation take the form name and date as illustrated. These operands are omitted for subsequent assignment operations.

2.2.3 PROGRAM HEADER (PROG)

The PROG header informs TRIM III that the operations to follow are program operations as distinguished from control operations. The PROG header must precede the first statement of a program. The PROG header operation on paper tape must always be preceded by a carriage return (see paragraph 3). A program header has the following format:

$$
\begin{array}{cccc}
L & W & V_0 & V_1 \\
\begin{bmatrix}\text{Program} \\ \text{name}\end{bmatrix} \rightarrow [\text{PROG}] & \bullet & [\text{name}] & \bullet & [\text{date}]
\end{array}
$$

1) L - The label of the PROG header operation is optional; however, when present it is considered to be the name of the program.

2) W - The operator of this header operation is always PROG and must be present.

3) V - The V operands of this header operation normally take the form name and date as illustrated. The operands are optional and completely flexible in number and length within the maximum line length.

2.2.4 CORRECTION HEADER (CORREC)

The CORREC header informs TRIM III that the operations following are source language corrections to be integrated into the source language program under assembly. A maximum of 192 correction operations is permitted for any one assembly run. Three types of correction operations are provided by TRIM III:

1) Insertions or additions.
2) Replacements.
3) Deletions.

Although the correction feature is primarily intended for use with paper tape input mode, it may be used with any combination of input modes, the only restriction being that all corrections must be read in prior to read-in of the source program.

The format of correction operations is identical to that required by the TRIM corrector (refer to the TRIM corrector description contained in this manual). Figure III-C-5 shows a sample of correction coding which may be used with the CORREC leader in the TRIM III assembler.

Corrections are always made on the basis of the sequential line identifier associated with each source program statement. This sequential identifier appears on TRIM III outputs 2, 12, and 14. If assembly consists of multiple source programs, it must be remembered that the sequential identifiers are cumulative and correction is based upon these cumulative identifiers in any given assembly. If two or more correction operations bear the same integral and fractional identifier, the last one read will supersede the preceding one with the same identifier, permitting a programmer to correct a correction. Only the last such correction will count towards the 192 maximum.

2.2.5 DEBUG DECLARATIVE

TRIM III accepts the declarative operation:

L	W	N
[label] ⟶	DEBUG ⟶	

UNIVAC
TAPE CORRECTION FORM

LABEL	OPERATOR	OPERANDS AND NOTES
MANL	→ HEADER TYPE CORREC	• W. C. Roos ⟨ 8Nov 1964
112 • 05	↙	
MANL17	→ MOVE	• 10 • MANL8 • MANL99 → INSERT CORRECTION
47 • 0	↙	
	→ ENTALK	• 501 → REPLACE CORRECTION
6 • 05	↙	
	→ BUFIN	• CHANL • MAD • 100D • MANL ↘
6 • 05	↙	↙
	→ BUFIN	• CHAN • MAD • 100 • MANL80 → CORRECTS A CORRECTION

201 •	↙	
	→ DELETE	• 18D → DELETES THIS AND NEXT 17
17 •	↙	
	→ DELETE	• → DELETES THIS ONE ONLY
315 • 05	↙	
MANL99	→ RESERV	• 8D ADDITION TO END OF PROGRAM
315 • 10	↙	
MANLAU	→	0 • ↘ ↙

315 • 15	↙	
MANLAL	→	0 • ↘ ↙
316 •	↙	↙
MANLB	→	0 • ↘ ↙
• • •	↙	↙
	→	•
•	↙	
	→	•

Figure III-C-5. Sample Correction Coding

The DEBUG operator informs TRIM III that generation is to be performed for
debugging operations contained in the source program. If the DEBUG operator
is absent, no generation will occur for such debugging operations. The DEBUG
operation when used must be loaded prior to the first PROG header. It normally
appears on the CONTR tape.

2.2.6 OUTPUT DECLARATIVE

TRIM III accepts the declarative operation:

$$[label] \rightarrow \overset{\text{W}}{\text{OUTPUT}} \quad \bullet \quad \overset{V_0}{[n]} \quad \bullet \quad \overset{V_1}{[n]} \quad \ldots \quad \overset{V_n}{[n]} \quad \overset{\text{N}}{\rightarrow}$$

The OUTPUT operation permits the user to specify the assembler outputs he
desires. The outputs are specified by number in the V_0 through V_n position.
Up to eight outputs may be requested by the OUTPUT operation. Requests in
excess of eight will be ignored and multiple OUTPUT statements are not permitted.
An example of a legal OUTPUT operation is given below.

$$\rightarrow OUTPUT \bullet 1 \quad \bullet \quad 15 \bullet 6 \bullet 2 \bullet 5 \rightarrow$$

2.2.7 DECKID DECLARATIVE

TRIM III accepts the declarative operation:

$$[label] \rightarrow \overset{\text{W}}{\text{DECKID}} \quad \bullet \quad \overset{V_0}{[name]} \quad \overset{\text{N}}{\rightarrow}$$

The DECKID operation permits the user to specify card identification on printer
or source card outputs he may select from TRIM III. From one to four alphanumeric
characters may be specified in the V_0 position. These characters together with
a 4-digit sequential octal number beginning with 0001, are added to each TRIM
III statement that is also assigned a sequential line identifier. This card
information will appear on the side-by-side printer listing output of the
program (output 12) and the punched card output in source language (output 15).
The new card identification and numbering preempts that which might be
present if the input source program is on cards. Any number of DECKID state-
ments may be inserted anywhere in the source program; however, each DECKID
operation affects only those statements following that DECKID statement, and
the card numbering will always begin with 0001.

2.2.8 ENDATA DECLARATIVE

The ENDATA operation is used with card input to TRIM III. It informs the
assembler of the end of a card deck. It does not mean the end of all input.
The ENDATA operation does not cause any object language generation. It may
have a label and notes. One blank card must follow each ENDATA card.

$$[L] \rightarrow \overset{\text{W}}{\text{ENDATA}} \rightarrow$$

2.3 MONO-OPERATIONS

Mono or one-to-one operations consist of the mnemonic function codes in the instruction repertoire and symbolic addresses, absolute machine codes, or constants. Mono-operation statements may be in one of the following formats:

2.3.1 FORMAT A

$$L \qquad\qquad W \qquad\qquad V_0$$

$$[\text{label}] \quad \longrightarrow \quad [\text{operator}] \quad \bullet \quad [\text{operand}] \longrightarrow$$

L - The label is optional.

W - The operator is the f or fm portion of the operation statement and is the mnemonic representation of the desired function code of the computer instruction repertoire.

V_0- Represents the u or k portion of the statement and may be a tag, a tag $\pm$ an integer, or an integer only. Integers may be in octal or decimal representation. When decimal representation is used, the integer must be followed by the letter D. Incrementing or decrementing of integers is not permitted. If V_0 is absent, TRIM III generates zeros for the operand without error indications.

Examples:

$\longrightarrow$ ENTAL•CAT $\longrightarrow$

$\longrightarrow$ STRADR•CAT+1 $\longrightarrow$

$\longrightarrow$ CMAL•CAT-8D $\longrightarrow$

$\longrightarrow$ ENTBK•28D $\longrightarrow$

$\longrightarrow$ ENTALK•7776 $\longrightarrow$ Minus 1 to AL

$\longrightarrow$ STOP•DOG $\longrightarrow$ DOG defined by an ALLOC opn

$\longrightarrow$ SKPOIN•7 $\longrightarrow$

$\longrightarrow$ CPAU $\longrightarrow$ Results in 506200

$\longrightarrow$ JP•LOK-10 $\longrightarrow$ LOK signifies this address

$\longrightarrow$ OUT•6 $\longrightarrow$ Output transfer channel 6

$\longrightarrow$ 0•CHEESE+1 $\longrightarrow$ Buffer terminal address

$\longrightarrow$ 0•CHEESE $\longrightarrow$ Buffer initial address

$\longrightarrow$ ENTAL   Results in 120000

$\longrightarrow$ JP •   Results in 340000

2.3.2 FORMAT B

TRIM III also accepts programs coded with absolute function codes and absolute or symbolic addressing. Normal instructions are represented by a 2-digit function code followed by a point separator and the desired u or mk operand. However, absolute instructions may also be represented by 6 consecutive digits without a point separator.

Examples:

$$\longrightarrow 12 \bullet 3505 \longrightarrow$$
$$\longrightarrow 63 \bullet CAT+6 \longrightarrow$$
$$\longrightarrow 50 \bullet 1305 \longrightarrow$$
$$\longrightarrow 50 \bullet 6200 \longrightarrow$$
$$\longrightarrow 506200 \longrightarrow$$

2.3.3 FORMAT C

Constants may be represented in a number of ways:

$\longrightarrow 7 \longrightarrow$	Results in 000007
$\longrightarrow 7 \bullet 0 \longrightarrow$	Results in 700000
$\longrightarrow 77 \longrightarrow$	Results in 000077
$\longrightarrow 77 \bullet 0 \longrightarrow$	Results in 770000
$\longrightarrow 777 \longrightarrow$	Results in 000777
$\longrightarrow 77070 \longrightarrow$	Results in 077070
$\longrightarrow 777 \bullet 7 \longrightarrow$	Illegal*
$\longrightarrow 777 \bullet \longrightarrow$	Results in 000777*
$\longrightarrow 123456 \longrightarrow$	Results in 123456

Two special mono-operations are available for the programmer's use; STOP and SKP. If either of these operators is used without a k operand, TRIM III will automatically generate an unconditional instruction of 50 56 40 or 50 50 40 respectively.

2.4 POLY-OPERATIONS

Frequently groups of instructions which perform a specific function appear iteratively in a program. A single poly-operation generates a unique sequence of instructions designed to perform some such specified function. This is the one-to-many relationship between instructions herein termed poly-coding; the parent instruction is termed a poly-operation. TRIM III provides for several poly-operations. In some cases TRIM III generates only a single instruction or,

---

*Whenever there is an expressed value following the point separator, only 1 or 2 digits are permitted in the operator position.

as in the case of REMARK and CALL operation, no instructions. It is permissible when coding a routine to intermix mono- and poly-operations in any desired order.

The CLEAR and MOVE poly-operations use the currently active B register and the MOVE poly-operation also uses AU in the generated coding. If the programmer does not wish the data in these registers to be destroyed, he must store and restore the data around a MOVE or CLEAR operation. The MOVE and CLEAR operations store and restore the programmer's special register setting. Since poly-operations generate more than one machine instruction, the tag LOK $\pm$ an integer must not be used for poly-operation coding.

2.4.1 RESERVE OPERATION (RESERV)

$$L \qquad\qquad W \qquad\qquad V_0$$

$$\begin{bmatrix} label \end{bmatrix} \longrightarrow RESERV \bullet \begin{bmatrix} Number \\ of\ words \end{bmatrix} \longrightarrow$$

The RESERV operation causes the desired number of sequential words to be reserved within a program. The operation generates the number of zero words* specified by the V_0 operand.

1) L - The label for this operation is optional.

2) W - RESERV must always be present.

3) V_0 - Specifies by an octal or decimal integer the number of zero words to be generated. V_0 may never equal zero.

   Examples:

   Assume CAT = 1000 and DOG = 2000

   CAT $\longrightarrow$ RESERV $\bullet$ 12 $\longrightarrow$ Generates zeros at addresses 1000-1011

   DOG $\longrightarrow$ RESERV $\bullet$ 10D $\longrightarrow$ Generates zeros at addresses 2000-2011

2.4.2 CLEAR OPERATION

$$L \qquad\qquad W \qquad\qquad V_0 \qquad\qquad\qquad\qquad V_1$$

$$\begin{bmatrix} label \end{bmatrix} \longrightarrow CLEAR \bullet \begin{bmatrix} Number \\ of\ words \end{bmatrix} \bullet \begin{bmatrix} starting\ address \end{bmatrix} \longrightarrow$$

The CLEAR operation clears to zero those memory addresses specified in the operation.

1) L - The label for this operation is optional.

2) W - CLEAR must always be present.

---

*TRIM III outputs 2, 12, and 14, used primarily for hard-copy debugging and documentation, reflect only the first generated zero word of each RESERV operation. All other object language outputs contain the requested number of zero words.

3) V_0- Specifies by an octal or decimal integer the number of consecutive memory locations to clear. V_0 may never exceed 4000 octal or 2048D. A V_0 of zero is not permitted.

4) V_1- Specifies the first address of the area to be cleared. The address may be expressed as an absolute octal number or as a symbolic tag plus or minus an octal or decimal integer; that is, CAT-12D or CAT-14. All the words to be cleared must be wholly contained within one memory bank.

Examples of coding for CLEAR operations are given below.

[label] ⟶ CLEAR • 18D • FLIP+12D ⟶

[label] ⟶ CLEAR • 22 • FLIP+14 ⟶

[label] ⟶ CLEAR • 100D • FLAP-5 ⟶

[label] ⟶ CLEAR • 4000 • 130000 ⟶

Examples of CLEAR operations and the absolute coding generated by the assembler are given below.

Assume EXAM1 = 1000, EXAM2 = 1006, and CAT = 10123

Input Operation				Generated Coding
EXAM1 ⟶ CLEAR • 70 • 7000 ⟶				36 0067
				75 1005
				50 7300
				41 7000
				73 1003
				50 7300
EXAM2 ⟶ CLEAR • 21D • CAT ⟶				36 0024
				75 1013
				50 7311
				41 0123
				73 1011
				50 7300

A symbolic representation of the instructions generated is given below.

⟶ ENTBK • [No. of locations -1] ⟶ Set B for No. of locations
⟶ STRSR • LOK+4 ⟶ Store current SR
⟶ ENTSR • [Bank No. of clear area] ⟶ Set SR to clear area
⟶ CLB • [First location] ⟶ Clear word at first location + B
⟶ BJP • LOK-1 ⟶ Decrement B and repeat loop
⟶ ENTSR • 0 ⟶ Return to current bank when B is zero

2.4.3 MOVE OPERATION

$$\begin{array}{cccccc}
\text{L} & \text{W} & \text{V}_0 & \text{V}_1 & \text{V}_2 \\
\left[\text{label}\right] \longrightarrow & \text{MOVE} \quad \bullet & \left[\begin{array}{l}\text{number}\\ \text{of words}\end{array}\right] \bullet & \left[\text{from address}\right] \bullet & \left[\text{to address}\right] \longrightarrow
\end{array}$$

1) L - The label for this operation is optional.

2) W - MOVE must always be present.

3) V_0- Specifies by an octal or decimal integer the number of sequential words to be moved. V_0 may never exceed 4000 octal or 2048D. A V_0 of zero is not permitted.

4) V_1- Specifies the first address of a block of data to be moved. It may be expressed as an absolute address in octal or as a symbolic tag plus or minus an octal or decimal integer. All the words to be moved must be wholly contained within one bank.

5) V_2- Specifies the first address to which the block of data is to be moved. It is expressed the same as the V_1 operand. All the destination addresses into which data are to be moved must be wholly contained within one bank.

Examples of coding for MOVE operations are given below.

$\left[\text{label}\right] \longrightarrow$ MOVE • 78D • CAT • DOG-7 $\longrightarrow$

$\left[\text{label}\right] \longrightarrow$ MOVE • 10 • HORSE+10 • COW+8D $\longrightarrow$

$\left[\text{label}\right] \longrightarrow$ MOVE • 4000 • CAT • PIG $\longrightarrow$

$\left[\text{label}\right] \longrightarrow$ MOVE • 100D • 0 • 10000 $\longrightarrow$

$\left[\text{label}\right] \longrightarrow$ MOVE • 100D • 30000 • CAT $\longrightarrow$

Examples of move operations and the absolute coding generated by the assembler are given below.

Assume EXAM3 = 1014, EXAM4 = 1024, and CAT = 1056

Input Operation	Generated Coding
EXAM3 $\longrightarrow$ MOVE • 10 • CAT • 7000 $\longrightarrow$	36 0007
	75 1023
	50 7300
	11 1056
	50 7300
	47 7000
	73 1016
	50 7300

Input Operation	Generated Coding
EXAM4 ⟶ MOVE • 100 • 12000 • CAT-100 ⟶	36 0077
	75 1033
	50 7311
	11 2000
	50 7310
	47 0756
	73 1026
	50 7300

A symbolic representation of the instructions generated is given below.

⟶ ENTBK • [No. of locations-1] ⟶ Set B for No. of words
⟶ STRSR • LOK+6 ⟶ Store current SR
⟶ ENTSR • [Bank No. of from address] ⟶ Set SR to origin bank
⟶ ENTAUB • [from address] ⟶ Get word at from address + B
⟶ ENTSR • [Bank No. of to address] ⟶ Set SR to destination bank
⟶ STRAUB • [To address] ⟶ Store word at to address + B
⟶ BJP • LOK-4 ⟶ Decrement B and repeat loop
⟶ ENTSR • 0 ⟶ Return to current bank when B is zero

2.4.4 I/O OPERATIONS

L	W	V_0	V_1	V_2	V_3
[label] ⟶ EXFCT •		[channel number] •	[AD / MAD / BK / MBK] •	[number of buffer words] •	[buffer start- ing address] ⟶
[label] ⟶ BUFIN •		[channel number] •	[AD / MAD / BK / MBK] •	[number of buffer words] •	[buffer start- ing address] ⟶
[label] ⟶ BUFOUT •		[channel number] •	[AD / MAD / BK / MBK] •	[number of buffer words] •	[buffer start- ing address] ⟶

1) L - Label for these operations is optional.

2) W - The operator must always be present.

3) V_0- Specifies the channel number expressed as an integer or a symbolic tag.

4) V_1- Specifies the buffer mode and must be present:

    a) AD - Advance without monitor.
    b) MAD - Advance with monitor.

c) BK - Back without monitor.
d) MBK - Back with monitor.

5) V_2 - Specifies as an octal or decimal integer the number of buffer words involved. Maximum of five digits.

6) V_3 - Specifies the address in memory at which buffering is to begin. V_3 may be expressed absolutely or as a symbolic tag plus or minus an octal or decimal integer.

Examples:

Assume CHAN = 07 and CAT = 10000

[label] → EXFCT • CHAN • AD • 1 • 30000 →   Generates   501307
                                                                 030001
                                                                 030000

[label] → BUFIN • 6 • MAD • 10 • CAT →   Generates   501106
                                                                 210007
                                                                 210000

[label] → BUFOUT • 3 • BK • 7 • CAT+7 →   Generates   501203
                                                                 410000
                                                                 410007

[label] → BUFOUT • 0 • MBK • 100D • CAT+99D → Generates   501200
                                                                 607777
                                                                 610143

NOTE: The examples above illustrate the fact that,
        for output and external function buffers, the
        inclusive buffer limits define a number of words
        which is one greater than the actual number of
        words to be transferred. These buffers terminate
        before transferring the word located at the ter-
        minal address.

2.4.5 LIBRARY CALL OPERATION

```
        L           W          V0         V1          Vn
    [label]  →  CALL  •    [n]     •    [n]  ···  [n]  →
```

The CALL operation permits the programmer to specify by name (label of the PROG header) the subroutines he wishes the assembler to retrieve from the library of subroutines. A single CALL operation may name up to eight such subroutines. If the user requires more than eight subroutines, he may specify them with additional CALL operations. Subroutines retrieved from the library are automatically added to the end of the source program and assembled with it. The user has complete control of their address allocation if he wishes via ALLOC operations.

Whenever a CALL operation follows the CONTR header, TRIM III will honor the calls, but the CALL operation itself will not appear on a side-by-side output listing. Only those operations following a PROG header appear on such listings. If a subroutine retrieved from the library contains CALL operations, these calls will also be retrieved and added to the end of the composite program until the last CALL operation has been honored. A request for output No. 7 causes all library CALL operations to be ignored.

The CALL operation causes no object program generation.

Examples:

⟶ CALL•TYPT•FLP•SINE•TYPC ⟶

⟶ CALL•PCHC ⟶

2.4.6 REMARK OPERATION

$$\begin{array}{ccc} L & W & V_0 \end{array}$$

$\begin{bmatrix}\text{label}\end{bmatrix}$ ⟶ REMARK • $\begin{bmatrix}\text{desired statement}\end{bmatrix}$ ⟶

The REMARK operation causes no object program generation. It is simply an aid to the programmer in expanding normal program notes.

The REMARK statement may not exceed one line or one card in length.

2.4.7 DATA OPERATION

$$\begin{array}{ccc} L & W & V_0 \end{array}$$

$\begin{bmatrix}\text{label}\end{bmatrix}$ ⟶ DATA • $\begin{bmatrix}\text{integer, binary point specification}\end{bmatrix}$ ⟶

The DATA operation allows the programmer to specify a positive or negative data integer and its binary point position. The bits are numbered from right to left 0-17D. The binary point specification must be separated from its associated integer by a comma. The absence of a minus sign implies a positive integer. The label is optional.

Examples:

$\begin{bmatrix}\text{label}\end{bmatrix}$ ⟶ DATA • 24D, 9D ⟶   Generates 030000

or

$\begin{bmatrix}\text{label}\end{bmatrix}$ ⟶ DATA • 30, 11 ⟶   Generates 030000

The binary representation is:

17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0

$[label]$ → DATA • 123,4 → Generates 002460

The binary representation is:

17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	1	0	1	0	0	1	1	0	0	0	0

2.4.8 PUNCH CONTENTS OPERATION (PCHC)

$$L \qquad\qquad W \qquad\qquad V_0$$

$[label]$ → PCHC • $\begin{bmatrix} \text{information to be punched} \\ \text{and/or typewriter commands} \end{bmatrix}$ →

The PCHC operation results in generated coding which, when run on the computer with the PCHC* subroutine, causes the octal contents of A, AU, AL, B or any memory location to be punched on the high-speed paper tape punch. The words to be punched may be interspersed with the following typewriter control symbols to provide subsequent listing in the desired format.

Operand	Performance
• ∣CR∣ •	carriage return, line feed
• Δ • or • ∣SP∣ •	space

The vertical bars indicate the information enclosed is a special symbol directing the typewriter. Each CR and SP must begin and end with the vertical bar. Controls are separated from other operands by point separators.

1) L - The label is optional.

2) W - The operator PCHC must be present.

3) V_0- Specifies the operands in the order in which they are to be punched. Except for the typewriter commands, all operands imply their contents are to be punched. Such operands may be A, AU, AL, active B, a tag or a tag ± an absolute value, or an absolute address.

Examples:

CAT → PCHC • A • Δ • 7070 • Δ • DOG-11D • ∣CR∣ →

→ PCHC • DOG • Δ • B • AL →

→ PCHC • ∣CR∣ • AU • ∣SP∣ • AL →

→ PCHC • DOG+10 • ∣SP∣ • CAT →

---

*See paragraph 3.4 6).

2.4.9 PUNCH TEXT OPERATION (PCHT)

$$\begin{array}{ccc} L & W & V_0 \end{array}$$

$$[\text{label}] \longrightarrow \text{PCHT} \bullet [\text{text and/or typewriter commands}] \longrightarrow$$

The PCHT operation results in generated coding which, when run on the computer with the PCHT* subroutine, causes the text and/or typewriter commands in the V_0 operand position to be punched by the high-speed paper tape punch. The text may be interspersed with the following typewriter control symbols as desired; each CR and SP must be set off between two vertical bars.

Operand	Performance
|CR|	carriage return, line feed
Δ or |SP|	space

1) L - The label is optional.

2) W - The operator PCHT must be present.

3) V_0- Is the text to be punched interspersed with typewriter commands desired by the programmer. If the text is too long for one PCHT operation, the programmer can write successive operations.

    Examples:

    CAT $\longrightarrow$ PCHT $\bullet$ PROFIT $\quad\Delta\quad$ AND $\quad\Delta\quad$ LOSS $\quad\Delta\quad$ FOR $\longrightarrow$

    $\longrightarrow$ PCHT $\bullet$ JULY $\quad\Delta\quad$ 10, $\quad\Delta\quad$ 1967 $\quad$ |CR|$\longrightarrow$

    NOTE: Point separators are not required within V_0; they will be punched if present.

2.4.10 TYPE CONTENTS OPERATION (TYPC)

$$\begin{array}{ccc} L & W & V_0 \end{array}$$

$$[\text{label}] \longrightarrow \text{TYPC} \bullet \begin{bmatrix} \text{information to be typed and/or} \\ \text{typewriter commands} \end{bmatrix} \longrightarrow$$

The TYPC operation results in generated coding which, when run on the computer with the TYPC* subroutine, causes the octal contents of A, AU, AL, current B, or any memory location to be typed on the typewriter. The words to be typed may be interspersed with the following typewriter commands.

---

*See paragraph 3.4 6).

Operand	Performance
• |CR| •	carriage return, line feed
• Δ • or • |SP| •	space (may be used for formatting)

The vertical bars indicate the information enclosed is a special symbol directing the typewriter. Each, CR or SP must begin and end with a vertical bar.

1) L - The label is optional.

2) W - The operator TYPC must be present.

3) V_0 - Specifies the operands in the order in which they are to be typed. Except for the typewriter commands, all operands imply their contents are to be typed. Such operands may be A, AU, AL, active B, a tag or a tag $\pm$ an absolute value, or an absolute address.

Examples:

CAT ⟶ TYPC • A • Δ • Δ • 7070 • Δ • Δ • DOG-11D • \|CR\| ⟶

⟶ TYPC • AU • Δ • AL • \|SP\| • B • HORSE ⟶

2.4.11   TYPE TEXT OPERATION (TYPT)

$$\begin{array}{ccc} L & W & V_0 \end{array}$$

[label]   ⟶   TYPT •   [text and/or typewriter commands] ⟶

The TYPT operation results in generated coding which, when run on the computer with the TYPT* subroutine, causes the text and/or commands in the V_0 operand position to be typed by the typewriter. The text may be interspersed with the following typewriter commands:

Operand	Performance
|CR|	carriage return, line feed
Δ or |SP|	space (may be used for formatting)

1) L - The label is optional.

2) W - The operator TYPT must be present.

3) V_0 - Is the text to be typed interspersed with typewriter commands. If the text is too long for one TYPT operation, the programmer may use successive operations to complete the text.

---

*See paragraph 3.4 6).

Examples:

CAT ⟶ TYPT • PROFIT |SP| AND △ LOSS △ FOR ⟶

⟶ TYPT • JULY △ 10, △ 1967 |CR| ⟶

NOTE: Point separators are not required within V_0; they will be typed if present.

2.4.12 DOUBLE SET OPERATION

L   L

[label] ⟶ DBLSET ⟶

The DBLSET operation insures that the Y of a double add or subtract instruction is located at an even address. The DBLSET operation is normally followed by a Y constant. TRIM III examines the address to which the following constant (or instruction) would normally be assigned. If the address is odd, a word of zeros is first generated to insure that the constant (or instruction) will be assigned to an even address. If the address is even, no generation results.

2.4.13 SETSR OPERATION

L   W V_0

[label] ⟶ SETSR • [alphanumeric tag] ⟶

The SETSR operation enables the programmer to place responsibility for setting k of an ENTSR instruction upon TRIM III. Based upon an ALLOC operation or the assembled address of the referenced tag, TRIM III generates an ENTSR instruction (5073 k) with the proper k value for each SETSR operation.

1) L - Label is optional.

2) W - SETSR must be present.

3) V_0- Must be an alphanumeric tag corresponding to a program label or an allocated value. The tag may not be incremented or decremented.

Examples:

Assume CAT is a label at 36421 and DOG is a label at 70460 and COW is allocated to 010000, then:

⟶ SETSR • CAT ⟶ Generates 507313

⟶ SETSR • DOG ⟶ Generates 507317

⟶ SETSR • COW ⟶ Generates 507310

2.5 DEBUGGING OPERATIONS

TRIM III provides two debugging operations for punching a paper tape output of either the contents of registers AU, AL, and current B, or the contents of specified sequential memory locations. TRIM III recognizes these operations only if a DEBUG declarative operation is read prior to the first PROG header operation. When recognized, these operations generate a set of three or five instructions in the object program which, when run on the computer with the DEBUG* subroutine, produce the desired dump. Each set of instructions is assigned a sequential identifying number which appears with each punched output, thereby enabling programmer recognition of repeated times through given coding paths. The debugging operations take the following form.

$$
\begin{array}{lll}
\text{L} & \text{W} & \text{N} \\
[\text{label}] \longrightarrow & \text{DUMPR} \longrightarrow &
\end{array}
$$

$$
\begin{array}{lllll}
\text{L} & \text{W} & V_0 & V_1 & \text{N} \\
[\text{label}] \longrightarrow & \text{DUMPM} \bullet & \begin{bmatrix}\text{number of} \\ \text{words to dump}\end{bmatrix} \bullet & \begin{bmatrix}\text{address of first} \\ \text{word to dump}\end{bmatrix} \longrightarrow &
\end{array}
$$

1) L - Label is optional.

2) W - DUMPR or DUMPM must always be present.

3) V_0- Applicable to the DUMPM operation only. Specifies the total number of memory locations to be dumped. The number may be expressed in octal or in decimal followed by the letter D.

4) V_1- Applicable to the DUMPM operation only. Expresses the address of the first word to be dumped. It may be expressed as an integer or a tag plus or minus an integer.

Examples of coding for DUMPR or DUMPM operations are given below.

$$
\begin{array}{llll}
[\text{label}] \longrightarrow & \text{DUMPR} \longrightarrow & & \\
[\text{label}] \longrightarrow & \text{DUMPM} \bullet & 12 \bullet & 10000 \longrightarrow \\
[\text{label}] \longrightarrow & \text{DUMPM} \bullet & 10D \bullet & 10000 \longrightarrow \\
[\text{label}] \longrightarrow & \text{DUMPM} \bullet & 10D \bullet & \text{CAT+28D} \longrightarrow \\
[\text{label}] \longrightarrow & \text{DUMPM} \bullet & 12 \bullet & \text{CAT-15} \longrightarrow \\
[\text{label}] \longrightarrow & \text{DUMPM} \bullet & 64D \bullet & \text{CAT} \longrightarrow
\end{array}
$$

---

*See paragraph 3.4 6).

Examples of the DUMPR and DUMPM operations and the coding generated by the assembler are given below.

Examples:

Assume EXAM5 = 1000, EXAM6 = 1050, and DEBUG = 30000.

Input Operation	Generated Coding
EXAM5 ⟶ DUMPR ⟶	301001
	030000
	000001
EXAM6 ⟶ DUMPM • 5 • 10000 ⟶	301051
	030000
	400002
	000005
	010000

A symbolic representation of the instructions generated is given below. The first three instructions apply to both DUMPR and DUMPM. The last two instructions apply to DUMPM only.

⟶ IRJP • DEBUG ⟶ Indirect return jump to DEBUG
⟶ 0 • DEBUG ⟶ Address of DEBUG
⟶ X • [Y]   X = 0 for DUMPR, 4 for DUMPM
            Y = No. of DUMPR or DUMPM operation
                in this program
⟶ [No. of words] ⟶ No. of words to be dumped
⟶ [First address] ⟶ Address of first word to be dumped

Both DUMPR and DUMPM operations preserve existing values in AU, AL, and the current B register.

TRIM III also provides two additional debugging operations for programmer use: DSTOP and DTYPT.

L   W

[label]   ⟶   DSTOP ⟶

The DSTOP operation permits the programmer to intersperse strategic debugging stops within his program. If the DEBUG operator is read in the assembly prior to the PROG header, the DSTOP will generate an unconditional stop (505640); otherwise, TRIM III will ignore the operation.

L   W V_0

[label]   ⟶   DTYPT • [text and/or typewriter commands] ⟶

The DTYPT operation performs the same function as the TYPT operation. If the DEBUG operator is read in the assembly prior to the PROG header, TRIM III will perform the generation; otherwise, the operation will be ignored.

2.6 TRIM III OUTPUTS

TRIM III provides 13 different outputs of the assembled and/or source program. The user selects his outputs in accordance with his needs and the available peripheral devices.

The available outputs are listed under the output device on which they are produced.

Monitoring typewriter:

    No. 1 – Program summary consisting of the number of memory locations used and inclusive addresses.

Paper tape punch: Except for outputs 6 and 11, all paper tapes are loadable via the utility packages. Outputs 6 and 11 may be used as input to TRIM III.

    No. 2 – Absolute assembled program, sequential line identifier, source program, and assembly error alarms when applicable. This is a side-by-side listing in source code preceded by a program summary consisting of the number of memory locations used and inclusive addresses.

    No. 3 – Absolute assembled program in source code, consisting of a carriage return, 88, carriage return, addresses and instructions, a carriage return, double period and checksum.

    No. 4 – Absolute assembled program in bioctal format, consisting of a 76 code, inclusive area addresses followed by the instructions only, and a checksum.

    No. 5 – Relocatable assembled program in bioctal format starts with a 75 code followed by the assembled program relative to base 00000, and terminates with a checksum. The output tape may be loaded starting at any desired memory location.

    No. 6 – Allocation output in source code consisting of an ALLOC header, followed by all program tags and labels and addresses in allocation format.

    No. 11– The source program only, produced in source code.

High-speed printer:

    No. 12– Absolute assembled program, sequential line identifier, deck and card number if applicable, source program, and assembly error alarms when applicable. This is a side-by-side listing suitable for hard-copy editing and documentation.

No. 14 - This is the same as output No. 12 except that there is no card information and page size is assumed to be 11 inches wide by 8½ inches long.

Card processor:

No. 13 - Relocatable assembled program on Hollerith-coded 80-column cards. The first card contains only the base load address. Subsequent cards contain up to 8 computer words, a cumulative checksum, and a card sequence number.

No. 15 - Source program only, on Hollerith-coded 80-column cards. Each card contains one TRIM III statement as well as any card deck identification and sequence number.

Magnetic tape unit:

Relocatable object program. Assembled object program in assemble table 3 format.

No. 10 - During assembly TRIM III automatically produces this output on the magnetic scratch tape. The tape data can be loaded into the computer memory absolutely or relocated to any specified base address by the utility packages.

No. 16 - Source program on magnetic tape. This output does not include declarative operations such as ALLOC, OUTPUT, or DECKID. Output No. 16 may be used as input to TRIM III.

Miscellaneous:

No. 7 - Output No. 7 is not itself an output, but does affect all other requested outputs, since it causes TRIM III to ignore all library CALL operations of the input program.

3. PROGRAMMING PROCEDURES

3.1 PAPER TAPE INPUT FORMAT

Two versions of TRIM III are available; one version accepts a source program paper tape prepared in field data code, the other version accepts a source program paper tape prepared in ASCII code (refer to Appendix A, Tables A-2 and A-3).

Each source tape must begin with a carriage return and terminate with a carriage return and two periods.

3.1.1 KEYBOARD CORRECTION METHODS

Typing-error correction procedures have been incorporated in both versions of the TRIM I, TRIM II, and TRIM III assemblers, and the TRIM corrector for deleting immediate keyboard errors that might be made in the preparation of input tapes for these programs on the UNIVAC 1232 and 1532 I/O consoles. These procedures are described under TRIM I, paragraph 8.

3.2 80-COLUMN CARD INPUT FORMAT

For those installations whose peripheral equipment configuration includes an on-line card reader, TRIM III accepts source programs prepared in Hollerith code on standard 80-column cards as well as source programs prepared on paper tape. The two input types may be intermixed.

Basically the coding format is similar for either card or paper tape unit. Interpretation of coding separator symbols for card input is given in Table III-C-2.

TABLE III-C-2. CODING SYMBOLS FOR CARD INPUT

Symbol		Key	Rows Punched
→	(start statement)	SKIP	(none)
→	(start notes)	≡ ≡ ≡	4,8 4,8 4,8
↙	(carriage return)	REL	(none)
|	(vertical line)	$ *	11, 3, 8
•	(point separator)	$ *	11, 4, 8
∘	(period)	• (	12, 3, 8
,	(comma)	')	0, 3, 8
+	(plus)	+ P	12
–	(minus)	SKIP	11

The straight coding arrow is interpreted according to its format position; it represents a SKIP key at the beginning of a statement and three dashes at the end of a statement. The point separator is represented by the * key in all card input.

Card control: The ENDATA operation card followed by one blank card denotes the end of a card input deck.

Coding format: The examples on the top of page III-C-30 illustrate the basic coding format for card input: (Also see Figure III-C-6).

CODING FORM

PROGRAMMER ___ EXT ___ MS.

PLT. ___

TITLE __PAN__ PAGE __1__ of __1__

CARD	INS	LABEL	OPERATOR	OPERANDS AND NOTES
0000		PAN	→ PROG	● CMB ● 4 OCTOBER 1964 XS-3 TO OCTAL INTEGER CONVERSION
0001	P	PAN	→ 0	●
0002	A		→ ENTAU	● PTMP XS-3 INTEGER TO AU
0003	N		→ ENTBK	● 3 SHIFT INDEX
0004	S		→ ENTALK	● 0
0005	K	PAN1	→ LSHA	● 6
0006	DECK		→ JPALZ	● PAN2
0007			→ ADDALK	● 7774 CHANGE TO 6-BIT BCD
0008			→ LSHAL	● 14
0009	P	PAN2	→ BJP	● PAN1
0010	A		→ ENTBK	● 2 SET INDEX FOR 3 CHARACTERS
0011	N	PAN3	→ ENTAL	● INCR CONVERSION LOOP
0012	S		→ LSHAL	● 2
0013	K		→ ADDAL	● INCR (INCR) INITIALLY CLEARED
0014	DECK		→ LSHAL	● 1
0015			→ STRAL	● PTMP
0016			→ ENTALK	● 0
0017	P		→ LSHA	● 6
0018	A		→ ADDAL	● PTMP
0019	N		→ STRAL	● INCR
0020	S		→ BJP	● PAN3
0021	K		→ IJP	● PAN CONVERTED VALUE IN AL AND INCR
0022	DECK	PTMP	→ 0	● TEMPORARY
0023		INCR	→ 0	● INTEGER WORD
0024			→ ENDATA	●

Figure III-C-6. Typical Coded Programmer Card Input

DECK ID	CARD NO.	INS NO.	L	W	V	N
B017	0005		CAT4 ——► ENTAUB • DOG5—►MASK FOR SEARCH			
B017	0005	05	——► CMSK • CAT10			
B017	0006		——► JPNOT • LOK-3 ——► LOOK AGAIN			

Card Format: Card format uses card columns 1-4 for deck identifier, 5-8 for
card number, 9-10 for card insert number, 11-16 for the label,
and 21-80 for statement and notes.

1 ◄——————► 4	5 ◄——————► 8	9 ◄——► 10	11 ◄——————► 16		21 ◄—————————————————
Deck Id	Card No.	Ins. No.	Label		Statement – – – Notes
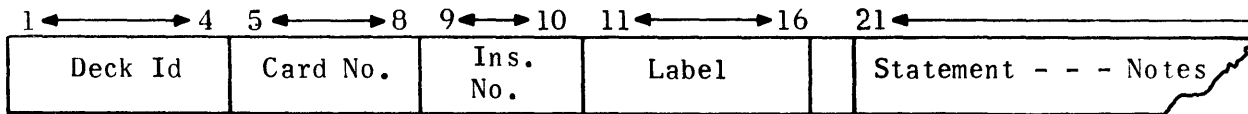
Three dashes (---), punched code 4,8, always follow the statement whether or
not there are notes. The REL (release) key terminates the card. TRIM III
makes no provision for the statement and notes to overflow one card. Any
attempt to continue notes on a second card results in improper generation
for that card.

The column-skip feature on the key punch provides a convenient means to bypass
unused columns reserved for the label. The keypunch operator begins a label
with column 11 and skips any unused columns between the end of the label and
column 21. If no label is present, the operator depresses the SKIP key and
the card is automatically positioned at column 21. The statement always begins
at column 21. Figure III-C-7 shows a typical input operation in punched card
format.
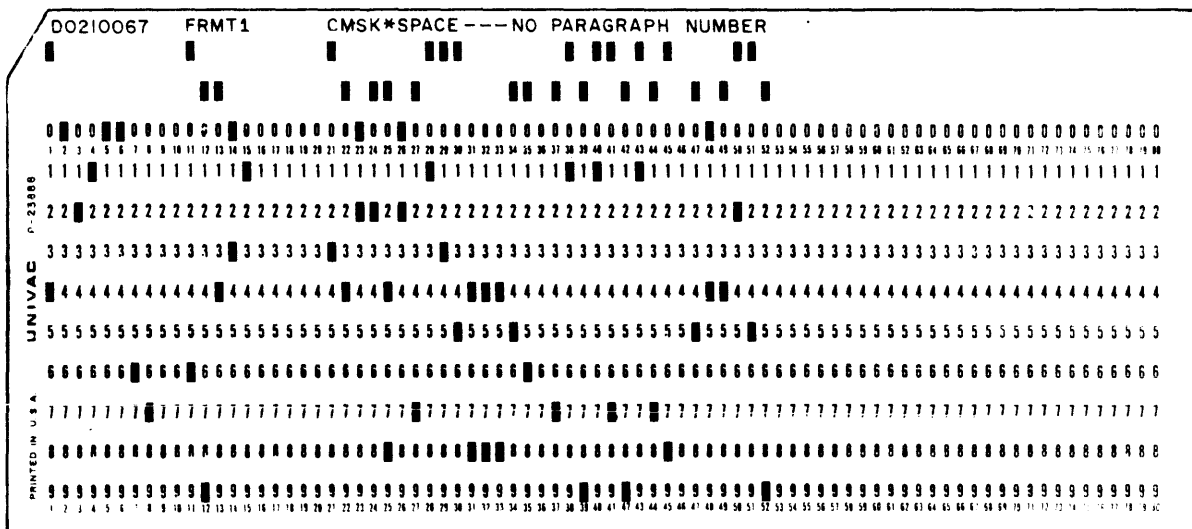
Figure III-C-7. Typical Punched Card Input Operation

3.3 MAGNETIC TAPE INPUT FORMAT

TRIM III accepts source programs recorded on magnetic tape in the following formats:

1) TRIM III Output No. 16 format.

2) All Magnetic tape outputs from the Card-to-Tape (CART) Processor Program.

The TRIM III Output No. 16 consists only of source program operations; therefore, any required declarative operations (for example, OUTPUT, ALLOC or DECKID) must be provided by the user via cards as a separate tape. The output of CART includes all operations contained on the card deck used in the card-to-tape operation.

3.4 SOURCE PROGRAM CORRECTIONS

When so directed by a CORREC header operation, TRIM III will perform source program corrections in conjunction with an assembly run. Outputs from the assembly will include the requested corrections. The following rules govern the use of the correction capability:

1) A maximum of 192 corrections per assembly is allowed.

2) Corrections must follow the CORREC header.

3) All corrections must be loaded prior to the loading of the source program to be corrected.

4) Corrections need not be in any special order. TRIM III will sort the correction items prior to merging them with the source program.

5) If two or more correction operations bear the same integral and fractional value, the last such operation overrides. This permits programmers to correct an erroneous correction.

6) Corrections are based on the assembler-assigned sequential line identifier for each source statement (see paragraph 2.2.4).

7) The integral and fractional portions of the correction identifier must be expressed in octal notation only. The integer is limited to a maximum of 6 digits; the fraction (which indicates insertion or addition) is limited to 3 digits. The fraction is a straight binary magnitude, (for example, the correction identifiers 12•2, 12•20, and 12•200 all have the same value).

8) Corrections may be prepared on punched cards or punched paper tape.

3.4.1 PAPER TAPE CORRECTION FORMAT

The format of correction operations prepared on paper tape is identical to
that required by the TRIM corrector (refer to input formats in the TRIM
corrector description contained in this manual) with the following exceptions:

1) Corrections must follow a CORREC header.

2) A maximum of six integral digits is permitted.

3.4.2 CARD CORRECTION FORMAT

Special formatting rules apply to correction operations prepared on 80-column
punched cards. Except for the CORREC header each correction action requires
two cards. The first card contains the integer and fraction of the sequential
identifier while the second card contains the correction operation itself. If
the correction operation is an insertion or replacement, the first ten columns
of this second card may also contain card identifications which are included in
the outputs 12 and 15 of the assembled corrected program, unless a DECKID
operator is used.

On the first card, the integral portion of the sequential identifier must begin
in column 11. A point separator (asterisk) must not be used. The fractional
portion, if any, must begin in column 21 and must be terminated with the con-
ventional three dashes. If there is no fraction, one zero code followed by the
three dashes must still be punched beginning at column 21.

An ENDATA card followed by a blank card must always follow the last correction
card even if other cards are to follow in the assembly.

The following example of card correction format makes changes to the program
illustrated in Figure III-C-8.

Column:	11	21
Card 9		
Card 8		ENDATA ---
Card 7	LIBX00191	0*77 ---
Card 6	22	1 ---
Card 5		DELETE *3
Card 4	14	0 ---
Card 3	LIBX0011 CAD	0*1000 --- Current Address
Card 2	12	0 ---
Card 1	BITSUM	CORREC ---

===
                                OUTPUT 12
===

MEM. STRG. USED   11061
00240 THRU 00404
01000 THRU 07743
20000 THRU 21747

LOK	INSTR	L1	ID	DECK	CARD	LO		
		0	LIBX	0001	BITSUM	PROG*JRS*6NOV64	FLEX	
00240	34 0364	1	LIBX	0002	BITSUM	JP*SEOK	LIBBLD BOTTSTRAP LOAD	
00241	76 0355	2	LIBX	0003	UPAKX	RJP*ERP		
00242	40 0247	3	LIBX	0004	UPAK	CL*CHECK	CLEAR ACCUM CKSUM	
00243	42 0254	4	LIBX	0005		STRB*BNASTY	SAVE B	
00244	76 0345	5	LIBX	0006	UPAK1	RJP*RF	READ FRAME	
00245	61 0244	6	LIBX	0007		JPALZ*UPAK1	IGNORE LEADER	
00246	34 0270	7	LIBX	0008		JP*BILD		
00247	00 0000	10	LIBX	0009	CHECK	0*	ACCUM CKSUM	
00250	00 0000	11	LIBX	0010	SUM	0*	ZERO CONST	
00251	00 0000	12	LIBX	0011	CAD	0*	CURRENT ADDR	
00252	00 0000	13	LIBX	0012	FAD	0*	FINAL ADDR	
00253	00 0076	14	LIBX	0013	BIO	76*	BIOCTAL CODE	
00254	00 0000	15	LIBX	0014	BNASTY	0*	BKEEPER	
00255	00 0000	16	LIBX	0015	BUWD	0*	DATA I/O BUFFER	
00256	00 0000	17	LIBX	0016		0*		
00257	00 0000	20	LIBX	0017		0*		
00260	00 0000	21	LIBX	0018		0*		
00261	00 0000	22	LIBX	0019		0*		
00262	00 0000	23	LIBX	0020		0*		
00263	00 0000	24	LIBX	0021		0*		
00264	00 0000	25	LIBX	0022		0*		
00265	00 0000	26	LIBX	0023		0*		
00266	00 0000	27	LIBX	0024		0*		
00267	00 0000	30	LIBX	0025		0*		
00270	40 0247	31	LIBX	0026	BILD	CL*CHECK		
00271	10 0250	32	LIBX	0027		ENTAU*SUM	CL AU	
00272	76 0335	33	LIBX	0028		RJP*BILD7	6 DIGITS OF ADDRS	
00273	50 4717	34	LIBX	0029		LSHA*17		
00274	46 0251	35	LIBX	0030		STRAU*CAD		
00275	10 0250	36	LIBX	0031		ENTAU*SUM		
00276	50 4703	37	LIBX	0032		LSHA*3		
00277	70 0001	40	LIBX	0033		ENTALK*1		
00300	74 0336	41	LIBX	0034		STRADR*BILD7+1		
00301	76 0335	42	LIBX	0035		RJP*BILD7	GET REST OF ADDRS	
00302	44 0252	43	LIBX	0036		STRAL*FAD		
00303	70 0002	44	LIBX	0037		ENTALK*2		
00304	74 0336	45	LIBX	0038		STRADR*BILD7+1		
00305	76 0335	46	LIBX	0039	BILD2	RJP*BILD7		
00306	32 0251	47	LIBX	0040		ENTB*CAD		

Figure III-C-8. TRIM III Output 12 from Card Input

3.5 GROUND RULES

Regardless of the input format, there are certain conventions which the pro-
grammer must bear in mind when coding for TRIM III.

1) No label may exceed six characters. The label must not begin with a
   number, the letter O, nor may it consist only of letters LOK. The label
   may never contain a +, -, comma, or point separator code.

2) The maximum size program which TRIM III can assemble is limited only by
   the number of memory locations above address 13000$_8$ used for label/tag
   storage (3 words per label or tag).

3) Each break in addressing sequence constitutes a program area. A total
   of 64 such areas is permitted.

4) TRIM I operators SETADR and EQUALS are ignored by TRIM III. The ALLOC
   operation replaces these two functions.

5) When specifying a decimal integer, the letter D occupies one digit position:
   therefore, the maximum decimal integer that can be expressed is 99999D.

6) Assembler support subroutines TYPT, TYPC, PCHT, PCHC, and the debugging
   package, DEBUG, are included in the TRIM III library of subroutines. The
   programmer uses a CALL operation to retrieve them from the library. The
   programmer may allocate these subroutines through normal ALLOC operations.
   If he does not allocate them, TRIM III will assign them sequential
   addresses immediately following the principal program. If these sub-
   routines are not assembled with the principal program and the programmer
   has not provided for their allocation, TRIM III will arbitrarily assign
   all references to them to the following fixed addresses:

TYPT	17000
TYPC	17160
PCHT	16400
PCHC	16560
DEBUG	17470

   Each of these five subroutines uses the tag CHAN for all I/O instructions.
   It is the programmer's responsibility to provide an ALLOC operation
   equating CHAN to the appropriate I/O channel.

7) If a program contains ADDA, ADDAB, SUBA, or SUBAB instructions, regardless
   of whether or not a DBLSET operation was used, the following restrictions
   shall apply to loading a TRIM III output No. 5, 10, or 13 into computer
   memory:

   a) If the program was assembled starting at an even address, it must be
      loaded starting at an even address.

b) If the program was assembled starting at an odd address, it must be loaded starting at an odd address.

8) TRIM III informs the user of a duplicate label via a typeout on the on-line typewriter. The typeout includes the sequential line identifier, the warning, DUP LBL, and the label name. Except for the warning typeout TRIM III will normally ignore duplicate labels equating all references to the address of the first such label. However, if the user has allocated a label which is in fact a duplicate, that allocation is lost with unpredictable results in address assignment.

9) Any program to be assembled by TRIM III must be assembled for only one 32K segment of memory, either 000000 through 077777 or 100000 through 177777. This means that the assembled program must reside in one 32K segment or the other, but not both. This does not preclude inter-segment references which would be implemented exactly as for inter-bank references. The TRIM assemblers do not provide any alarm indications for this condition.

4. TRIM III LOADING AND OPERATING PROCEDURES

4.1 BASIC INFORMATION

TRIM III is a magnetic-tape-stored assembly system which accepts source programs written with absolute or mnemonic function codes and symbolic addressing and produces assembled output programs suitable for loading into the computer and/or hard copy editing and documentation. TRIM III has been designed to fit the channel and equipment configuration of the center in which it is used. The assembler provides the user with a simple means for selecting three optional modes of input, and each mode is represented by a number code.

Input Mode	Number Code
Cards	000001
Paper Tape	000002
Magnetic Tape	000003

Each TRIM III has one normal mode built into it (the one prevailing at the center where it is used). If only the normal mode is required, the user need not concern himself with the modes at all. However, if different input modes are to be used (for example, card and paper tape in combination) TRIM III users must familiarize themselves with the input mode codes and their use.

Prior to loading and operating TRIM III, the computer and the I/O equipment (magnetic tape unit, I/O console or card processor) must be placed in the operational state with all switches in the normal operating position.

4.2 LOADING TRIM III

1) For installations possessing a magnetic tape wired bootstrap:

   a) Mount the TRIM III assembler tape on magnetic tape cabinet 1,
      transport 1, and set the corresponding write enable button on
      the magnetic tape control panel.

   b) At the computer control panel, press MASTER CLEAR, LOAD, and
      START. The TRIM III executive will be loaded into memory and the
      computer will stop with P = 01404.

2) For installations possessing a paper tape wired bootstrap:

   a) Mount the TRIM III assembler tape on magnetic tape cabinet 1, trans-
      port 1, and set the corresponding write enable button on the magnetic
      tape control panel.

   b) Mount the TRIM III paper tape loader in the paper tape reader.

   c) At the computer control panel, press MASTER CLEAR, LOAD, and START.
      The TRIM III executive will be loaded into memory and the computer
      will stop with P = 01404.

4.3 INITIALIZING TRIM III

When the computer stops with P = 01404, it is necessary to identify the magnetic
tape configuration to be used for assembly. This identification need be made
only once for all subsequent assemblies unless it is necessary to change the
configuration. TRIM III expects tape information in the format OOXXCT where XX
is the channel number (bits 6 through 11), C is the tape cabinet number (bits
3 through 5), and T is the transport number (bits 0 through 2). Thus, 001512
represents channel 15, cabinet 1, transport 2, and 000112 represents channel
1, cabinet 1, transport 2.

The procedures required to initialize TRIM III are as follows:

1) Set the AU register to the number (OOXXCT) which identifies the location
   of the assembler tape.

2) Set the AL register to the number (OOXXCT) which identifies the transport
   to be used as the magnetic scratch tape.

3) Start the computer. The computer stops with AU and AL cleared.

4) If only two transports are to be used in the assembly, start the computer.
   When the computer stops with P set to 01400, TRIM III is ready for use
   (refer to paragraph 4.4).

5) If more than two transports are to be used in the assembly, perform the
   procedures below.

   a) If source input is to be from magnetic tape, set the AU register to
      the number (OOXXCT) which identifies the transport to be used for
      source input.

b) If a magnetic tape output other than output No. 10 is to be requested, set the AL register to the number (OOXXCT) which identifies the transport on which the output is to be produced. Output No. 10 is always produced on the scratch tape.

c) Start the computer. When the computer stops with P set the 01400, TRIM III is ready for use (refer to paragraph 4.4).

If, at any time during assembly, the user wishes to change the magnetic tape configuration, he may do so by setting P to 01404 and repeating the initialization procedures.

4.4 USING TRIM III

1) For paper tape input, mount the source tape in paper tape reader.

2) For card input, initialize the card reader.

3) For magnetic tape input, mount the input tape as follows:

a) If two tape transports are being used, mount the input tape on the scratch tape transport.

b) If three or four transports are being used mount the input tape on the transport designated for the input tape. Since this transport is used only for magnetic tape input, no warning typeout occurs.

4) Master clear.

5) Set P = 01400.

6) Set PROGRAM SKIP key 1.

7) For error typeout suppression, set PROGRAM SKIP key 3.

8) If source input is other than the normal mode, set AL to the proper code. (For normal mode, AL remains equal to zero.)

9) Start the computer.

10) TRIM III prepares to read input. If further operator action is necessary to continue assembly of the input currently being read, the computer stops after an operator instruction typeout occurs on the on-line typewriter. Refer to paragraph 4.5 to determine the operator action required.

11) When the current read-in is complete, the computer stops with AL equal to zero. At this time the operator must perform one of the following:

a) If additional input is required, mount the source input on the input device as directed in 1), 2), or 3) above and restart the procedure at step 8) above.

      b) If no additional input is required, release PROGRAM SKIP key 1 and
           start the computer. TRIM III begins assembling the program. If
           further operator action is necessary, the computer stops after an
           operator instruction typeout (refer to paragraph 4.5).

12) If for any reason the operator wishes to abort an output during process-
     ing, he must perform the following procedures:

     a) Stop the computer.

     b) Master clear the computer.

     c) Set the P register to 01401.

     d) Start the computer. TRIM III begins processing the next requested
         output. If all outputs have been processed, the typeout SELECT
         OUTPUTS IN A will occur.

NOTE: Since TRIM III includes correction features, it is possible to correct
       a source program and assemble it for the desired new outputs in the
       same assembly run. Although this feature is intended primarily for the
       paper tape input mode, it may be used with any combination of input
       modes. Correction tapes or cards must be read in before the source
       program(s). If the assembly consists of multiple source programs, it
       must be remembered that the sequential line identifiers are cumulative
       and the corrections are based upon these identifiers in any given
       assembly. The input of the corrections and source program proceeds as
       in a normal assembly run. After the corrections have been read in, the
       source program(s) must follow with PROGRAM SKIP key 1 still set. Any
       outputs selected will contain the requested corrections including source
       program outputs 15 (cards), 11 (paper tape), and 16 (magnetic tape).

4.5 OPERATOR INSTRUCTION TYPEOUTS

TRIM III contains limited error detection capability. The majority of programmer
errors are handled internally. However, it is desirable when practicable to
permit the user to take corrective action during the assembly process in order
to achieve an accurate assembly. The headings of the subparagraphs which follow
are instruction typeouts that may occur during assembly. Information given
in each subparagraph directs operator actions required by the typeout. When
PROGRAM SKIP key 3 is set, certain typeouts are suppressed.

4.5.1 SET KEY 1

This typeout will occur if the user has not set PROGRAM SKIP key 1 at the start
of the assembly process. To correct, set PROGRAM SKIP key 1 and start again.
PROGRAM SKIP key 3 has no effect on this error typeout.

4.5.2 IDENT. MTUS IN A

This typeout occurs when no magnetic tape configuration identification has been made prior to the first assembly. To correct this condition, perform the following steps:

1) Identify magnetic tape units exactly as outlined in paragraph 4.3 of this section.

2) Start the computer. PROGRAM SKIP key 3 has no effect on this error typeout.

4.5.3 IDENTIFY TAPE JOB IN AL

This typeout occurs when the magnetic tape input to be read is in the format produced by the CART (card-to-tape) program. Since this format may contain more than one source program on the same tape, the operator must identify the program to be read from tape by performing the following actions:

1) Set the AL register to the number which identifies the position of the program on the tape (1 for the first program, 2 for the second program, and so forth).

2) Start the computer. TRIM III directs the tape unit to pass tape until the selected program is reached and then begins reading the input program. If two or more programs on the same tape are to be assembled together, these procedures are repeated for each program.

4.5.4 REMOVE INPUT TAPE TO SAVE

This typeout occurs when input has been read from the magnetic tape transport identified as the scratch transport. The output No. 10 will also be written on this transport during the assembly process. Therefore, if the input tape is to be saved, the operator must change the tape on the scratch transport before the output No. 10 is produced. This typeout occurs each time a magnetic tape read-in is completed; therefore, if more than one input program is read from the same magnetic tape, the operator must remove the tape only after the last input has been read. After the last magnetic tape input has been read and the tape has been removed, the operator must mount a scratch tape on the scratch tape transport and proceed with the assembly at step 11) of paragraph 4.4.

.4.5.5 SELECT OUTPUTS IN A

This typeout may occur twice during an assembly. If it occurs before any outputs have been produced, it indicates that the programmer has neglected to select outputs via a programmed output operation. To correct this condition, perform the following steps:

1) Set AU_{5-0} and AL_{5-0} to desired output numbers.

2) Start the computer.

     a) The computer stores the outputs and stops. Repeat steps a) and
        b) until all desired outputs (not more than eight) have been
        selected. When the procedure is repeated with either AU or AL
        equal to zero, TRIM III assumes all selections have been made
        and proceeds. PROGRAM SKIP key 3 has no effect on this typeout.

If this typeout occurs after at least one output has been produced, it indicates
that the assembly is complete. If another program is to be assembled at this
time, the operator may elect to stack the output No. 10 for the next program
behind the output No. 10 for the previously assembled program on the scratch
tape. To select this option, perform the following steps:

1) Set PROGRAM SKIP key 2.

2) Start the computer. After TRIM III performs an index table adjustment,
   the computer stops with P set to 01400.

3) Release PROGRAM SKIP key 2, and begin assembly of the next program.

4.5.6 IF NECESSARY CHANGE SCRATCH TAPES FOR THIS OUTPUT

This typeout occurs when a magnetic tape output, other than output No. 10, has
been requested but no tape transport has been identified for magnetic tape
output. The typeout indicates that TRIM III is ready to write the magnetic
tape output on the scratch tape and destroy the output No. 10 in the process.

If the operator does not wish to save the output No. 10, he may start the
computer to continue assembly. If the operator does wish to save the output
No. 10, he must change the tape on the scratch tape transport before starting
the computer to continue assembly.

4.5.7 MTU ERROR CTXX IMPR. COND.

This typeout indicates an error condition on the XX tape unit. The user
must correct the condition before restarting the assembly. PROGRAM SKIP key 3
has no effect on this error.

4.5.8 SET BASE ADDR. IN AL

This typeout indicates that the programmer has neglected to allocate the first
program label. To correct, set AL_{15-0} to the desired base address and start.
If PROGRAM SKIP key 3 is set, TRIM III arbitrarily allocates the program to
address 01200, and no typeout occurs.

4.5.9 NNNNN DUP. LBL XXXXXX

This typeout occurs when the source program contains at least two identical
labels. TRIM III equates all references to a duplicate label to the address
of the first such label. TRIM III does not stop after the typeout. NNNNN is
the sequential program line identifier number and XXXXXX is the duplicate
label. If PROGRAM SKIP key 3 is set, the typeout does not occur.

4.5.10   UNALLOC TAGS
          NNNNN XXXXX AAAAA

By far the most common programmer error is the use of a tag for which no
allocation was made and which does not appear anywhere in the source program
as a label. TRIM III will stop after typing NNNNN XXXXX (sequential line
identifier and tag name). To correct, perform the following steps:

   1) Set AL to the address at which the tag is to be allocated (if the
       tag is to be allocated to zero, leave AL clear).

   2) If the tag refers to an instruction contained within the program
       being assembled, set AL_{17} to a 1.

   3) If the user wishes he may allocate all future unallocated tags to the
       address in AL by setting AU to any nonzero value.

   4) Start the computer. TRIM III types the manual allocation and uses it to
       continue assembly.

The typeout UNALLOC tags occurs once only. Thereafter, only the identifier
and the tag are typed. If the user elects to allocate all unallocated tags
to a fixed address, only the first such tag is typed.

If PROGRAM SKIP key 3 is set, TRIM III arbitrarily allocates all unallocated
tags to address 00000.

4.5.11   TCS ERR XX TBL XX

This typeout indicates the table control system (TCS) of TRIM III has detected
an error while attempting to operate on the indicated table. When meaningful,
the number of the item being manipulated when the error occurred is displayed
in AU. Table III-C-3 describes the errors which TCS detects.

TRIM III has been designed so that a table overflow error will seldom occur.
If a table overflow error does occur, the programmer may extend the limits of
the table as set in the table design and restart at the TCS entrance 10012.

If the limits cannot be extended, the programmer must eliminate the cause of
the overflow or reassemble his program in smaller segments. PROGRAM SKIP key 3
has no effect on errors of this type.

4.5.12   POLY-CODE BANK OFL

This typeout indicates that generation resulting from a poly-code used in the
source program has overflowed from one bank to the next. TRIM III does not
stop following this typeout but will produce the selected outputs even though
they will require correction. PROGRAM SKIP key 3 has no effect on this typeout.

TABLE III-C-3. TCS ERRORS

Error Number	Meaning	Usual Cause
1	Illegal Table Number*	
2	Illegal Media Designation*	
3	Illegal TCS Function Code*	Assembler error, Bad TRIM III tape
4	Misused Q-Replace	
5	Illogical TCS Function Sequence	
6	Table Not Found*	CALL used but no Library Directory on tape
7	Table Overflow	Too many labels, segments, or corrections
8	Too Many Tape Units Referenced or Item Length of Zero	Loose cabling
9	Unrecoverable Tape Error	
	Table 1	Bad scratch area behind TRIM III
	Table 3	Bad scratch tape

*Incorrect table design or control parameters.

SECTION IV. OPERATOR SERVICE ROUTINES

Operator service routines are those routines used by the computer operator, under manual control, to perform computing center operations. Such routines perform handling service to the user; however, they do not become integrated into his programs. This category includes routines such as load/dump packages, trace debugging routines, and program corrector routines.

UPAK I - This is a paper tape utility package which loads assembled program tapes and makes memory dumps on paper tapes. The package provides other console conveniences such as inspect and change memory cell contents, store constant in memory, and so forth.

UPAK III - This is an expanded modular utility package. The modules of the package operate normally under manual control; however, they can also be activated under program control. A control routine loads one or all modules as specified by parameters. The package has the following capabilities:

1) Paper tape handler.
2) Computer control panel operations such as inspect and change.
3) Magnetic tape handler, UMTH (basic handler for UNIVAC magnetic tape system).
4) Magnetic tape duplicator.
5) Loader for assembler produced magnetic tape object programs.
6) Memory dump on the high-speed printer.
7) Card/load/dump.
8) Print image on magnetic tape and tape-to-printer.
9) Magnetic tape handler, JOSH (complete handler for magnetic tape system).

Since UPAK III is modular, additional utility functions may be added without changing the general characteristics of the package.

TRIM corrector - This routine corrects source programs. It reads correction tapes and erroneous source tapes into the computer, makes the necessary corrections, and punches a corrected tape. The routine is a companion to the TRIM I and TRIM II assemblers.

TRIM library builder - This routine updates source magnetic tape libraries which are used with the TRIM III assembler. It also has editing capabilities.

Trace debugging program - This program traces the execution sequence of a program during a processing run. It produces serial information pertaining to the address and contents of the instruction executed, operand if applicable, B register content, and the entire A register.

CART - This routine performs a punched-card-to-magnetic tape conversion, with optional correction and listing capability.

SECTION IV-A. UPAK I PAPER TAPE UTILITY PACKAGE

1. GENERAL INFORMATION

UPAK I is a collection of routines combined into one program which is designed to perform utility functions for computer operators and programmers. The routines which comprise UPAK I are:

1) Paper tape load.
2) Paper tape absolute bioctal dump.
3) Paper tape absolute typewriter code dump.
4) Typewriter dump.
5) Inspect and change.
6) Store constant.
7) Search memory.
8) Copy paper tape.

The paper tape load and dump routines can be operated either manually from the computer control panel or under control of a user's program. All other routines must be operated manually from the computer control panel. When any of the routines are operated manually, interrupts are locked out.

UPAK I occupies approximately 2250 (octal) memory locations. It may be loaded at any desired address in memory above 01000; however, the entire package must be loaded within one memory bank. UPAK I is supplied with a self-loader on the front of the tape. The computer's wired paper tape bootstrap loads the self-loader, which then may be used to load UPAK I. The self-loader uses addresses 00540 through 00777 in core memeory.

2. PROGRAM DESCRIPTION

When UPAK I is loaded in memory, each individual routine can be used to per-form one or more specific functions. The routines are accessible through entrance addresses, which are octal increments relative to the base address (address at which UPAK I is loaded). Table IV-A-1 specifies the entrance addresses for all UPAK I routines. If a routine is operated manually, the operator enters the routine by setting the P register to the entrance address after manually setting any necessary initial parameters. If the routine is operated under program control, the controlling program must first set initial parameters and then execute a return jump to the programmed entrance address.

TABLE IV-A-1. UPAK I ENTRANCE ADDRESSES

Entrance Address	Type of Entry and Routine Entered
Base Address + 0	Programmed entrance. Paper tape load
+ 2	Programmed entrance. Paper tape absolute bioctal dump

TABLE IV-A-1.   UPAK I ENTRANCE ADDRESS (CONT.)

Entrance Address	Type of Entry and Routine Entered
Base Address + 4	Programmed entrance. Paper tape absolute typewriter code dump
+ 6	Manual entrance. Paper tape load
+10	Manual entrance. Paper tape absolute typewriter code dump
+12	Manual entrance. Paper tape absolute bioctal dump
+14	Inspect and change
+16	Store constant in memory
+20	Search memory
+22	Copy paper tape
+26	Typewriter dump

2.1   PAPER TAPE LOAD

The paper tape load routine consists of a load selector and three load sub-routines. The load selector reads paper tape until a nonzero frame is found. It then examines the nonzero frame to determine the type of tape to be loaded, and calls one of the three load subroutines to load the tape. The table below lists the criteria used to select the load subroutine.

Nonzero Frame	Subroutine Called
75	Load relocatable bioctal code
76	Load absolute bioctal code
All other codes	Load absolute typewriter code

2.1.1   LOAD ABSOLUTE TYPEWRITER CODE

A carriage return followed by either an 8 or an 88 and another carriage return normally activates this load subroutine. A single 8 indicates the input tape has no checksum; and 88 indicates the input tape has a checksum. The subroutine ignores any data which may precede either of these two combinations. Each tape instruction is preceded by a 5-digit address. Addresses need not be sequential; however, all five digits must be present. The next 6 digits constitute the instruction to be loaded; all six digits must be present. The

load subroutine, in effect, accumulates the first 11 octal digits following
a carriage return and loads the last 6 digits at the address specified by
the first 5 digits. All other character codes, including notes, are ignored.
A carriage return signals the end of the instruction. If less than 11
octal digits are accumulated, the instruction is not loaded. A final carriage
return followed by a double period (..) terminates the load and initiates a
checksum verification when required. The tape format is shown below.

No checksum format	Checksum format
8	88
AAAAA IIIIII	AAAAA IIIIII
AAAAA IIIIII	AAAAA IIIIII
.	.
.	.
.	.
.	.
AAAAA IIIIII	AAAAA IIIIII
..	..
	CCCCCC

PROGRAM SKIP keys may be set to provide several options during the load (refer
to paragraph 3.2.1).

2.1.2 LOAD ABSOLUTE BIOCTAL CODE

The absolute bioctal tape must begin with a 76 (code for absolute bioctal tape).
Immediately after the 76 code are the initial and final addresses consisting
of five digits each. 6-digit instructions follow without further addressing,
and the tape is terminated by a 6-digit checksum. The tape format is shown
below.

Absolute Bioctal Tape Format

76	Absolute Bioctal Code
II	
II	
IF	I - Initial address
FF	
FF	F Final address
XX	
XX	X - Instruction words
XX	
XX	
..	
..	
XX	
XX	
XX	
CC	C - Checksum
CC	
CC	

The load subroutine stores the initial and final addresses and begins load-
ing the instructions into memory starting at the initial address. As each
three frames of tape are read, they are loaded into the next sequential address
in memory. After the final address has been loaded, the subroutine verifies
the checksum. PROGRAM SKIP keys may be set to provide several options during
the load (refer to paragraph 3.2.1).

2.1.3 LOAD RELOCTABLE BIOCTAL CODE

The relocatable bioctal tape must begin with a 75 (code for relocatable bioctal
tape). The entire program must be relative to base zero. Six-digit instruc-
tions follow the 75 code without addressing. Each instruction is preceded by
a 1-digit modification code which tells the load subroutine how to modify the
instruction for storage. The tape is terminated by a 6-digit checksum preceded
by a code of 7. The computer operator specifies the load base address in AU;
the load routine then uses this information to accomplish the tape load. The
tape can be loaded anywhere in computer memory. The tape format is shown
below:

Relocatable Bioctal Tape Format

75	Relocatable bioctal code
MX	M – Modification code
XX	X – Instruction words
XX	
XM	
XX	
XX	
XX	
MX	
XX	
•	
•	
X7	7 – Checksum code
CC	C – Checksum
CC	
CC	

Modification codes appearing on a relocatable bioctal tape are:

Code	Meaning	Type of Instruction
0	No modification	Constant or 4-digit Y unmodified
1	Add base address to Y_{11-0}	4-digit Y modified
2	No modification	5-digit Y unmodified
3	Add base address to Y_{14-0}	5-digit Y modified (bit 15 is set to 0 or 1 depending on specified base address)
4	Increment current load address by instruction value.	Negative or positive increment
5, 6	Not used	Not used
7	Checksum follows	Tape checksum

PROGRAM SKIP keys may be set to provide several options during the load (refer to paragraph 3.2.1).

2.2 PAPER TAPE ABSOLUTE TYPEWRITER CODE DUMP

This routine punches the contents of a specified memory area on paper tape in absolute typewriter code. The format in which the tape is dumped is the same as the "88" format (with checksum at the end) described in paragraph 2.1.1. The output tape includes both the addresses and the contents of the memory locations being dumped. The routine provides the option of typing the output rather than punching it on tape.

2.3 PAPER TAPE ABSOLUTE BIOCTAL CODE DUMP

This routine punches the contents of a specified memory area on paper tape in absolute bioctal code. The format in which the tape is dumped is the same as the format described in paragraph 2.1.2. If more than one program area is dumped successively on the same tape, the areas may be separated by either a 2-frame leader or a 256-frame leader. The 2-frame option permits the tape to be loaded continuously (without stopping between areas).

2.4 INSPECT AND CHANGE

The inspect and change routine causes the contents of memory location specified in AU to be displayed in AL. The contents of AL may then be changed manually. (AL) is then returned to the memory address from which it was taken. The inspection address need be entered only the first time since (AU) is increased by 1, and the contents of sequential addresses will be brought into AL with each successive performance of the inspect and change function. If the user wishes to inspect the contents of some addresses other than the next sequential address, he may do so by setting the new address in AU before returning AL to memory. The routine provides the option of punching and/or typing the inspect addresses and their contents. The format of the punched tape and typeout is given below.

READ-WRITE

8

AAAAA NN NNNN FF FFFF
.
.
.
AAAAA NN NNNN FF FFFF

AAAAA is the inspect address, NN NNNN is the new content of the address, and FF FFFF is the former content of the address. The punched tape may be used as an errata tape for the program in memory.

2.5 STORE CONSTANT IN MEMORY

This routine stores a constant in a specified number of consecutive memory addresses. The operator manually enters the constant and the limits of the memory area. If a constant of zero is specified, the memory area is cleared; however, the routine never clears addresses occupied by UPAK I.

2.6 SEARCH MEMORY

This routine searches a specified memory area for all values which satisfy the conditions specified by a mask and a searchand. All values in the memory area that satisfy the conditions are typed on the console typewriter. The format of the typeout is given on the next page.

```
                    SEARCH 1219

                    LLLLL TO UUUUU
                    MM MMMM SS SSSS
                    AAAAA CC CCCC
                    AAAAA CC CCCC


                           .
                           .
                           .

                    AAAAA CC CCCC
                    END SEARCH
```

LLLLL and UUUUU are the lower and upper limits of the search area; MM MMMM is the mask; SS SSSS is the searchand; and AAAAA is the address of a word, CC CCCC, which satisfies the conditions of the mask and searchand. An example is given below to demonstrate the capability provided by this routine.

Example:

Assume that the operator wishes to find all I/O instructions in a program loaded in memory beginning at address 010000 and ending at address 020000. The typeout below illustrates the limits, mask, and searchand to be specified.

```
        SEARCH 1219

        10000 to 20000 }      Memory limits
        77 4000 50 0100}      Mask and searchand
        10123 50 1306
        12777 50 1106
        13011 50 1203         Instructions which
        14000 50 3600         satisfy conditions
        16520 50 1305         of mask and searchand
        16540 50 1105
        17000 50 1505
        17500 50 3200
        END SEARCH
```

2.7 COPY PAPER TAPE

This routine produces on exact copy of a paper tape. It is a faster and more reliable method than off-line duplication.

2.8 TYPEWRITER DUMP

This routine produces a typed hard copy of the contents of a specified memory area. The format of the output is shown on the next page.

```
CORE DUMP FROM LLLLL TO UUUUU
LLLLL    CCCCCC_0     CCCCCC_1              . . .   CCCCCC_7
BBBBB ---
     .
     .
     .
XXXXX                                       . . .   CCCCCC_n
```

LLLLL and UUUUU are the lower and upper limits of the memory area to be dumped; $CCCCCC_0$ is the contents of the lower limit; $CCCCCC_1$ through $CCCCCC_7$ are the contents of the next seven sequential addresses; and BBBBB is 10 (octal) greater than LLLLL. $CCCCCC_n$ is the contents of the upper limit. Addresses are typed only for every 10 (octal) words. If the content of any address is zero, it is indicated by a blank space rather than zeros. If one entire line contains all zeros, it is indicated by a blank line. If two or more consecutive lines contain all zeros, only one blank line appears on the output. The routine provides the option of punching the information on paper tape.

3. LOADING AND OPERATING PROCEDURES

3.1 LOADING UPAK I

UPAK I is provided on punched paper tape. The tape is subdivided into two parts: a self-loader in absolute bioctal format and UPAK I in relocatable bioctal format. To load UPAK I, the following procedure is used:

1) Place the tape in the reader.

2) Master clear the computer and read-punch unit.

3) Press the LOAD button to activate paper tape bootstrap.

4) Start the computer. The computer stops with AU and AL equal to zero after the self-loader is in memory.

5) Set AU to the desired UPAK I base address.

6) Start the computer. The computer stops with AU and AL equal to zero after UPAK I has been loaded.

After UPAK I has been loaded into memory, addresses 00540-00777, occupied by the UPAK I loader, are available for use. It should be noted, however, that any subsequent loading of UPAK I will use these addresses to accomplish the load.

Automatic checksum verification by the paper tape load subroutines is the only error detection function performed by UPAK I. If tape and load checksums agree, the computer comes to a normal stop with AU and AL equal to zero. If they do not agree, the computer stops with (AU) = load checksum and (AL) = tape checksum.

The format of the UPAK I tape is such that only the self-loader is loaded
via paper tape bootstrap. Control is then given to a temporary checksum
verification routine which verifies that the self-loader is properly contained
in memory. If verification is correct, the computer stops with AU and AL
clear. Upon restarting, control is given to the self-loader which loads
UPAK I. If verification is incorrect, the computer stops with AU equal to
the load checksum and AL equal to the tape checksum. After UPAK I has been
loaded, the same checksum indication is provided in AU and AL.

3.2 USING UPAK I

The procedures required to operate all routines of UPAK I are given in the
following subparagraphs. All procedures are for manual operation and
assume that the computer and I/O console have been placed in the operational
state with all switches in the normal operating position. If paper tape
load and dump routines are to be operated under program control, input
parameters must be set by the controlling program before the program
executes a return jump to the programmed entrance of the routine.

3.2.1 PAPER TAPE LOAD

1) Master clear the computer.

2) Set the P register to the UPAK I base address +6.

3) Mount the tape to be loaded in the reader.

4) If the tape to be loaded is relocatable, set the AU register to
    the starting address of the load.

5) Set the PROGRAM SKIP key options desired (refer to 7) below).

6) Start the computer. The computer stops when the tape is loaded.
    Load error indications are dependent upon the load options selected.
    If no options are selected, the computer stops with AU equal to the
    computed checksum and AL equal to the tape checksum. If checksums are
    equal, AU and AL are clear.

7) Load Options. The following options are selectable via PROGRAM SKIP
    keys:

    a) Absolute Bioctal Load

        1. PROGRAM SKIP key 0 - If this key is set, a tape checksum veri-
            fication is performed but information read from tape is not
            stored in memory. At the completion of the read, AU is set to
            the computed checksum and AL is set to the tape checksum. If
            the checksums are equal, AU and AL are cleared.

        2. PROGRAM SKIP key 3 - If this key is set, program areas are

typed on the on-line typewriter. The format for each area is:

BBBBB   TO   EEEEE

BBBBB is the beginning address and EEEEE is the end address.

3. PROGRAM SKIP key 4 - If this key is set, a tape verify is
    performed. This option causes information read from tape to
    be compared to information stored in core memory. Differences
    are indicated in one of three ways depending upon the setting
    of PROGRAM SKIP keys 1 and 2.

    a. If PROGRAM SKIP key 1 is set, differences are typed on
        the on-line typewriter in the following format:

        AAAAA   TT   TTTT   CC   CCCC

        AAAAA is the core address, TTTTTT is the tape word and
        CCCCCC is the core word.

    b. If PROGRAM SKIP key 2 is set, differences are punched on
        paper tape for off-line listing. The format is the same
        as for on-line typeout.

    c. If PROGRAM SKIP keys 1 and 2 are not set, differences are
        displayed in AU and AL during computer stops. During the
        first step, the core address is displayed in AU and the
        tape word is displayed in AL. After the computer is re-
        started, a second stop occurs with the core word displayed
        in AU and the tape word displayed in AL. After the com-
        puter is again restarted the tape word is stored in core
        memory (unless PROGRAM SKIP key 0 is set) and the verify
        continues.

b) Relocatable Bioctal Load

    1. PROGRAM SKIP keys 0, 1, 2, and 4 provide the same option as
        for absolute bioctal load. PROGRAM SKIP key 3 is ignored.

c) Absolute Typewriter Code Load

    1. PROGRAM SKIP keys 0, 1, 2, and 4 provide the same options as
        for absolute bioctal load.

    2. PROGRAM SKIP key 3 - If this key is set, the subroutine permits
        the loading of a typewriter code tape which does not contain
        the tape identifier (carriage return, 8 or 88, carriage return).

3.2.2 PAPER TAPE ABSOLUTE TYPEWRITER CODE DUMP

1) Master clear the computer.

2) Set the P register to the UPAK I base address +10_8.

3) Set AU to the first address to be dumped.

4) Set AL to the last address to be dumped.

5) Set PROGRAM SKIP key 1 if output is to be typed rather than punched.

6) Start the computer. After punching the contents of the specified addresses on tape, the computer stops. To obtain another dump begin at step 3).

3.2.3 PAPER TAPE ABSOLUTE BIOCTAL DUMP

1) Master clear the computer.

2) Set the P register to the UPAK I base address +12_8.

3) Set AU to the first address to be dumped.

4) Set AL to the last address to be dumped.

5) Set PROGRAM SKIP key 4 if a 2-frame trailer is to be punched after the dump.

6) Start the computer. The computer stops after punching the contents of the specified addresses. ·To obtain another dump begin at step 3). If a 2-frame trailer is selected in step 5) and more than one dump is punched on the same tape, the tape can be loaded, without stops, by the paper tape load routine. If the 2-frame trailer is not selected, a 256-frame trailer is punched after the dump.

3.2.4 INSPECT AND CHANGE

1) Master clear the computer.

2) Set the P register to the UPAK I base address +14_8.

3) Set AU to the first address to be inspected.

4) Set the PROGRAM SKIP key options desired (refer to 8) below).

5) Start the computer. The computer stops with the address in AU and its contents in AL.

6) If desired, change the address in AU and/or the contents in AL.

7) Start the computer.

a) If the contents only were altered, the new contents are stored at the original address and the computer stops with the next sequential address in AU and its contents in AL.

b) If the address only was changed, the old contents are restored to the proper memory location, and the computer stops with the new address in AU and its contents in AL.

c) If both the address in AU and its contents in AL were changed, the new contents are stored at the original address, and the computer stops with the new address in AU and its contents in AL.

8) Inspect and change options.

The following options are selectable via PROGRAM SKIP keys.

a) PROGRAM SKIP key 1 - If this key is set, the addresses inspected and their contents are typed on the on-line typewriter. The format of the typeout is:

           READ-WRITE
           AAAAA NN NNNN FF FFFF

AAAAA is the inspected address, NN NNNN is the new content, and FF FFFF is the former content.

b) PROGRAM SKIP key 2 - If this key is set, an errata tape of the changes is punched. The tape is punched in paper tape absolute typewriter code format and can be loaded via the UPAK I paper tape load routine. To finalize the tape after all changes have been punched, set P to UPAK I base address $+24_8$ and start the computer. Two periods and a trailer are then punched on tape. If the errata tape is to be punched without a leader set PROGRAM SKIP keys 2 and 3 before the first address is inspected.

3.2.5 STORE CONSTANT IN MEMORY

1) Master clear the computer.

2) Set the P register to the UPAK I base address $+16_8$.

3) Set the first storage address in AU.

4) Set the last storage address in AL.

5) Start the computer. The computer stops with AU cleared.

6) Set the constant in AU.

7) Start the computer. The computer stops after storing the constant in all addresses of the specified memory area.

3.2.6 SEARCH MEMORY

1) Master clear the computer.

2) Set the P register to the UPAK I base address +20$_8$.

3) Set AU to the search mask.

4) Set AL to the searchand.

5) Start the computer. The computer stops with AU and AL cleared.

6) Set AU to the lower limit of the area to be searched.

7) Set AL to the upper limit of the area to be searched.

8) Start the computer. The computer compares the logical product of mask and searchand with the logical product of the mask and each memory location in the specified area. If the logical products are equal, the address of the memory location and its contents are typed on the on-line typewriter (refer to paragraph 2.6 for the format of the typeout).

3.2.7 COPY PAPER TAPE

1) Master clear the computer.

2) Set the P register to the UPAK I base address +22$_8$.

3) Mount the tape to be copied in the reader.

4) Start the computer. The computer reads the input tape and punches an exact copy.

5) To stop the copy routine set PROGRAM SKIP key 0. This key should be set while the routine is copying the trailer on the input tape. If more trailer is required on the output tape, depress the tape feed button on the I/O console.

3.2.8 TYPEWRITER DUMP

1) Master clear the computer.

2) Set the P register to the UPAK I base address +26$_8$.

3) Set AU to the address of the first word to be dumped.

4) Set AL to the address of the last word to be dumped.

5) Set PROGRAM SKIP key 0 if the output is to be punched rather than typed on the on-line typewriter.

6) Start the computer. The computer stops after typing (or punching) the contents of the specified memory locations (refer to paragraph 2.8 for the format of the output).

SECTION IV-B. UPAK III UTILITY PACKAGE III

1. GENERAL INFORMATION

UPAK III is a modular utility system comprised of stacked programs in the
TRIM III output No. 10 format on magnetic tape. A control program loaded
by magnetic tape bootstrap accomplishes loading of one or more of the compo-
nent modules. UPAK III presently has the following modules:

Module	1	Paper tape handler (PTHAN).
Module	2	Magnetic tape handler (UMTH).
Module	3	Magnetic tape duplication (MTDUP).
Module	4	TRIM III output 10 load (LOAD10).
Module	5	Inspect and change and store constant (ICH-STC).
Module	6	Print memory contents (PRINTC).
Module	7	Card handler (DATCD).
Module	10	Printer line image on tape and tape-to-printer (POTPOP).
Module	11	Magnetic tape handler (JOSH).

The modular framework of UPAK III, however, will permit the incorporation of
additional modules with a minimum of effort. Refer to paragraph 1.2.3,
expanding UPAK III.

UPAK III assumes a minimum equipment configuration of one computer with
a minimum of 16,384D words of core memory, one magnetic tape unit with two
transports, an I/O console, and a card processor. UPAK III may be loaded
anywhere in computer memory above address 01000 with the single restriction
that any module must be entirely contained within a single memory bank. Most
UPAK III functions can be operated under program control as well as manually
from the computer control panel.

2. CONTROL PROGRAM

2.1 PROGRAM DESCRIPTION

The control program of UPAK III is basically a module loader. It may be
loaded by magnetic tape bootstrap anywhere in core memory above address
01000. However, it must be wholly contained within one bank of memory.

Through manual parameter specification, the control program loads one module
or all modules at a given time. If one specific module is requested, the
load address must be specified. If more than one module but not the
entire package is desired at specific addresses, the user must repeat the
load procedure for each particular module. If module number zero is specified
UPAK III loads all of the modules at the base addresses denoted in Table
IV-B-1; the base address parameter is ignored.

Module entrances are determined by standard increments to their load address
as shown in Table IV-B-1.

Care must be exercised in specifying a module base load address to the control
program so that the module is loaded entirely within one memory bank.

TABLE IV-B-1. ENTRANCE ADDRESSES AND ASSIGNED BASES

Module	No.	Base	Entrance Increments to Base
PTHAN	1	01100	+ 3 - ICH Entrance
Size: 1060_8			+ 4 - STC Entrance
			+ 6 - Programmed PT Load
			+ 0 - Manual PT Load
			+12 - Programmed Typewriter Code Dump
			+ 2 - Manual Typewriter Code Dump
			+10 - Programmed Bioctal Dump
			+ 1 - Manual Bioctal Dump
UMTH	2	02200	+ 0 - Programmed Entrance
Size: 600_8			+ 2 - Manual Entrance
MTDUP	3	03000	+ 0 - Programmed Entrance
Size: 712_8			+ 2 - Manual Entrance
LOAD10	4	04000	+ 0 - Programmed Entrance
Size: 600_8			+ 2 - Manual Entrance
ICH-STC	5	04700	+ 0 - ICH Entrance
Size: 77_8			+ 1 - STC Entrance
PRINTC	6	05000	+ 0 - Programmed Entrance
Size: 450_8			+ 2 - Manual Entrance
DATCD	7	06000	+ 0 - Programmed Card Load
Size: 626_8			+ 2 - Manual Card Load
			+ 3 - Programmed Card Dump
			+ 5 - Manual Card Dump
POTPOP	10	07000	+ 0 - Programmed line image on tape
Size: 434_8			+ 2 - Manual line image on tape
			+ 3 - Programmed tape-to-printer
			+ 5 - Manual tape-to-printer
JOSH	11	10000	+ 0 - Programmed Entrance
Size: 640_8			+ 2 - Check status entrance
			+ 4 - Check busy entrance
			+10 - Manual Entrance

2.2 LOADING UPAK III

To load the UPAK III control program via magnetic tape bootstrap, mount the UPAK III tape on transport 1 of cabinet 1. Then,

1) Master clear the computer.

2) Push the LOAD button to activate bootstrap.

3) Start the computer. When the computer stops, a parameter is displayed in AU. The parameter format is shown in Figure IV-B-1.

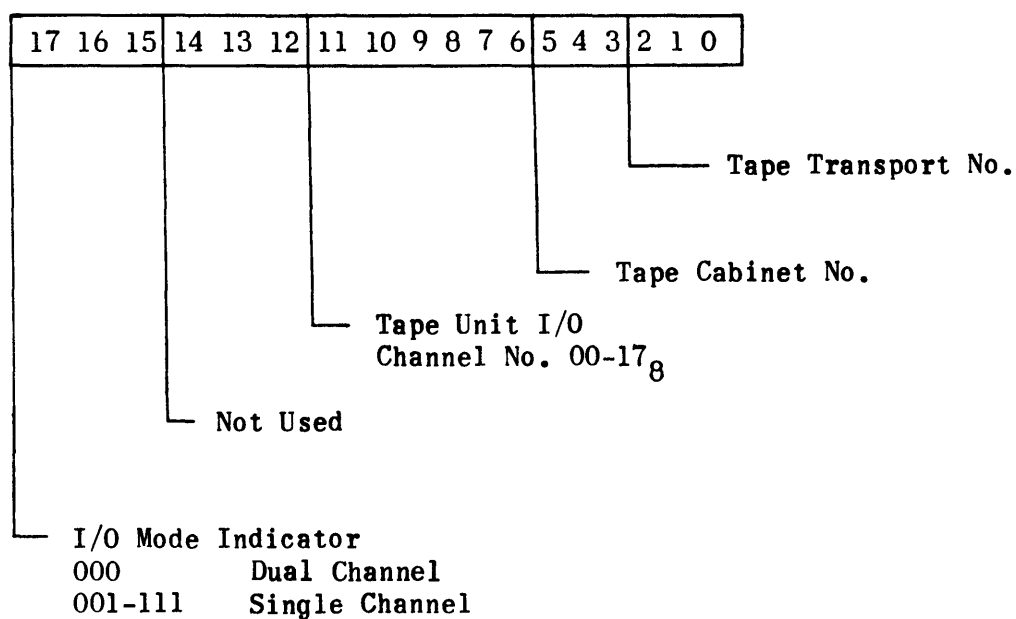| 17 16 15 | 14 13 12 | 11 10 9 8 7 6 | 5 4 3 | 2 1 0 |

Tape Transport No.

Tape Cabinet No.

Tape Unit I/O
Channel No. 00-17$_8$

Not Used

I/O Mode Indicator
000   Dual Channel
001-111   Single Channel

Figure IV-B-1. Tape Address Parameter

If the parameter does not specify the tape channel, address, and mode desired, change AU as desired.

> NOTE: The logical selection of the tape address is not restricted to cabinet 1, transport 1 at this time.

4) Enter a 16-bit control program load address in AL if desired. If AL is zero, the control program will be loaded at its assigned address, 012000.

5) Start the computer. When the computer stops, the control routine is loaded.

UPAK III may be loaded via paper tape bootstrap through use of a core-stored magnetic tape bootstrap program. To accomplish the load perform the following steps:

1) Mount the UPAK III magnetic tape on cabinet 1, transport 1.

2) Place the PTMTBS* No. 4 paper tape in the paper tape reader.

3) Master clear the computer.

4) Press the LOAD button on the control panel.

5) Start the computer. When the computer stops, a parameter is displayed in AU. The parameter format is shown in Figure IV-B-1.

   If the parameter does not specify the tape channel, address, and mode desired, change AU as desired.

   > NOTE: The logical selection of the tape transport is not restricted to cabinet 1, transport 1 at this time.

6) Enter a 16-bit control program load address in AL if desired. If AL is zero, the control program will be loaded at this assigned address, 012000.

7) Start the computer. When the computer stops, the control routine is loaded.

To load UPAK III modules, perform the following steps:

1) Master clear the computer.

2) Set P to the base address of the control routine.

3) Set AU to MMXXCT where MM is the module number; XXCT is the UPAK III tape address (channel XX, cabinet C, and transport T). If MM is zero, all modules will be loaded at their assigned addresses (refer to Table IV-B-1).

---

*Magnetic tape bootstrap on paper tape.

4). Set a 16-bit load address for the requested module in AL, if desired. If AL is zero, the control program will load the module at the assigned address (refer to Table IV-B-1).

5) Start the computer. The computer stops after the module is loaded. To operate specific modules,refer to paragraphs 3 through 11.

6) To load another module, repeat steps 3) through 5) above.

2.3 EXPANDING UPAK III

User expansion of the UPAK III system to incorporate one or more additional modules may be accomplished through use of TRIM III. When expanded, UPAK III operates normally except when specifying the control routine parameter MM = 00. To validate this parameter it is necessary to change the number of modules in the system by errata to the UPAK III control routine (address, base + 66$_8$) before specifying MM = 00. This word should contain 7000XX where XX is the number of modules minus one in octal. Without this errata, MM = 00 results in loading the original UPAK III modules only.

To add modules to UPAK III:

1) Mount the TRIM III tape.

2) Mount the UPAK III tape as the TRIM III scratch tape.

3) Load TRIM III.

4) Change TRIM III address 01057 from zero to the number of files on the UPAK III tape (actual number of modules plus two).

5) Assemble the program that is to become the next UPAK III module. Care should be taken so that the base address is compatible with other UPAK III module base addresses.

6) After the last desired output has been produced (one output other than a source output - 11, 15 or 16 - must be selected) and the SELECT OUTPUTS IN A typeout occurs, set PROGRAM SKIP key 2 and start the computer. The computer stops at address 01400.

7) Release PROGRAM SKIP key 2.

8) Repeat steps 5) and 7) until all desired additional modules have been assembled.

9) Rewind the expanded UPAK III tape and remove it from the scratch transport.

3. PAPER TAPE HANDLER MODULE (PTHAN)

3.1 PROGRAM DESCRIPTION

PTHAN is a collection of subroutines which provides paper tape I/O functions and examination and alternation of memory for program debugging. The seven subroutines are:

1) Load absolute typewriter code.
2) Load absolute bioctal code.
3) Load relocatable bioctal code.
4) Dump absolute typewriter code.
5) Dump absolute bioctal code.
6) Inspect and change.
7) Store constant in memory.

PTHAN may be loaded anywhere in computer memory above address 01000 with the single restriction that the entire package must be entirely contained within a single memory bank. All functions operate either manually or under program control except for the inspect and change and store constant in memory functions, which are manually operable only.

Entrance addresses to the several subroutines are assigned relative to the PTHAN base address as shown in Table IV-B-1.

PTHAN subroutines use the currently active B register but store and restore their original value. The load subroutines when operating under program control also store and restore the user's special register setting.

1) Inspect and change.

   The inspect and change routine causes the contents of the memory location specified in AU to be displayed in AL. The contents of AL may then be changed manually. (AL) is then returned to the memory address from which it was taken. The inspection address need be entered only the first time, since (AU) is increased by one and the contents of sequential addresses will be brought into AL with each successive performance of the inspect and change function. Should the user wish to inspect the contents of some address other than the next sequential address, he may do so by setting the new address in AU before returning (AL) to memory.

2) Store constant in memory.

   The store constant in memory function permits the user to load a specified area of memory with a value manually entered into AU. If (AU) = 0 the area is cleared.

3) Load absolute typewriter code.

   A carriage return, followed by either an 8 or an 88 activates the load routine. A single 8 indicates that the tape has no checksum (user

prepared tapes). An 88 indicates that the tape is terminated with a 6-digit checksum (TRIM outputs 2 and 3 and UPAK typewriter code dumps). A carriage return must follow the 8 or 88 code.

Once the load routine has been activated, it accumulates the first 5 digits following each carriage return and assembles them as the address. It then accumulates the next six digits and stores them at the accumulated address. All characters following the first 11 digits are ignored until another carriage return is found. If less than 11 digits are accumulated, the instruction will not be loaded. Examples of tape formats are shown below.

Each tape to be loaded must terminate with a carriage return and a double period (..), which terminates the load and initiates a checksum verification when required. If the checksum verification is correct, the load terminates with (AU) and (AL) = 0. When it is incorrect, the load terminates with (AU) = computed checksum and (AL) = tape checksum.

If PROGRAM SKIP key 1 is set the load subroutine performs checksum verification without storing the information in memory.

Format With No Checksum	Format With Checksum
8	88
XXXXX XX XXXX	XXXXX XX XXXX
XXXXX XX XXXX	XXXXX XX XXXX
. . .	. . .
. . .	. . .
XXXXX XX XXXX	XX XXXX(Checksum)
..	..

4) Load absolute bioctal code.

The absolute bioctal tape must begin with a 76 (code for absolute bioctal tape with 5-digit addressing). Immediately after the 76 or 77 code are the initial and final addresses consisting of 5 digits each. 6-digit instructions follow without further addressing, and the tape is terminated by a 6-digit checksum.

Absolute Bioctal
76 Tape Format

76	Absolute Bioctal Code
II	
II	I-initial address
IF	
FF	F-final address
FF	
XX	X-instruction words
XX	
XX	
..	
..	
XX	
XX	
XX	
CC	C-checksum
CC	
CC	

If PROGRAM SKIP key 1 is set, the absolute bioctal load subroutine will
perform a checksum verification without loading into memory.

5) Load relocatable bioctal code.

The relocatable bioctal tape must begin with a 75 (code for relocatable
bioctal tape). The entire program must be relative to base zero. Six-
digit instructions follow the 75 code without addressing. Each instruc-
tion is preceded by a 1-digit modification code which tells the load
routine how to modify the instruction for storage. The tape is terminated
by a 6-digit checksum preceded by a code of 7. The computer operator
specifies the load base address in AU; the load routine then uses this
information to accomplish the tape load.

Relocatable Bioctal Tape Format

75	Relocatable Bioctal Code
MX	M-Modification code
XX	X-Instruction words
XX	
XM	
XX	
XX	
XX	
MX	
XX	
.	
.	
.	
X7	7-Checksum code
CC	C-Checksum
CC	
CC	

Modification codes appearing on a relocatable bioctal tape are:

Code	Meaning	Type of Instruction
0	No modification	Constant or 4-digit Y unmodified
1	Add base address to Y_{11-0}	4-digit Y modified
2	No modification	5-digit Y unmodified
3	Add base address to Y_{14-0}	5-digit Y modified (bit 15 is set to 0 or 1 depending on specified base address)
4	Increment current load address by instruction value	Negative or positive increment
5, 6	Not used	Not used
7	Checksum follows	Tape checksum

If PROGRAM SKIP key 1 is set, the relocatable bioctal load subroutine will perform a checksum verification without loading into memory.

6) Dump absolute typewriter code.

This dump is initiated manually or under program control for output on punched paper tape. The 88 code format (with checksum at the end) is the only one dumped. The output tape includes both the addresses and the contents of the memory locations being dumped.

7) Dump absolute bioctal code.

The absolute bioctal dump is initiated manually or under program control for output on punched paper tape. The tape format is a 77 code followed by:

a) The initial and final addresses of the area being dumped.
b) The contents of the inclusive memory addresses.
c) The checksum.

If more than one program area is dumped successively on the same tape, each such area is formatted as described.

3.2 OPERATION PROCEDURES

3.2.1 OPERATION OF INSPECT AND CHANGE

1) Set P to PTHAN base address +3.

2) Set the desired memory address to be inspected in AU.

3) Start the computer. The computer stops with the address in AU and its contents in AL.

4) The user may now change the address in AU and/or the contents in AL.

5) Start the computer.

   a) If the contents only were altered, the new contents are stored at the original address and the computer stops with the next sequential address in AU and its contents in AL.

   b) If the address only was changed, the old contents are restored to the proper memory location and the computer stops with the new address in AU and its contents in AL.

   c) If both the address in AU and its contents in AL were changed, the new contents are stored at the original address, and the computer stops with the new address in AU and its contents in AL.

6) Any number of such sequences may be executed starting with step 4).

3.2.2 OPERATION OF STORE CONSTANT IN MEMORY

1) Set P to PTHAN base address + 4.

2) Set the first storage address in AU.

3) Set the last storage address in AL.

4) Start the computer. PTHAN records these addresses and the computer stops with AU cleared.

5) Set the desired constant in AU.

6) Start the computer. PTHAN stores (AU) at successive memory locations within the parameters established in steps 2) and 3).

7) Additional entrances may be made starting from step 2).

3.2.3 MANUAL OPERATION OF ALL PAPER TAPE LOADS

1) Mount a tape in the reader.

2) Set P to PTHAN base address.

3) Set PROGRAM SKIP key 1 if checksum verification only is desired.

4) For relocatable bioctal load only set the starting address in AU. (Not required if PROGRAM SKIP key 1 is set.)

5) Start the computer. When the computer stops, AU is set to the computed checksum and AL is set to the tape checksum. If the checksums are equal AU and AL are both clear.

6) Successive tapes may be loaded without resetting P.

3.2.4 PROGRAM OPERATION OF ALL PAPER TAPE LOADS

1) The tape to be loaded must be mounted in the reader.

2) For relocatable bioctal load only, the controlling program must enter AU with the starting address of the load and then execute a return jump or indirect return jump to PTHAN base address + 6.

3.2.5 MANUAL DUMP OF TYPEWRITER CODE

1) Set P to PTHAN base address + 2.

2) Set the first address to be dumped in AU.

3) Set the last address to be dumped in AL.

4) Start the computer.

5) Successive dumps may be taken by starting from step 2).

3.2.6 PROGRAM OPERATION OF DUMP TYPEWRITER CODE

The controlling program must:

1) Enter AU with the first address to be dumped.

2) Enter AL with the last address to be dumped.

3) Execute a return jump or indirect return jump to PTHAN base address + 12 (octal).

3.2.7 MANUAL DUMP OF ABSOLUTE BIOCTAL CODE

1) Set P to PTHAN base address + 1.

2) Set the first address to be dumped in AU.

3) Set the last address to be dumped in AL.

4) Start the computer.

5) Successive dumps may be taken by starting from step 2).

3.2.8 PROGRAM OPERATION OF DUMP BIOCTAL CODE

The controlling program must:

1) Enter AU with the first address to be dumped.

2) Enter AL with the last address to be dumped.
    Execute a return jump or indirect return jump to PTHAN base address
    + 10 (octal).

4. MAGNETIC TAPE HANDLER MODULE (UMTH)

4.1 PROGRAM DESCRIPTION

UMTH provides the user with the basic magnetic tape handling services of read,
write, write tape mark, search, pass n records, space file, rewind. These
services may be recorded in single or dual channel operation, high or low
density, bioctal or octal coding and odd or even parity (even parity may be used
with BCD only). Only forward buffering is permitted.

This module may be loaded anywhere in core memory above address 01000 with the
restriction that the entire module must be wholly contained within one memory
bank. Both manual and programmed entrances are provided to UMTH (refer to
Table IV-B-1).

4.2 INPUT PARAMETERS

Six parameter entries govern operation of UMTH. These parameters may be set
manually or under program control. The format of the parameters is shown in
Figure IV-B-2.

Parameter 1
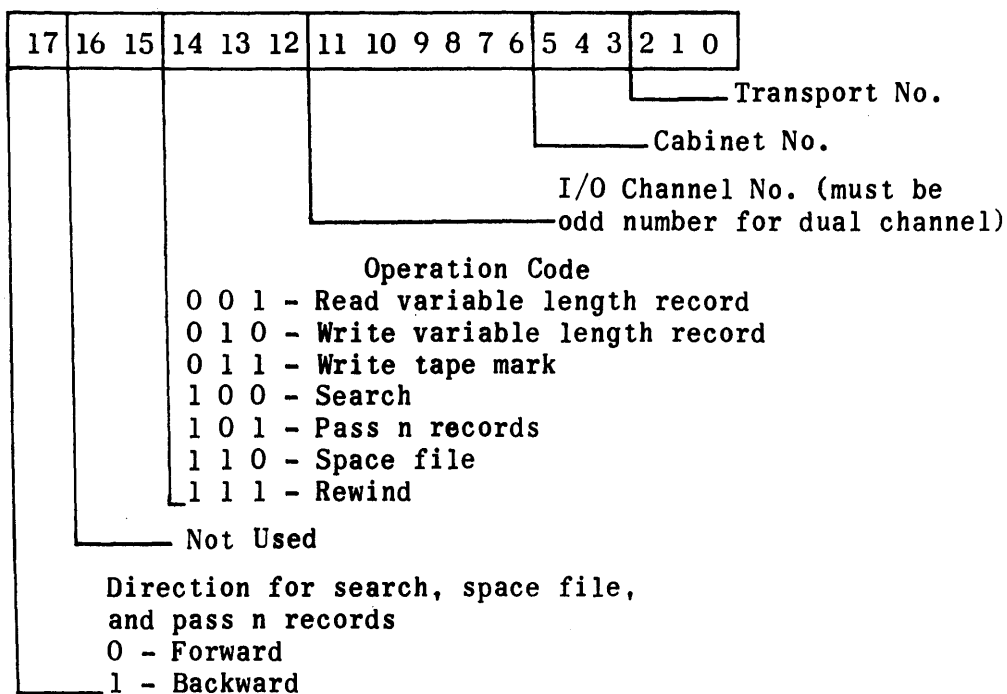
```
┌──┬─────┬────────┬─────────────┬─────┬─────┐
│17│16 15│14 13 12│11 10 9 8 7 6│5 4 3│2 1 0│
└──┴─────┴────────┴─────────────┴─────┴─────┘
                                          └────── Transport No.
                                    └──────── Cabinet No.

                              I/O Channel No. (must be
                              odd number for dual channel)

                    Operation Code
          0 0 1 - Read variable length record
          0 1 0 - Write variable length record
          0 1 1 - Write tape mark
          1 0 0 - Search
          1 0 1 - Pass n records
          1 1 0 - Space file
          1 1 1 - Rewind

          ──── Not Used

     Direction for search, space file,
     and pass n records
     0 - Forward
     1 - Backward
```

Figure IV-B-2. UMTH Input Parameters (Sheet 1 of 3)

Parameter 2

```
┌──────┬──┬──┬─────┬──────────────┬──────────────┐
│17 16 │15│14│13 12│11 10 9 8 7 6 │5 4 3 2 1 0   │
└──────┴──┴──┴─────┴──────────────┴──────────────┘
```

No. of records to
pass or No. of
files to space
(77_8 maximum)
000001 through 111111

Not Used

Density
0 0 - 200 frames/inch (low for 1240)
1 0 - 556 frames/inch (high for 1240)
0 1 - 800 frames/inch
1 1 - Same density as last instruction

Parity
0 - Even (used with BCD only)
1 - Odd

Coding
0 - Bioctal
1 - Octal

Modulus
0 0 - Modulus 3 (18 bits)
0 1 - Modulus 4 (24 bits)
1 0 - Modulus 5 (30 bits)
1 1 - Modulus 6 (36 bits)

Figure IV-B-2. UMTH Input Parameters (Sheet 2 of 3)

Parameter 3

17 16	15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

Buffer initial
address for read,
write, or search

Not Used

Parameter 4

17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

No. of words (octal)
for read, write
or search

Parameter 5

17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

Search constant for
single channel
search, first half
of search constant
for dual channel
search

Parameter 6

17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

Second half of
search constant for
dual channel
search
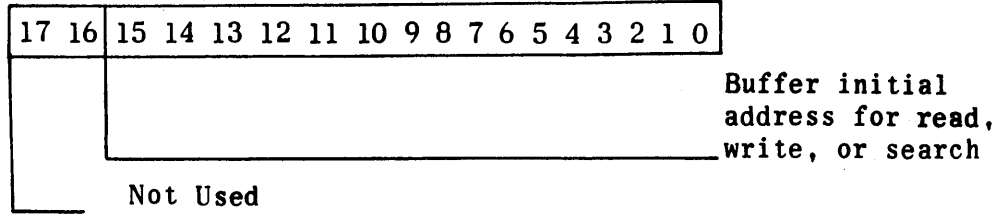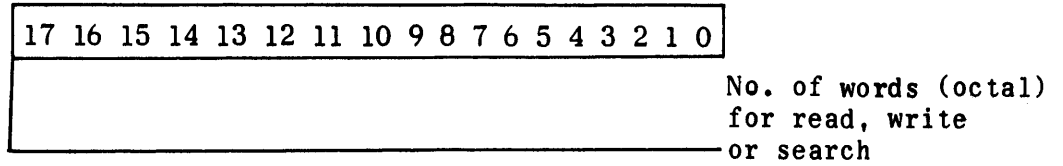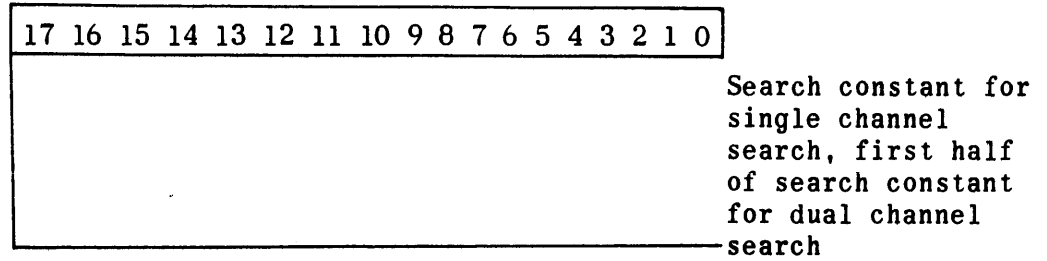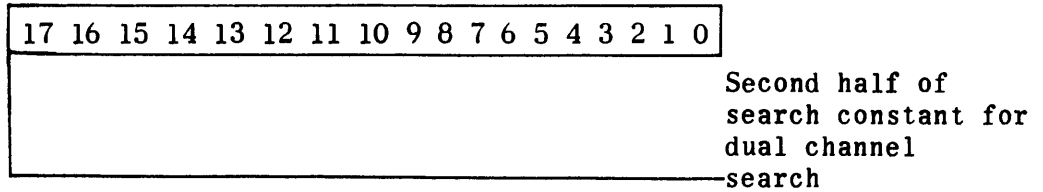
Figure IV-B-2. UMTH Input Parameters (Sheet 3 of 3)

4.3 OPERATING PROCEDURES

4.3.1 OPERATION UNDER PROGRAM CONTROL

The calling program must enter UMTH with parameter 1 in AU and parameter 2 in AL. If parameters 3, 4, 5 and 6 are required they must be stored in consecutive memory addresses and the active B register must contain the address of parameter 3 upon entering UMTH. These consecutive addresses may consist of two, three, or four words depending upon the operation to be performed. Parameters 3 and 4 are needed for read and write operations; parameters 3, 4, and 5 are needed for single-channel search operation; parameters 3, 4, 5, and 6 are needed for dual-channel search operations. The calling program enters UMTH by executing a return jump or indirect return jump to UMTH base address + 1. The magnetic tape to be handled must be mounted on the proper transport before starting the calling program.

Upon resumption of control, the calling program may check the contents of the A register to verify a successful operation (see paragraph 4.4).

4.3.2 MANUAL OPERATION

1) Master clear the computer.

2) Set P to UMTH base address.

3) Mount the magnetic tape to be handled on a tape transport.

4) Set parameter 1 in AU; set parameter 2 in AL.

5) Start the computer.

6) When the computer stops, set parameter 3 in AU; set parameter 4 in AL.

7) Start the computer. If the operation to be performed is a search, the computer stops again.

8) Set parameter 5 in AU, and, if needed set parameter 6 in AL.

9) Start the computer.

If an improper condition is encountered, correct the condition and start the computer. UMTH then re-executes the operation.

After completion of a successful operation the computer stops. To execute another operation, start again and the computer stops with parameters 1 and 2 in AU and AL. Proceed with the next operation from step 3) above.

4.4 ALARMS AND STATUS INDICATIONS

Under both manual and program operation, UMTH uses the A register to indicate the status of the operation attempted.

If (AL) = zero, a successful operation is indicated.

If (AL) = 777777, an unsuccessful operation is indicated.

(AU) = status word. Bit indications for the status word are as follows:

Bit Set	Indication
17	UMTH tried to recover seven times unsuccessfully
16-15	not used
14	improper condition
13	duplex control (1540 only, 1 = no, 0 = yes)
12	transport ready (1540 only)
11	xirg detected (1540 only)
10	output timing error
9	input timing error
8	incorrect frame count
7	lateral parity error
6	longitudinal parity error
5	last motion of tape (1 = backward, 0 = forward)
4	tape mark (end of file)
3	no write enable
2	end of tape
1	low tape
0	load point

UMTH does not try to recover upon detection of an input or output timing error.

UMTH assumes the following:

1) That search constants for a backward search must be reversed character-wise (1240 only).

2) That whenever dual-channel operation is selected, the number of words to input or output (parameter 4) must be even. The buffer initial address (parameter 3) may be even or odd.

3) That address 00141 is reserved for use by UMTH as an indirect interrupt address.

4) That the contents of AU, AL, and B prior to entering UMTH need not be restored upon exit.

5. MAGNETIC TAPE DUPLICATION MODULE (MTDUP)

5.1 PROGRAM DESCRIPTION

The MTDUP module of UPAK III may be used with the computer when operating with one magnetic tape unit with two transports and an I/O console. The module copies the content of one magnetic tape (From tape) onto another (To tape). The normal copy process continues until MTDUP encounters two consecutive tape marks or until end-of-tape, whichever occurs first.

MTDUP performs the duplication in the following sequence:

1) Rewind From tape and To tape.

2) Read one record from From tape into user-specified buffer.

3) Use buffer control words to determine buffer limits, and write the record on the To tape. Repeat steps 2) and 3) until two successive tape marks have been found or end-of-tape is detected.

4) Rewind both tapes if verification option is selected.

5) Read one record from From tape into user-specified buffer and checksum the record.

6) Read one record from To tape into user-specified buffer and checksum the record. Compare record checksum for From and To tapes. If not equal, add one to error counter; repeat steps 5) and 6) until two successive tape marks or end-of-tape have been found.

7) Rewind both tapes if rewind option was selected.

8) If the error counter is zero, type COPY-OK; otherwise type ERR-XXX where XXX is the number of copy errors detected.

NOTE: The user may alter the number of successive tape marks terminating his From tape. To do this, he must store the exact number desired at MTDUP label CAT.

MTDUP occupies approximately 712_8 memory locations and may be loaded anywhere in computer memory above address 01000 with the restriction that the entire module must be loaded entirely within one memory bank. The module operates either under program control or manually from the computer control panel.

5.2 INPUT PARAMETERS

The six input parameters to MTDUP are shown in Figure IV-B-3. Parameters 4 and 6 need not be exact terminal addresses but they must be large enough to accommodate the largest record on the From tape. Parameters 5 and 6 are required only for verification.

Parameter 1

17	16	15 14	13	12	11 10	9 8 7 6	5 4 3	2 1 0

Transport No.
for From Tape

Cabinet No. for
From Tape

Channel No. for Magnetic
Tape (odd for dual channel)

Density
00 – 200 Frames/Inch (Low for 1240)
10 – 556 Frames/Inch (High for 1240)
01 – 800 Frames/Inch
11 – Same as last instruction

Parity
0 – Even
1 – Odd

Coding
0 – Bioctal
1 – Octal

Modulus
0 0 – Modulus 3 (18 bits)
0 1 – Modulus 4 (24 bits)
1 0 – Modulus 5 (30 bits)
1 1 – Modulus 6 (36 bits)

Rewind Option
0 – Do not rewind after MTDUP
1 – Rewind after MTDUP

Mode
0 – Single channel
1 – Dual channel

Figure IV-B-3. MTDUP Input Parameters (Sheet 1 of 3)

Parameter 2

```
| 17 16 15 14 13 12 11 10 9 8 7 6 | 5 4 3 | 2 1 0 |
```

Transport No. for
─To tape

Cabinet No. for
─To tape

─Not Used

Parameter 3

```
| 17 16 | 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0 |
```

Buffer Initial Address
─ for From Tape

─Not Used

Parameter 4

```
| 17 16 | 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0 |
```

Buffer Terminal Ad-
─dress for From Tape

─Not Used

Figure IV-B-3. MTDUP Input Parameters (Sheet 2 of 3)

Parameter 5

```
┌─────┬──────────────────────────────────────┐
│17 16│15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0│
└─────┴──────────────────────────────────────┘
                                          Buffer Initial Ad-
                                          dress for To Tape

              Not Used
```

Parameter 6

```
┌─────┬──────────────────────────────────────┐
│17 16│15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0│
└─────┴──────────────────────────────────────┘
                                          Buffer Terminal Ad-
                                          dress for To Tape

              Not Used
```

Figure IV-B-3. MTDUP Input Parameters (Sheet 3 of 3)

5.3 OPERATING PROCEDURES

5.3.1 OPERATION UNDER PROGRAM CONTROL

The calling program enters the active B register with the address of the first of six successive computer words containing parameters 1 through 6 in the required order, and executes a return jump or indirect return jump to MTDUP base address. Upon resumption of control, the calling routine may check the contents of AU and AL for errors (refer to paragraph 5.4). Both the From tape and the To tape must be mounted on the proper transports before starting the calling program.

5.3.2 MANUAL OPERATION

1) Master clear the computer.

2) Set P to MTDUP base address + 2.

3) Mount the From tape and the To tape on tape transports (same I/O channel).

4) Set parameter 1 in AU; set parameter 2 in AL.

5) Start the computer.

6) When the computer stops, set parameter 3 in AU; set parameter 4 in AL.

7) Start the computer.

8) When the computer stops, set parameter 5 in AU; set parameter 6 in AL.

9) Start the computer. When the duplication process is complete and successful, the typeout COPY-OK occurs on the on-line typewriter. If the typeout ERR-XXX occurs, it indicates that XXX errors were detected during the process.

5.4 ALARMS AND STATUS INDICATIONS

An improper condition of a tape unit or a status word error after seven tries causes MTDUP to exit (or stop) with AU equal to the status word from the tape system. This indicates that MTDUP did not complete the duplication. If the tape duplication is completed without tape system errors, the computer stops with AU clear.

If the duplication process was completed without tape system errors, but checksum errors were detected upon check-reading the new tape against the old, the number of such checksum errors (for each separate record on tape) is displayed in AL.

MTDUP assumes the following:

1) That both the From and To tapes are mounted on tape transports served by a common I/O channel.

2) That the I/O console is on-line with the computer.

3) That address 00141 is reserved for use by MTDUP as an indirect interrupt address.

4) That the contents of AU, AL, and B, prior to entering MTDUP, need not be restored upon exit.

6. TRIM III OUTPUT 10 LOAD MODULE (LOAD10)

6.1 PROGRAM DESCRIPTION

In the course of its assembly process, TRIM III produces the object program in tabular form on magnetic tape. This table serves as the source for all assembler outputs reflecting the object program. The table, called output No. 10, may also be loaded into computer memory by this UPAK III module. TRIM III can stack a maximum of 40_8 of these outputs on one tape. Each output is a file on the output tape. Files are numbered 1 through 40 in the order of their appearance on tape. The user has two load options:

1) That the program shall be loaded absolutely with addressing assigned at assembly time.

2) That the program shall be loaded relative to any specified base address.

TRIM III writes each output 10 file on magnetic tape in a specified format. The format consists of a 30-word sentinel record, some number of 140_8-word instruction records, a 1-word sentinel (included in the last record), and a tape mark. All sentinel words consist of the octal characters 747474. The instruction records consist of a 1-word file count, 37_8 instruction items, and two spare words. The file count specifies the number of items from the beginning of the file to the end of the record in which it appears. Each item has the following format:

Word 0	E	Sequential Identifier
Word 1	M	Address
Word 2		Instruction

} Used by TRIM III only

1) Word 0 contains a line error counter, E, and a sequential line identifier used by TRIM III but not by UPAK III.

2) Word 1 contains a modification code, M, and an address. M tells the load routine how to modify the instruction for storage for a relocatable load. M may be any one of the following codes:

Code	Meaning	Type of Instruction
0	No modification	Constant or 4-digit Y unmodified
1	Add base address to Y_{11-0}	4-digit Y modified
2	No modification	5-digit Y unmodified
3	Add base address to Y_{14-0}	5-digit Y modified (bit 15 is set to 0 or 1 depending on specified base address)

The address is the address for this instruction assigned at assembly time.

3) Word 2 contains the machine instruction or constant to be stored.

6.2 INPUT PARAMETERS

Two parameters govern the operation of the TRIM III output 10 load module. They are shown in Figure IV-B-4.

Parameter 1

17 16 15 14 13 12	11 10 9 8 7 6	5 4 3	2 1 0

Transport No. of
—Output 10 Tape

Cabinet No. of
—Output 10 Tape

I/O Channel
—No. $(0-17_8)$

File No. of Output
—10 to be Loaded $(0-40_8)$

Parameter 2

17 16	15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

Base Address for Relocatable Load,
——Zero for Absolute Load

—Not Used
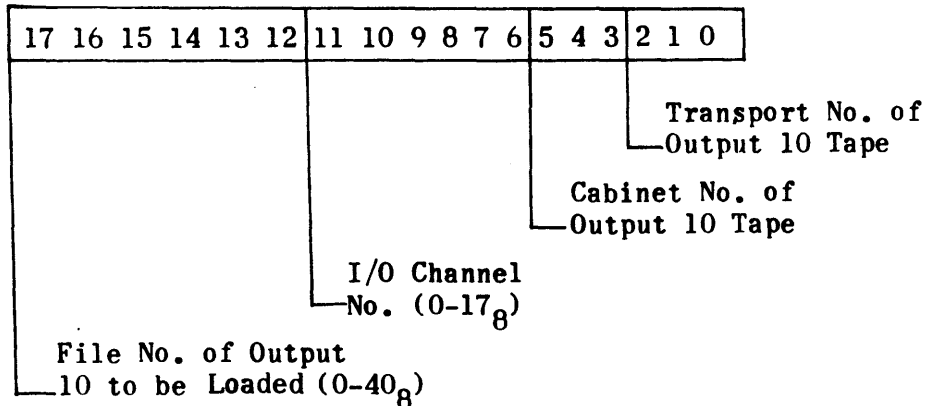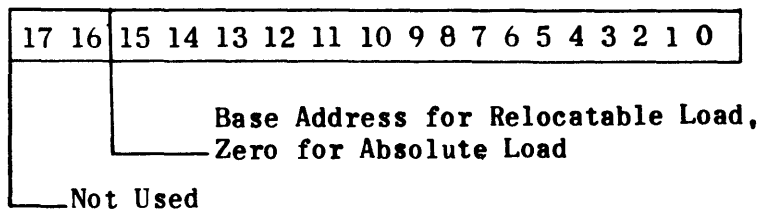
Figure IV-B-4. LOAD10 Input Parameters

6.3 OPERATING PROCEDURES

6.3.1 OPERATION UNDER PROGRAM CONTROL

The calling program must enter AU with parameter 1, enter AL with parameter 2 if the load is relocatable, and execute a return jump or indirect return jump to this module base address. The output 10 tape must be mounted on the proper transport before starting the calling program.

6.3.2 MANUAL OPERATION

1) Master clear the computer.
2) Set P to the output 10 load base address + 2.
3) Mount the output 10 tape on a transport.
4) Set parameter 1 in AU; set parameter 2 in AL.
5) Start the computer. When the computer stops with AL clear, the load is complete.

6.4 ALARMS AND STATUS INDICATIONS

After completing (or attempting) the load, the module returns control to the using program with AL containing one of the following status conditions:

1) (AL) = 0, indicates a successful load.
2) (AL) positive, indicates the load routine has attempted to read the same record seven times and failed. AL contains the status word.

LOAD 10 always stops if an improper condition arises. AL contains the tape status word normalized left (sign bit set).

LOAD 10 uses address 00141 as an indirect interrupt address.

LOAD 10 does not restore the original contents of AU, AL, or B.

7. INSPECT AND CHANGE AND STORE CONSTANT MODULE

7.1 PROGRAM DESCRIPTION

This module consists of two routines: inspect and change, and store constant in memory. The inspect and change routine causes the contents of the memory location specified in AU to be displayed in AL. The contents of AL may then be changed manually. (AL) is then returned to the memory address from which it was taken. The inspection address need be entered only the first time, since (AU) is increased by one and the contents of sequential addresses are brought into AL with each successive performance of the inspect and change function. Should the user wish to inspect the contents of some address other than the next sequential address, he may do so by setting the new address in AU before returning (AL) to memory.

The store constant in memory routine permits the user to load a specified area of memory with a value manually entered into AU. If (AU) = 0 the area is cleared.

7.2 OPERATING PROCEDURE

7.2.1 INSPECT AND CHANGE

1) Master clear the computer.

2) Set P to the module base address.

3) Set AU to the memory address to be inspected.

4) Start the computer. The computer stops with the address in AU and its contents in AL.

5) The user may now change the address in AU and/or the contents in AL.

6) Start the computer. One of the following occurs:

    a) If the contents only were altered, the new contents are stored at the original address and the computer stops with the next sequential address in AU and its contents in AL.

    b) If the address only was changed, the contents of AL are restored and the computer stops with the new address in AU and its contents in AL.

    c) If both the address in AU and its contents in AL were changed, the new contents are stored at the original address, and the computer stops with the new address in AU and its contents in AL.

7) Any number of such sequences may be executed starting with step 5).

7.2.2 STORE CONSTANT IN MEMORY

1) Master clear the computer.

2) Set P to the module base address + 1.

3) Set the first storage address in AU.

4) Set the last storage address in AL.

5) Start the computer. The computer stops with AU cleared.

6) Set the desired constant in AU.

7) Start the computer. The specified constant is stored in all memory locations within the parameters established in steps 3) and 4).

8) Additional entrances may be made by starting from step 3).

8. PRINT MEMORY CONTENTS (PRINTC)

8.1 PROGRAM DESCRIPTION

The PRINTC module of UPAK III may be used when the computer is operating with a high-speed printer. This module lists a specified area of computer memory on the high-speed printer, suppressing the printout of any zero words. PRINTC prints each line in the following format:

1) Nine columns of data. separated from each other by five spaces, except for the first column, which is separated by six spaces.

2) The first column contains the 5-digit octal address of the memory word in the second column.

3) Columns two through nine contain the contents of eight (10_8) consecutive addresses beginning with the address shown in the first column. The first 2 digits (function code position) of these memory words are separated from the other four by one space.

4) If the contents of any memory address to be listed is zero (000000), the zero codes are not printed and that column is left blank. If the contents of the addresses of an entire line are zeros, the address in column one is not printer either.

5) If the contents of addresses for two or more consecutive lines are zeros, PRINTC still only allows one blank line between the last printed line and the next group of eight (10_8) addresses containing a nonzero quantity, no matter how much memory area lies in between.

PRINTC occupies approximately 450_8 memory locations and may be loaded anywhere in computer memory above address 01000 with the restriction that the entire module shall be contained entirely within one memory bank. The module operates either under program control or manually from the computer control panel.

8.2 OPERATING PROCEDURE

Prior to operating this module the operator must initialize the high-speed printer and ensure that the printer is loaded with enough paper to print the required information.

8.2.1 OPERATION UNDER PROGRAM CONTROL

The calling program must set AU to the first address to be printed and AL to the last address to be printed before executing a return jump or indirect return jump to the PRINTC base address.

8.2.2 MANUAL OPERATION

1) Master clear the computer.

2) Set AU to the first address to be printed; set AL to the last address to be printed.

3) Set P to the PRINTC base address + 2.

4) Start the computer. When the designated area has been processed, PRINTC stops at the base address + 6 with AU and AL cleared. Another memory area may be designated at this point and the dump started without changing the P register.

9. CARD HANDLER (DATCD)

9.1 PROGRAM DESCRIPTION

DATCD consists of two subroutines which provide the input and output of memory data in a specific format on 80 column punched cards via the card processor. The two subroutines are:

1) CLOD - Load absolute or relocatable cards (TRIM III, Output No. 13).
2) CDMP - Dump absolute cards (TRIM III, Output No. 13).

DATCD may be loaded any where in computer memory above address 01000 with the single restriction that the entire module must be contained within a single memory bank. Both functions operate either manually or under program control.

CLOD is a subroutine of DATCD that loads instruction words in relocatable format which it processes from 80-column Hollerith-coded cards read via the card processor. The user specifies the memory area or areas into which the information is loaded.

The input format consists of three types of cards.

1) Address card.

The address card contains an address in columns 1 through 6 and a control digit of 1 in column 80. CLOD examines the 80th column control digit of the first card that it buffers in. If it is a 1, CLOD takes the address contained in columns 1 through 6 of that card as the initial load address. If it is a 4, CLOD assumes that no address card is present and uses the contents of the upper half of the A register (AU) as the initial load address. Any digit other than a 1 or 4 in the 80th column of the first card causes CLOD to ignore that card and continue to read cards until a control digit of 4 is found.

Other than the first 6 columns and column 80, the rest of the columns on the address card are normally blank (per TRIM III, Output 13, or the output from CDMP - the memory dump on cards). However, if the user wishes to load manually prepared (key punched) cards in compatible format, he may cause CLOD to skip its checksum procedure by punching the three letters ICS in columns 10, 11, and 12 of the address card (ignore checksum).

Hence, the address card is an optional card with an optional feature on the card itself.
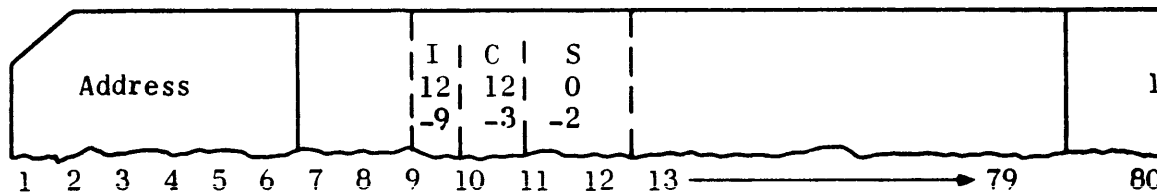
Figure IV-B-5. DATCD Address Card Format

2) Instruction cards.

Following the address card (if any), each card except the last card has the following format:

a) Eight (10 octal) 6-digit computer instructions or constants in columns 1 through 48 inclusive.

b) Eight (10 octal) relocatable modification digits (one for each data word) in column 58 through 65. These digits have the same meaning as those on a relocatable bioctal paper tape:

0 - No modification.
1 - Add base address to Y_{11-0} (4-digit Y modified).
2 - No modification.
3 - Add base address to Y_{14-0} (5-digit Y modified bit 15 is set to 0 or 1 depending on requested base address for relocatable load).
4 - Increment current load address by constant word value (negative or positive).
5,6 - Not used.
7 - End of load (corresponding data word not loaded). Naturally, this digit appears only upon the last card of the load.

c) One 6-digit cumulative checksum, generated by the TRIM III number 13 output or CDMP, in columns 67 through 72. For cards prepared on the key punch with ICS in columns 10 through 12 of the address card, these columns are ignored. Cumulative means that the checksum of the preceding card is added to the present card's checksum and punched, and so forth.

d) One 4-digit card number in either decimal or octal notation in columns 75 through 87. CLOD makes no reference, examination, or use of this number, as it is optional.

e) A control digit of 4 in column 80. This is mandatory for all cards to be loaded by CLOD; any other digit causes the data of that card to be ignored and the next card read.

1st Instruction Word	2nd thru 8th Instruction Words (6 Columns Each)	8 Modification Digits	Cumulative Checksum (Optional)	Card Number (Optional)	4

1 2 3 4 5   6 7 ——————▸ 48   58 ———▸65   67 ———▸ 72   75 ———▸ 78   80

Figure IV-B-6. DATCD Instruction Card Format

3) Last instruction card.

The last card of any relocatable or absolute card load has almost exactly the same format as the other data cards. The number of loadable instruction words on the last card varies from seven to none. The last card must always contain a constant word of any value which has a corresponding modification digit of 7. CLOD ignores this last word and terminates.

CDMP is a subroutine of DATCD that dumps a specified area of core memory on 80 column cards. It first converts the octal words of the specified memory area into excess-three (XS-3) code. It converts enough words to fill the card buffer or finish the specified area, and then it punches the data via the card processor onto 80-column Hollerith-coded cards. This process continues until the entire specified area has been punched.

The format of the cards punched by CDMP is given below.

1) Address card.

The first card punched by CDMP is the address card which contains the initial dump address in columns 1 through 6, and a control digit, 1, in the 80th column. This is the same format as shown in Figure IV-B-5 except that no information is ever punched in columns 10, 11, and 12.
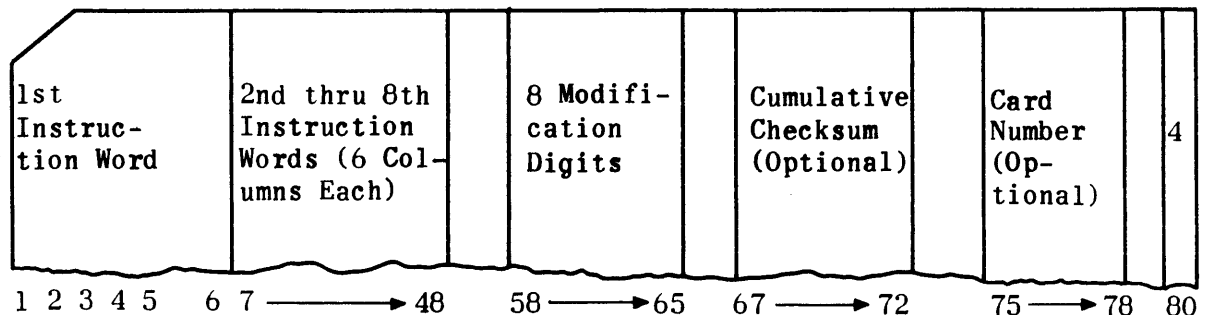
2) Instruction cards.

Following the address card, each card except the last card has the following format:

a) Eight (10 octal) consecutive 6-digit computer memory words, in columns 1 through 48 inclusive.

b) Eight (10 octal) relocatable modification digits, (one for each memory word) in columns 58 through 65. Because this is an absolute dump, all of these modification digits are zero.

c) One 6-digit cumulative checksum in columns 67 through 72. Cumulative means that the checksum from the previous card is added to the checksum of the present card before being punched.

d) One 4-digit card number in columns 75 through 78. The cards are numbered in octal notation.

e) A control digit of 4 in column 80.

3) Last instruction card.

The last card in any specific memory dump has almost exactly the same format as the normal instruction cards. The difference is that the last card contains from zero to seven memory words with a like number of zero modification digits. Following the last memory word of a dump (either on the same card or the beginning of the next) CDMP punches a word of 6 zeros with a corresponding modification digit of 7.

9.2 INPUT PARAMETERS

The card dump section of DATCD (CDMP) requires as input parameters the first and last address of the memory area to be punched on cards. These addresses must be present in AU and AL in any order (first-last, or last-first).

The card load section of DATCD (CLOD) requires as an input parameter the address to begin the memory load. The address may be specified in AU or on an address card.

9.3 OPERATING PROCEDURE

To load 80-column cards in TRIM III, Output No. 13 format:

1) Place the cards in the card reader input hopper and initialize the card reader.

2) If the first card is an address card, no parameters are required as input to DATCD. If no address card is present, AU must be set (manually or by the calling program) to the base load address.

3) If operation is under program control, the calling program must execute a return jump or indirect return jump to DATCD base address.

4) For manual operation, set P to the DATCD base address + 2, and start the computer.

To dump memory on 80-column cards:

1) Make certain the input hopper of the punch unit contains enough blank cards to punch the data and initialize the card punch.

2) If operation is under program control, the calling program must first set AU and AL to the first and last addresses of the area to be dumped, and then execute a return jump or indirect return jump to DATCD base address + 3.

3) If operation is manual, set AU and AL to the first and last addresses of the area to be dumped, set P to DATCD base address + 5, and start the computer.

9.4 ALARMS

The DATCD card load routine (CLOD) always stops before loading any information from a card if the load address is below 01000. Data from the card can be loaded at the request address by restarting the computer. However, DATCD stops before loading information from the next card if the current load address is still below 01000. This cycle continues until the load address exceeds 01000.

Unless the ignore checksum option is being employed, DATCD's load routine stops if it detects an error in the cumulative checksum during a load. Since the checksum total of all cards preceding any specific card is included in that card's checksum, the omission or disorder of any cards from a given dump results in a checksum error during an attempted load.

10. PRINTER LINE IMAGE ON TAPE AND TAPE-TO-PRINTER MODULE (POTPOP)

10.1 PROGRAM DESCRIPTION

POTPOP is a module of UPAK III used when operating with magnetic tape units and a high-speed printer. One routine (POT) of the module writes a pre-formatted, XS-3 coded, 132-decimal character buffer on magnetic tape for subsequent listing on the printer. POT also writes the tape mark which identifies the end of the listing for the listing routine.

The other routine (POP) lists a POT-generated magnetic tape on the printer. POP continues to read and print line-records from the designated magnetic tape until it detects a tape mark.

10.2 INPUT PARAMETERS

The input parameters required by this module are illustrated in Figure IV-B-7. Both parameters 1 and 2 are used by POT. The POP routine uses only the lower order 10 bits of parameter 1.

Parameter 1
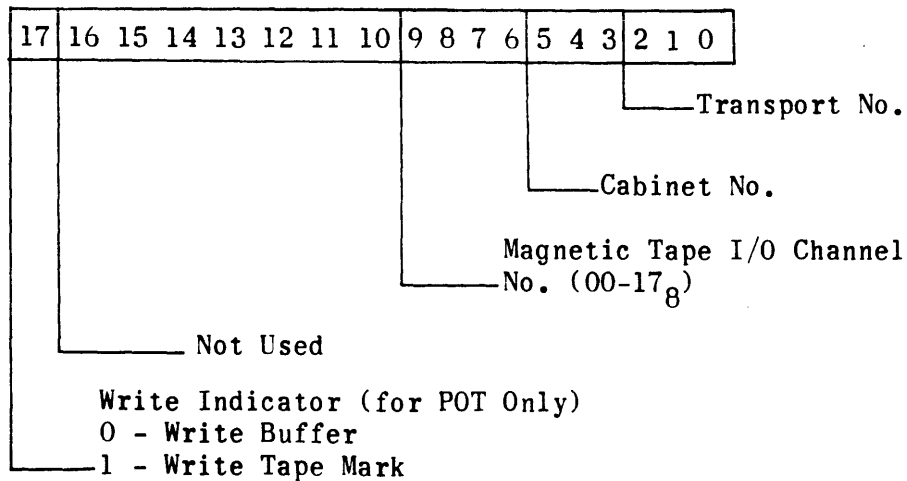
Figure IV-B-7. POTPOP Input Parameters (Sheet 1 of 2)

Parameter 2

```
      | 17 16 | 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0 |
      |       |                                       |
      |       |   Buffer Initial Address
      |       └──(54₈-word Buffer)
      └──Not Used
```
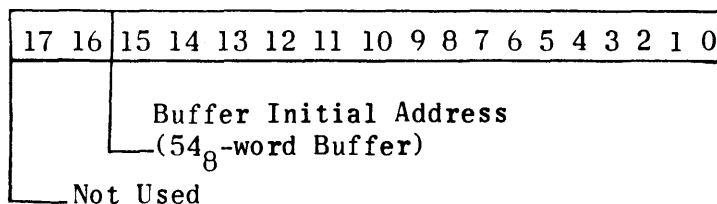
Figure IV-B-7. POTPOP Input Parameters (Sheet 2 of 2)

10.3 POT OPERATING PROCEDURES

To write a printer line image or tape mark on magnetic tape, perform the procedures listed under paragraphs 10.3.1 or 10.3.2.

10.3.1 UNDER PROGRAM CONTROL

The tape on which information is to be written must be mounted on the transport specified by parameter 1 and the tape unit must be placed in the ready condition before starting the calling program. The calling program must set parameters 1 and 2 in AU and AL respectively and execute a return jump or indirect return jump to the POTPOP base address. Each time the POTPOP module is called in this manner it writes either a 54_8-word buffer or a tape mark on tape. If a buffer is to be written, the calling program must store the information in the buffer area in XS-3 code. If the information is written without error, POTPOP returns to the calling program with AU and AL clear.

10.3.2 MANUAL OPERATION

1) Master clear the computer.

2) Mount the tape on which information is to be written on a tape transport.

3) Set parameter 1 in AU.

4) Set parameter 2 in AL.

5) Set P to the POTPOP base address + 2.

6) Start the computer. Either a tape mark or a 54_8-word buffer beginning at the address set in AL is written on tape. After the information is written the computer stops with AU and AL clear.

10.4 POP OPERATING PROCEDURES

To list a POT-generated tape on the printer, perform the procedures listed under paragraphs 10.4.1 and 10.4.2.

10.4.1 UNDER PROGRAM CONTROL

The POT-generated tape must be mounted on the tape transport specified, the magnetic tape unit must be placed in the ready condition, and the printer must be initialized with sufficient paper in the printer before the calling program is started. The calling program must enter AU with the address of the tape transport (parameter 1) and execute a return jump or indirect return jump to the POTPOP base address + 3. The POP routine reads tape and sends the information (coded in XS-3) to the printer in 54_8-word buffers. When the POP routine reads a tape mark the routine returns control to the calling program with AU and AL clear.

10.4.2 MANUAL OPERATION

1) Master clear the computer.

2) Mount the POT-generated tape on a tape transport.

3) Set parameter 1 in AU.

4) Initialize the high-speed printer.

5) Set P to the POTPOP base address + 5.

6) Start the computer. The POP routine reads tape and prints the information on the printer. When a tape mark is read the computer stops with AU and AL clear.

10.5 ALARMS AND STATUS INDICATIONS

POT or POP tries one recovery upon detecting an error in writing or reading a printer line record on magnetic tape. If an error occurs on the second attempt, the routines ignore it and proceed normally.

An improper condition status word from the tape unit causes POT or POP to stop with:

    (AU) = 777777   (AL) = 777777

The user may remedy the problem and restart from that point. POT will try writing the last buffer it was given; POP will attempt to read the next record on tape.

11. MAGNETIC TAPE HANDLER MODULE (JOSH)

11.1 PROGRAM DESCRIPTION

The JOSH magnetic tape handler provides users with the ability to implement all of the hardware capabilities of UNIVAC magnetic tape systems while in on-line operation with the computer.

Under program operation JOSH provides for three types of user control via three separate entrances.

1) JOSH base address + 0 - **Initiate tape function.**

   JOSH initiates the requested tape function and returns control to the calling program allowing it to proceed while the tape action is in process. The calling program may wait for an interrupt and process it or let JOSH process it by immediately jumping to the JOSH base address + 2.

2) JOSH base address + 2 - **Job completion status check.**

   The tape handler interrupt routine stores a coded status indicator in the first word of the tape function request packet. The user may perform his own completion check by analyzing the contents of this word. More simply, he can enter JOSH at base address + 2 and JOSH will perform the check for him and indicate completion status via a good or bad exit. A bad exit is executed by an indirect jump to the next instruction in the calling program. A good exit is executed by an indirect jump to the next instruction + 1 in the calling program.

3) JOSH base address + 4 - **Check busy status.**

   If the user wishes, he may execute JOSH at this entrance to sample current status of the tape function. JOSH returns control to the calling program immediately with the indication in AL. If the function is still in progress (busy), AL is set to a nonzero value; if the function is complete (hardware interrupt status in), AL is clear.

JOSH provides automatic recovery capability by performing up to 10 recovery attempts in an effort to complete a tape function successfully.

JOSH may be operated either under program control or manually from the computer control panel. This UPAK III module may be loaded anywhere in computer memory above address 01000 with the single restriction that the entire module must be wholly contained within one memory bank. The JOSH program occupies approximately 640_8 memory locations.

11.2 INPUT PARAMETERS

When operating under control of a user's program, JOSH requires up to seven words of input parameters. These words must be stored in sequential memory addresses. The first word must always be zero; it is used as the storage location for the magnetic tape status word. The next six words are used to store parameters 1 through 6 shown in Figure IV-B-8. Only those parameters required by the operation codes to be performed need be stored. However, the parameters must be stored in order; that is, if parameter 6 is required it must be stored in the seventh address of the parameter area. When JOSH is operated manually, parameters shown in Figure IV-B-8 are entered manually into AU and AL.

Parameter 1

```
┌──┬────────────────┬─────────────┬─────┬─────┐
│17│16 15 14 13 12 │11 10 9 8 7 6│5 4 3│2 1 0│
└──┴────────────────┴─────────────┴─────┴─────┘
```

└─Transport No.

└─Cabinet No.

└─Magnetic Tape I/O Channel No. (0-17$_8$)

Operation Code

00000 - Ø - Read
00001 - 1 - Read Selective
00010 - 2 - Read - Ignore Error Halt
00011 - 3 - Read - Retain Control (Tape Handler Holds
                        Until Function is Complete)
00100 - 4 - Search - Type I
00101 - 5 - Search - Type II - Identical
00110 - 6 - Search File - Type I
00111 - 7 - Search File - Type II - Identical
01000 - 10 - Write
01001 - 11 - Write - Extended Inter-Record Gap
01010 - 12 - Write - Ignore Error Halt
01011 - 13 - Write - Extended Inter-Record Gap -
                        Ignore Error Halt
01101 - 14 - Write Tape Mark
01101 - 15 - Write Tape Mark - Extended Inter-Record Gap
01110 - 16 - Pass N Files Forward
01111 - 17 - Pass N Records Back
10000 - 20 - Pass N Records Forward
10001 - 21 - Backspace
10010 - 22 - Backspace - Read
10011 - 23 - Pass N Files Back
10100 - 24 - Back Search - Type I
10101 - 25 - Back Search - Type II - Identical
10110 - 26 - Back Search File Type I
10111 - 27 - Back Search File Type II - Identical
11000 - 30 - Rewind
11001 - 31 - Rewind - Clear Write Enable
11010 - 32 - Rewind - Read
11011 - 33 - Rewind - Read - Clear Write Enable

Mode

0 - Dual Channel
1 - Single Channel

Figure IV-B-8. JOSH Input Parameters (Sheet 1 of 3)

IV-B-36

Parameter 2

```
┌──────────┬──────────┬─────────┬─────┬─────────────┐
│ 17 16 15 │ 14 13 12 │ 11 10 9 │ 8 7 6│ 5 4 3 2 1 0 │
└──────────┴──────────┴─────────┴─────┴─────────────┘
```

                                            No. of Records to Pass or
                                        └─Files to Space (0-77₈)

                                    Density

                                    000 - 200 Frames/Inch (Low for 1240)
                                    010 - 556 Frames/Inch (High
                                    001 - 800 Frames/Inch
                                  └─011 - Same as Last Instruction

                            Parity

                            000 - Even
                          └─001 - Odd

                    Coding

                    000 - Bioctal (2 Characters/Frame)
                  └─001 - Octal (Redundant)

            Modulus

            000 - Modulus 3 (18 bits)
            001 - Modulus 4 (24 bits)
            010 - Modulus 5 (30 bits)
          └──011 - Modulus 6 (36 bits)

No. of Records to Pass or Files to Space (0-77$_8$)

Density
000 - 200 Frames/Inch (Low for 1240)
010 - 556 Frames/Inch (High
001 - 800 Frames/Inch
011 - Same as Last Instruction

Parity
000 - Even
001 - Odd

Coding
000 - Bioctal (2 Characters/Frame)
001 - Octal (Redundant)

Modulus
000 - Modulus 3 (18 bits)
001 - Modulus 4 (24 bits)
010 - Modulus 5 (30 bits)
011 - Modulus 6 (36 bits)

Parameter 3

```
┌───────┬──────────────────────────────────┐
│ 17 16 │ 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0 │
└───────┴──────────────────────────────────┘
```

                Buffer Initial Address
                for Read, Write, and
              └──Search Operations

        └────Not Used
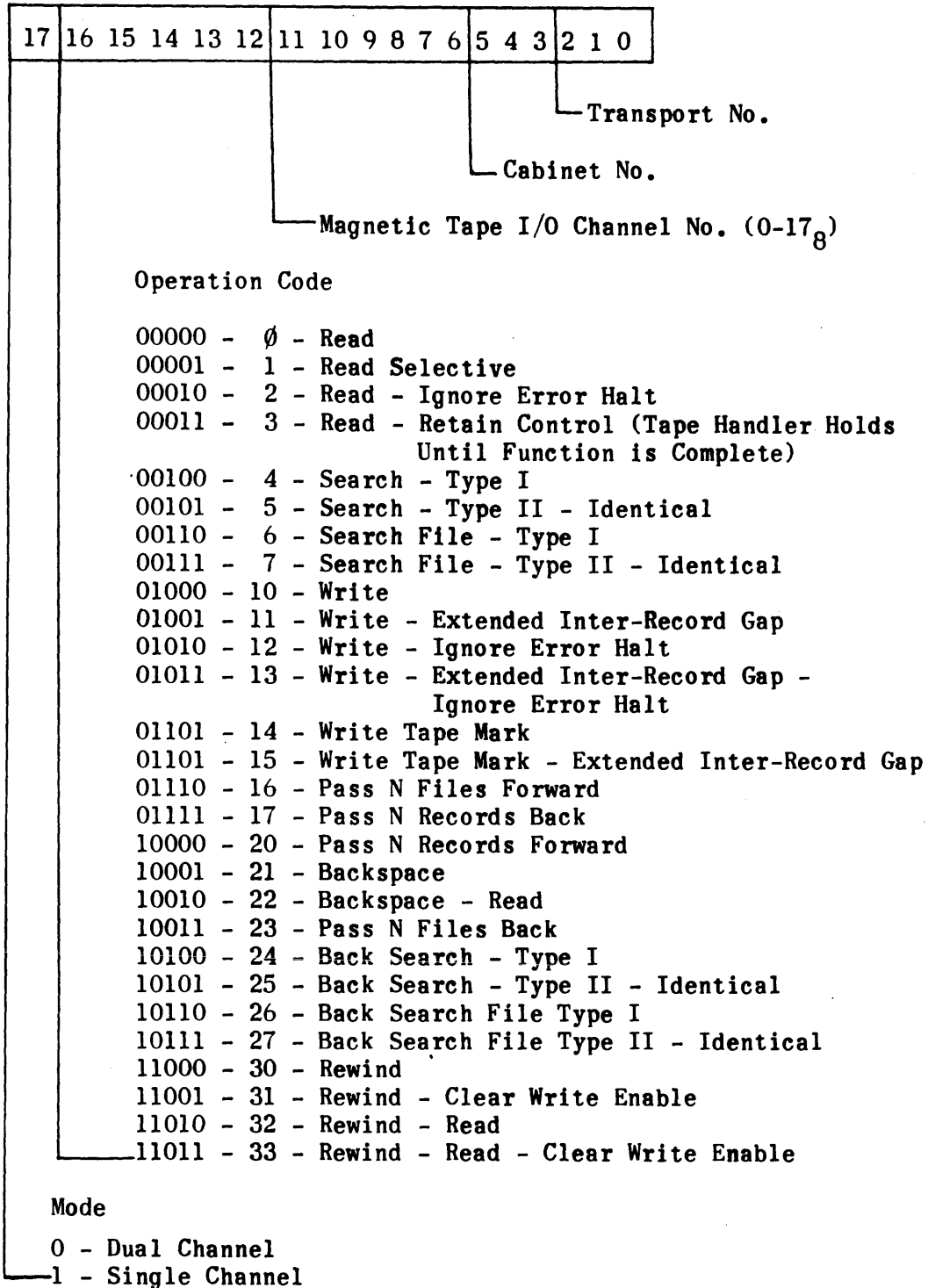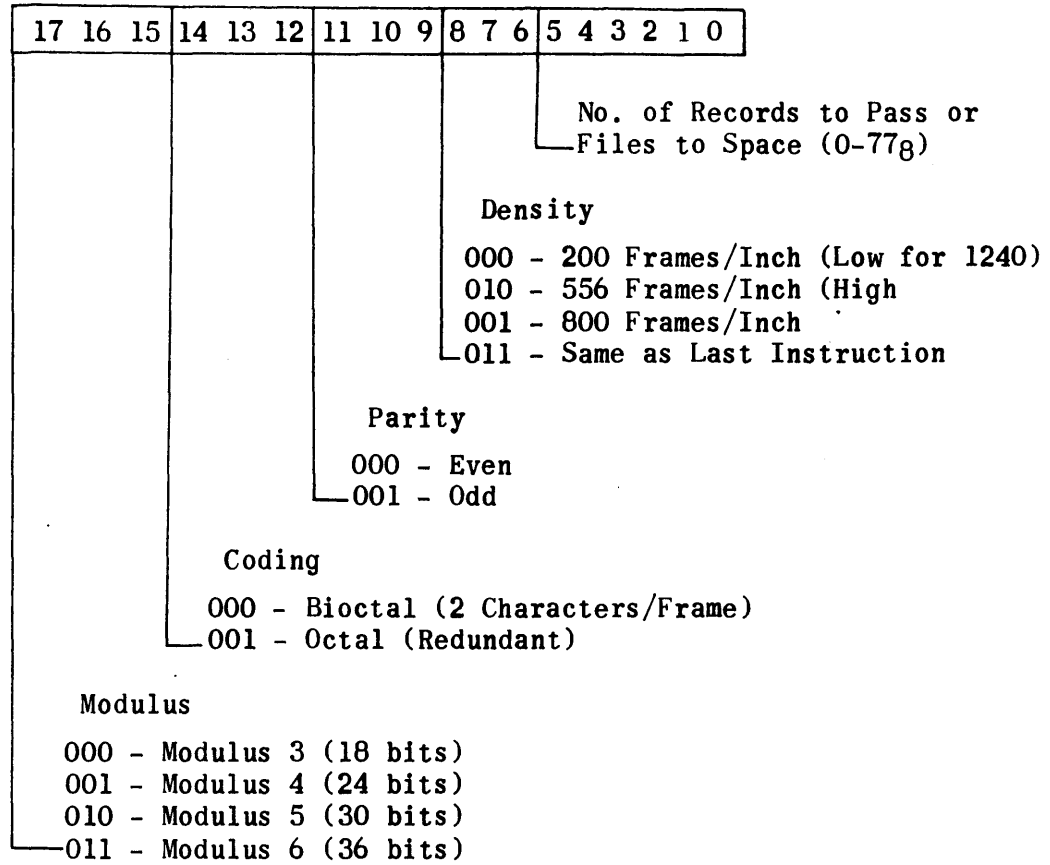
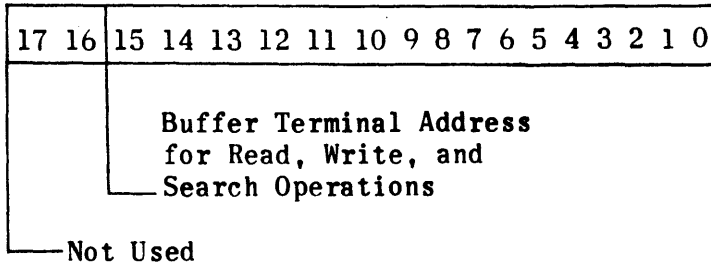Figure IV-B-8. JOSH Input Parameters (Sheet 2 of 3)

Parameter 4

17 16	15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

Buffer Terminal Address
for Read, Write, and
Search Operations

Not Used

Parameter 5

17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

Search Constant for Single
Channel Search, First Half
of Search Constant for Dual
Channel Search

Parameter 6

17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

Second Half of Search
Constant for Dual Channel
Search

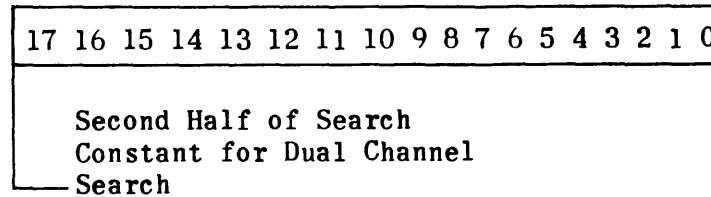Figure IV-B-8. JOSH Input Parameters (Sheet 3 of 3)

11.3 OPERATING PROCEDURES

11.3.1 OPERATION UNDER PROGRAM CONTROL

Before starting the calling program, any required tape must be mounted on the specified transports and the magnetic tape unit must be placed in the ready condition. The calling program must contain the required parameters in consecutive memory locations. The calling program defines the parameter locations by entering the active B register with the first address of the parameter packet before executing a return jump or indirect return jump to the JOSH base address. The first word of the packet must be a blank word (zero). JOSH automatically extracts the required parameters, initiates the tape function, and returns control to the calling program.

At any time the calling program may check to determine if the tape function is completed by executing a return jump or indirect return jump to JOSH base address + 4. JOSH checks the tape function and returns control to the calling program with:

1) AL clear if the tape function is complete,
2) AL set to a nonzero value if the tape function is still in process.

The calling program may check the status of an attempted tape operation by either of two methods:

1) If the calling program executes a return jump or indirect return jump to JOSH base address + 2, JOSH waits for the interrupt, interprets the status word, and returns control to the calling program via a good or bad exit.

   a) A good exit returns control to the next instruction + 1 (skips one instruction) in the calling program. AU and AL are both clear upon return.

   b) A bad exit returns control to the next instruction in the calling program with the following status indications in AU and AL:

      (AU) ⇒ tape unit status word
      (AL) ⇒ 1, 2, 3, 4, 5, or 6
               1 ⇒ tape mark
               2 ⇒ improper condition
               3 ⇒ unrecoverable tape error
               4 ⇒ parameter error
               5 ⇒ low tape
               6 ⇒ end of tape

2) The calling program may wait for the interrupt and interpret the status word by checking for the status codes, defined above, in the first word of the parameter packet (0 indicates successful completion). If this method is used, the calling program must ensure that the interrupt has been received before attempting to interpret the status. This can be done by looping through an RJP or IRJP to JOSH base address + 4 and checking status only when control is returned with AL clear.

11.3.2 MANUAL OPERATION

1) Mount any required tape on a tape transport and place the tape unit in the ready condition.

2) Master clear the computer.

3) Set P to the JOSH base address + 10.

4) Set AU to parameter 1; set AL to parameter 2.

5) Start the computer. The computer stops with AU and AL clear.

6) Set AU to parameter 3; set AL to parameter 4. If these parameters are not necessary for the operation code selected in parameter 1, leave AU and AL clear.

7) Start the computer. The computer stops with AU and AL clear.

8) Set AU to parameter 5; set AL to parameter 6. If these parameters are not necessary for the operation code selected in parameter 1, leave AU and AL clear.

9) Start the computer. When the tape function is completed, the computer stops with the status word displayed in AU for operator interpretation.

11.3.3 SPECIAL CONSIDERATIONS

1) Search constants for a backward search must be reversed characterwise.

2) Whenever dual-channel operation is selected, the buffer must specify an even number (2, 4, 6, 8, and so forth) of memory words.

3) Address 00141 is reserved for use by JOSH as an indirect interrupt address.

4) The contents of AU, AL, and active B are not restored.

5) If the user elects to use the status checking feature of JOSH base address + 2, he must do so before re-entry into JOSH to do the next tape function.

SECTION IV-C. TRIM CORRECTOR

1. GENERAL INFORMATION

The TRIM corrector program is a service routine designed to provide operators and programmers with the means to correct source programs without manually preparing a new source program tape. The TRIM corrector first reads and stores a list of correction operations from punched tape. Each correction operation contains an identifier which relates the correction to a specific operation in the source program. The correction operations may be read in any order and may be contained on any number of tapes; however, all correction tapes must be read before any source program tapes are read. After each tape is read, the program stops to allow the next tape to be mounted in the reader. After all corrections have been read, the program sorts the corrections by identifier into a sequential list in ascending order. The program then reads the uncorrected source program one operation at a time. The source program may be read from source program tapes (manually punched or TRIM Output No. 11) or from program listing tapes (TRIM Output No. 2); however, all tapes used in a correction run must be of the same type. After each tape is read, the program stops to allow the next tape to be mounted in the reader.

If source program tapes are read, the program assigns a sequential identification number to each operation as the operation is read. The first source operation read is assigned the number 0, the second operation is assigned the number 1, and so forth. No attempt is made by the program to determine the type of operation being read. If the source program is read from a program listing tape (TRIM output No. 2) the program discards all information on the tape except the L1ID numbers and the source operations. The L1ID numbers are used as the identification numbers.

As each source operation is read (from either type of tape), the program determines the identification number of the operation and compares the number with the identifier of the next correction to be processed. If the source operation identification number is smaller, the source operation is punched on the corrected source tape without alteration. If the correction identifier is smaller, the correction operation is punched on the corrected source tape. If the two numbers are equal, the correction operation is inspected to determine the type of correction specified. The correction is then processed and the next source operation is read. When all correction and source operations have been processed, the program punches a double period and the checksum on the corrected source tape and comes to a final stop. The corrected source tape is ready for input to either the TRIM I or II assembly system.

2. INPUT FORMATS

Correction tapes must begin with a carriage return and terminate with a carriage return and two periods. Each correction on the tape must consist of an identifier and one of three types of correction operations. The general format for a correction is as shown on the next page.

$$\left[\text{Identifier}\right]_{\curvearrowleft}$$

$$\left[\text{Correction operation}\right]_{\curvearrowleft}$$

The identifier may consist of an integer only or an integer and a fraction. The integer defines the operation in the source program at which the correction is to be applied. The fraction is used only with insert type corrections to define the sequence in which instructions are to be inserted into a source program. The maximum size of the integer is five characters and the maximum size of the fraction is three characters. When the source program is to be read from tapes generated by a TRIM assembly system, the integer portion of the identifiers used with correction operations may be taken directly from the L1ID numbers on the hard-copy listing of the tape. When the source program is to be read from manually prepared source tapes, identifiers may be determined by sequentially numbering all operations contained on the tapes to be read. One method of accomplishing this numbering process is to first obtain a hard copy of all information on the source tapes and then number each operation sequentially (starting with zero and using octal numbers only). The number of an operation may then be used as the identifier for a correction which applies to that operation.

2.1 DELETE CORRECTION

The identifier of a delete correction is the number of the operation (or first operation of a group of sequential operations) to be deleted from the source program. The correction operation consists of a ⟶, the word DELETE, and a carriage return when only one program operation item is to be deleted. If more than one sequential program operation is to be deleted, the word DELETE is followed by a point separator and the number (in octal) of operations to be deleted. A carriage return must follow the operation. Two examples are given below.

Example 1 deletes item 127 from the source program:

127·0 ⤸

    ⟶ DELETE ⤸

Example 2 deletes items 127 through 133 from the source program:

127·0 ⤸

    ⟶ DELETE·5 ⤸

2.2 REPLACE CORRECTION

The identifier of a replace correction is the identification number of a source program operation to be replaced by the correction operation. The correction operation may be any TRIM source language operation. The correction operation must not exceed 80 characters in length (including control codes)

and must be terminated with a carriage return. The example which follows replaces the source program operation 105 with the correction operation.

105·0 ⤶

CAT ⟶ ENTBK 35 ⟶ INITIALIZE INDEX ⤶

2.3 INSERT CORRECTION

The identifier of an insert correction contains an integer and a fraction separated by a point separator. The integer appears before the point separator and specifies the source program operation after which the correction operation is to be inserted. The fraction defines the correction as an insert correction and also specifies the order in which operations are inserted when more than one operation is to be inserted at the same point in the source program. For example, if two corrections with identifiers 12·1 and 12·2 are read by the TRIM corrector, the correction operation associated with 12·1 is inserted after source operation 12 and the correction operation associated with 12·2 is inserted after correction operation 12·1. The correction operation to be inserted may be any TRIM source language operation. The correction operation must not exceed 80 characters in length (including control codes). An example of the coding required to insert three instructions after operation 106 of a source program follows.

Example:

106·1 ⤶

DOG ⟶ ENTAU·SUM ⤶

106·2 ⤶

⟶ LSHA·6 ⤶

106·3 ⤶

GETOUT ⟶ RJP·DOIT ⤶

Additions may be made to the end of a source program by means of either the insert or replace correction. For example, if the last source program operation is number 320, then either of the methods in the following two examples may be used to add operations.

Example 1 (insert):

320·1 ⤶ .

CAT ⟶ 0·0 ⟶ CLEAR STORAGES SUBR ⤶

320·2

⟶ ENTBK 10

320·3

CAT1 ⟶ CLB·WSTRG

320·4

⟶ BJP·CAT1

320·5

⟶ IJP·CAT

Example 2 (replace):

321·0

CAT ⟶ 0·0 ⟶ CLEAR STORAGES SUBR

322·0

⟶ ENTBK·10

323·0

CAT1 ⟶ CLB·WSTRG

324·0

⟶ BJP·CAT1

325·0

⟶ IJP·CAT

3. PREPARATION OF CORRECTION TAPES

Correction tapes must be punched in TRIM source code (same code as the program
to be corrected). Any device capable of punching this code may be used to pre-
pare correction tapes; however, this description covers the preparation of tapes
on the UNIVAC I/O consoles only. If another device is used, care must be taken
to ensure that the proper codes are generated for the required control codes.
When a UNIVAC I/O console is used, the console must be placed in the off-line
mode and the operator must select the option which permits punching tape while
simultaneously typing on paper by keyboard entry.

The maximum number of correction operations used in a correction run is deter-
mined by the computer memory size. Approximately 200_8 corrections are allowed

for every memory bank (4096_{10} words of core storage). These corrections may be punched on any number of correction tapes.

Every correction tape must begin with a carriage return code and end with a carriage return code followed by two consecutive period codes. Except for special control symbols, the keyboard entries must correspond to the characters which appear in the operations; that is, key A is activated for an A, key B for a B, and so forth. Certain special control symbols used for formatting opertions are not reflected on the keyboard. These symbols are defined in Appendix A, Table A-1, which also lists the key used to represent the symbol and the code generated by that key. In addition to the keys identified in Table A-1, the operator must use the line feed key to advance the paper in the typewriter after each carriage return. The code generated by the line feed key is not used by the TRIM corrector.

A limited amount of in-process corrections may be made while a correction tape is being prepared. Specifically, two types of corrections are allowed. One type deletes the last code punched and the other type deletes all information punched since the last carriage return code. Refer to TRIM I, paragraph 8, for the description of these in-process corrections.

4. OPERATING PROCEDURES

The TRIM corrector is designed to operate with a minimum equipment configuration of a computer and an on-line reader-punch unit. Prior to operating the TRIM corrector the computer and reader-punch unit must be placed in the operational state with all switches in the normal operating position. The TRIM corrector absolute bioctal tape must be loaded into computer memory via a utility package. The procedures for operating the TRIM corrector are given below.

1) Master clear the computer.

2) Set the P register to 00400.

3) Set PROGRAM SKIP keys 1 and 2.

4) Set PROGRAM SKIP key 3 if a side-by-side listing tape is to be used as input.

5) Mount a correction tape in the reader.

6) Start the computer.

7) When the computer stops with the AL and AU registers clear, repeat steps 5) and 6) for the next correction tape. When all correction tapes have been read, proceed to step 8). If, at any time, the computer stops with the AL register set to 007777, the maximum number of corrections has been exceeded. The operator may terminate the operation or process the corrections already stored. To process the corrections already stored, remove the correction tape from the reader and proceed to step 8).

8) Release PROGRAM SKIP key 1.

9) Mount the tape on which the source program is contained. This may be either a source program tape or a side-by-side listing tape. If the source program is contained on more than one tape, the tapes must be read in the proper order (determined by the program sequence and the correction identifiers).

10) Start the computer. If no correction tape has been read, the computer stops immediately with the AL register set to 000007; restart at 1).

11) As the input tape is read, the corrected source program tape is punched. When the computer stops after an input tape is read, repeat 9) and 10) for the next tape. When all tapes which contain the source program have been read, proceed to step 12).

12) Release PROGRAM SKIP key 2.

13) Release PROGRAM SKIP key 3 if this key was set at step 4).

14) Start the computer.

15) When the computer stops, punch a trailer on the output tape (corrected source tape) and remove the tape from the punch. The checksum on the corrected source tape may be verified by performing procedures a) through f) under step 16).

16) Verify the checksum on the corrected source tape as follows:

a) Master clear the computer.

b) Set the P register to 00400.

c) Set PROGRAM SKIP key 4.

d) Mount the tape to be verified in the console reader.

e) Start the computer.

f) When the computer stops with the P register set to 00400, the AU and AL registers display the checksum indication. If both the AU and AL registers are clear, the computed checksum and the tape checksum are the same and the tape may be used as input to a TRIM assembly system. If AU and AL are not both zero, AU contains the computed checksum and AL contains the tape checksum. This indicates that an error occurred during the correction run.

SECTION IV-D. TRIM LIBRARY BUILDER (LIBBLD)

1. GENERAL INFORMATION

The TRIM III library builder, LIBBLD, is a program by means of which a user may add, delete, or replace programs on the assembler library. LIBBLD also has the capability to furnish listings of the library directory and any or all of the programs in the library. The library builder is designed for use with the following minimum equipment configuration:

   1 computer with 16K words of memory
   1 magnetic tape system with two transports
   1 I/O console with paper tape reader, paper tape punch, and typewriter

Optional equipment includes:

   1 card reader and high-speed printer
   1 card punch

Input to and output from the library builder is identical to TRIM III source language formats.

A TRIM III library consists of two parts, the directory which records the names of the library subroutines ordered by their physical appearance on the tape, and the library of subroutines themselves. The library directory is stored on magnetic tape, preceding the library.

2. INPUTS

2.1 BUILDING OR UPDATING

The command tape or deck must always be preceded by a LIBUPD header control operation and a library number identification operation. For example:

   → LIBUPD   •   [name]   •   [date] →

   → LIBNO   •   [library number] →

When building a new library, the library number is an assignment of this number to the library. Once a library has been built, LIBBLD checks the assigned number against that of the LIBNO operation to ensure that they correspond. If they do not, an error typeout occurs.

The operations allowed in updating a library are:

   REC (record - add program at beginning of library)

   DEL (delete - delete program from library)

   WDG (wedge - insert program in library, not at the beginning)

RPL (replace - program with new program)

Only the REC operation is used when building a new library.

These operations are accomplished in two steps:

1) Directory building or updating.
2) Library building or updating.

To insure proper calling, it is the responsibility of the user to enter all programs in the proper order in the directory. This is dictated by the fact that the magnetic tape can never be rewound when calling programs from the library during an assembly run. This requires any program internally calling on another to precede the called program on the library.

The LIBBLD checks all calls within a program to determine if they are listed in proper order in the directory, and if not, prints out a call error on the typewriter.

Input to LIBBLD may be either on paper tape or punched cards.

The format for paper tape input is illustrated in the following example of a command sequence:

    ⟶ LIBUPD•JRS•JUNE 65 ⟶   building or updating header.

    ⟶ LIBNO•35 ⟶   command to update library 35.

    ⟶ REC•DOG ⟶   command to enter program DOG at beginning.

    ⟶ DEL•CAT ⟶   command to delete program CAT.

    ⟶ WDG•MUTT∘JEFF ⟶   command to insert program MUTT following program JEFF.

    ⟶ RPL•MOP ⟶   command to replace old program MOP with new.

    ••   end of commands.

Each entry must be a separate statement. The tab at the end of each statement may be omitted; however, a carriage return must precede each entry (each line shown above). A label may precede the beginning tab on the LIBUPD and LIBNO statements.

The format for card input is illustrated in the following example of command sequence:

```
          _____
         /                                        /|
        /          ENDATA ---                    / |
       /_____/  |
      /                                        /|  /|
     /          RPL*MOP ---                    / | / |
    /_____/  |/  |
   /                                        /|  /|  /|
  /       WDG*MUTT*JEFF ---                 / | / | / |
 /_____/  |/  |/  |
/                                        /|  /|  /|  /
|          DEL*CAT ---                   / | / | / | /
|_____/  |/  |/
|                                        /|  /|  /
|          REC*DOG ---                   / | / | /
|_____/  |/
|                                        /|  /
|          LIBNO*35 ---                  / | /
|_____/
|                                        |
|       LIBUPD*JRS*JUNE 65 ---           |
|                                        |
|_____|
```

Column 21 is the starting point of the command. A blank card must appear as the last card in the command deck.

2.2 LISTING

The command tape or deck must always be preceded by a LIBLST header control operation and a library number identification operation. For example:

→ LIBLST • [name] • [date] →

→ LIBNO • [library number] →

The user may request a listing of the directory (DIR), an individual subroutine, or the entire library of subroutines (LIB).
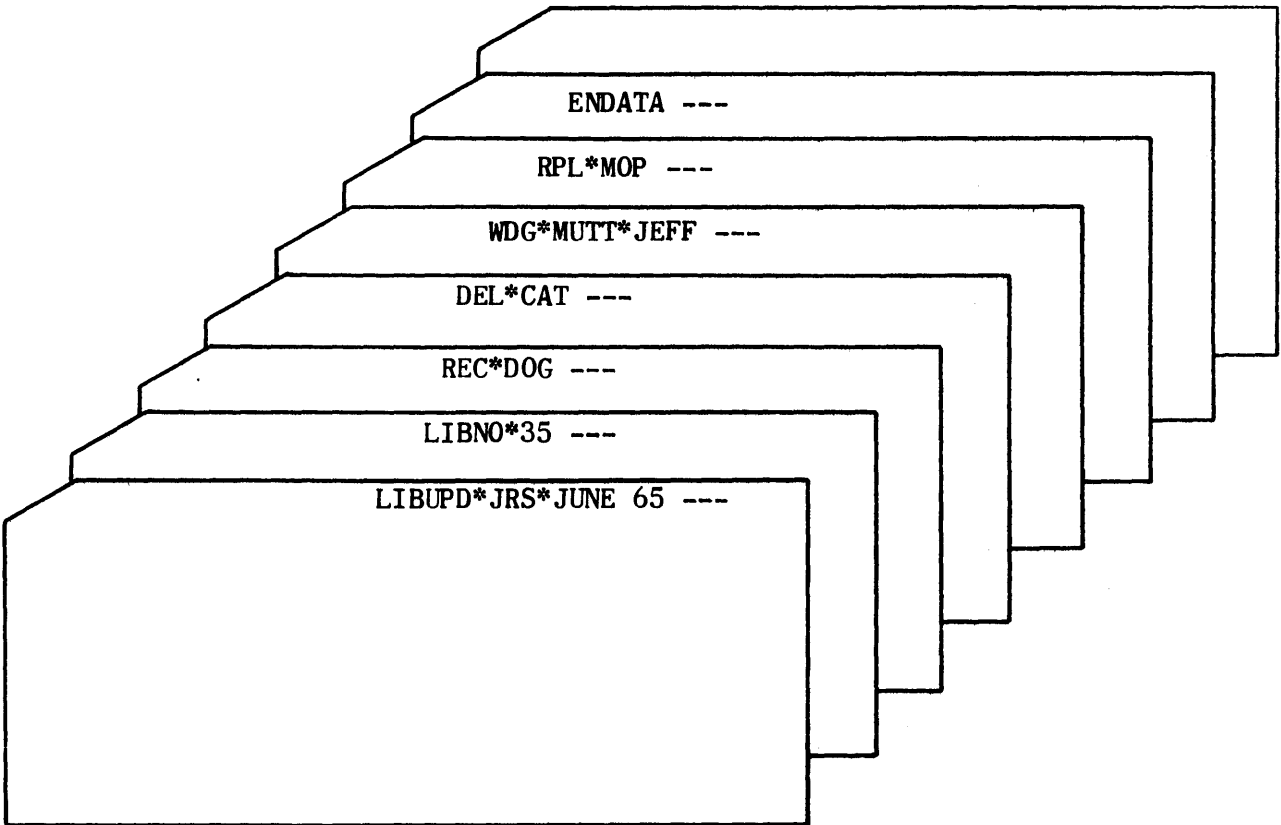
Input to LIBBLD may be either on paper tape or punched cards.

The format for paper tape input is illustrated in the following example of a command sequence:

⟶ LIBLST•JRS•JUNE 65 ⟶   listing header.

⟶ LIBNO•40 ⟶   command to list from library 40.

⟶ LST•DIR ⟶   command to list directory.

⟶ LST•LIB ⟶   command to list whole library.

⟶ LST•CAR•COW•RAT ⟶   command to list programs, CAT, COW, and RAT.

     ••   end of commands.

The tab at the end of each statement may be omitted; however, a carriage return must procede each entry (each line shown above). A label may precede the beginning tab on the LIBLST and LIBNO statements.

The format for card input is illustrated in the following example of a command sequence:

IV-D-4
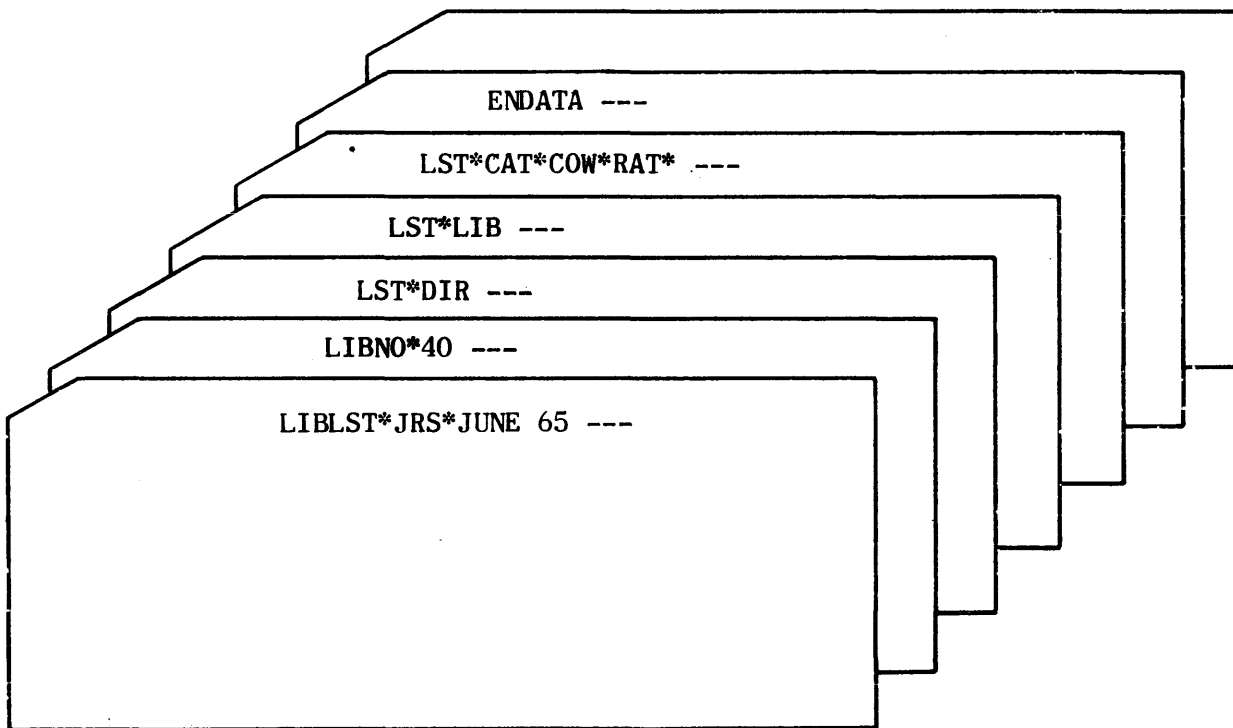
3. OUTPUTS

3.1 BUILDING OR UPDATING

When using LIBBLD to build or update a library, the output is the new library tape, in the format illustrated in Figures IV-D-1 and IV-D-2.

Word	1	7 7 0 0 7 7
	⋮	
	30_8	7 7 0 0 7 7
Word	0	LIBRARY NUMBER
	1	PROG
	2	
	3	LABEL 1
	4	PROG
	5	LABEL 2
	N − 3	PROG
	N − 2	LABEL N
	N − 1	α α α
	N	α α α
	N + 1	7 7 0 0 7 7

A 30_8 word block of 770077 precedes the library directory, and it is terminated by two words of alphas and one word of 770077.

Figure IV-D-1. Library Directory

1	1	1	1	1	1
1	1	1	1	1	1

PROG 1

0	1	0	1	0	1
0	0	0	0	0	2

PROG 2

0	1	0	1	0	1
0	0	0	0	0	2

PROG N

0	1	0	1	0	1
0	0	0	0	0	2
1	1	1	1	1	1

A 30_8 word block of 111111 precedes the library, and it is terminated by one word of 111111. Each major program is followed by a word of DELTAS and a word of 000002.

Figure IV-D-2. Library Routines Format

3.2 LISTING

When using LIBBLD to list the directory or any or all of the library sub-routines, the output may be requested on punched paper tape, punched cards, or a high-speed printer.

4. OPERATING PROCEDURES

Prior to operating the library builder, the computer, magnetic tape unit, and I/O console must be placed in the operational state with all switches in the normal operating position. If the card processor is to be used, it must also be placed in the normal operating position. The library builder absolute bioctal tape must be loaded into computer memory via a utility package. The procedures required to operate the library builder are given below. While operating the program the user must observe typeouts on the console typewriter and refer to the directions provided in paragraph 5.

4.1 LIBRARY BUILDING AND UPDATING PROCEDURES

1) Mount the tape, on which the new or updated library is to be written, on a tape transport.

2) If the process is an update, mount the old library tape on a second tape transport.

3) If input is to be from paper tape, mount the tape containing the control and command operations in the paper tape reader.

4) If input is to be from cards, mount the control and command card deck in the card input hopper, initialize the card reader, and set PROGRAM SKIP key 3 on the computer.

5) If a new library is to built, set **PROGRAM SKIP** key 1.

6) If the operation is to be performed in dual-channel mode, set the appropriate channel mode switch to the dual position. If a library was written on tape in dual-channel mode, it must be updated in dual-channel mode.

7) Master clear the computer.

8) Identify the new and old library tapes in AU and AL, respectively, by setting the magnetic tape channel No. in bits 9 through 6, the cabinet No. in bits 5 through 3, and the transport No. in bits 2 through 0. If dual-channel mode is to be used, the channel No. must be odd.

9) Set P to 20000.

10) Start the computer. The control and command input is read and the computer stops.

11) Set **PROGRAM SKIP** key 2.

12) If any new programs are to be written on the new library, mount the input program on the appropriate input device (paper tape reader or card reader). If input is from cards, **PROGRAM SKIP** key 3 must be set;

if input is from tape, it must not be set. If no new programs are to be written, proceed with step 15).

13) Start the computer. The input program is read, and the computer stops.

14) Repeat steps 12) and 13) for each new program to be written on the library.

15) Release PROGRAM SKIP key 2.

16) Start the computer. The building or updating process is performed and the new or updated library is written on tape. When the process is complete, the computer stops after the typeout DONE occurs.

4.2 LIBRARY LISTING PROCEDURES

1) Mount the library tape on a tape transport.

2) If listing commands are on paper tape, mount the tape in the reader.

3) If listing commands are on cards, mount the cards in the card input hopper, initialize the card reader, and set PROGRAM SKIP key 3 on the computer.

4) Set PROGRAM SKIP key 0 to select output on the high-speed printer; set PROGRAM SKIP key 4 to select output on the card punch. If neither of these keys are set, output is on punched paper tape. Only one of the keys may be set during a given run.

5) Ensure that the output device is ready (paper in the printer, tape in the tape punch, cards in the card punch).

6) If card or printer output is selected and input is from tape, initialize the selected output device.

7) If the operation is to be performed in dual-channel mode, set the appropriate channel mode switch to the DUAL position. If a library was written in dual-channel mode it must be listed in dual-channel mode.

8) Master clear the computer.

9) Identify the library tape address in AL by setting the magnetic tape channel No. in bits 9 through 6, the cabinet No. in bits 5 through 3, and the transport No. in bits 2 through 0. If dual-channel operation is selected, the channel No. must be odd.

10) Set P to 20000.

11) Start the computer. The listings defined by the input command operations are produced on the selected output device and the computer stops after the typeout DONE occurs.

5. TYPEOUTS

1) CALL ERR XXXXXX.

Program XXXXXX was called but is not found in the directory or is not in the proper order. It must follow the calling program on the library because the tape is never rewound while calling programs during an assembly run. Correction must be made before processing is restarted.

2) DIRECTORY.

This typeout, followed by a listing of the directory, occurs whenever a listing of the directory is requested on paper tape.

3) DONE.

The building, updating, or listing process is complete.

4) ILL-OPERATOR XXX.

An illegal updating command has been detected. Processing continues and the illegal command it ignored.

5) IMP COND.

An improper condition has been detected on a magnetic tape unit. Correct the improper condition and start the computer. The magnetic tape function will be attempted again.

6) LIBNO XXX.

Library XXX is the one being updated or listed.

7) NO HEADER.

No legal header has been read. Correct the tape or card deck, place the corrected tape or card deck in the reader, and start over.

8) NO LIBNO.

The library to be updated or listed does not contain an identifying library number. Correct the tape or card deck, place the corrected tape or card deck in the reader, and start over.

9) READ XXXXXX NEXT.

The wrong source program tape or deck was mounted in the reader. Mount the correct program and start the computer.

10) TCS ERR Z TBL Y.

The table control system has detected an error while attempting to operate on the indicated table Y. The error is unrecoverable and is the result of internal trouble or a magnetic tape error (X = 9).

11) WRONG LIBNO XXX.

XXX is the library mounted on the tape transport but it is not the one requested for updating or listing. However, if the user starts the computer, library XXX will be updated or listed. If updating is requested, the new library assumes the library number which was input via the command card deck or paper tape.

12) XXXXXX NOT IN DIR.

Program XXXXXX is either not found in the directory or has been deleted by command.

SECTION IV-E. TRACE DEBUGGING PROGRAM (TRACK)

1. GENERAL INFORMATION

The trace debugging program (TRACK) traces the course of up to five designated
areas of a program being executed on the computer. TRACK inserts an IRJP to
itself in the first two instructions of each area and then exits to the start-
ing address of the program being traced. When the program re-enters TRACK at
the first area limit, the original first two instructions are restored, inter-
rupts are locked out, and the trace begins.

TRACK permits the computer to execute a monitored instruction; temporarily stops
the monitored program action; and then causes the instruction just executed,
together with the contents of pertinent registers, to be listed on the monitor-
ing typewriter or punched on paper tape. A version of TRACK is also available
for use with a high-speed printer. Besides displaying the traced instruction
in absolute (octal) coding, TRACK also prints the instruction with its TRIM
symbolic operator for easier identification by the user. TRACK also provides
the options of ignoring subroutines in the trace area or tracing only those
instructions which modify a specified address.

TRACK compares the address of each instruction it executes against the final
address given for the area it is processing. When a match is found, TRACK
exits to the traced program's next designated address.

Examination and analysis of the printed register contents permit the programmer
to follow the progress of the running program and to determine the location and
source of errors.

TRACK is available in relocatable (bioctal) format and requires approximately
2,340 (octal) storage locations in the same bank of memory.

Because every monitored instruction requires the execution of at least a
hundred TRACK instructions, and a relatively great amount of time is required
by the typewriter or punch, it is obvious that real-time problems cannot be
traced in the true sense with TRACK.

TRACK does not use or alter the interrupt registers, and it locks out all
interrupts while it is tracing a program area. However, this should not deter
users from tracing programs with interrupts, since the interrupt routine itself
may be specified as one of the areas to trace, or the I/O segments of the
program may be left out of the designated trace areas.

2. INPUT

The input to the TRACK program consists of instructions extracted from desig-
nated areas of the user's program. Input parameters are entered manually
by the program operator. These parameters define the first address of the
user's program, the desired output option, and the areas in the user's program
which are to be traced.

3. OUTPUT

The output of the TRACK program consists of the traced instructions and corresponding addresses together with the contents of pertinent registers. This output may be a typed or printed hard copy or a punched tape.

The TRACK output for each instruction consists of the information which follows. Then the information is typed or printed, it appears from left to right across the page in the order listed below.

1) Address of the executed instruction.
2) Executed instruction in absolute (octal) coding.
3) Executed instruction with operator in TRIM symbolic language.
4) Contents of upper half of accumulator register (AU).
5) Contents of lower half of accumulator register (AL).
6) Present index register being used and its contents (B).
7) Operand of executed instruction.

There are exceptions to this format. In cases 4), 5), and 6), the contents of the A and B registers are not printed unless a change was made to them; the identifiers, however, are always printed. In cases 6) and 7), index register and operand printouts are omitted for instructions where B registers are never referenced, and the operand is always a constant appearing in the instruction itself. Also, in case 7) if a fault condition arises (function codes of 00, 01, 77, 5000, or 5077), the word, fault, appears in the operand position.

4. USER GROUND RULES

The user must keep in mind that TRACK replaces the first two instructions of each designated area with entry instructions to TRACK before starting the monitored routine and, upon returning to TRACK, those original first two instructions are replaced. Caution must therefore be exercised in choosing the trace area initial address so that either one of the first two instructions is not modified by the running program before the program executes those instructions. Also, a trace output results only once for each entire specified area since the TRACK entry instructions are replaced by the original program instructions. If the trace of a repetitive loop is desired, the initial address to the area must be selected so that the area includes the entire loop process.

TRACK stops prior to executing each instruction if PROGRAM STOP 4 is set. During the stop, the next instruction of the traced program is displayed in AL and the address of this instruction is displayed in AU. Any desired changes may be made to them at this time. After restarting the computer, the trace continues. TRACK also stops if a fault instruction is traced or if a program stop in the traced program is fulfilled.

Although TRACK locks out all interrupts when initially entered from a program area and removes this lockout only when control is returned to the traced program upon completion of the upper limit instruction, TRACK executes remove interrupt lockout instructions. Care should be exercised by the user in this regard, expecially when it is desired to trace interrupt routines. If TRACK is interrupted while monitoring a program area, and in turn monitors the

interrupt subroutine, the flags and storage relating to the first area will be destroyed, resulting in an erroneous trace.

The addresses given to define a trace area for TRACK are not output area limits outside of which nothing is traced. The trace may proceed anywhere throughout computer memory, and TRACK relinquishes control only after the instruction at the higher of the two specified area addresses is executed. Therefore, the contents of all area limit addresses must be legitimate instructions.

5. OPERATING PROCEDURES

The computer and I/O console must be placed in the operational state with all switches in the normal operating position. If the TRACK program is to be operated with a high-speed printer for output, the printer must also be initialized. Prior to operating the TRACK program, both the user's program to be traced and the TRACK program must be loaded via a utility package. TRACK may be loaded at any address in memory with the single restriction that the entire program must be loaded within one memory bank. The operating procedures are given below.

1) Master clear the computer.

2) Set the P register to the starting address of TRACK.

3) Set the AU register to the starting address of the user's program.

4) Set the AL register for the desired output option:

   a) **AL set to zero** - Track traces all instructions starting with first instruction designated and ending with last instruction designated.

   b) **AL set negative** - TRACK traces all instructions except those executed between a return jump (direct or indirect) and the return from the jump; that is, subroutines are ignored. If the first instruction to be traced is an RJP or IRJP to a subroutine, that subroutine will be traced.

   c) **AL set positive** - TRACK considers the value in AL as an address and produces a trace output only for those instructions which alter the contents of that address.

5) Start the computer. The computer stops with the AU and AL registers cleared.

6) Set the addresses of the first and last instructions to be traced in AU and AL (either address in either register). The area defined by these addresses may be the entire program or any portion thereof. If only one instruction is to be traced, the address must be set in both AU and AL.

7) Start the computer. The computer stops with AU and AL cleared.

8) **Repeat steps** 6) and 7) for each desired trace area. A total of five areas may be designated.

9) If less than five areas have been entered, leave AU and AL clear and start the computer. The computer stops with AL set to a nonzero value.

10) Set the AU and AL registers to the initial parameters required by the user's program (if any are required).

11) Set **PROGRAM SKIP** key 0 if output is to be punched (no action is required for typewriter output or high-speed printer output).

12) Set **PROGRAM STOP** key 4 if it desired to stop the program prior to executing each traced instruction.

13) Start the computer. The user's program starts and runs normally until a trace area is reached. Then the TRACK program controls the execution of all traced instructions in the user's program. After executing each traced instruction, TRACK types or punches the trace information for that instruction.

14) If **PROGRAM STOP** key 4 is set, the computer stops prior to executing each instruction. While the computer is stopped, the operator may change the instruction which is displayed in the AL register. After changing the instruction, or if no change is desired, start the computer to continue the trace operation. If a stop condition in the traced program is fulfilled, TRACK allows the computer to stop. To continue the trace operation, start the computer.

15) After TRACK has traced the execution of all instructions in the first trace area, control is returned to the user's program until the next trace area is reached. When all designated areas have been traced, the user's program runs to completion and terminates in the normal manner.

16) If the **TRACK** output is on punched tape, punch a trailer on the tape, remove the tape from the punch, and obtain a typed hard copy of the punched tape via off-line operation of the console.

17) If the **TRACK** output is on the typewriter or printer, remove the hard copy from the typewriter or printer.

18) Examination and analysis of the hard copy will permit the programmer to follow the progress of the running program and to determine the location and source of errors in that program.

SECTION IV-F. CARD-TO-TAPE PROCESSOR (CART)

1. GENERAL

The punched card-to-tape processor, CART, is a computer stored program which
has the ability to stack ordered card decks on magnetic tape for subsequent
continuous processing. By means of appropriate control cards, CART directs the
stacking of cards on tape, with the following options:

1) Card-to-tape, a single job.
2) Card-to-tape, consecutive jobs.
3) Replace jobs on a previous card-to-tape.
4) Insert job on a previous card-to-tape.
5) Delete jobs on a previous card-to-tape.
6) Punch jobs from a previous card-to-tape.
7) List jobs from a previous card-to-tape.
8) Withdraw specified jobs from a previous card-to-tape.
9) Resequence jobs.
10) Correct jobs from a previous card-to-tape.

The CART program is designed for use with a minimum configuration of the
following UNIVAC equipment:

1) Computer with 16K memory.
2) Magnetic tape system with two transports.
3) I/O console with typewriter.
4) Card processor.

2. INPUT

Input to the CART program consists of two types of punched cards and, in some
cases, information read from magnetic tape. The two types of punched cards
are control cards and job cards. Control cards direct the operation of the
CART program and must conform to a specific format. Job cards contain in-
formation to be written on tape. This information may be in any 80-column
format and may be written as a separate job or merged with other jobs read
from magnetic tape. The general format for control cards is shown below.
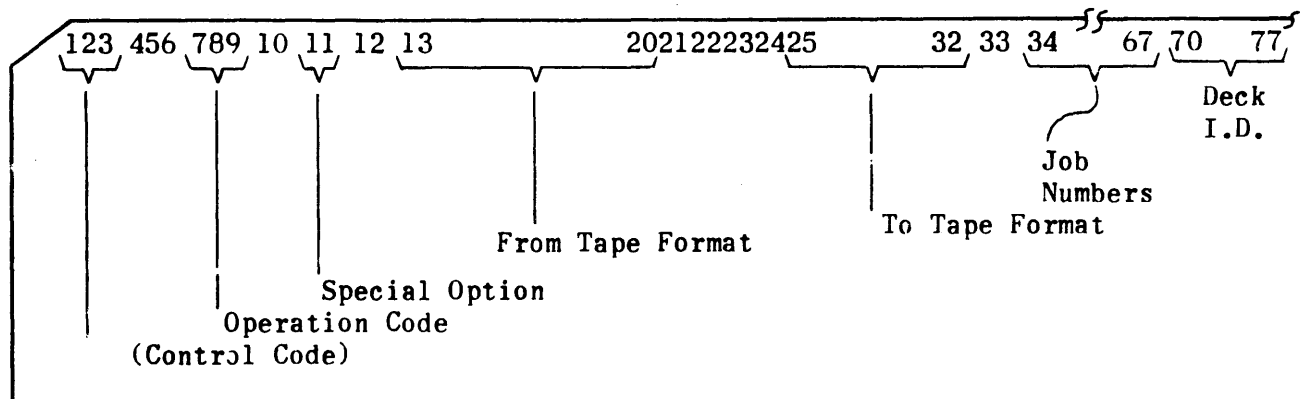
1) Control code CTT (columns 1, 2, and 3).

The control code CTT must appear in columns 1, 2, and 3 of all control cards except those with an operation code of ENJ, END, and DEL (cards with these operation codes must not have any code punched in columns 1, 2, and 3).

2) Operation code (columns 7, 8, and 9).

The code in columns 7, 8, and 9 defines the operation to be performed by the CART program. The list of legal operation codes given below describes the operations which can be performed by the CART program.

Operation Code	Definition
RPF	Replace the job(s).
DEF	Delete the job(s).
ADD	Add the job(s) to present library tape.
WDR	Withdraw the job(s) from current library tape.
LST	List the selected job(s) on the printer.
OLY	Correct the job(s) specified and Write on output tape.
ALL	Correct the job(s) and copy all jobs on output tape.
PUN	Punch the selected job(s).
ENJ	End of this job.
END	End of this CART run.
DEL	Delete the card(s).

3) Special options (column 11).

The presence of a code in column 11 permits simultaneous performance of two operations. One of two legal codes may be used. The codes are L and R and their meaning to the CART program is as follows:

    L – List the selected jobs.
        This option is allowed only on a control card with a WDR operation code. The L in column 11 directs CART to list the information withdrawn from tape while the withdraw operation is being performed.

R – Resequence the selected job, with the new deck identification contained in columns 70-77 on the CTT card.
If columns 70-77 are blank, the deck name portion of the deck ID is not changed, but the card numbers for each deck name are resequenced. A CTT card must have only one job number in the job number field when the resequence option is selected. The resequence option may be selected on control cards with operation codes at RPF, ADD, WDR, ALL, OLY, or PUN. If the resequence option is selected with RPF, ADD, or ALL operation codes, only the new job or the corrected job is resequenced.

4) From tape format (columns 13 through 20).

The information in columns 13 through 20 defines the format and address of information to be read from tape by the CART program. The codes required are as follows:

Card Column	Code and Meaning
13	H = 800 FPI (1540, 1219 mode only) M = 556 FPI L = 200 FPI
14	O = Octal B = Bioctal
15	S = Single Channel D = Dual Channel
16	O = Odd Parity E = Even Parity
17 & 18	Channel number of FROM tape
19	Cabinet number of FROM tape
20	Transport of FROM tape

NOTE: If no information is to be read from tape, the codes described above apply to the To tape; that is, the tape on which information is to be written.

5) To tape format (columns 25 through 32).

The information in columns 25 through 32 defines the format and address of the information to be written on tape by the CART program. If the control card contains either an LST or PUN operation code these columns must be left blank. If the control card contains an ADD operation code and no information is to be read from tape, the To tape information is coded in columns 13 through 20 and this portion of the card is left blank.

The codes required are as follows:

Card Column	Code and Meaning
25	H = 800 FPI (1540, 1219 mode only) M = 556 FPI L = 200 FPI
26	O = Ocatl B = Bioctal
27	S = Single Channel D = Dual Channel
28	O = Odd Parity E = Even Parity
29 & 30	Channel Number of To tape
31	Cabinet Number of To tape
32	Transport of To tape

6) Job numbers (columns 34 through 67).

The information in columns 34 through 67 defines the jobs to be processed. Job numbers must be octal numbers and must not be longer than two digits.. A comma, dash, or period must follow each job number specified. A comma serves to separate two job numbers. A dash between two job numbers indicates that all jobs from the first number through the second number are to be processed. A period must follow the last job number in the job-number field. Job numbers must be specified in numerical order for all operation codes except LST, WDR, and PUN. An example of a typical job number field is as follows:

Example: 3, 6-10, 12, 14

This field indicates that jobs 3, 6, 7, 10, 12, and 14 are to be processed.

7) Deck I.D. (columns 70 through 77).

When the resequence option (R) is coded in column 11, the deck identification name may be contained in columns 70 through 77.

3. OPERATIONS

The CART program reads a control card and performs the operation specified by the operation code. Job cards which may be required for the operation must follow the control card in the input deck. If a tape previously written by CART is to be revised, it must be defined in the From Tape field of the control card. The card sequences for the various operations of the CART program are defined with examples in the following subparagraphs.

3.1 CARD-TO-TAPING A SINGLE JOB

For a single job the card sequence is a CCT control card with an ADD operation code and blank To tape and job number fields, a job card deck, and ENJ card, and an END card. This sample deck also resequences the job as it is being written on tape. The first card I.D. is NUID0001.

3.2 CARD-TO-TAPING CONSECUTIVE JOBS

This sample deck illustrates the card sequence for writing two jobs on tape.

3.3 REPLACING JOBS

This sample deck replaces jobs 1, 3, 4, and 5 and yields a new tape with a
new format. The input tape is written in single channel, medium density,
odd parity, and bioctal character coding. The output tape is written in
single channel, low density, odd parity, and octal character coding. The input
tape is on transport 2; the output tape is on transport 3.

```
        END
              INPUT CARDS TO REPLACE JOB5 ON TAPE
        END
              INPUT CARDS TO REPLACE JOB4 ON TAPE
        END
              INPUT CARDS TO REPLACE JOB3 ON TAPE
        END
              INPUT CARDS TO REPLACE JOB1 ON TAPE
 CTT   RPF   MBS00512    LOS00513 1,3-5.
```

3.4 INSERTING JOBS

In the example below, the input cards become the new job 6 on the output
tape. The inserted job is resequenced as each new deck name is encountered
since there is no deck I.D. specified in columns 70-77. When inserting a
job, CART always inserts the input cards ahead of the job number specified.

```
    ENJ
            RESEQUENCED BUT DECK NAMES WILL NOT BE CHANGED.
            INPUT CARDS TO BE INSERTED AHEAD OF JOB6. JOB6 WILL BE
CTT    ADD R MODOO511    MODOO514 6.
```

3.5 DELETING JOBS

The sample CTT card below directs CART to delete jobs 5, 7, 10, and 12.
The new tape is on transport 3.

```
CTT    DEF    LOSOO514    LOSOO513 5,7-12.
```

3.6 PUNCHING JOBS

The sample CTT card below directs **CART** to punch cards for job 5. The new card deck is resequenced beginning with a deck I.D. of CARD0001. Resequencing is done in conjunction with other card-to-tape functions. A new CTT card is required for each job unless an old sequence number contained on tape is to be used. The new sequence number is entered in columns 70-77 of the CTT card.

```
CTT    PUN R LOS00513              5.                              CARD0001
```

3.7 LISTING JOBS

The sample CTT card below directs **CART** to list jobs 1, 3, and 5.

```
CTT    LST   MP000512             1,3,5.
```

3.8 WITHDRAWING JOBS

This operation gives the user the capability to build a library by extracting only those specified jobs from previously written tape and placing them on a new tape. A CTT card is required each time a different input tape is referenced.

The sample CTT card below directs **CART** to write jobs 5, 6, and 2 on the output tape, in that order. The L in column 11 directs the printer to list the jobs that are being withdrawn.

```
CTT   WDR L MBS00513    MBS00511 5,6,2.
```

3.9 CORRECTING JOBS

The correct option gives the user the capability to update current library tapes by deleting, inserting, and replacing cards.

A listing accompanies the correction operations. It consists of the corrected statements and the statement that follows any correction. When a delete is performed, the first statement that is deleted, DELETE, and the first statement after the deleted statements are printed. When a replace is performed, the new statement and the following statement are printed. When an insert is performed, the inserted cards and one statement after the new cards are printed.

The CTT card must include either the ALL or the OLY operator. ALL directs the CART program to copy all jobs from an input tape while correcting the jobs specified in the job number field of the CTT card. OLY directs the program to copy only the jobs that it is correcting.

Corrections are made to a job according to the deck I.D. specified on each correction card. These cards must be in the same order as they are on the tape. An ENJ card must follow the correction cards for each job. Correction cards for insertion or replacement are simply the new source language cards. The new source cards replace incorrect cards and insert new cards according to the deck I.D. in columns 1-10. Spaces are not equated with zeros by CART. Thus, the user must be certain that the deck I.D. on his correction card has spaces and zeros that exactly correspond to those on his original card and written on the tape. The correction card for deleting cards must include DEL in columns 7-9, the deck I.D. of the first card to be deleted in columns 13-22, left justified and the number of cards to be deleted in columns 25-27, right justified. If a deck name appears in two or more separate segments of a job, any corrections are made to the segment that appears first, unless a correction card(s) for a segment with a different deck name separates the correction cards for the segments with identical deck names. Two examples of the organization for correction card decks are shown below.

1) ALL operator.

   The sample correction deck on the top of page IV-F-11 directs CART to:

   a) Correct job 1 with a replacement and an insert.

   b) Copy job.

   c) Correct job 3 by deleting the card with a deck I.D. of TEST0008 and the following 3 cards.

   d) Copy the rest of the input tape.

2) OLY operator.

This sample correction deck directs CART to correct job 4 with a
replacement and correct job 5 by deleting 3 cards and inserting 1 card.
Only jobs 4 and 5 are written on the output tape.

4. OPERATING PROCEDURES

The procedures below assume that all equipment to be used has been placed in the operational state with all switches in the normal operating position. The CART program must be loaded into memory starting at address 01000. The procedures required to operate the CART program are given below.

1) Master clear the computer.

2) Mount tapes as follows (all tapes must be at load point):

   a) Card-to-taping single or consecutive jobs.
   Mount a scratch tape on the transport selected as the From tape on the CTT card and set the write enable button.

   b) LST and PUN operations.
   Mount the input tape on the transport selected as the From tape on the CTT card.

   c) Other operations.
   Mount the input tape on the specified From tape transport and a scratch tape on the tape transport selected as the To tape on the CTT card and set the write enable button.

3) Place the input card deck in the card reader input hopper.

4) Place two blank cards at the end of the input card deck and initialize the card reader.

5) Set the P register to 01000.

6) Start the computer. The CART program provides informative typeouts on the on-line typewriter while performing the requested operations. Error conditions are also indicated by typeouts. Observe the typeouts as they occur and refer to paragraph 5.

The CART program continuously performs the operations specified on CTT cards. In order to stop the program, the user must place an end card (in addition to any that are necessary for an option) in the input card deck. The CART program resumes operation upon restarting the computer. When the CART program is stopped in order for the user to change tapes or change the address of tape units, he must reinitialize the program. This is done by rewinding all user tapes, setting P = 01000, and starting the computer.

When CART writes on a tape, the tape is positioned before the end of job sentinel upon completion of an operation. This position gives the user the capability to merge tapes by using the WDR operator.

NOTE: If the punch is used for output, perform the procedure necessary to remove the last card punched from the punch.

5. INFORMATIVE AND ERROR TYPEOUTS

   1) SET SINGLE DUAL SWITCH TO --- FOR CHANNEL ---.

      This typeout always occurs to remind the user of the mode and channel of the tape unit that he has specified on the CTT card.

   2) END OF INPUT TAPE.

      This typeout indicates that the CART program has detected an end of tape sentinel block.

   3) Typeouts will indicate the operation that is currently being performed and the job numbers as the jobs are being processed.

      a) CORRECT AND COPY ONLY.
          CORRECT JOB NUMBER.
          JN (job number)
          JN

      b) CORRECT AND COPY ALL.
          CORRECT JOB NUMBER.
          JN
          JN

      c) REPLACE JOB NUMBER.
          JN
          JN

      d) ADD JOB NUMBER.
          In card-to-taping jobs, no job numbers are typed out. In inserting jobs, the job numbers are typed for the jobs from the input tape that are placed before the new jobs.

      e) DELETE JOB NUMBER.
          JN
          JN

      f) LIST JOB NUMBER.
          JN
          JN

      g) WITHDRAW JOB NUMBER.
          JN
          JN

      h) PUNCH JOB NUMBER.
          JN
          JN

4) ILLEGAL OPERATOR.

The operation code is either incorrect or not placed in the proper columns of the CTT card. When the condition is corrected, start the computer.

5) ILLEGAL JOB NUMBER.

A job number is greater than two digits, job numbers are not properly separated by punctuation, or the first job number does not begin in column 34 of the CTT card.

6) ILLEGAL TAPE FORMAT.

The components of the From and/or the To tape format(s) are improperly represented by the letter codes or the letter codes are placed incorrectly in the columns of the CTT card.

7) INCORRECT CTT CARD.

This typeout occurs after ILLEGAL JOB NUMBER, ILLEGAL OPERATOR, and ILLEGAL TAPE FORMAT. It also occurs when the job number field of a CTT card is not terminated by a period or when any information on a CTT card is placed in an incorrect column. When the condition is corrected, initialize the card reader and start the computer.

8) BAD INPUT TAPE.
NO RECOVERY.

This typeout indicates that the input tape is in error, and a new input tape is required. If any manual manipulation of tape units is necessary, the CART program must be reinitiated. Otherwise, restart the computer.

9) MTU ERROR.
IMPROPER CONDITION.

Correct an improper condition on the tape transports that are designated in the From and To tape fields on the CTT card. If the position of any tapes being used is changed manually, the CART program must be reinitiated. Otherwise, restart the computer.

10) MTU ERROR.
INPUT TIMING ERROR.
NO RECOVERY.

The requirements for correction of this condition are the same as specified for BAD INPUT TAPE.

11) MTU ERROR.
OUTPUT TIMING ERROR.
NO RECOVERY.

This typeout indicates a problem within CART or incorrect hardware operation. The input tape is not in error.

V. PROGRAMMER SERVICE SUBROUTINES

Programmer service subroutines are those subroutines which the programmer assembles with his routine to perform a subordinate function. These subroutines exist in source assembler language for easy integration into user's programs.

Programmer service subroutines include mathematical subroutines, conversion subroutines, and assembler support subroutines.

SECTION V-A. MATHEMATICAL SUBROUTINES

1. FIXED POINT SQUARE ROOT (SQR)

GENERAL DESCRIPTION: The fixed point square root subroutine computes the square root of a positive number, N, where N may be zero or $.000001_8 \leq N \leq 177777_8$.

The input N is reduced or raised to a number $X \cdot 2^f$ where $2 \leq X < 4$. Within these limits, the subtraction of the hyperbola $C/X+D$ from the straight line $A(X+D)+B$ yields a very close approximation to the curve $Y = \sqrt{X}/\sqrt{2}$. Therefore $\sqrt{N}$ is computed:

$$\sqrt{N} \cong I = Y = \sqrt{2}\ (2)^{f/2}\ \text{where}$$

$$Y = A(X+D)+B-C/(X+D)$$

If the shifting to reduce or raise N to X was odd, then

$$I = Y\ (2)^{f/2} = Y\ \sqrt{2}\ (2)^{(f-1)/2}$$

If the shifting was even, then $I = Y\ \sqrt{2}\ (2)^{f/2}$

One pass through Newton's formula, $\sqrt{N} = 1/2\ \left[(N/I) + I\right]$, yields $\sqrt{N}$ accurate from 13 to 16 binary bits.

MEMORY USED: 150_8

APPROXIMATE RUN TIME: 220-290 microseconds.

INPUT PARAMETERS: Positive number (N) scaled 2^{18} in A (integral bits in AU, fractional bits in AL). The first 16 significant bits only are used.

OUTPUT PARAMETERS: $\sqrt{N}$ scaled 2^{18} in A. There are 16 significant bits maximum.

CONSTANTS USED: $A = 0.104327 = 065325_8$ scaled 2^{18}

$B = 1.19815 = 046256_8$ scaled 2^{14}

$C = 3.38816 = 154330_8$ scaled 2^{14}

$D = 2.82785 = 132374_8$ scaled 2^{14}

$\sqrt{2} = 132405_8$ scaled 2^{15}

2. FIXED POINT SINE AND COSINE (SINCOS)

GENERAL DESCRIPTION: This subroutine computes sine and cosine of an angle X in BAMS (binary angular measurement), where $0^\circ \leq X < 360^\circ$. The first three significant bits are stored to determine the quadrant that X lies in. The remainder of X, which is less than 45°, is further divided. The first seven bits are stored in an index register for referencing a SINE table and a

COSINE table, each 200_8 long with function values for the range 0^0-45^0; the last seven bits are converted to radians, and since they represent an angular measurement of only 0^0-$0^020'56"$, the SINE of this small angle (SIN B) equals the angle in radians (SIN B=B); the COSINE may be represented by the formula: COS B = 1 - B^2/2.

Substituting in the formulas:

  SIN(A+B) = SIN A COS B + SIN B COS A
  COS(A+B) = COS A COS B - SIN A SIN B

We obtain:

  SIN(A+B) = SIN A(1-B^2/2) + B COS A
  COS(A+B) = COS A(1-B^2/2) - B SIN A

Where SIN A and COS A are obtained by table-lookup. The original three significant bits are referenced for the correct quadrant, and the sine and cosine values just computed are or are not interchanged, and they are or are not complemented (negated), accordingly, for SIN X and COS X.

MEMORY USED: 545_8 (400_8 Table)

APPROXIMATE RUN TIME: 251-349 microseconds.

INPUT PARAMETERS: An angle (X) in BAMS scaled 2^{17}, less than 360^0 in AU.

OUTPUT PARAMETERS: SIN X in AU, COS X in AL, both scaled 2^{16}.

REMARKS: Tested accuracy over 360^0 range = $\pm.000014_8$ (14-bit accuracy)

3. FIXED POINT ARCTANGENT (RTAN)

GENERAL DESCRIPTION: This subroutine computes the angle α whose tangent is Y over X, where $0^0 \leq \alpha < 360^0$. Y and X are made absolute and the smaller of $[Y/X]$ -- $[X/Y]$ (=Z=TANθ) is computed. The most significant six bits of this tangent value (=A) are stored in an index register for referencing an angle table which is expressed in BAMS (binary angular measurement) in 64 non-linear increments of 54 to 27 minutes from 0^0 to 45^0. The absolute angle θ is then computed using the formula:

$$\theta = TAN^{-1} A + 1/2 \pi (\frac{Z-A}{1+AZ})$$

If Y was greater than X, then:

  α = 90^0 - θ for first quadrant
  α = 90^0 + θ for second quadrant
  α = 270^0 - θ for third quadrant
  α = 270^0 + θ for fourth quadrant

If X was greater than or equal to Y, then:

$\alpha = \theta$ for first quadrant
$\alpha = 180^0 - \theta$ for second quadrant
$\alpha = 180^0 + \theta$ for third quadrant
$\alpha = 360^0 - \theta$ for fourth quadrant

The original sign values of Y and X determine the quadrant.

MEMORY USED: 226_8

APPROXIMATE RUN TIME: 235 microseconds, average.

INPUT PARAMETERS: Y in AU, X in AL, both to the same scale, positive or negative.

OUTPUT PARAMETERS: An angle (α) in BAMS scaled 2^{17} in AL.

REMARKS: Tested accuracy over all quadrants is to $+10''$, $-20''$.

4. FIXED POINT ARCSINE (ARCSIN)

GENERAL DESCRIPTION: This subroutine converts the sine of an angle into its corresponding angle in the range -90^0 to 90^0. The input (18-bit word) is the sine of the angle expressed in octal digits and scaled 2^{16}. The output (18-bit word) gives the corresponding angle in BAMS and is scaled 2^{18}.

METHOD: The approach taken here utilizes the results (decimal digits) obtained by Kogbetliantz [1, 2] :

(1) $\text{Arcsin } X = \pm |X| \left\{ A_0 + \dfrac{a_1}{b_1 - |X|^2} + \dfrac{a_2}{b_2 - |X|^2} \right\}$ for $0 \le |X| \le 1/2$

where: $A_0 = 0.5249978317$ $\qquad\qquad$ $b_1 = 1.270451499$

$\qquad\quad a_1 = 0.09425018578$ $\qquad\qquad$ $b_2 = 3.702672882$

$\qquad\quad a_2 = 1.484093006$

(2) $\text{Arcsin } X = \pm \left\{ \dfrac{\pi}{4} + 1/2 \text{ arcsin } (2 |X|^2 - 1) \right\}$ for $\dfrac{1}{\sqrt{2}} \le X \le \sin \dfrac{(3\pi)}{8}$

where arcsin $(2 |X|^2 - 1)$ is then evaluated by using equation (1) above.

since $0 \le 2 |X|^2 - 1 \le \dfrac{1}{\sqrt{2}}$.

(3) $\text{Arcsin } X = \pm \left\{ \dfrac{\pi}{2} - 2 (12.73421816 - |X|) \cdot 2^{-h} R_2 \right\}$

$\qquad$ for $\sin \dfrac{(3\pi)}{8} \le |X| \le 1$ $\qquad$ where h and R_2 are defined by

(3a) $E^2 \cdot (1 - |X|^2)/2 = 2^{-2h} \cdot f$ with $.25 \leq f < 1, \ h \geq 5$

(3b) $R_2 = 1/2 \ (R_1 + f/R_1)$ and $R_1 = C_1 \left(C_2 + f - \dfrac{C_3}{f + C_4} \right)$

where $C_1 = .33431261$ $C_4 = .53164106$

       $C_2 = 2.76913454$ $E = 0.0852176716$

       $C_3 = 1.19031245$

NOTE: In equations (1), (2), and (3), the sign is $\pm$ if $0 \leq X \leq 1$, $-$ if $-1 \leq X \leq 0$

INPUT PARAMETER: The sine of an angle in octal digits scaled 2^{16} is entered in AL.

OUTPUT PARAMETER: The corresponding angle in BAMS scaled 2^{18} in AU. If the computer stops with 377777 in AU, this indicates that the input sine is greater than 200000 (1 scaled 2^{16}) or less than -200000 (-1 scaled 2^{16}) and is therefore illegal.

COUPLING PROCEDURE:

  ⟶   ENTAL $\cdot$ [octal value of the sine scaled 2^{16}]

  ⟶   IRJP $\bullet$ ARCSIN

  ⟶   [abnormal exit]

  ⟶   [make use of the output value; that is, arcsine, in BAMS scaled 2^{18} in AU]

COMPUTATION TIME:

for $0 \leq |\text{ARCSIN } X| < \pi/4$,	208 1/3 - 244 1/3 microseconds
for $\pi/4 \leq |\text{ARCSIN } X| < 3\pi/8$,	249 2/3 - 297 2/3 microseconds
for $3\pi/8 \leq |\text{ARCSIN } X| \leq \pi/2$,	301 2/3 - 348 1/3 microseconds

MEMORY USED: 206_8

ACCURACY: The absolute error is less than or equal to $.000004_8$ BAMS. This corresponds to an accuracy of five octal digits.

REMARKS: To use this subroutine as an independent program, change the instructions at addresses "ARCSIN" and "FIRST-1" to "STOP" and "JP $\bullet$ ARCSIN", respectively.

REFERENCES:

1) A. Ralston and H. S. Wilf, Mathematical Methods for Digital Computers, John Wiley and Sons, Inc., New York, 1960, pp. 31-33.

2) E.G. Kogbetliantz, Computation of Arcsine N for $0 < N < 1$ Using an Electronic Computer, I.B.M. Research and Devel., Vol. 2, No. 3, July, 1958, pp. 218-222.

5. FIXED POINT NATURAL LOGARITHM (NATLOG)

GENERAL DESCRIPTION: This subroutine computes the natural logarithm of any number N where $377777_8 \times 2^{-0377757_8} \leq N \leq 377777_8 \times 2^{0377757_8}$. The number N should be expressed in the input format (1) $N = N_1 \times 2^{N_2} = N_1 \times 2^{-S}$ where $1 \leq N_1 \leq 377777_8$ and S = scaling factor of $N = -N_2$

Both N_1 and S should be given in octal notation. N_1 is entered in AU and S is entered in AL. The output ln N appears in A in octal notation and is scaled 2^{2010}.

METHOD: Using Maehly's method [1] in the reduced range $2^{-1/2} < F^* < 2^{1/2}$ where $N = 2^M \cdot F$, $1 < F < 2$ and $F^* = F/\sqrt{2}$, six correct significant places may be obtained with the following approximation:

$$(2) \quad \ln N = \log_e N \approx (M+1/2) \ln 2 + Q_3(F).$$

Here (3) $\quad Q_3(F) = P_0 + \sum_{k=1}^{3} \frac{-P_k}{|F+T_k|}$

with the coefficients:

$P_0 = 3.681656603_{10}$

$P_1 = 34.410692910_{10}$

$P_2 = 8.126503834_{10}$

$P_3 = 0.2665195666_{10}$

$T_1 = 10.379672140_{10}$

$T_2 = 2.051212813_{10}$

$T_3 = 0.4249952497_{10}$

INPUT PARAMETER: With N expressed in the proper input format.

$$N = N_1 \cdot 2^{N2} = N_1 \cdot 2^{-S}$$

where $1 \leq N_1 \leq 377777_8$

and S = scaling factor of $N = -N_2$

$(N_1$ and S are in octal notation), N_1 is entered in AU and S in AL.

Thus consider the number:

$$N = 100.146_8 = 100146_8 \times 2^{-9}$$

Input format of N: $N = 100146_8 \times 2^{-11}{}_8$

$N_1 = 100146_8$ is entered in AU and $S = -N_2 = 11_8$ is entered in AL.

An equally valid input format for $N = 100.146_8$ is $N = 200314_8 \cdot 2^{-12}{}_8$, with $N_1 = 200314_8$ and $S = 12_8$

OUTPUT PARAMETER: The natural logarithm of N is given in A in octal notation and scaled $2^{20}10$.

EXECUTION SEQUENCE:

⟶ ENTAU • $\left[\text{octal value of } N_1\right]$

⟶ ENTAL • $\left[\text{octal value of scaling } S\right]$

⟶ IRJP • NATLOG´

⟶ $\left[\text{use output value found in A}\right]$

RESTRICTIONS: The starting address of this subroutine, which corresponds to the label "NATLOG", must be even-numbered.

MEMORY USED: 76_8

COMPUTATION TIME: 232 2/3 - 243 2/3 microseconds.

ACCURACY: In various trial runs the absolute errors obtained were less than 4×2^{-14}. This assures at least five correct places in the output value scaled 2^{20}.

REMARKS: To use this subroutine as an independent program, change the instructions at addresses NATLOG and EXIT to STOP and JP NATLOG, respectively

REFERENCE: A.Ralston and H.S. Wilf, Mathematical Methods for Digital Computers, John Wiley & Sons, Inc., New York, 1962, pp. 28-30.

6. FIXED POINT EXPONENTIAL (EXPON)

GENERAL DESCRIPTION: This subroutine computes e^N for $-37.7777_8 \leq N \leq 37.7777_8$. The input (18-bit word) is the octal value of N scaled 2^{12} in AL. The output consists of the six most significant octal digits of e^N in AL and its scaling in AU.

METHOD: A rational approximation to e^N adapted for use on computers [1] is capable of attaining high accuracy in a relatively short time. The range of N is reduced by multiplying the binary representation of N by $\log_2 e$.

Denoting the integral part of the product by M and the fractional part by F, we have

(1) $e^N = e^{(M+F)/\log_2 e} = 2^M \cdot e^{F \cdot \ln 2}$ where $0 < F < 1$

Further reduction of the range $(0, \ln 2)$ of $F \cdot \ln 2$ leads to

(2) $e^N = 2^M \cdot 2^{A(K)} \cdot e^{P_K \cdot \ln 2} = 2^M \cdot 2^{A(K)} \cdot e^Z$ where $P_K = F - A(K)$

$$A(K) = \sum_{i=0}^{K-1} S_i/2^{i+1}$$

S_i = sign of P_K; $S_0 = 1$ since $P_0 = F > 0$

$Z = P_K \cdot \ln 2$

Finally, e^Z is approximated by the rational function $P_m(Z)/P_m(-Z)$

where $(2m)! \; P_m(Z) = M! \sum_{j=0}^{M} (2m-j)! \; Z^j / \left[j! \; (m-j)! \right]$

The degree of accuracy of the approximations varies with K and M. Thus, for K = 2, m = 2, the theory predicts an accuracy of six places.

For K = 2, m = 2, the rational approximation becomes

(3) $e^N \approx 2^M \left[2^{A(2)} + \dfrac{A_1 \cdot P}{(P-A_2) \; P + A_3} \right]$

where $P = P_2 = F - A(2)$

$$A(2) = \begin{cases} 1/4 & \text{if } 0 < F < 1/2 \\ 3/4 & \text{if } 1/2 \leq F \leq 1 \end{cases}$$

$$A_1 = \begin{cases} 20.58795\ 84469_{10} & \text{if } 0 < F < 1/2 \\ 29.11577\ 00557_{10} & \text{if } 1/2 \leq F \leq 1 \end{cases}$$

$A_2 = 8.65617\ 02340_{10}$

$A_3 = 24.97642\ 77120_{10}$

INPUT PARAMETER: The octal representation of N, scaled 2^{12}, is entered in AL.

OUTPUT PARAMETER: The six most significant octal digits of e^N appear in AL and its scaling factor appears in AU. Thus

$\left. \begin{array}{l} \text{AU} = 17_8 \\[1em] \text{AL} = 104102_8 \end{array} \right\}$ means $e^N = 104102_8 \times 2^{-15} = 1.04102_8$

and $\left.\begin{array}{l} AU = -14_8 \\ AL = 125062_8 \end{array}\right\}$ means $e^N = 125062_8 \times 2^{12} = 1250620000_8$

COUPLING SEQUENCE:

$\longrightarrow$ ENTAL • [octal value of N, scaled 12]

$\longrightarrow$ IRJP • EXPON

$\longrightarrow$ [Make use of output value in A]

MEMORY USED: 72_8 (50_8 instructions and 22_8 constants)

COMPUTATION TIME: 192 2/3 - 228 2/3 microseconds

ACCURACY: The absolute error is less than three bits. This corresponds to an accuracy of five significant octal places.

REMARKS: To use this subroutine as an independent program, change the instructions at addresses EXPON and EXIT to STOP and JP EXPON, respectively.

REFERENCE: A. Ralston and H. S. Wilf, Mathematical Methods for Digital Computers, John Wiley and Sons, Inc. New York, 1960, pp. 19-22.

7. FLOATING POINT ARITHMETIC PACKAGE

GENERAL DESCRIPTION: This program provides the basic floating point arithmetic operations for addition, multiplication and division. This package is normally used where floating capability and compact data structure are desired and where medium speed arithmetic operations are tolerable.

NUMBER REPRESENTATION: The floating point format utilized is described below with each operand occupying 36 bits or two computer words.

The floating point word structure is as follows:

S	C	M
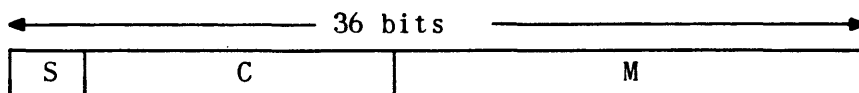
$\longleftarrow$ 36 bits $\longrightarrow$

where S (1 bit) indicates the sign
 C (8 bits) is the biased characteristic
 M (27 bits) is the mantissa.

The number notation used is ones complement (that is, to obtain the algebraic complement of a floating point value, all 36 bits are complemented rather than the sign bit, S, alone.

PARAMETER RANGE: A floating point number N must satisfy one of the following conditions:

a) N = 0

b) $2^{-129} \leq |N| < 2^{127}$

If $N = 0$, then $S = C = M = 0$. If $N \neq 0$, then N is represented by the expression $M \cdot 2^X$, where (by definition) $1/2 \leq M \leq 1$ and $X = C-128$. The binary point for M is assumed between C and M; therefore, given any $N \neq 0$, the most significant bit of M must contain a one; this condition is referred to as N being normalized (or M containing the maximum number of significant bits). The characteristic, C, represents the true binary characteristic of the number N biased by 128 $(X+128)$.

OPERATIONS AVAILABLE:

Operation	Algorithm	Execution Time (1219)
ADD	$(A) + (B) \longrightarrow A$	220-235 microseconds
MULTIPLY	$(A) \cdot (B) \longrightarrow A$	310-315 microseconds
DIVIDE	$(A) \div (B) \longrightarrow A$	272-321 microseconds

INPUT PARAMETERS: Two normalized floating point input parameters are required for each operation, operand A and operand B.

    A. Operand A is located in the A (AU, AL) register, with the most significant half in AU.

    B. Operand B is stored in memory with its address in the currently selected (ICR) B register. The least significant half of operand B is stored in address (B) with the most significant half in (B) +1.

OUTPUT PARAMETER: One output parameter is computed for each operation. The normalized floating point Result is located in the A register.

STORAGE REQUIRED: This package occupies a total of 324_8 cells, including temporary storage.

ERRORS: An error occurs:

1) When division by zero or an unnormalized number is attempted.

2) When a characteristic overflow occurs denoting a result where $|N| \geq 2^{127}$. (If characteristic underflow occurs; that is, $|N| < 2^{-129}$ no error will occur and the result will be +0.

ALGEBRAIC FORMULAS: The following formulas are used to maintain 1 1/2 precision (27 bits) required by the 1-8-27 floating point format:

$$A = A_2 + A_1$$

$$B = B_2 + B_1$$

$$A \cdot B \simeq A_2 B_2 + A_2 B_1 + B_2 A_1$$

$$A \div B \simeq \frac{A}{B_2} \left[1 - \frac{B_1}{B_2} \right]$$

PROGRAMMING CONSIDERATIONS:

In order to execute a floating point operation, the programmer must perform must perform the following steps:

1) Select a B register and load it with the address of the B operand

2) Load the A register with the A operand

3) Perform a direct or indirect return jump to the add (FADD), multiply (FMUL), or divide (FDIV) routine. The return jump instruction must be followed by an error exit and a normal exit in that order.

Upon completion of a normal operation, control is transferred to the return jump instruction +2 with the normalized packaged result in the A register. The original contents of SR, ICR and B0-B7 remain unaltered. An error exit will restore control to the return jump instruction +1.

REMARKS: The floating point package is structured so that the entrance cell for add is the first cell of the package; multiply entrance address is ADD +66; and divide entrance address is ADD +147 (octal).

This package does not provide subtract. However, the programmer obviously need only complement the appropriate operand and perform an add.

The package can be located anywhere in core as long as it is contained wholly within one bank, and the base address is at an even core location.

8. FLOATING POINT TO FIXED POINT CONVERSION

GENERAL DESCRIPTION: This subroutine converts a floating point number to a scaled, fixed point number.

INPUT: Floating point number in register A, with the most significant half in AU and the least significant half in AL.

OUTPUT: Fixed point number in A, its scaling in the current B register. A scaling of zero means the binary point is at the right end of register A. A positive scaling indicates that the point is to the left and a negative scaling that the point is to the right.

METHOD: The 9 bits of the characteristic are masked off and the 27 bit mantissa retained as the fixed point number. The characteristic is used to evaluate the scaling.

CALLING SEQUENCE:

Address	Mnemonic		Operand
L	ENTAL	.	(N least)
L+1	ENATU	.	(N most)
L+2	IRJP	.	(FLTFIX)
L+3	( )	.	(Return)

RESTRICTIONS: Must be wholly contained within one bank.

9. FIXED POINT TO FLOATING POINT CONVERSION

GENERAL DESCRIPTION: This subroutine converts a scaled fixed point number, N, to a floating point. The scaling reference point is the right end of register A; positive to the left and negative to the right.

INPUT: Fixed point number in register A; scaling of number in current B register.

OUTPUT: Floating point number in A; with the most significant half in AU and the least significant half in AL.

CALLING SEQUENCE:

Address	Mnemonic		Operand
L	ENTAL	.	(N least)
L+1	ENTAU	.	(N most)
L+2	ENTBK	.	(scaling)
L+3	IRJP	.	(FIXFLT)
L+4	( )	.	(Return)

NOTE: If N = $\pm$ 0, the resultant floating point number will equal all 0's.

RESTRICTIONS: All of the routine must be in one bank. The scaling S must satisfy the inequality:

$$0 \leq 243 - S - X \leq 377 \quad \text{(all octal)}$$

where X is the number of shifts required to make the contents of bit positions 34 and 35 unequal.

10. FLOATING POINT COMPARE

GENERAL DESCRIPTION: This subroutine compares algebraically two floating point numbers.

INPUT: Floating point number, A, in register A, with the most significant half in AU and the least significant half in AL. The floating point number, B, must be in core locations, Y, and Y + 1, with the most significant half in Y + 1 and the least significant half in Y.
The current B register must contain the address, Y.

OUTPUT: The routine returns control to one of three locations depending on the outcome of the compare.

CALLING SEQUENCE:

Address	Mnemonic	Operand
L	IRJP	. (CMAFLT)
L+1	JPB	. LOK
L+2	(JP)	. (Return here if $A > B$
L+3	(JP)	. $A = B$
L+4	(JP)	. $A < B$)

RESTRICTIONS: The routine must be entirely contained within the same bank.

11. FLOATING POINT SQUARE ROOT

GENERAL DESCRIPTION: This subroutine computes the square root of a normalized floating point number, N. $N = 0$ or $0.14693679 . 10^{-38} \le N \le 0.17014118 . 10^{39}$.

METHOD: N is scaled so that $N = X \cdot 2^S$, $2 \le X < 4$. Since the subtraction of the hyperbola $C/(X+D)$ from the straight line $A(X+D) + B$ yields a good approximation to $\sqrt{X} / \sqrt{2}$, a close guess to the $\sqrt{N}$ can be found:

$$A(X+D)+B-C/(C+D) \cong Y = \sqrt{X} / \sqrt{2}$$

$$\sqrt{N} \cong Y \cdot 2^{S/2} \quad \text{(S is odd)}$$

$$\sqrt{N} \cong Y \cdot \sqrt{2} \cdot 2^{S/2} \quad \text{(S is even)}$$

Letting I equal this guess, one pass through Newton's Formula,

$$\sqrt{N} \cong 1/2 \left(\frac{N}{I} + I \right)$$

yields the $\sqrt{N}$ to at least 26 binary bits.

INPUT: N must be in register A with the most significant half in AU and the least significant half in AL.

OUTPUT: $\sqrt{N}$ in register A with the most significant half in AU and the least significant half in AL.

CALLING SEQUENCE:

Address	Mnemonic		Tag
L	ENTAU	.	N most
L+1	ENTAL	.	N least
L+2	IRJP	.	(FSQRT)
L+3	JPAUNG	.	(ERROR CHECK)

RESTRICTIONS: Location FSQRT must be at an even core storage address. The routine may be in any bank, but it must be entirely contained within that bank. The routine assumes N is normalized.

12. FLOATING POINT TANGENT

GENERAL DESCRIPTION: This subroutine computes the tangent of a floating point argument in radians.

METHOD: This routine uses FLOATA, FSIN, and FCOS.

$$FTAN(X) = \frac{FSIN(X)}{FCOS(X)}$$

CALLING SEQUENCE:

Address	Mnemonic		Operand
L	IRJP	.	(FTAN)
L+1	0	.	TAG 1
L+2	0	.	TAG 2
L+3	(RETURN)	.	

TAG 1 is the address of the least significant half of X and TAG 2 is the address of the least significant half of FTAN(X). Register A also contains FTAN(X) upon return from the routine.

RESTRICTIONS: FTAN must be wholly contained with one bank. All registers except A are saved. TAG 1 and TAG 2 must be in the same bank as the return jump instruction.

13. FLOATING POINT SINE AND COSINE

GENERAL DESCRIPTION: This subroutine computes SIN and COS of an angle in radians expressed in floating point.

METHOD: For the SIN, 90° is subtracted from the argument. For both SIN and COS, the argument is reduced to

$$0 \leq |Arg| \leq 2$$

If the argument falls within the range, $\frac{n \pi}{2} \pm 0.0004$, n + 0, 1, 2, or 3, the correct value is taken from a table and used for the function value. Otherwise the angle is reduced to the 1st quadrant with appropriate adjustment of sign and used in

$$f(X) = 1 + \sum_{i=1}^{5} a_{2i} X^{2i}$$

CALLING SEQUENCE:

Address	Mnemonic		Operand
L	IRJP	.	(FSIN, FCOS)
L+1	0	.	TAG 1
L+2	0	.	TAG 2
L+3	(Return)	.	

TAG 1 is the address of the least significant half of the argument and TAG 2 the address of the least significant half of the function value. Register A also contains the answer upon return from the routine.

RESTRICTIONS: This routine must be wholly contained within one bank. TAG 1 and TAG 2 must be in the same bank as the return jump instruction. This routine also uses FLOATA.

14. FLOATING POINT ARCSINE AND ARCTANGENT

GENERAL DESCRIPTION: This subroutine computes ARCSIN or ARCTAN given an (X,Y) coordinate.

METHOD:

For ARCSIN, $Z = Y/\sqrt{X^2 + Y^2}$

then FASIN $(Z) = FATAN(Z/\sqrt{1-Z^2})$

For ARCTAN, $Z = \dfrac{Y}{X}$

$$f(Z) = \pi/4 + \sum_{i=0}^{7} C_{2i+1} \left(\frac{Z - 1}{Z + 1} \right)^{2i + 1}$$

The appropriate adjustment is made for the quadrant in which (X,Y) appears.

CALLING SEQUENCE:

Address	Mnemonic	Operand
L	IRJP	. (FASIN, FATAN)
L+1	0	. TAG 1
L+2	0	. TAG 2
L+3	0	. TAG 3
L+4	(Return)	. TAG

TAG 1 is the address of the least significant half of the Y-coordinate, TAG 2 the address of the least significant half of the X-coordinate, and TAG 3 the. address of the least significant half of the answer. Register A also contains the answer upon return from the routine.

RESTRICTIONS: The routine must be wholly contained with one bank. TAG 1, TAG 2, and TAG 3 must be in the same bank as the return jump instructions. FLOATA and FSQRT are used by FASIN and FATAN.

15. FLOATING POINT NATURAL LOGARITHM

GENERAL DESCRIPTION: This subroutine computes $\ln(X)$. X is in floating point format.

METHOD: X is changed to the range $1 \leq Z \leq 10$, $X = Z \cdot 10^n$
Then $\ln(X) = \log(X) \ln(10) = \left[\log(Z) + n\right] \ln(10)$ where

$$\log(Z) = 1/2 + \sum_{i=0}^{4} c_{2i+1}\left(\frac{z - \sqrt{10}}{z + \sqrt{10}}\right)^{2i+1}$$

CALLING SEQUENCE:

Address	Mnemonic	Operand
L	IRJP	. (FLN)
L+1	0	. TAG 1
L+2	0	. TAG 2
L+3	(Return)	.

TAG 1 is the address of the least significant half of X and TAG 2 the address of the least significant half of $\ln(X)$. Register A also contains $\ln(X)$ upon return from the routine.

RESTRICTIONS: FLN must be wholly contained within 1 bank. FLOATA, and FIXFLT are used by FLN. TAG 1 and TAG 2 must be in the same bank as the return jump instruction.

16. FLOATING POINT ARITHMETIC

GENERAL DESCRIPTION: This subroutine performs add, subtract, multiply, and
divide operations on floating point numbers. This is the same subroutine as
FLOAT except for I/O parameters and the addition of FSUB.

CALLING SEQUENCE:

Address	Mnemonic	Operand
L	IRJP	. (FADD, FSUB, FMUL, OR FDIV)
L+1	0	. TAG 1
L+2	0	. TAG 2
L+3	0	. TAG 3
L+4	(Return)	.

TAG 1 and TAG 2 are the addresses of the least significant halves of operands 1
and 2. TAG 3 is the address of the least significant half of the answer.

For FADD and FMUL, the order of operands 1 and 2 is not important. For FSUB,
operand 1 is the minuend and operand 2 the subtrahend. For FDIV, operand 1
is the dividend and operand 2 the divisor. Register A also contains the
answer upon return from the routine.

RESTRICTIONS: The routine must be wholly contained with one bank. TAG 1,
TAG 2, and TAG 3 must be in the same bank as the return jump
instruction.

SECTION V-B. CONVERSION SUBROUTINES

1. CONVERT OCTAL TO TYPEWRITER-CODED DECIMAL (TODE)

GENERAL DESCRIPTION: The TODE subroutine converts an octal number, with a given number of fractional binary bits, to a coded decimal number with the required number of decimal fractional digits. It can handle six octal digits, including the decimal point, if present. In addition, it handles negative and positive numbers by using a third output word, containing only the minus code. Roundoff is optional.

INPUT PARAMETERS:

Variable Words	Contents
TODEN	Octal input number for conversion
TODEN1	Contains the number of binary bits to the right of the radix point in the input number TODEN.
TODEN2	Number of output decimal digits to right of decimal point.
TODEN3	Round option: 1 = round; 0 = no round

OUTPUT PARAMETERS:

Variable Words	Contents
TODET TODET1	Hold resulting decimal number, normalized left, in typewriter code. Output words are also in AL and AU.
TODET-1	Contains minus sign code if input is negative and zeros if input is positive.

EXECUTION SEQUENCE:

1) Load input octal number for conversion in location TODEN.
2) Load number of input fractional binary bits in location TODENL.
3) Load output fractional decimal digit indicator in location TODEN2.
4) Load output roundoff indicator in location TODEN3.
5) Return jump to TODE.
6) Use converted values in AU and AL or in TODET and TODET1.

NOTE: The decimal point appears as a period code in output. For example, 3.5 would be written 63 75 65 in Field data code and as 63 56 65 in ASCII code.

2. DECIMAL TO OCTAL ROUTINE (DOCTL)

GENERAL DESCRIPTION: The DOCTL subroutine, using typewriter code, converts decimal numbers, both positive and negative, to octal. It can handle up to 6 typewriter-coded digits, plus, by means of another word, a minus sign if required. The input number may contain a decimal point as one of the six digits. The number of fractional binary bits in the output can be selected as well as the roundoff option.

RESTRICTIONS: Maximum size of input words to be converted is 2 computer words or 6 digits in code; this includes code for decimal point if there is one. (Range is 131071D to -131071D).

INPUT PARAMETERS:

Variable Words	Contents
DOCTN & DOCTN +1	Contain the 6-digit typewriter-coded number.
DOCTN2	Holds number of fractional binary bits requested in answer.
DOCTN3	Contains roundoff option. 1 = no round; 0 = round.
DOCTN -1	Contains minus sign code if number is negative, otherwise zero.

OUTPUT PARAMETERS:

Variable Words	Contents
DOCTOT and AL	Resulting octal digits, scaled and rounded as directed by input parameters at memory location DOCTOT and AL. If B0, error indicator, is nonzero, then no output is produced.

ALARMS: If an error condition occurs, the computer jumps to address RESTOR and the error is identified by a number in B0.

B0 = 4, Two decimal points in input number
B0 = 3, Input number too large
B0 = 2, Too many fractional bits requested
B0 = 1, Illegal code in input

EXECUTION SEQUENCE:

1) Load sign at DOCTN-1.
2) Load first part of input number at DOCTN.
3) Load remainder of input number at DOCTN+1, normalized left.
4) Load output fractional binary bits requested in DOCTN2.
5) Load output roundoff indicator in location DOCTN3.
6) Return jump to DOCTL.
7) Use converted value in A1 or in DOCTOT.

NOTE: The decimal position is known by the number of binary bits shown in DOCTN2.

SECTION V-C. ASSEMBLER SUPPORT SUBROUTINES

1. TRIM DEBUGGING PACKAGE (DEBUG)

GENERAL DESCRIPTION: The TRIM debugging package is activated by generation in
the operational program resulting from the use of the DUMPM or DUMPR operators.
The package dumps the contents of the AL, AU, and current B registers (DUMPR)
or consecutive addresses and their contents (DUMPM) onto punched paper tape
for subsequent off-line listing. Below are samples of input to and output
from the debugging package.

DUMPR

   1) Operational program (shown here, for clarity, in mnemonics)

        IRJP•LOK+1
        0•DEBUG
        0•00002   (0 = DUMPR code; 2 = unique identity number)

   2) DEBUG output

        2. AU 252525   AL 525252   B 007103

DUMPM

   1) Operational program (shown here, for clarity, in mnemonics)

        IRJP•LOK+1
        0•DEBUG
        4•00007   (4 = DUMPM code; 7 = unique identity number)
        0•00004   (number of words to be dumped)
        0•00240   (address of first word to be dumped)

   2) DEBUG output

        7. 00240   340245
            00241   340645
            00242   340747
            00243   340703

The debugging package is available in three formats:

   1) As a TRIM III library subroutine.
   2) As a punched paper tape in source language format.
   3) As a punched paper tape in relocatable bioctal format.

Formats 1) and 2) use the tag CHAN to reference the paper tape I/O channel.
The programmer is responsible for allocating CHAN to the appropriate I/O
channel number.

2. TYPE TEXT SUBROUTINE (TYPT)

GENERAL DESCRIPTION: TYPT is a special TRIM library subroutine which types out pre-designated statements at any point in the user's program where he has inserted a TYPT operator.

The TYPT subroutine processes the object program data words which the type-text generator of the TRIM assembler produces from the poly-operation TYPT source statement. Since the TYPT subroutine is called by an IRJP instruction within the user's object program, the source program must be assembled with the TYPT subroutine or with the TYPT subroutine allocated to an address at which it must be loaded when the user's object program is run. If a TYPT statement is used within a program and the TYPT subroutine is not called or programmer allocated, the TRIM assembler will automatically allocate it to a fixed address in the object program.

TYPT unstrings and types the six bit typewriter characters in the words immediately following the indirect return jump to the TYPT subroutine. It continues to extract and type from sequential words until an end sentinel code is encountered. TYPT then exits to the next address of the object program.

TYPT is available in three formats:

   1) As a TRIM III library subroutine.
   2) As a punched paper tape in source language format.
   3) As a punched paper tape in relocatable bioctal format.

Formats 1) and 2) use the tag CHAN to reference the paper tape I/O channel. The programmer is responsible for allocating CHAN to the appropriate I/O channel number.

3. TYPE CONTENTS SUBROUTINE (TYPC)

GENERAL DESCRIPTION: TYPC is a special TRIM library subroutine which types out the present contents of any register or memory address designated after a TYPC operator at any point in a user's program.

The TYPC subroutine interprets the object program code words which the type-contents generator of the TRIM assembler produces from the poly-operation TYPC source statement; the TYPC subroutine then types the designated register or memory location contents.

Since the TYPC subroutine is called by an indirect return jump within the user's object program, the user's source program must be assembled with the TYPC subroutine or with the TYPC subroutine allocated to an address at which it must be loaded when the user's object program is run. If a TYPC statement is used within a source program and the TYPC subroutine is not called or programmer allocated, the TRIM assembler will automatically allocate it to a fixed address in the object program.

TYPC is available in three formats:

1) As a TRIM III library subroutine.
2) As a punched paper tape in source language format.
3) As a punched paper tape in relocatable bioctal format.

Formats 1) and 2) use the tag CHAN to reference the paper tape I/O channel.
The programmer is responsible for allocating CHAN to the appropriate I/O
channel number.

4. PUNCH TEXT SUBROUTINE (PCHT)

GENERAL DESCRIPTION: PCHT is a special TRIM library subroutine which punches
out pre-designated statements at any point in the user's program where he has
inserted a PCHT operator.

The PCHT subroutine processes the object program data words which the punch-
text generator of the TRIM assembler produces from the poly-operation PCHT
source statement. Since the PCHT subroutine is called by an IRJP instruction
within the user's object program, the user's source program must be assembled
with the PCHT subroutine or with the PCHT subroutine allocated to an address
at which it must be loaded when the user's object program is run. If a PCHT
statement is used within a program and the PCHT subroutine is not called or
programmer allocated, the TRIM assembler will automatically allocate it to
a fixed address in the object program.

PCHT unstrings and punches the 6-bit typewriter characters in the words imme-
diately following the indirect return jump to the PCHT subroutine. It continues
to extract and punch from sequential words until an end sentinel code is en-
countered. PCHT then exits to the next address of the object program.

PCHT is available in three formats:

1) As a TRIM III library subroutine.
2) As a punched paper tape in source language format.
3) As a punched paper tape in relocatable bioctal format.

Formats 1) and 2) use the tag CHAN to reference the paper tape I/O channel.
The programmer is responsible for allocating CHAN to the appropriate I/O
channel number.

5. PUNCH CONTENTS SUBROUTINE (PCHC)

GENERAL DESCRIPTION: PCHC is a general TRIM library subroutine which punches
out the present contents of any register or memory address designated after a
PCHC operator at any point in a user's program.

The PCHC subroutine interprets the object program code words which the punch-
contents generator of the TRIM assembler produces from the poly-operation
PCHC source statement; the PCHC subroutine then punches the designated
register or memory location contents.

Since the PCHC subroutine is called by an indirect return jump within the user's object program, the user's source program must be assembled with the PCHC subroutine or with the PCHC subroutine allocated to an address at which it must be loaded when the user's object program is run. If a PCHC statement is used within a source program and the PCHC subroutine is not called or programmer allocated, the TRIM assembler will automatically allocate it to a fixed address in the object program.

PCHC is available in three formats:

   1)   As a TRIM III library subroutine.
   2)   As a punched paper tape in source language format.
   3)   As a punched paper tape in relocatable bioctal format.

Formats 1) and 2) use the tag CHAN to reference the paper tape I/O channel. The programmer is responsible for allocating CHAN to the appropriate I/O channel number.

TABLE A-1. EQUIVALENT INPUT FORMAT CODES

Software Name	Software Symbol	UNIVAC 1532 Symbol Substitution	ASCII Codes	UNIVAC 1232 Symbol Substitution	Field Data Codes
Carriage return	↙		1 5		0 4 0 3
Tab	→	←	1 3 7		7 6
Point separator	•	*	5 2	Special ▢	7 2
Double period	..		5 6 5 6	Apostrophe '	7 5 7 5
Space	△		4 0		0 5
Comma	,		5 4		5 6
Vertical bar					4 1	Exclamation !	5 5
Plus	+		5 3		4 2
Minus	-		5 5		4 1

TABLE A-2. FIELD DATA CODE (6 BITS), UNIVAC 1232
KEYBOARD AND TYPEWRITER

	0	1	2	3	4	5	6	7
0	Master Space	Upper Case	Lower Case	Line Feed	Car. Return	△	A	B
1	C	D	E	F	G	H	I	J
2	K	L	M	N	O	P	Q	R
3	S	T	U	V	W	X	Y	Z
4	)	-	+	<	=	>	_	$
5	*	(	"	:	?	!	' (Comma)	Ⓢ STOP
6	0	1	2	3	4	5	6	7
7	8	9	' (Apos.)	;	/	. (Period)	▭ SPEC	↑ IDLE

△ = SPACE

NOTE: Master space indicates an absence of information.

A-2

TABLE A-3. ASCII CODE (7 BITS), UNIVAC 1532
KEYBOARD AND TYPEWRITER

	0	1	2	3	4	5	6	7
00			-					
01			Line feed			Carriage return		
02								
03								
04	Space	!	"	#	$	%	&	' (apos.)
05	(	)	*	+	, (comma)	-	. (period)	/
06	0	1	2	3	4	5	6	7
07	8	9	:	;	<	=	>	?
10	@	A	B	C	D	E	F	G
11	H	I	J	K	L	M	N	O
12	P	Q	R	S	T	U	V	W
13	X	Y	Z	[	/	]	↑	←

TABLE A-4. TRIM INTERNAL CHARACTER CODE CHART (6 BITS)

	0	1	2	3	4	5	6	7
0	i	Δ	Data Control Characters					!
1	0	1	2	3	4	5	6	7
2	8	9	A	B	C	D	E	F
3	G	H	I	J	K	L	M	N
4	Not Used	P	Q	R	S	T	U	V
5	W	X	Y	Z	(	)	<	>
6	-	+	'	/	:	* / &	□ / ←	?
7	=	"	$	— / *	.	,	;	Not Used

i = ignore

Δ = space

Data control characters are used as internal controls.

Code 10 is used for zero and letter O.

▱ = Field data code symbol.

◿ = ASCII code symbol.