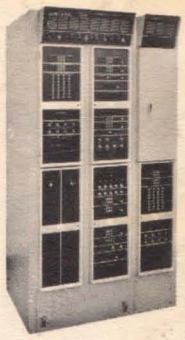
UNIVAC® 1219

MILITARY COMPUTER



PROGRAMING STUDY GUIDE

C. Whitehead Jr.

P. McLaughlin

1219 COMPUTER PROGRAMING STUDENT STUDY GUIDE

FEBRUARY 1967

PX 3943-0-2

Prepared by: Defense Systems Training

UNIVAC | DEFENSE SYSTEMS DIVISION
DIVISION OF SPERRY RAND CORPORATION

LIST OF EFFECTIVE PAGES

PAGE NUMBER	CHANGE IN EFFECT	PAGE NUMBER	CHANGE IN EFFECT
Title	Original	1.5-1 thru 1.5-18	Original
ii thru vi	Original	1.6-1 thru 1.6-26	Original
1.1-1 thru 1.1-8	Original	2.1-1 thru 2.1-30	Original
1.2-1 thru 1.2-42	Original	2.2-1 thru 2.2-36	Original
1.3-1 thru 1.3-20	Original	2.3-1 thru 2.3-56	Original
1.4-1 thru 1.4-32	Original	3.1-1 thru 3.1-50	Original

USE OF YOUR STUDY GUIDE FOR PROGRAMING TRAINING

PUBLICATIONS

This publication is for your use while studying programing of the 1219 Computer. You may place in this book any notes that will help you for reviewing later on in the class, or that will help you when you are on the job. You may retain this publication for your personal use.

OTHER PUBLICATIONS

This publication was designed to help you study the available UNIVAC Manuals and allow you easier access to particular information desired. The study guides in this publication are to be used for training purposes only and are not meant to supersede or replace any other publications.

PRESENTATION OF COURSE MATERIAL

The presentation of course material on the 1219 Computer is according to study guide topics. Each topic will be presented by the instructor. He will assign certain sections of the study guides to enable you to study more effectively the information that has been presented or will be presented in class. The study guides contain training goals and objectives for the particular topic, information supplementing existing manuals, references, and study questions included to help you measure your understanding and help you learn the material.

OUTLINE OF CONTENTS

SECTION 1 - UNIVAC 1219 COMPUTER FUNCTIONAL DESCRIPTION

1.1 General Description
1.2 Instruction Repertoire
1.3 Operator's Information
1.4 Program Development
1.5 Detailed Block Diagram of the 1219 Computer
1.6 Functional Input/Output

SECTION 2 - SOFTWARE

2.1 Trim II Assembler
2.2 Trim III Assembler
2.3 Operator Service Routines

SECTION 3 - PERIPHERAL EQUIPMENT

3-1. Peripheral Equipment

SECTION 1 - UNIVAC 1219 COMPUTER FUNCTIONAL DESCRIPTION

1.1. GENERAL DESCRIPTION

1.1-1. OBJECTIVES

To give the student additional information on the general characteristics of the 1219 computer.

1.1-2. INTRODUCTION

The 1219 computer is a versatile, stored-program, medium-scale, general-purpose digital computer with solid state, parallel circuitry designed for maximum reliability.

1.1-3. REFERENCES

PX 3288, Programers Reference Manual, Section I-A.

1.1-4. INFORMATION

a. General. The UNIVAC 1219 computer is a medium-scale, general-purpose, digital computer. The 1219 has a standard computer word length of 18 binary digits or bits, with the capacity to combine two binary words to do 36-bit arithmetic operations. The computer uses one's complement binary arithmetic and logical operations to manipulate data. This data is contained in the computer in two types of storage devices.

One of these storage devices is the flip-flop register. The flip-flop is a bistable electrical device. One of these two stable states is used to represent a binary one and the other a binary zero. A series of these flip-flops is combined to form a device for holding binary data. This is referred to as a hardware register.

The second device for storing binary data is the magnetic core memory. This is a nonvolatile storage device where binary information is represented by the direction of magnetic flux in each core.

To solve a problem, a digital computer executes a series of instructions stored within its memory. This series of instructions, called a program, tells the computer which functions to perform. The instructions are performed by the transfer of the instruction from memory into hardware registers and the manipulation of that information and/or additional information from memory, other registers, or external sources.

The computer has a repertoire of more than 100 flexible, single-address instructions with provisions for address or operand modification by eight index registers.

b. Operational. The computer consists of four functionally definable sections as shown in figure 1.1-1. These sections, classified by the function which they perform, are the control section, the I/O section, the arithmetic section and the memory section.

1. Control Section. The control section provides the timing, instruction translation, and operational sequencing required for performance under either manual operation or operation under program control.

For manual operation, control panels contain those switches and indicators with which the computer may be sequenced, under operator control, through the functions of an instruction, allowing each operation and its results to be visually displayed and examined.

2. I/O Section. The I/O section contains optional 4, 8, 12, or 16 input/output channels used for communication between the computer and associated peripheral devices or other computers. Circuits contained within the I/O section are the I/O registers, priority circuits, and associated circuits required to logically initiate the input and output signal and data transfers. Thus, the I/O section provides the necessary buffer between the main computer circuitry and the peripheral communicating devices.

3. Arithmetic Section. The arithmetic section of the computer performs the arithmetic and logic functions to present a solution to a given problem. The arithmetic functions include addition, subtraction, multiplication, division, shifting, and scaling. The logic functions include masking, selective substitution, comparisions, and word transfers between the components of the arithmetic section or between the arithmetic section and the computer core memory. The circuits contained within the arithmetic section include a 36-bit A-register which is subdivided into two 18-bit registers, AU and AL, the X-register, the D-register, the W-register, and the subtractive type adder.

The arithmetic section employs one's complement binary arithmetic to perform these arithmetic and logic functions. The control section utilizes the arithmetic section to modify operands and addresses.

4. Memory Section. The memory section consists of a main core storage memory, a core storage control memory, a bootstrap memory, and the associated addressing, control, and transfer logic. The main memory has the capability of storing optional 8k, 16k, or 32k, 18-bit binary words with provisions for expansion to 65k. It is used to store programs, constants, and input/output data. It is bit-oriented and operates on the coincident current principle.

The bootstrap memory is word-organized, permanent storage for fixed instructions and constants and provides for automatic initial loading of programs and for automatic recovery in the event of program failure. The contents of the bootstrap memory, 32, 18-bit words, are unalterable.

The control memory contains either 128 or 256 storage addresses, depending upon the optional operating modes contained within the equipment. It provides storage for the buffer control words and contains the index registers, fault registers, real-time clock registers, and the scale factor shift count register. The control memory is a word-organized magnetic core memory with a cycle time of 500 nanoseconds.

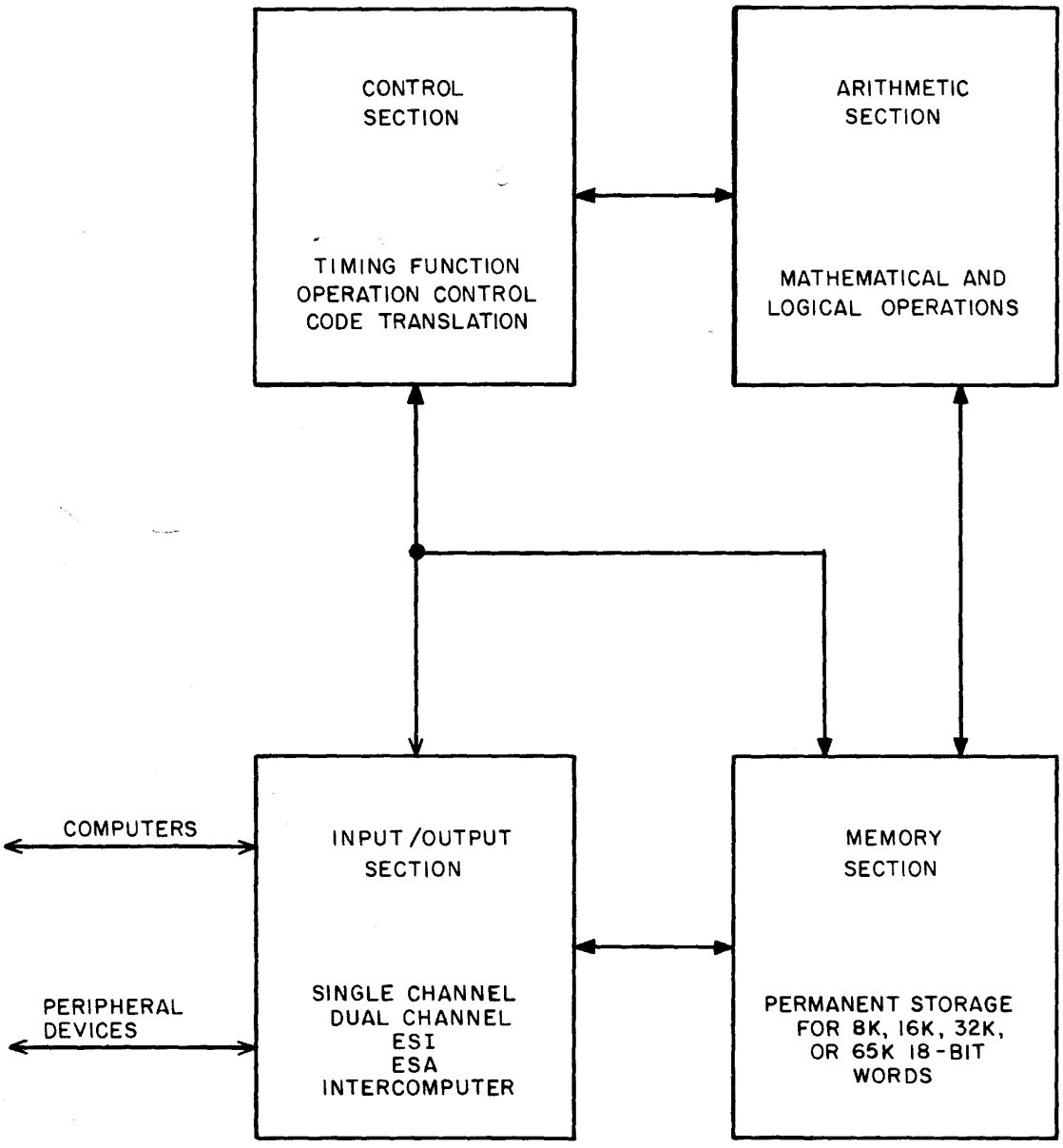


Figure 1.1-1. UNIVAC 1219 Computer Simplified Block Diagram

c. Registers and Their Contents. All registers in the computer may be classified as addressable or unaddressable. Those discussed here will be other than the normal core storage registers. Addressable registers are directly available to the programer through computer instructions. The other functional registers are nonaddressable.

1. Addressable Registers

a) 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

b) 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

c) 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

d) B. The contents of the active index register in control memory which are used to modify y to form an address or an odd-numbered instruction less than 50_8 and, therefore, not illustrated on the block diagram. 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 are transmitted.

e) 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.

f) P. Program address register, P, (i.e., the address of the instruction currently being entered for execution). The contents of P are 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.

g) SR. A 5-bit special register through which the program has control of the 4096 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 the 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 $2^3 = 1$.) Refer to paragraph 1.3.

2. Nonaddressable Registers

a) B_U. An 18-bit FF-register associated with control memory. It is used to hold and update buffer limits during I/O operations and to hold contents of the currently active index or B register, located in control memory, prior to updating.

b) C_0. Two 18-bit output buffer registers for transferring data or instruction words (external function) to external devices which may include other computers. The C_0-register is the buffer register for the odd-numbered channels (1, 3, 5 and 7) and the C_E-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.

c) C_0 & C_E. Two 18-bit output buffer registers for channels 10_8 through 17_8 (optional).

d) D. An 18-bit arithmetic exchange register holds an operand for the adder during arithmetic and I/O operations.

e) 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 1 instructions and m for format 2 instructions). The most significant bit will be set for format 2 instructions only. Computer control is directed from this register.

f) S_1. A 16-bit address register which receives the address of a main 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 input/output 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.

g) S_0. A 7- or 8-bit address register which receives the address of control memory locations. The S_0-register may receive addresses from the S_1-register, ICR-register, the I/O section , and the arithmetic section as well as peripheral equipment capable of specifying an address.

h) W. An 18-bit shifting register in the arithmetic section.

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

j) Z_1. 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 and data.

k) Z_0. An 18-bit control memory buffer register for all transfers to and from control memory locations. It communicates with all sections of the computer.

NAME: _____

1.1-5. STUDY QUESTIONS

a. What are the four major sections of a computer and what is the function of each?

I/O
CONTROL
ARITHMETIC
MEMORY

b. Define a program.

INSTRUCTIONS TO COMPUTER. STEP BY STEP

c. What is meant by one's complement binary arithmetic?

i.e, ADD ~~8+7~~ 8 − 7 = 01000 + 11000 = 1 00000 → +1 = 00001 = 1

00111
11000

d. Explain the difference between addressable and nonaddressable registers.

e. Explain the difference between the B-register and the BU-register.

SECTION 1 - UNIVAC 1219 COMPUTER FUNCTIONAL DESCRIPTION

1.2. INSTRUCTION REPERTOIRE

1.2-1. OBJECTIVES

a. To provide a means of study of the instruction repertoire by classified types to improve your understanding and use of the various instructions.

b. To provide study questions on the use of the instructions.

1.2-2. INTRODUCTION

In order to program the 1219 it is necessary that the programer have a complete understanding of the instruction repertoire. It is the purpose of this section to aid you in the study of the instruction repertoire and its use.

1.2-3. REFERENCES

PX 3288, Programers Reference Manual, Section I-C & I-D

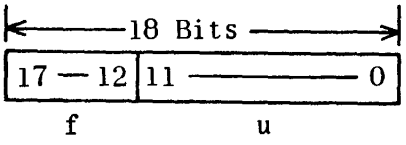
1.2-4. INFORMATION

a. Symbol Conventions. See table 1.2-1.

b. Philosophy of Memory Address Control. The computer main core memory is built in modules of 4096 memory locations. The twelve bit constant u of the instruction word can designate one of these locations in any bank of memory. In order to designate a specific bank of memory in addition to a location within the bank the value u must be extended upward. For a memory size of 4096 times 2^n the computer must be capable of extending u upward in magnitude n bits. These n bits are taken from those bits of SR $2^0 - 2^2 + 2^4$ or from those bits of P above the twelve lower order bits. The programer has control over the activity of SR through which he may specify any address in memory. Refer to paragraph 1.2-4c, Word Format, for detailed uses of P and SR.

c. Instruction Word Formats. Two basic instruction word formats are used by the computer, format I, and format II.

1. Format I Instructions.



f: function code, six high order bits
u: twelve low order bits

TABLE 1.2-1 SYMBOL CONVENTIONS

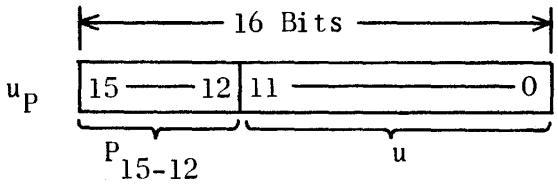
SYMBOL	DESCRIPTION
AU	Upper accumulator, 18-bit arithmetic register.
AL	Lower accumulator, 18-bit arithmetic register.
A	AU and AL linked together to form one 36-bit arithmetic register.
B	Contents of the active index register, 18-bit one's complement.
f	Function code, high order six bits of all instruction 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, 5-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 the table: | | 0 | 1 | |---|---|---| | 0 | 0 | 0 | | 1 | 0 | 1 |
( ) v ( )	Logical sum, or inclusive OR defined by the table: | | 0 | 1 | |---|---|---| | 0 | 0 | 1 | | 1 | 1 | 1 |
( ) + ( )	Half add, Half subtract, or exclusive OR defined by the table: | | 0 | 1 | |---|---|---| | 0 | 0 | 1 | | 1 | 1 | 0 |
( )' 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.
→	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, i.e., $xY = \underbrace{u_{11} \cdots u_{11}}_{\text{6 Bits}} \underbrace{u_{11} \cdots u_0}_{\text{12 Bits}}$

The definition and usage of u are determined by the function code utilizing u in two distinct manners:

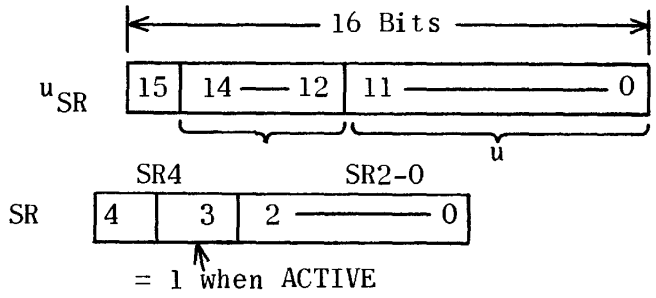
a) u Used as a Constant. In this case, u itself is the operand and requires no further memory reference; however, u is extended to 18 bits. (Refer to paragraph 1.2-4d, List of Instructions.)

b) u Used as an Address. In this case, u is used as the lower order 12 bits of the base address referring to a memory cell. The base address is 16 bits, designated as u_P or u_{SR}, and is described below:

u_P is defined as a 16-bit address whose four high order bits consist of the four higher order bits of P and whose twelve low order bits are u.

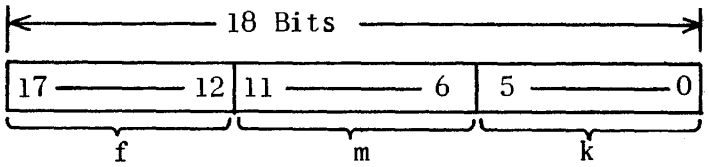


u_{SR} is defined as a 16-bit address whose four high order bits consist of the three lower bits and the high order bit of SR and whose twelve low order bits are u.



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. (Refer to paragraph 1.2-4d, List of Instructions.)

2. Format II Instructions.



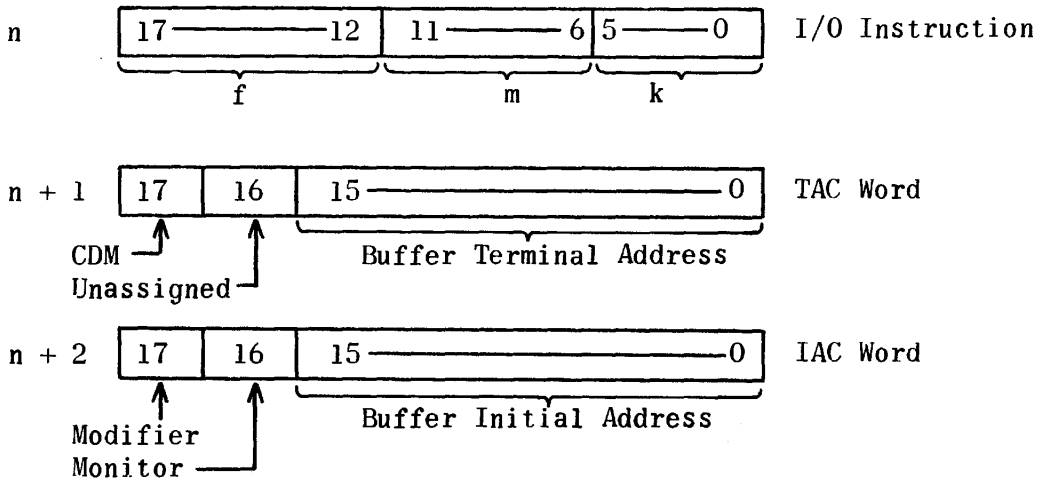
f: six-bit function code (always equal to octal 50)
m: six-bit minor function code
k: six low order bits

Format II instructions perform a variety of operations and can be classified in two instruction categories:

a) No Memory Address Needed. In this case, the information existing in the computer's arithmetic or control registers and the value k are sufficient to perform the specified operation.

b) Initiate Input/Output 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; the format is as follows:

Any Address



Bit 17: CDM of n + 1 (terminal address control word) is the continuous data mode identifier. If equal to one the computer I/O section operates in the continuous data mode. If equal to zero a normal buffer is executed.

Bit 17: Modifier of n + 2 (initial address control word); if equal to one the buffer current (initial) address is decremented for each word transferred in or out; if equal to zero the buffer current address is incremented.

Bit 16: Monitor of n + 2 (initial address control word); if equal to one the monitor interrupt occurs upon successful completion of the last transfer; if equal to zero no monitor interrupt will occur.

NOTE

Normal buffer termination occurs when the incremented/decremented buffer current address is equal to the buffer terminal address. A buffer is terminated when tests in the control section detect buffer control address equality.

3. I/O Buffer Initiating Instructions. During the execution of any instruction that initiates a buffer three main memory references are involved.

a) The I/O instruction is extracted from memory and interpreted by the control section and sets I/O active on the specified channel.

b) The terminal address control word is transferred from the location following the I/O instruction to the control memory location assigned to that type buffer terminal address control word.

c) The initial address control word is transferred from the location following the TAC word in main memory to the control memory location assigned to that type buffer current address word.

Computer control reads the next sequential instruction and continues the program leaving the input/output section with the task of handling the transfers. The input/output section generates the addresses in control memory to examine the control words placed there by the preceding steps when it receives a request for word transfer from the device on the activated channel. For the actual word transfer the I/O section robs one main memory cycle from the program being executed.

d. List of Instructions (By Classified Types). Tables 1.2-3 through 1.2-15 list the repertoire of instructions for the computer. Common usage and examples are included with instructions where the meaning may not be obvious. No attempt has been made to indicate more sophisticated use. The instructions are listed and defined in the following format.

(Octal code) (Instruction name) (TRIM code) (Symbolic summary)

(Execution time)

(Definition of the y address or constant)

(Text defining the instruction in detail)

(Examples and/or notes if any)

The symbolic summary expression will use the symbol Y to include y or y + B whichever is stated in the text for that instruction.

1. Enter Type Instructions. See table 1.2-3.

2. Store Type Instructions. See table 1.2-4

3. Modifying Type Instructions. See table 1.2-5.

4. Jump Instructions. See table 1.2-6.

   a) Unconditional Jumps. See table 1.2-6.

   b) Conditional Jumps. See table 1.2-7.

TABLE 1.2-2. UNIVAC 1219 COMPUTER REPERTOIRE OF INSTRUCTIONS

CODE	SYMBOL	DESCRIPTION	TIME IN μs
02	CMAL	Compare Y	4
03	CMALB	Compare Y + B	4
04	SLSU	Selective substitute	4
05	SLSUB	Selective substitute Y + B	4
06	CMSK	Masked compare Y	4
07	CMSKB	Masked compare Y + B	4
10	ENTAU	Enter AU, Y	4
11	ENTAUB	Enter AU, Y + B	4
12	ENTAL	Enter AL, Y	4
13	ENTALB	Enter AL, Y + B	4
14	ADDAL	Add Y, 18 bit	4
15	ADDALB	Add Y, + B, 18 bit	4
16	SUBAL	Subtract Y, 18 bit	4
17	SUBALB	Subtract Y + B, 18 bit	4
20	ADDA	Add Y, 36 bit	6
21	ADDAB	Add Y + B, 36 bit	6
22	SUBA	Subtract Y, 36 bit	6
23	SUBAB	Subtract Y + B, 36 bit	6
24	MULAL	Multiply Y	14
25	MULALB	Multiply Y + B	14
26	DIVA	Divide, Y	14
27	DIVAB	Divide, Y + B	14
30	IRJP	Indirect RJP, Y	6
31	IRJPB	Indirect RJP, Y + B	6
32	ENTB	Enter B, Y	4
33	ENTBB	Enter B, Y + B	4
34	JP	Jump, Y	2
35	JPB	Jump, Y + B	2
36	ENTBK	Enter, B, U	2
37	ENTBKB	Modify, B, U	2
40	CL	Store zero, Y	4
41	CLB	Store zero, Y + B	4
42	STRB	Store, B, Y	4
43	STRBB	Store B, Y + B	4
44	STRAL	Store AL, Y	4
45	STRALB	Store AL, Y + B	4
46	STRAU	Store AU, Y	4
47	STRAUB	Store AU, Y + B	4
51	SLSET	Selective set (IOR), Y	4
52	SLCL	Selective clear (AND), Y	4
53	SLCP	Selective complement (XOR), Y	4
54	IJPEI	Indirect jump (RIL), Y	4
55	IJP	Indirect jump, Y	4
56	BSK	Increment B, skip, Y	4
57	ISK	Decrement index, skip, Y	6
60	JPAUZ	Jump AU zero, Y	2
61	JPALZ	Jump AL zero, Y	2
62	JPAUNZ	Jump AU not zero, Y	2
63	JPALNZ	Jump AL not zero, Y	2
64	JPAUP	Jump AU positive, Y	2

TABLE 1.2-2. UNIVAC 1219 COMPUTER REPERTOIRE OF INSTRUCTIONS (CONT.)

CODE	SYMBOL	DESCRIPTION	TIME IN μs
65	JPALP	Jump AL positive, Y	2
66	JPAUNG	Jump AU negative, Y	2
67	JPALNG	Jump AL negative, Y	2
70	ENTALK	Enter AL, Y	2
71	ADDALK	Add U, 12 bits	4
72	STRICR	Store ICR, Y	4
73	BJP	Decrement B, jump, Y	2
74	STRADR	Store address, Y	4
75	STRSR	Store SR, deactivate SR, Y	4
76	RJP	Return jump, Y	4
5001	SIN	Set input active	2
5002	SOUT	Set output active	2
5003	SEXF	Set external function active	2
5011	IN	Initiate input buff, k	6
5012	OUT	Initiate output buff, k	6
5013	EXF	External function	6
5014	RTC	Enable real-time clock	2
5015	INSTP	Terminate input, k	2
5016	OUTSTP	Terminate output, k	2
5020	SRSM	Set resume ff (Intercomp)	2
5021	SKPIIN	Skip input inact, k	2
5022	SKPOIN	Skip output inac, k	2
5024	WRFI	Wait for interrupt	2
5026	OUTOV	Force output one word, k	2
5027	EXFOV	Force Ext function one word, k	2
5030	RIL	Remove interrupt lockout	2
5032	EXL	Remove Ext interrupt lockout	2
5034	SIL	Set interrupt lockout	2
5036	SXL	Set Ext interrupt lockout	2
5041	RSHAU	Right shift AU, k	2-10
5042	RSHAL	Right shift AL, k	2-10
5043	RSHA	Right shift A, k	2-20
5044	SF	Scale A left, k, SF	2-20
5045	LSHAU	Left shift AU, k	2-10
5046	LSHAL	Left shift AL, k	2-10
5047	LSHA	Left shift A, k	2-20
5050	SKP	Skip console key, k	2
5051	SKPNBO	Skip no borrow	2
5052	SKPOV	Skip overflow	2
5053	SKPNOV	Skip no overflow	2
5054	SKPODD	Skip L(AU, AL) odd parity	2
5055	SKPEVN	Skip L(AU, AL) even parity	2
5056	STOP	Stop console key, k	2
5057	SKPNR	Skip no resume ff (intercomp)	2
5060	RND	Round AU	2
5061	CPAL	Complement AL	2
5062	CPAU	Complement AU	2
5063	CPA	Complement A	2
5072	ENTICR	Enter ICR, k	2
5073	ENTSR	Enter SR, k	2

TABLE 1.2-3. ENTER TYPE INSTRUCTIONS

10	ENTER AU (ENTAU) Execution time: $y = u_p$ or u_{SR} Clear AU. Then transmit (y) to AU.	(Y) ⟶ AU 4 microseconds
11	ENTER AU (ENTAUB) Execution time: $y = u_P$ or u_{SR} Clear AU. Then transmit (y + B) to AU.	(Y) ⟶ AU 4 microseconds
12	ENTER AL (ENTAL) Execution time: $y = u_P$ or u_{SR} Clear AL. Then transmit (y) to AL.	(Y) ⟶ AL 4 microseconds
13	ENTER AL (ENTALB) Execution time: $y = u_P$ or u_{SR} Clear AL. Then transmit (y + B) to AL.	(Y) ⟶ AL 4 microseconds
32	ENTER B (ENTB) Execution time: $y = u_P$ or u_{SR} Transmit (y) to B_{ICR} The full 18 bits of (y) are transmitted to the B-register (a normally addressable core cell).	(Y) ⟶ B Reg 4 microseconds
33	ENTER B (ENTBB) Execution time: $y = u_P$ or u_{SR} Transmit (y + B) to B_{ICR} The full 18 bits of (y + B) are transmitted to the B register (a normally addressable core cell).	(Y) ⟶ B Reg 4 microseconds

TABLE 1.2-3. ENTER TYPE INSTRUCTIONS (CONT.)

36 — ENTER B WITH CONSTANT (ENTBK) — xY⟶B — 2 microseconds

Execution time:
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 instruction; it does not refer to an address.
Example of enter B with constant when u = 7701:

B_i = any value
B_f = 777701

70 — ENTER AL WITH CONSTANT (ENTALK) — xY⟶AL — 2 microseconds

Execution time:
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 within the instruction; it does not refer to an address.

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

Execution time:
Clear the index control register. Then transmit the three, low-order bits of k to the ICR.

50 73 — ENTER SPECIAL REGISTER (ENTSR) — k_{4-0}⟶SR — 2 microseconds

Execution time:
Clear the special register. Then transmit the five low-order bits of k to the SR. (SR_3 = 1 activates the SR.)

TABLE 1.2-4. STORE TYPE INSTRUCTIONS

40	CLEAR Y (STORE ZERO) (CL) Execution time: y = u_P or u_{SR} Store an 18-bit word of zeros at storage address y.	0 ⟶ Y 4 microseconds
41	CLEAR Y (STORE ZERO) (CLB) Execution time: y = u_P or u_{SR} Store an 18-bit word of zeros at storage address y + B.	0 ⟶ Y 4 microseconds
42	STORE B(STRB) Execution time: y = u_P or u_{SR} Store B at storage address y.	B ⟶ Y 4 microseconds
43	STORE B (STRBB) Execution time: y = u_P or u_{SR} Store B at storage address y + B.	B ⟶ Y 4 microseconds
44	STORE AL (STRAL) Execution time: y = u_P or u_{SR} Store (AL) at storage address Y. $(AL)_f = (AL)_i$	(AL) ⟶ Y 4 microseconds
45	STORE AL (STRALB) Execution time: y = u_P or u_{SR} Store (AL) at storage address y + B. $(AL)_f = (AL)_i$	(AL) ⟶ Y 4 microseconds
46	STORE AU (STRAU) Execution time: y = u_P or u_{SR} Store (AU) at storage address y. $(AU)_f = (AU)_i$	(AU) ⟶ Y 4 microseconds
47	STORE AU (STRAUB) Execution time: y = u_P or u_{SR} Store (AU) at storage address y + B. $(AU)_f = (AU)_i$	(AU) ⟶ Y 4 microseconds
50	(See format 2 instructions immediately following function code 77.)	

TABLE 1.2-4. STORE TYPE INSTRUCTIONS (CONT.)

72	STORE INDEX CONTROL REGISTER (STRICR) Execution time: $y = u_P$ Replace the low-order six bits of (y) with a six-bit value consisting of the four lower-order bits equal to the address of the index register and the remaining two bits equal to zero. As this instruction effects a six-bit partial transfer, the upper 12 bits of (y) remain unchanged. $(ICR)_f = (ICR)_i$	$(ICR) \longrightarrow Y_{3-0}$ 4 microseconds
74	STORE ADDRESS (STRADR) Execution time: $y = u_P$ 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 six bits of (y) remain undisturbed. $(AL)_f = (AL)_i$ Examples of a store address instruction: $(AL)_i = 742504$ $(y)_i = 567777$ $(y)_f = 562504$	$(AL)_{11-0} \longrightarrow Y_{11-0}$ 4 microseconds
75	STORE SPECIAL REGISTER (STRSR) Execution time: $y = u_P$ Replace the low-order six bits of (y) with a six-bit value of which the five low-order bits are equal to special register with the remaining bit equal to zero, then clear SR. As this instruction effects a six-bit partial transfer, the upper 12 bits of (y) remain undisturbed. $(SR)_f = 0$	$(SR) \longrightarrow Y_{4-0}$ 4 microseconds

TABLE 1.2-5. MODIFYING TYPE INSTRUCTIONS

37	MODIFY B WITH CONSTANT (ENTBKB) Execution time: y = u (sign extended to 18 bits) Add y to B (add a constant to B). Note that u is a 12-bit, one's complement number contained within the instruction and can be used to increment or decrement B.	B_i + xY⟶B Reg 2 microseconds
56	B SKIP (BSK) Execution time: y = u_P Test B and (y) for equality. Skip next instruction if equal; otherwise increment B by one and read the next instruction.	If B = (Y), skip NI If B ≠ (Y), (B) + 1 ⟶B, read NI 4 microseconds

57	INDEX SKIP (ISK)
	
	Execution time:
	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) = 0, skip NI
		If (Y) ≠ 0, (Y) - 1 ⟶Y, read NI
		6 microseconds

73	B JUMP (BJP) Execution time: y = u_P If B ≠ 0, subtract one from B, then jump to y; otherwise take the next instruction leaving B unaltered (neg zero ≠ 0).	If B ≠ 0, B-1⟶ B-Reg & Y⟶P If B = 0, Execute NI 2 microseconds

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.

TABLE 1.2-6. UNCONDITIONAL JUMPS

30 INDIRECT RETURN JUMP (IRJP) (P) + 1⟶(Y); (Y) + 1⟶P

Execution time: 6 microseconds

instruction executed from running program:

$y = u_P$

Store (P) + 1 at the address which is the low-order 16 bits of (y), then increment that address by one and enter it into the program address register.

Instruction executed from entrance register on interrupt: y = u.

NOTE

u is 12 bits

Store (P) at the address which is the low-order 16 bits of (y), then increment that address by one 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 address 17421, but at the same time letting the subroutine know where to return control.

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

Execution time: 6 microseconds

Instruction executed from running program:

$y = u_P$

Store (P) + 1 at the address which is the low-order 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.

NOTE

u is 12 bits

Store (P) at the address which is the low-order 16 bits of (y + B) then increment that address by one and enter it into the program address register.

TABLE 1.2-6. UNCONDITIONAL JUMPS (CONT.)

34 DIRECT JUMP (JP) — Y⟶P; NI = (Y)
Execution time: 2 microseconds
$y = u_P$
Unconditional jump to y. (Set P = y.)

35 DIRECT JUMP (JPB) — Y⟶P; NI = (Y)
Execution time: 2 microseconds
$y = u_P$
Unconditional jump to y + B.

NOTE

Since B is an 18-bit, one's complement number, care must be taken when using this instruction. In addition, it is possible that the 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 ENABLE INTERRUPTS (IJPEI) — (Y)⟶P; enable interrupts
Execution time: 4 microseconds
$y = u_P$ Address = $(y)_{15-0}$

Remove interrupt lockout (enable interrupts). Then jump to the address which is the low-order 16 bits of (y). An application of this instruction is the termination of a subroutine activated by an interrupt.

55 INDIRECT JUMP (IJ) — (Y)⟶P
Execution time: 4 microseconds
$y = u_P$ Address = $(y)_{15-0}$
Jump to the address which is the low-order 16 bits of (y).

76 DIRECT RETURN JUMP (RJP) — (P) + 1⟶Y; Y + 1⟶P
Execution time: 4 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 16 bits being the address (P) + 1 with the upper two 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.

77 ILLEGAL CODE - Jump to fault entrance register, address 0, or address 500 (depending upon position of AUTO RECOVERY switch).
Execution time: 2 microseconds

TABLE 1.2-7. CONDITIONAL JUMPS

60 JUMP AU ZERO (JPAUZ)

If (AU) = 0, · Compare not set, ⊕ (AL) = M, · Compare set, ⊕ L(AL)(AU) = M, · Compare set, Y⟶P

Execution time: 2 microseconds

$y = u_P$

Jump to y, for example, set P = y, if:

1) Compare stage of the comparison designator is not set and (AU) = 0. (Negative zero acts as not zero.)

or

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

Otherwise, execute the next instruction.

61 JUMP AL ZERO (JPALZ)

(JPEQ)

If (AL) = 0, · Compare not set, ⊕ (AL) = M, · Compare set, + L(AL)(AU) = M, · Compare set, Y⟶P

Execution time: 2 microseconds

$y = u_P$

Jump to y, for example, set P = y, if:

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

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

Otherwise, execute next instruction.

62 JUMP AU NOT ZERO (JPAUNZ)

If (AU) ≠ 0, · Compare not set, ⊕ (AU) ≠ M, · Compare set, ⊕ L(AL)(AU) ≠ M, · Compare set, Y⟶P

Execution time: 2 microseconds

$y = u_P$

Jump to y, for example, set P = y, if:

1) Compare stage of comparison designator is not set and (AU) ≠ 0.

or

TABLE 1.2-7. CONDITIONAL JUMPS (CONT.)

	2) 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) JUMP AL NOT EQUAL M (JPNOT) Execution time: $y = u_P$ Jump to y, for example, set P = y, if: 1) Compare stage of comparison designator is not set and (AL) $\neq$ 0. or 2) Compare stage of comparison designator is set and the equals stage of the comparison designator is not set. Otherwise, execute next instruction.	If (AL) $\neq$ 0, · Compare not set, $\oplus$ (AL) $\neq$ M · Compare set, $\oplus$ L(AL)(AU) $\neq$ M, · Compare set, Y $\longrightarrow$ P 2 microseconds
64	JUMP AU POSITIVE (JPAUP) JUMP AL EQUAL OR GREATER THAN M (JPMLEQ) Execution time: $y = u_P$ Jump to y, for example, set P = y, if: 1) Compare stage of comparison designator is not set and (AU) $\geq$ 0. or 2) Compare stage of comparison designator is set and the less than stage of comparison is not set. Otherwise, execute next instruction.	If (AU) Pos · Compare not set, $\oplus$ (AL) $\geq$ M · Compare set, $\oplus$ L(AL)(AU) $\geq$ M · Compare set, Y $\longrightarrow$ P 2 microseconds
65	JUMP AL POSITIVE (JPALP) JUMP AL EQUAL OR GREATER THAN M (JPMLEQ)	If (AL) Pos, · Compare not set, $\oplus$ (AL) $\geq$ M · Compare set, $\oplus$ L(AL)(AU) $\geq$ M, · Compare set, Y $\longrightarrow$ P

TABLE 1.2-7. CONDITIONAL JUMPS (CONT.)

Execution time: 2 microseconds
$y = u_P$
Jump to y, for example, set P = y, if:

1) Compare stage of comparison designator is not set and (AL) ≥ 0.

2) 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)
JUMP AL LESS THAN M (JPMGR)

If (AU) Neg · Compare not set, ⊕ (AL) < M · Compare set, ⊕ L(AL)(AU) < M · Compare set, Y ⟶ P

Execution time:
$y = u_P$
Jump to y, for example, set P = y, if:

1) Compare stage of comparison designator is not set and (AU) less than zero.

or

2) 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)
JUMP AL LESS THAN M (JPMGR)

If (AL) Neg, ⊕ (AL) < M · Compare set, ⊕ L(AL)(AU) < M, · Compare set, Y ⟶ P

Execution time: 2 microseconds
$y = u_P$
Jump to y, for example, set P = y, if:

1) Compare stage of comparison designator is not set and (AL) less than zero.

or

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

Otherwise, execute next instruction.

c) Conditional Jump Features. The arithmetic conditional jump instructions may be used with associated compare instructions to obtain certain results according to the state of the comparison designator or they may be used independently for other results.

The comparison designator is a 3-state bistable register which records the results of a compare instruction (02, 03, 06 and 07) 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 (codes 60-67). Therefore, in order to set the compare stages desired, a compare instruction must immediately precede a single 60-67 instruction, or immediately precede the first of a consecutive string of 60-67 instructions. Otherwise, these jump instructions are executed without reference to the comparison designator.

The arithmetic conditional jump instructions 60-67 are used with or without an associated compare instruction (02, 03, 06 and 07). If used without a preceding compare instruction, the jump is executed upon satisfying the condition directly stated by the instruction. If a compare instruction is used in conjunction with one or more conditional jump instructions, the satisfaction of a jump condition is dependent upon the set or not set state of certain stages of the comparison designator.

Table 1.2-8 shows the jump or no-jump conditions resulting from the combined and separate uses of the compare and arithmetic conditional jump instructions. The compare instructions use the following operands for comparison with (AL):(y) for 02; (y + B) for 03; for comparison with L(AL)(AU):L(Y)(AU) for 06; and L(y + B)(AU) for 07.

5. Arithmetic Type Instructions. See Table 1.2-9.

*Note all compares are algebraic. Therefore when comparing a portion of or an entire word the sign of that computer word must be considered.

TABLE 1.2-8. JUMP OR NO-JUMP CONDITIONS

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 if (AL) = M	No JP	*	*	(AU) = 0 or (AL) = M
61	JP if (AL) = 0	JP if (AL) = M	No JP	*	*	(AL) = 0 or M
62	JP if (AU) ≠ 0	No JP	JP if (AL) ≠ M	*	*	(AU) ≠ 0 or (AL) ≠ M
63	JP if (AL) ≠ 0	No JP	JP if (AL) ≠ M	*	*	(AL) ≠ 0 or M
64	JP if (AU) ≥ 0	*	*	JP if (AL) ≥ M	No JP	(AU) POS. or (AL) ≥ M
65	JP if (AL) ≥ 0	*	*	JP if (AL) ≥ M	No JP	(AL) POS. or (AL) ≥ M
66	JP if (AU) ≥ 0	*	*	No JP	JP if (AL) < M	(AU) NEG. or (AL) < M
67	JP if (AL) < 0	*	*	No JP	JP if (AL) < M	(AL) NEG. or (AL) < M

*Does not apply, and the next sequential instruction is executed.

**(AL) = (AL) if compared as a result of 02, 03 instructions.
M = (y)
(AL) = L(AU)(AL) if compared as a result of 06, 07 instructions
M = L(AU)(Y)

TABLE 1.2-9. ARITHMETIC TYPE INSTRUCTIONS

14 ADD AL (ADDAL) — (AL)+(Y) ⟶ AL

Execution time: 4 microseconds

y = up 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) ⟶ AL

Execution time: 4 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) ⟶ AL

Execution time: 4 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.

17 SUBTRACT AL (SUBALB) — (AL) - (Y) ⟶ AL

Execution time: 4 microseconds

y = up 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 + L, Y) ⟶ A

Executive time: 6 microseconds

y = up 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 in y + 1. The sign of the double length number is indicated by the most-significant bit of (y + 1). Address y must be even; for example, the right-most octal digit must be 0, 2,4, or 6.

NOTE

The instruction is executed in the following manner. The AU and AL registers are linked to form a continuous 36-bit A register. Any

*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).

TABLE 1.2-9. ARITHMETIC TYPE INSTRUCTIONS (CONT.)

borrow required by AL comes from AU; any end-around-borrow required by 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 - The result may be incorrect since the addition of some numbers results in an end-around borrow. Since it is blocked, the result will be 1 larger than it should be (as in the example).

21 ADD A (ADDAB) (A) + (Y + 1, y)→A
Execution time: 6 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 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, instruction 20.)

22 SUBTRACT A (SUBA) (A) - (Y + L, Y)→A
Execution time: 6 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 designator if overflow occurs.* The least-significant half is in cell y and the most-significant half in cell y + 1. The sign of the double length number is the sign of (y + 1). Address y must be even. The computer executes subtract A in a manner analogous to the add A instruction. (See NOTE, 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).

TABLE 1.2-9. ARITHMETIC TYPE INSTRUCTIONS (CONT.)

23 SUBTRACT A (SUBAB) (A) - (Y + 1, Y)→A
Execution time: 6 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 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, instruction 20.)

24 MULTIPLY AL(MULAL) (AL) x (Y)→A
Execution time: 14 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 values of the factors, then corrected for algebraic sign.

25 MULTIPLY AL (MULALB) (AL) x (Y)→A
Execution time: 14 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); Quote→AL, Rem——AU
Execution time: 14 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.* 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).

TABLE 1.2-9. ARITHMETIC TYPE INSTRUCTIONS (CONT.)

27 DIVIDE A (DIVAB) — (A) ÷ (Y); Quot → AL, Rem → AU

Execution time: 14 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.)

71 ADD CONSTANT TO AL (ADDALK) — (AL) + xY → AL

Execution time: 2 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)

50:60 ROUND AU (RND) — If (AU) positive, (AU) + $(AL)_{17}$ → AL

If (AU) negative, (AU) - $(AL)'_{17}$ → AL

Execution time: 2 microseconds

If (AU) is positive, add bit position 17 of AL to (AU); if (AU) is negative, subtract the complement 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. Set Overflow FF if overflow occurs.

6. Logical Type Instructions.

a) Format 1 Instructions. See Table 1.2-10.

TABLE 1.2-10. FORMAT 1 LOGICAL TYPE INSTRUCTIONS

00 ILLEGAL CODE - Jump to fault entrance register, address 0 or address 500 (depending upon position of AUTO RECOVERY switch).
Execution time: 2 microseconds

01 ILLEGAL CODE - Jump to fault entrance register, address 0 or address 500 (depending upon position of AUTO RECOVERY switch).
Execution time: 2 microseconds

02 COMPARE AL (CMAL) (AL):(Y)
Execution time: 4 microseconds
Y = up or u_{SR}
Algebraically compare (AL) with (y) and set the comparison designator as follows:

1) Set the compare stage
2) Set the greater-than stage if (AL) > (Y)
3) Set the equals stage if (AL) = (Y)

$(AL)_f = (AL)_i$

NOTE

The comparison designator is cleared by the execution of any subsequent instruction other than codes 60-67, and no interrupt will be honored while the designator is set.

03 COMPARE AL (CMALB) (AL):(Y)
Execution time: 4 microseconds
y = up or u_{SR}
Algebraically compare (AL) with (y + B) and set the comparison designator as follows:

1) Set the compare stage
2) Set the greater-than stage if (AL) > (y + B)
3) Set the equals stage if (AL) = (y + B)

$(AL)_f = (AL)_i$

NOTE

The comparison designator is cleared by the execution of any subsequent instruction other than codes 60-67, and no interrupt will be honored while the designator is set.

04 SELECTIVE SUBSTITUTE (SLSU) L(AU)'(AL) + L(AU)(y) → AL or $(Y)_n \rightarrow AL_n$ for $(AU)_n = 1$
Execution time: 4 microseconds
y = up 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

TABLE 1.2-10. FORMAT 1 LOGICAL TYPE INSTRUCTIONS (CONT.)

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

Execution time: 4 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.

$(AU)_f = (AU)_i$

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

Execution time: 4 microseconds

$y = u_P$ or u_{SR}

Algebraically compare selected bits of (AL) with corresponding bits of (y) and set comparison designator as follows:

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

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

NOTE

The comparison designator is cleared by the execution of any subsequent instruction other than codes 60-67, and no interrupt will be honored while the designator is set.

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):(AU)(Y)

Execution time: 4 microseconds

$y = u_P$ or u_{SR}

Algebraically compare selected bits of (AL) with corresponding bits of (y + B), and set the comparison designator as follows:

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

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

NOTE

The comparison designator is cleared by the execution of any subsequent instruction other than codes 60-67, and no interrupt will be honored while the designator is set.

51 SELECTIVE SET (SLSET) $(AL) \vee (Y) \longrightarrow AL$ or set $(AL)_n$ for $(Y)_n = 1$

Execution 4 microseconds

$y = u_P$

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

TABLE 1.2-10. FORMAT 1 LOGICAL TYPE INSTRUSTIONS (CONT.)

Example of selective set:

$(AL)_i$ = 123456

(y) = 000077

$(AL)_f$ = 123477

52 SELECTIVE CLEAR (SLCL) L(AL)(Y) ⟶ AL or clear $(AL)_n$ for $(Y)_n = 0$

Execution time: 4 microseconds

y = up

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.

Example of selective clear:

$(AL)_i$ = 123456

(y) = 707070

$(AL)_f$ = 103050

53 SELECTIVE COMPLEMENT (SLCP) (AL) ⊕ (Y) ⟶ AL or complement $(AL)_n$ for $(Y)_n = 1$

Execution time: 4 microseconds

y = up

Complement the individual bits of (AL) corresponding to ones in (y), leaving the remaining bits of (AL) unaltered, for example, 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

50:61 COMPLEMENT AL (CPAL) (AL)' ⟶ AL

Execution time: 2 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)' ⟶ AU

Execution time: 2 microseconds

Complement (AU), leaving the result in AU; ignore k. (See NOTE, instruction 50:61.)

50:63 COMPLEMENT A (CPA) (AL)' ⟶ A

Execution time: 2 microseconds

Complement (A), leaving the result in A; ignore K. (See NOTE, instruction 50:61.)

b) Examples of logical instructions. The selective type instructions are explained more fully on the following pages.

SELECTIVE CLEAR
(LOGICAL PRODUCT) L(AL)(Y) $\longrightarrow$ AL

The most common use of the logical product in programing is an operation commonly referred to as "masking". Masking is the process of lifting out a selected portion of an operand and leaving undesired parts behind. This is done by placing a "mask" in a core location, consisting of ones in the desired bit positions, and zeros in the others. The logical product will then contain only bits from the other operand corresponding to the ones in the core location, the other bits being zero. The following example, using 12-bit registers, shows a masking of alternate octal digits from an operand in AL.

mask in Y = 111 000 111 000
(AL) = 001 010 011 100
L(AL)(Y) = 001 000 011 000

SELECTIVE SET A_n v Y_n = INCLUSIVE OR

The selective set instructions are used to force ones into selected bit positions of an operand contained in AL. If a one is already in the selected bit position, it will remain. The following example adds four to each octal digit 4 in AL.

(AL) = 001 010 011 100
(Y) = 100 100 100 100
(AL) v (Y) = 101 110 111 100

SELECTIVE COMPLEMENT (AL) + (Y) EXCLUSIVE OR

The selective complement instructions are used to complement selected bits of an operand contained in AL. In the following example, alternate octal digits of an operand in AL are complemented.

(AL) = 100 101 110 111
(Y) = 111 000 111 000
(AL) + (Y) = 011 101 001 111

SELECTIVE SUBSTITUTE L (AU) · (AL) + L(AU) · Y

The selective substitute instructions provide for replacing selected bits of an operand contained in AL with corresponding bits of an operand Y. The bits to be substituted are specified by ones in AU. In the following example, alternate octal digits of an operand in AL are replaced by the corresponding octal digits of an operand in AL are replaced by the corresponding octal digits in an operand Y.

(AU) = 000 111 000 111
($\overline{AU}$) = 111 000 111 000
(AL) = 001 101 011 110
(Y) = 110 010 101 100

} initial conditions

Our select bits in AU indicate that we wish to replace the second and fourth octal digits, (counting from the left), of (AL) with the corresponding octal digits of Y, yielding a result = 001 010 011 100.

First:	$(\overline{AU})$	= 111 000 111 000
	(AL)	= 001 101 011 110
	$L(\overline{AU}) \cdot (AL)$	= 001 000 011 000
Second:	(AU)	= 000 111 000 111
	(Y)	= 110 010 101 100
	L(AU) · (Y)	= 000 010 000 100

Finally:	$L(\overline{AU}) \cdot (AL)$	= 001 000 011 000
	L(AU) · (Y)	= 000 010 000 100
$L(AU) \cdot (AL) + L(AU) \cdot (Y)$		= 001 010 011 100

Note that the notation $L(AU)' \cdot (AL)$ v $L(AU) \cdot (Y)$ also is logically correct for Selective Substitute.

7. Shift Type Instructions. See table 1.2-11.

8. Skip Type Instructions.

   a) Arithmetic. See table 1.2-12.

   b) I/O (Skips). See table 1.2-13.

9. Input/Output Type Instructions. See table 1.2-14.

10. Program Stop and Fault Type Instructions. See table 1.2-15.

TABLE 1.2-11. SHIFT TYPE INSTRUCTIONS

50:41 RIGHT SHIFT AU (RSHAU)

Execution time: 2 microseconds (k = 0);
4-10 microseconds (k ≠ 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, for example, 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: 2 microseconds (k ≠ 0);
4-10 microseconds (k ≠ 0)*

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

50:43 RIGHT SHIFT A (RSHA)

Execution time: 2 microseconds (k = 0);
4-20 microseconds (k ≠ 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; for example, the low-order bits are lost upon completion of the shift.

Example of right shift A with k = 2:

	AU	AL
$(A)_i$ (positive)	= 370000	000000
After first shift	= 174000	000000
After second shift	= 076000	000000
$(A)_i$ (negative)	= 460000	000000
After first shift	= 600000	000000
After second shift	= 700000	000000

50:44 SCALE FACTOR (SF)

Execution time: 4 microseconds (k = 0);
4-20 microseconds (K ≠ 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.

TABLE 1.2-11. SHIFT TYPE INSTRUCTIONS (CONT.)

1) 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 - shift count (7-1) = the 18-bit quantity 000006 00017.

2) Example of scale factor with k = 3:
$(A)_i$ = 600000 000000 (negative, not normalized)
After first shift = 400000 000000 (negative, normalized)

The computer then stores the quantity 000002 $\longrightarrow$ 00017.

3) Example of scale factor with k = 1:
$(A)_i$ = 070000 000000 (positive, not normalized)
After first shift = 160000 000000 (positive, not normalized)

The computer, having exhausted k, stores the quantity 000000 $\longrightarrow$ 00017 leaving (A) only partially normalized.

50:45 LEFT SHIFT AU (LSHAU)
Execution time: 2 microseconds (k = 0);
4-10 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: 2 microseconds (k = 0);
4-10 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.

50:47 LEFT SHIFT A (LSHA)
Execution time: 2 microseconds (k = 0);
4-20 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) = 300000 000000
After first shift = 600000 000000
After second shift = 400000 000001

*Execution time for shifting is 2 microseconds for every 4 shifts after shifting has been initiated-therefore execution time for shifting any of the registers could be a maximum of 34 microseconds.

TABLE 1.2-12. ARITHMETIC SKIP TYPE INSTRUCTIONS

50:51 SKIP ON NO BORROW (SKPNBO)
Execution time: 2 microscconds skip or no skip
If the last previous add A or subtract A required a borrow, take the 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.
This instruction clears the borrow designator.

50:52 SKIP ON OVERFLOW (SKPOV)
Execution time: 2 microseconds skip or 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: 2 microseconds skip or no skip
If an overflow condition did not occur on the 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: 2 microseconds skip or 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 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: 2 microseconds skip or 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.

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

TABLE 1.2-13. I/O SKIP TYPE INSTRUCTIONS

50:21 SKIP ON INPUT INACTIVE (SKPIIN)
Execution time: 2 microseconds skip or 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: 2 microseconds skip or 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 (SKPEIN)
Execution time: 2 microseconds skip or no skip
Test for external function activity on channel k. If inactive, skip the next instruction; otherwise, take the next instruction.

50:50 SKIP ON KEY SETTING (SKP)
Execution time: 2 microseconds skip or 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, 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 4)	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.

50:57 SKIP ON NO RESUME (SKPNR)
Execution time: 2 microseconds skip or no skip
If the resume designator is not set (indicating unsuccessful transfer of a word to an output device), skip the next sequential instruction; otherwise take the next instruction.

TABLE 1.2-14. INPUT/OUTPUT TYPE INSTRUCTIONS

50:01 SET INPUT ACTIVE (SIN)
Execution time: 2 microseconds
Set input channel k to the active state. The buffer control words stored in memory locations 60 + 2k and 61 + 2k or as specified by the externally specified index or externally specified address will control the transfers.

50:02 SET OUTPUT ACTIVE (SOUT)
Execution time: 2 microseconds
Set output channel k to the active state. The buffer control words stored in memory locations 40 + 2k and 41 + 2k or as specified by the ESI or ESA will control the transfers.

50:03 SET EXTERNAL FUNCTION ACTIVE (SEXF)
Execution time: 2 microseconds
Set channel k external function mode active. The buffer control words stored in memory locations 40 + 2k and 41 + 2k will control the transfers.

50:11 INPUT TRANSFER (IN)
(P + 1)→61 + 2k (240 + 2k)*
(P + 2)→61 + 2k (241 + 2k)*
Set input active on channel k.
Execution time: 6 microseconds
Initiate input transfer mode on channel k.

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

50:12 OUTPUT TRANSFER (OUT)
(P + 1)→40 + 2k (220 + 2k)*
(P + 2)→41 + 2k (221 + 2k)*
Set output active on channel k.
Execution time: 6 microseconds
Initiate output transfer mode on channel k.

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

50:13 EXTERNAL FUNCTION (EXF)
(P + 1)→40 + 2k (220 + 2k)*
(P + 2)→41 + 2k (221 + 2k)*
Set external function active on channel k.
Execution time: 6 microseconds
Initiate external function mode on channel k.

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

*Buffer control registers for channels 8-15.

TABLE 1.2-14. INPUT/OUTPUT TYPE INSTRUCTIONS (CONT.)

50:14 ENABLE REAL-TIME CLOCK MONITOR (RTC)
Execution time: 2 microseconds
Enable the real-time clock monitor interrupt; ignore k. After execution of this instruction, equality between the RTC register (location 15) and the RTC monitor word register (location 14) will interrupt the computer program. The next instruction is taken from the RTC monitor interrupt entrance register (location 12) and the RTC monitor is disabled.

50:15 TERMINATE INPUT (INSTP) Clear input active channel k.
Execution time: 2 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 channel k.
Execution time: 2 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 channel k.
Execution time: 2 microseconds
Terminate external function on channel k.

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

50:20 SET RESUME (SRSM)
Execution time: 2 microseconds
Set the resume designator to permit honoring the next requesting output function. Loss of any information currently held by the output register(s) for a peripheral device is allowed by this instruction.

40:24 WAIT FOR INTERRUPT (WTFI)

or

50:25 Execution time: 2 microseconds
Stop the computer until any interrupt occurs; ignore k, then execute the instruction located in the interrupt entrance register designated by the interrupt.

50:26 OUTPUT OVERRIDE (OUTOV)
Execution time: 2 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

TABLE 1.2-14. INPUT/OUTPUT TYPE INSTRUCTIONS (CONT.)

will transfer a word whether the buffer is active or not. Also since the transfer takes place under control of the word in the buffer control register the two buffer control words must not be equal.

50:27 EXTERNAL FUNCTION OVERRIDE (EXFOV)
Execution time: 2 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. Also since the transfer takes place under control of the word in the buffer control register the two buffer control words must not be equal.

50:30 REMOVE INTERRUPT LOCKOUT (RIL)

or

50:31 Execution time: 2 microseconds
Remove the interrupt lockout - enable all interrupts, all channels; ignore k.

NOTE

A 50:30 or 50:31 instruction must be used in conjunction with a 50:34 or 50:35 instruction. It will not affect a 50:36 or 50:37 instruction.

50:32 REMOVE EXTERNAL INTERRUPT LOCKOUT (RXL)

or

50:33 Execution time: 2 microseconds
Remove the external interrupt lockout - enable external interrupts, all channels; ignore k.

NOTE

A 50:32 or 50:33 instruction must be used in conjunction with a 50:36 or 50:37 instruction. It will not affect a 50:34 or 50:35 instruction.

50:34 SET INTERRUPT LOCKOUT (SIL)

or

50:35 Execution time: 2 microseconds
Set the interrupt lockout - disable all interrupts, all channels; ignore k.

50:36 SET EXTERNAL INTERRUPT LOCKOUT (SXL)

or

50:37 Execution time: 2 microseconds
Set the external interrupt lockout - disable external interrupts, all channels; ignore k.

TABLE 1.2-15. PROGRAM STOP AND FAULT TYPE INSTRUCTIONS

50:56 STOP ON KEY SETTING (STOP)

Execution time: 2 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 one, 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.

00:01 ILLEGAL CODE - Jump to fault entrance register, address 0, or address 500 (depending on position of AUTO RECOVERY switch).

Execution time: 2 microseconds

77 ILLEGAL CODE - Jump to fault entrance register, address 0, or address 500 (depending on position of AUTO RECOVERY switch).

50:00 ILLEGAL CODE - Jump to fault entrance register, address 0, or address 500 (depending on position of AUTO RECOVERY switch).

Execution time: 2 microseconds

50:77 ILLEGAL CODE - Jump to fault entrance register, address 0, or address 500 (depending on position of AUTO RECOVERY switch).

Execution time: 2 microseconds

NAME _____

1.2-5. STUDY QUESTIONS

a. What are the two types of instructions?

b. What is the purpose of the SR and ICR registers?

c. What memory location is referenced during a B modification if the ICR register contains all zeros?

d. What is the main difference between a 54 and 55 instruction?

e. How are B modified instructions indicated?

f. Explain each of the instructions in the following list which starts at P = 3000. This is a nonsense program which emphasizes the use of the SR & ICR registers.

ADDRESS	CONTENTS	MNEMONIC	EXPLANATION
000001	00 0000		
000002	00 0002		
002773	00 0004		
002774	12 3456		
002775	77 0000		
002776	00 0000		
002777	00 0001		
003000	50 7201		
003001	36 0000		
003002	70 0200		
003003	14 2777		
003004	44 2776		
003005	50 7333		
003006	13 3400		
003007	71 0020		
003010	45 2000		
003011	56 2773		
003012	34 3006		
003013	75 6001		
003014	10 2775		
003015	06 2774		
003016	63 6000		

NAME: _____

f. (Cont.)

ADDRESS	CONTENTS	MNEMONIC	EXPLANATION
003017	50 5601		
.			
.			
.			
005777	00 0000		
006000	44 5777		
006001	50 7300		
006002	02 3401		
006003	61 6005		
006004	50 5602		
006005	50 7202		
006006	73 3000		
006007	50 5640		
.			
.			
.			
.			
132000	00 0000		
.			
.			
.			
.			
132004	00 0000		
.			
.			
.			
133400	00 0300		
133401	12 4736		
133402	00 1600		
133403	00 0020		
133404	00 0176		

g. Explanation each of the following set of instructions. This is a non-sense program which emphasizes I/O instruction.

ADDRESS	CONTENTS	EXPLANATION
063200	50 1306	
063201	17 7010	
063202	17 7010	
063203	50 2306	
063204	34 3203	
063205	50 1317	
063206	17 7007	
063207	17 7007	
063210	50 2717	
063211	50 1106	
063212	54 3000	
063213	34 2400	
063214	50 1217	
063215	40 1000	
063216	60 1500	
063217	50 5640	

NAME: _____

h. Write a program to count the total number of 1 bits in any of five adjacent memory cells. Put the total in AU and stop. The first address of the five should be put in A before the program is exhausted.

i. Transfer a block of data from one section of memory to another. For example, (1000 1075) to, (143600 143675).

j. Sum a group of numbers in 20 adjacent memory cells, (037150 037170) and take their average. Put the average in AL, the total in AU and stop. If overflow occurs during summation, handle it as a subroutine which could be expanded later. In this problem just put a stop on a key setting in the subroutine.

SECTION 1 - UNIVAC 1219 COMPUTER FUNCTIONAL DESCRIPTION

1.3. OPERATOR'S INFORMATION

1.3-1. OBJECTIVES

To familiarize the student with the switches and controls that affect the 1219 computer operation.

1.3-2. INTRODUCTION

The 1219 computer is basically an automatic machine; however, there are certain switches and controls that affect computer operation.

1.3-3. REFERENCES

PX 3316, Vol. I and Vol. II.

1.3-4. INFORMATION

a. General. A computer is basically an automatic machine; however, there are certain switches and controls that affect computer operation. The switches and controls provide a means of setting and clearing the registers, selecting computer operating speeds, selecting optional program jumps or stops, and selecting input/output modes. On the 1219 computer, these controls are on the front panels of each drawer. The panels are designated as input/output panel (A1 or optional A9), control panel 1 (A2), memory panel (A3 or optional A9), control panel 2 (A4), and power control panel (A5). These panels are shown in figures 1.3-1 through 1.3-4.

The computer register and control flip-flops are associated with indicator/switches on the operator panels. These indicator/switches serve a dual function. That is, the indicator comes on when the associated flip-flop is in the set or 1 condition. Also, pressing the indicator places the associated flip-flop in the set or 1 condition. The indicators are normally intended to provide a visual indication of status and operation. As such, they are intended primarily for maintenance personnel.

b. Description of Controls. Tables 1.3-1 through 1.3-5 list the switches and indicator/switches of the computer with a functional description of each.

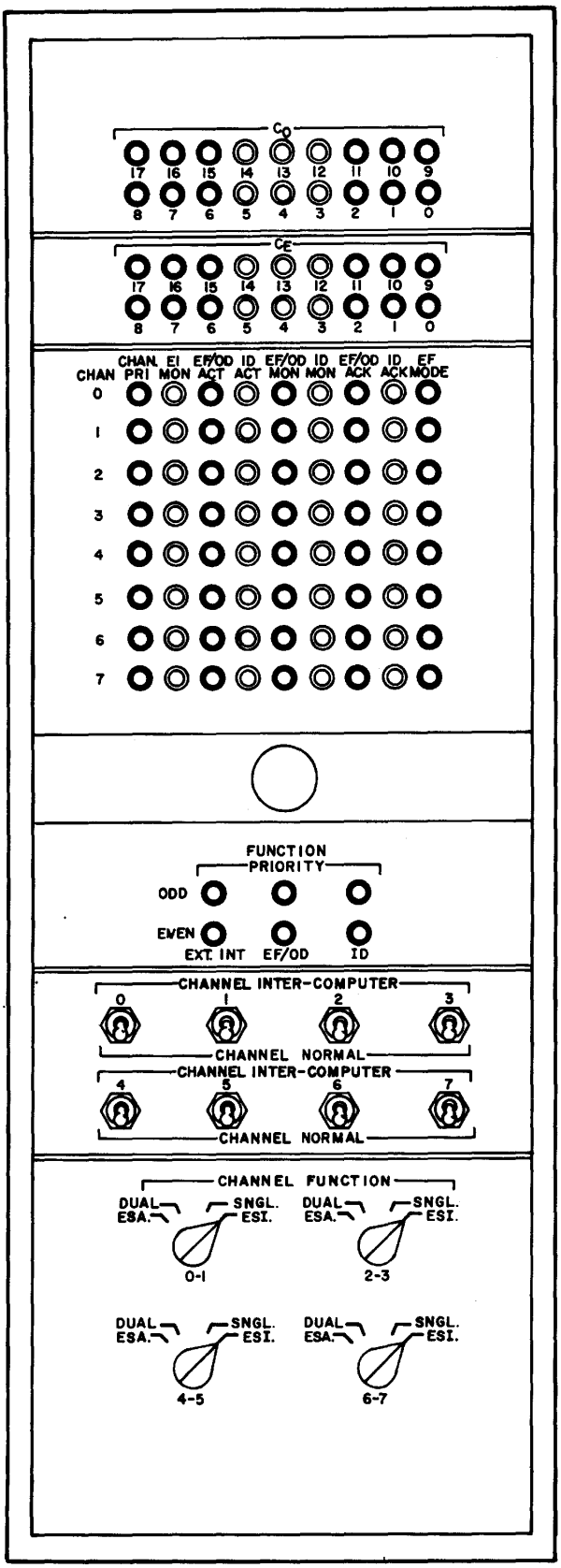


Figure 1.3-1. Input/Output Panel (A1 or optional A9)

TABLE 1.3-1. INPUT/OUTPUT PANEL, A1 OR OPTIONAL A9 (FIGURE 1.3-1)

TITLE	FUNCTION/USE
C_O 0-17	Displays the contents and allows manual control of the 18 bits of the communications register common to all odd-numbered output channels. It contains the upper 18 bits of the word in the dual channel mode.
C_E 0-17	Displays the contents and allows manual control of the 18 bits of the communications register common to all even-numbered output channels. It contains the lower 18 bits of the word in the dual channel mode.
I/O Channel and Status Grid	The indicator/switches indicate the function and channel of that coordinate.
CHAN 0-7	Channels 0 through 7 or 10 through 17.
CHAN PRI	The indicated channel is requesting priority.
EI MON	An external interrupt monitor has been detected and is being processed.
EF/OD ACT	The external function or output data mode has been made active (able to communicate).
ID ACT	The input data mode has been made active.
EF/OD MON	An external function or output data monitor has been detected and is being processed.
ID MON	An input data monitor has been detected and is being processed.
EF/OD ACK	An external function or output data acknowledge signal has been placed on an output control line.
ID ACK	An input data acknowledge signal has been placed on an output control line.
EF MODE	External function mode; when on, it indicates the EF mode for any EF/OD function; when off, it indicates the OD mode.
	NOTE Pressing the above indicator/switches except CHAN PRI will set the associated flip-flops.
FUNCTION PRIORITY Grid	The indicator/switches indicate the function and channel group of the coordinate.
ODD	The odd-numbered I/O channels
EVEN	The even-numbered I/O channels

TABLE 1.3-1. INPUT/OUTPUT PANEL, A1 OR OPTIONAL A9 (FIGURE 1.3-1) (CONT.)

TITLE	FUNCTION/USE
EXT INT	An external interrupt request for priority has been made or is being processed.
EF/OD	An external function or output data request has been made or is being processed.
ID	An input data request for priority has been made or is being processed. A request for priority cannot appear simultaneously on the odd and even channel group. NOTE Pressing the FUNCTION PRIORITY indicator/ switches will set the associated flip-flops.
CHANNEL INTER-COMPUTER/CHANNEL NORMAL 0-7	Up position: Enables the corresponding numbered input and output channels to be used as an intercomputer channel. Down position: Allows normal use of that numbered channel for other peripheral equipment.
CHANNEL FUNCTION Switches 0-1 2-3 4-5 6-7	Allows the two channels selected to operate in the externally specified address mode (dual mode forced).
ESA	
DUAL	Connects adjacent input/output channels so they operate in double length (36-bit) mode; for example, switch 0-1 in DUAL position allows channels 0 and 1 to operate in the dual mode.
SINGLE	Allows I/O channels to operate in single (18-bit) mode.
ESI	Allows the two channels selected to operate in the externally specified index mode (dual mode forced).

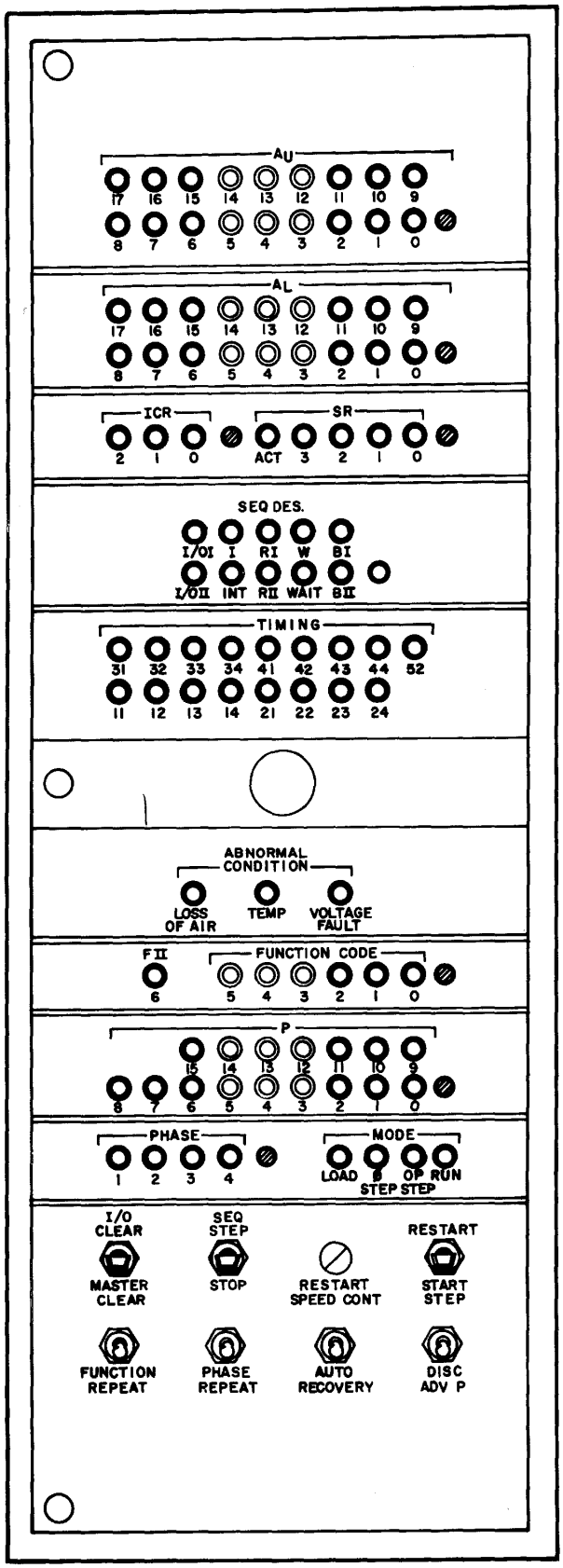


Figure 1.3-2. Control Panel 1 (A2)

TABLE 1.3-2. CONTROL PANEL 1 (A2) (FIGURE 1.3-2)

TITLE		FUNCTION/USE
A_U	0-17 and clear	Displays the contents and allows manual control of the A_U register (the upper eighteen bits of the A register). Each bit may be set by pressing the appropriate indicator/ switch. The clear button clears all 18-bit positions.
A_L	0-17 and clear	Displays the contents and allows manual control of the A_L register (the lower eighteen bits of the A register). Each bit may be set by pressing the appropriate indicator/switch. The clear button clears all 18-bit positions.
ICR (Index Control Register)	0-2 and clear	Indicates which of eight index registers is to be used as a modifier. Clear button clears entire register.
SR (Special)	bits 0-2 and 4 and clear	A four-bit register used to specify the memory bank currently being used. The clear button clears the register.
	ACT bit 2^3	Comes on when SR is active. When cleared, SR is inactive.
SEQ DES	I/O 1 (I/O I on front panel)	Indicates the performance of the first I/O sequence.
	I/O 2 (I/O II on front panel)	Indicates the performance of the second I/O sequence.
	I	Indicates the performance of the instruction sequence which is common to all instructions. Manually selecting the I-sequence clears all other sequences.
	INT	Indicates the performance of the interrupt sequence.
	R 1 (R I on front panel)	Indicates the performance of the first R (read) sequence.
	R 2 (R II on front panel)	Indicates the performance of the second R (read) sequence.
	W	Indicates the performance of the W (write) sequence.
	WAIT	Indicates the performance of the wait sequence (for example, the computer waits for an interrupt.)
	B 1 (B I on front panel)	Indicates the performance of the first B-sequence which reads the TACW and stores it in the control memory.
	B 2 (B II on front panel)	Indicates the performance of the second B-sequence which reads the IACW and stores it in the control memory.

TABLE 1.3-2. CONTROL PANEL 1 (A2) (Figure 1.3-2) (CONT.)

TITLE		FUNCTION/USE
		NOTE All of the sequence designator indicators can be manually set.
TIMING 11-14 21-24 31-34 41-44 and 52		Indicates the setting of the main timing cycle flip-flops, T11 through T52. Thus, as the indicators come on and go off the progression of the cycle time of the computer is indicated. Indicator 52 comes on only during the 50:61 and 50:63 instructions.
ABNORMAL CONDITION	LOSS OF AIR	This indicator comes on when the air flow sensor detects the absence of air circulation.
	TEMP	This indicator comes on when either low temperature thermostat detects an internal air temperature higher than 115°F (46°C).
	VOLTAGE FAULT	This indicator comes on when any of the logic voltage (±5V, -3V) or memory voltage (±10V) fluctuates outside of the preset limits.
F II (Format 2) 6		Bit 2^6 indicator of the function code register indicates the function code is of the Format 2 type (for example, 50:XX). May be manually set or cleared.
FUNCTION CODE 0-5 and clear		Bit 2^0-2^5 indicators of the function code register octally display the function code in the F-register. All six bits may be manually set or cleared.
P 0-15 and clear		Displays the contents and allows manual control of the sixteen bits of the program address register. Each bit may be set by pressing the appropriate indicator/switch. The clear button clears all 16 bit positions.
MODE	LOAD	Indicates the load mode. Pressing the MODE LOAD button sets the load mode, clears all other modes, and forces a jump to address 00500 for loading at high speed.
	PHASE STEP	Indicates the phase step mode. Pressing the MODE PHASE STEP button sets the phase step mode, clears all other modes, and enables the PHASE indicator/switches and computer for clock phase operation. Inhibits memory.

TABLE 1.3-2. CONTROL PANEL 1 (A2) (Figure 1.3-2) (CONT.)

TITLE		FUNCTION/USE
MODE (CONT.)	OP STEP	Indicates the operation step mode. Pressing the MODE OP STEP button sets the operation step mode, clears all other modes, and enables the computer for execution of one operation at a time. This will be one instruction or one sequence depending upon the position of the SEQ STEP/STOP switch.
	RUN	Indicates the run mode. Pressing the MODE RUN button sets the run mode, clears all other modes, and enables the computer for normal high speed operation.
PHASE	1-4 and clear	Indicates the phase selected. Selecting the one phase by pressing the indicator/switch enables computer to issue phase pulses in conjunction with the phase step mode. Phase pulses are issued individually beginning with the selected phase. Pressing the clear button clears all four phases.
I/O CLEAR/MASTER CLEAR Switch		Center position: Neutral. Momentary up position (I/O CLEAR): Clears I/O section of computer only and sets all stages of channel priority. Momentary down position (MASTER CLEAR): With computer not running and not in phase stop mode, clears all sections, registers, and control flip-flops of central computer. With computer running, it clears the VOLTAGE FAULT and/or PROGRAM FAULT indicators only.
SEQ STEP/STOP Switch		Center position: Neutral. Momentary up position (SEQ STEP): Enables execution of a single sequence in conjunction with the operation step mode. Momentary down position (STOP): Stops high speed operation of the computer; RUN indicator goes off.
RESTART SPEED CONT		Varies the speed of the low-speed oscillator.
RESTART/START STEP Switch		Center position: Neutral. Momentary up position (RESTART): Enables the selected mode to be executed at the restart speed control rate setting. Momentary down position (START/STEP): Load mode selected - initiates high-speed start at address 00500. Run mode selected - initiates high-speed start at address designated in P. Phase step mode selected - initiates the issuance of the phase or phases indicated by the PHASE indicators.

TABLE 1.3-2. CONTROL PANEL 1 (A2) (Figure 1.3-2) (CONT.)

TITLE	FUNCTION/USE
RESTART/START STEP Switch (CONT.)	Op step mode selected - initiates execution of one instruction or one sequence, depending upon the position of the SEQ STEP/STOP switch.
FUNCTION REPEAT Switch	Up position: Forces a repeat of the instruction in the F-register at mode rate, and disables the clearing of the S-register in any sequence other than I, except in the W-sequence of instruction 50:10 - 50:13.
PHASE REPEAT Switch	Up position: Forces the repeat, at high speed, of the phase or phases selected by the PHASE indicators. (Phase step mode must be selected.)
AUTO RECOVERY Switch	Up position: Computer fault results in a jump to address 00500. Down position: Computer fault results in a jump to address 00000.
DISC ADV P Switch	Up position: Inhibits incrementing the P-register.

TABLE 1.3-3. MEMORY PANEL (A3 OR OPTIONAL A9)

TITLE	FUNCTION/USE
MARGINAL CHECK Switches	Up position (HIGH): Places a high bias on memory sense amplifier circuits. Center position (NORMAL): Places a normal bias on memory sense amplifier circuits. Down position (LOW): Places a low bias on memory sense amplifier circuits. Left-hand switch affects the upper 16K addresses. Right-hand switch affects the lower 16K addresses.

TABLE 1.3-4. CONTROL PANEL 2 (A4) (Figure 1.3-3)

TITLE	FUNCTION/USE
PROGRAM STOP indicators 0-5	Comes on when a program stop occurs as a result of a 50:56 instruction. Indicator 5 lights for an unconditional stop; the rest are dependent upon the stop switches.
PROGRAM STOP Switches 0-4	On position (up): Enables a program stop on switch setting for a 50:56 instruction if the corresponding bit of the instruction is a binary one.
PROGRAM SKIP Switches 0-4	On position (up): Enables a skip of the next instruction on a 50:50 instruction if the corresponding bit of the instruction is a binary one.
S_1 0-15 and clear	Displays the contents and allows manual control of the 16 bits of the memory address register. Each bit may be set by pressing the appropriate indicator/switch. The clear button clears all bit positions simultaneously.
Z_1 0-17 and clear	Displays the contents and allows manual control of the 18 bits of the main memory exchange register. Each bit may be set by pressing the appropriate indicator/switch. Pressing the clear button clears all 18-bit positions.
B 0-17 and clear	Displays the contents and allows manual control of the 18 bits of the buffer control register. Each bit may be set by pressing the appropriate indicator/switch. The clear button clears all bit positions simultaneously.
K 0-5	Displays the contents and allows manual control of the six-bit register used for shift, multiply, divide, stop, and skip instructions. Must be cleared by MASTER CLEAR switch.
REG +10V/-10V	Manual adjustable potentiometers setting the outputs of the $\pm$10V regulators.
ADV P SEQ 0-2	Indicates the set condition of the advance P sequence flip-flops and the operation of the sequence. Each flip-flop may be set by pressing the appropriate indicator/switch.
INTERRUPT INST	Indicates that an instruction fault (f = 00, 01, 77) has been detected by the fault circuitry, and a fault interrupt address is being generated to the computer.

TABLE 1.3-4. CONTROL PANEL 2 (A4) (Figure 1.3-3) (CONT.)

TITLE		FUNCTION/USE
INTERRUPT (CONT.)	RESUME FAULT	Indicates that the resume signal was not received during a minimum period of one second and a maximum of two seconds after data was placed on an inter-computer channel.*
	RTC MON	Indicates that memory address 15, RTC word, is equal to memory address 14, RTC monitor; and that an interrupt is being generated to inform the computer.*
	RTC OVERFLOW	Indicates that the 18 bits of the RTC word have changed from all binary ones to all binary zeros; and that an interrupt is being generated to inform the computer.*
	SYNC	Indicates that a synchronizing interrupt has been received from a peripheral device and is being processed.
EXT SYNC DISC Switch		Up position: The external synchronizing input is disconnected. Down position: The external synchronizing input may be received.
RTC	SEQ Indicator	Indicates that the operation of updating the real-time clock word is in process.
	DISC Switch	Up position: The RTC interrupting signal is disconnected (disabled). Down position: The RTC interrupting signal is operational (enabled).
I/O TRANSLATOR	ACTIVE	Indicates the performance of any I/O operation or instruction (50:01 - 50:27).
	ESA	Indicates that the current I/O operation is in the externally specified address mode.
	DUAL	Indicates that the current I/O operation is in the dual (Dual channel, 36-bit words) mode.
	ESI	Indicates that the current I/O operation is in the ESI (externally specified index) mode.
FUNCTION	0 and 1	These indicator/switches dipslay the binary value which represents the I/O functions. The values are assigned as follows: 00 - Ext Interrupt 10 - Output 01 - Unassigned 11 - Input
CHANNEL	0-3 and clear	These indicator switches display the octal value of the active I/O channel. Each bit may be set by pressing the appropriate indicator/switch. The Clear button clears all bits of CHANNEL and FUNCTION.
MULT/DIV SEQ	0-6	Indicate the set condition of the multiply, divide, shift, and scale sequence flip-flops and the operation of the sequence. Each flip-flop may be set by pressing the appropriate indicator/switch.

*Dependent on the RTC being operational (RTC DISC switch in the down position).

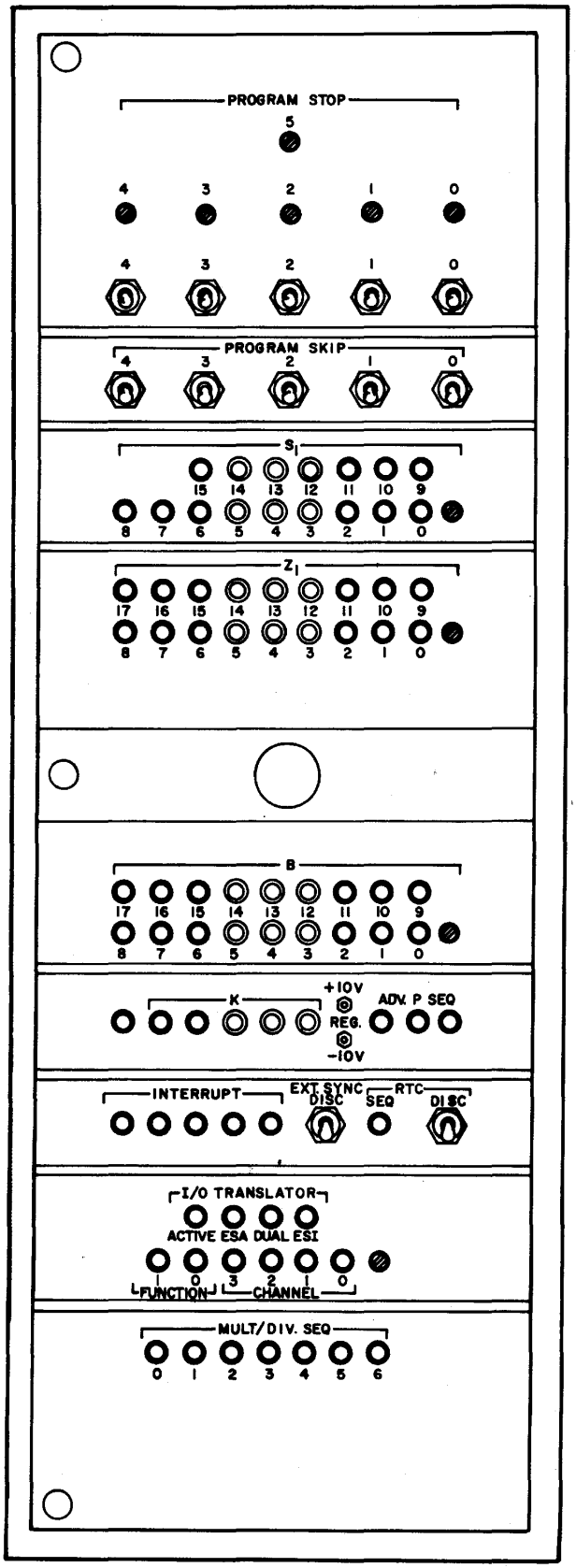


Figure 1.3-3. Control Panel 2(A4)

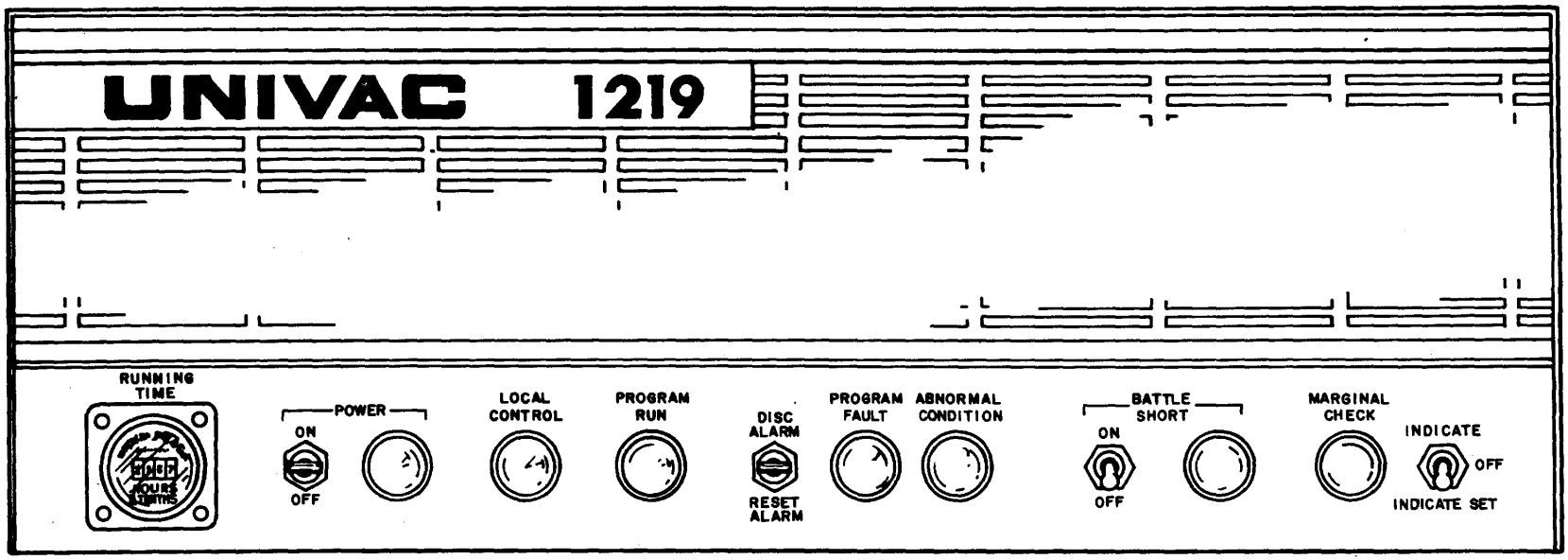


Figure 1.3-4. Power Control Panel (A5)

TABLE 1.3-5. POWER CONTROL PANEL (A5) (FIGURE 1.3-4)

TITLE	FUNCTION/USE
RUNNING TIME METER	Cumulatively records the time that power is supplied to the computer. Range of meter is 0 to 9999.9 hours and cannot be reset.
POWER ON/OFF Switch	Under normal conditions (proper interlock and operating temperatures), when momentarily in the ON position, power is applied to the computer; in the momentary OFF position power is removed.
POWER Indicator	Comes on when power is being applied to the computer.
LOCAL CONTROL Indicator	Comes on when computer may be controlled from the control panels of the computer (not the remote control panel).
PROGRAM RUN Indicator	Comes on when the computer is in any mode of operation other than stop. (For example, the run flip-flop is set.)
DISC ALARM/RESET ALARM Switch	Center position: Allows the horn to sound on program, voltage, blower, or temperature fault. Momentary down position (RESET ALARM): Silences horn. Momentary up position (DISC ALARM): Disables horn for all faults.
PROGRAM FAULT Indicator	Comes on when the computer detects an illegal function code of 00, 01, 77 (format 1) in the F-register. Goes off by setting the I/O CLEAR/MASTER CLEAR switch to the MASTER CLEAR position, regardless if the computer is running or not.
ABNORMAL CONDITION Indicator	Comes on when one of the three abnormal condition indicators, TEMP, LOSS OF AIR, or VOLTAGE FAULT on panel A2 is on.
BATTLE SHORT Switch	ON position: Disables memory protective circuitry, temperature and blower sensor circuitry thus removing all automatic computer shutdown.
BATTLE SHORT Indicator	Comes on when BATTLE SHORT switch is in the ON position.
MARGINAL CHECK Indicator	Comes on when one of the memory MARGINAL CHECK switches is in the HIGH or LOW position or when either CLOCK NARROW/NORMAL switch is in NARROW position.
INDICATE-OFF-INDICATE/ SET Switch	Center position (OFF): Disconnects voltage to all incandescent lamps on all panels. Up position (INDICATE): Supplies voltage to all incandescent lamps on all panels. Down position (INDICATE/SET): Supplies voltage to all incandescent lamps and pushbutton switches on all panels.

c. Manual Reading and Writing. The following procedures for manually reading and writing can be used to test and debug a program. The manual operations principally involve the correction and checking of the stored program being run. Although all of the procedures can be used, it is not intended that they should be rigidly followed for all cases of program correction and checking.

1. Manual Reading from Addresses. Manual reading into the AL register involves the use of the ENTER AL instruction (f = 12). With this instruction the contents of any memory address can be observed by using the following procedures.

STEP 1. Set the SEQ STEP/STOP switch momentarily to the STOP position.

STEP 2. Set the I/O CLEAR/MASTER CLEAR switch momentarily to the MASTER CLEAR position.

STEP 3. Set the P-register equal to one less than the address of the word to be entered into the AL register.

STEP 4. Set the FUNCTION CODE register equal to 12_8.

STEP 5. Set the FUNCTION REPEAT switch to the up position.

STEP 6. Press the MODE OP STEP button.

STEP 7. Set the RESTART/START STEP switch momentarily to the START STEP position (repeat Step 7 for first-time operation).

With Step 7 completed, the computer stops and the desired word is available for inspection in the AL register. To repeat the procedure, perform Step 7 only.

2. Manual Loading (Writing) into Addresses. The procedure for manually entering a word into storage can be varied. One way is with a STORE AL (f = 44) instruction using the following procedures.

STEP 1. Set the SEQ STEP/STOP switch momentarily to the STOP position.

STEP 2. Set the I/O CLEAR/MASTER CLEAR switch momentarily to the MASTER CLEAR position.

STEP 3. Set the P-register equal to one less than the address of the desired storage address.

STEP 4. Set the FUNCTION CODE register equal to 44_8.

STEP 5. Set the FUNCTION REPEAT switch to the up position.

STEP 6. Set the word to be stored in the AL register.

STEP 7. Press the MODE OP STEP button.

STEP 8. Set the RESTART/START STEP switch momentarily to the START STEP position (repeat Step 8 for first-time operation).

With Step 8 completed, the computer stops and the word in the AL register will have been transferred to the desired memory location. To repeat the procedure, perform Step 8 only.

3. Manual Inspect and Change Routine. Many times it is necessary to check the contents of consecutive addresses in storage. This can be done by manually loading the short program in table 1.3-6. Load each instruction of the program using the procedure outlined in paragraph 1.3-4c2.

TABLE 1.3-6. INSPECT AND CHANGE ROUTINE

ADDRESS	INSTRUCTION	EXPLANATION
010	507201	Select address 01 for index register (B1) modification.
011	440001	(AL) to B1
012	110000	B1 to AU
013	505640	STOP
014	470000	(AU) $\longrightarrow$ (B1)
015	710001	Increment AL
016	340011	Jump

When the program has been loaded it can be used to read consecutive memory addresses by use of the following procedures.

STEP 1. Set the SEQ STEP/STOP switch momentarily to the STOP position.

STEP 2. Set the I/O CLEAR/MASTER CLEAR switch momentarily to the MASTER CLEAR position.

STEP 3. Set the P register equal to 00010_8.

STEP 4. Set the desired starting address in the AL register.

STEP 5. Press the MODE RUN button.

STEP 6. Set the RESTART/START STEP switch momentarily to the START STEP position.

With Step 6 completed the computer will stop and AU will display the contents of the address specified in AL. Each time the RESTART switch is pressed, AL will be incremented and the computer will stop with the contents of the next consecutive address displayed in AU. If at any time the operator decides to change the contents of an address as displayed in AU, he can make the change manually (set AU to

the new value) and press the RESTART switch. This will automatically change the contents of that address (as specified in AL) and then display the contents of the next consecutive address.

d. Wire Memory (Bootstrap).

1. General. The 1219 computer has 32_{10} memory address locations (00500_8 through 00537_8) which contain the wired memory bootstrap program for an initial load or bootstrap program. The computer can have a bootstrap written for paper tape, magnetic tape, or teletype input. The following paragraphs describe the paper tape bootstrap as a typical example.

The wired load program provides the ability to enter an initial package of utility routines which may subsequently be used to enter and debug more sophisticated programs. These memory address locations have unique characteristics in that they operate in a special type of nondestructive read-out mode. They are not accessible to the programmer for store-type instructions.

The wired memory bootstrap program first locks out all interrupts then loads in the 1219 utility package. The utility package is formulated so that only the basic load routine is read into the memory addresses immediately following the bootstrap program via wired memory load program. Control is then given to a temporary checksum verification routine of the utility package which verifies the basic load routine. A utility package routine then reads the balance of the utility package. If the checksum verification is incorrect, the computer will come to an unconditional stop with AU equal to the tape checksum and AL equal to the load checksum.

2. Operating Instructions. Instructions for the load mode and automatic recovery mode are described in the following paragraphs.

a) Load Mode. The load mode is used when the manual selection of the wired load program is desired and the computer is in the master clear state. The procedure is as follows.

STEP 1. Place paper tape utility package in reader (assuming paper tape input).

STEP 2. Master clear the computer.

STEP 3. Press the MODE LOAD button.

STEP 4. Set the RESTART/START STEP switch momentarily to the RESTART position. If tape and load checksums agree the computer will come to an unconditional stop with AU and AL cleared.

b) Automatic Recovery Mode. If a fault condition occurs during the running of a program and the AUTO RECOVERY switch is in the up position, an interrupt will address 00500_8 (starting address of the wired load program). This locks out all interrupts and initiates the wired load program which loads in the utility package. For paper tape input, the paper tape utility package must have been placed in the reader (may be positioned on any leader frame) for the recovery to be completed.

NOTE

A fault condition occurring during the running of a program with the AUTO RECOVERY switch in the center position will cause a jump to address 00000. Action will continue as programmed.

NAME: _____

1.3-5. STUDY QUESTIONS:

a. Explain the uses of the four modes of the 1219 computer.

b. What is the purpose of the AUTO RECOVERY switch?

c. What are the different uses of the SEQ STEP/STOP switch?

d. Explain the different uses of the RESTART/START STEP switch.

c. What is the purpose of the F II bit?

SECTION 1 - UNIVAC 1219 COMPUTER FUNCTIONAL CHARACTERISTICS

1.4. PROGRAM DEVELOPMENT

1.4-1. OBJECTIVES

To acquaint the student with techniques for developing programs.

1.4-2. INTRODUCTION

Development of real-time operational programs consists of several phases that are completed following an analysis of the program's requirements. The programing activity for the development of an operational program consists of the following seven phases:

a. Program definition and design.

b. Flow charting.

c. Coding.

d. Compiling and assembly.

e. Program checkout.

f. Program integration.

g. Program installation.

The satisfactory completion of any operational programing task is based on the assumption that two types of information are available to the programer(s), a description of the problem and a description of the system interfaces. The desired information is preferably documented in problem documents known as "Program Requirements" and "Interface Specifications." The information is furnished to the programer(s) by those responsible for system design.

1.4-3. REFERENCES

None.

1.4-4. INFORMATION

a. Program Definition and Design.

1. Introduction. The work associated with the definition portion of this phase consists of the analysis and study necessary to define the problem to be solved. This work results in the preparation of a "Program Design Report" to show

the over-all characteristics of the program(s) to be generated and to provide the basic outline for the development of specific, detailed plans for the computer solution to the problem. The design portion of this phase of the work results in the preparation of one or more documents called "Coding Specification" - one for each program or program module.

The proposed computer solution to a problem is verified, modified, and even redesigned, if necessary, to permit easier computer solution. System constraints are identified along with a thorough data analysis of the computer solution.

Program definition and design is the first phase of program development. Its inputs are program requirements and interface specifications, and its outputs are a Program Design Report, Data Designs (if a compiler is to be used), and one or more Coding Specifications.

2. Definition Stage. In the definition stage of this phase, the program requirements and interface specifications are analyzed in the programing activity to verify the proposed computer solution of the mathematical model, equations, or functions described. Functions and mathematical presentations may be rearranged, modified, or redesigned to permit easier computer solution of the problems.

Completion of the analysis results in preparing a Program Design Report (see paragraph 1.4-4a4) for each program or program system showing the specific, detailed plans for the problem solution.

3. Design Stage. In the design stage of this phase, the general plan of problem solution, as defined in the Program Design Report, is broken down into details, and the exact steps to be followed in solving the problem are spelled out. The proposed solution is checked and modified, if necessary, in the desire to produce the best method for the solution of the problem.

This stage of the development results in preparing one or more Coding Specifications (see paragraph 1.4-4a5), depending upon the complexity of the problem, in which system constraints and a detailed explanation of the computer solution are included.

4. Program Design Report. During the initial stage of the program definition and design phase, the organization of the operating procedures (processes) and the organization of the data (and data tables) of the program are determined and documented in the Program Design Report. In addition, the document may contain some design and organization information primarily of interest to the individuals developing the program.

The typical topics included in a Program Design Report are as follows:

a) Programing conventions.

b) Data table organization.

c) Descriptions of data.

d) Identification of program sections (modules, routines, etc.).

e) Organization of program.

f) Identification of common data tables.

g) Functional descriptions of program sections.

h) Outline of over-all program control.

i) Identification of common subroutines.

j) Outline of operation and application procedures.

k) Scheduling.

l) Overall test organization for check-out and integration.

m) Test requirements.

n) Program production requirements.

The specific format and content of a "Program Design Report" vary with each program being developed; the above-listed topics are examples of typical topics and may or may not be included in specific reports.

5. Coding Specification. The purpose of a Coding Specification is to define a program. It normally includes:

a) The method and processes of problem solution.

b) The restrictions imposed on the program by external conditions.

c) The external interfaces to the program.

External interfaces are concerned primarily with data but may include such interface information as:

a) Programs.

b) Processes.

c) Equipment.

d) Other devices.

A Coding Specification is written for each level of a program (development). Individual Coding Specifications are prepared:

a) For a program system.

b) For Parts of the same system.

c) Eventually for each program written at the task level.

A Coding Specification contains three main categories of information:

1) Title page.

2) Purpose of program.

3) Coding requirements.

a) Title Page. The title page includes (but is not limited to) the following items.

1) Name of the company.

2) Location of the company.

3) Name of the producing department.

4) Program title.

5) Data produced.

6) Author(s).

7) Customer.

b) Purpose of Program. The purpose of program of a Coding Specification is a summary statement describing the major functions performed by the program. It is a concise description of the programed solution for a problem. Reference to other documents and identification of any other sources of information are listed following the statement of the purpose.

c) Coding Requirements. Coding requirements are described in terms of four major classes of information:

1) Input data.

2) Output data.

3) Methods and processes.

4) Restrictions.

The information is presented in sufficient detail so that all information necessary to code the program is available in the specification. All Coding Specifications for the same program follow a uniform organization; the four major classes of information serve as the basis for organizing Coding Specifications.

All program input and program output data are defined in the detailed items of:

1) Item name.

2) Item symbol.

3) Tag assignment.

4) Source of destination.

5) Bit designations.

6) Number of bits (or words).

7) Scaling.

8) Accuracy.

9) Granularity.

b. Program Flow Charting.

1. Introduction. Graphic symbols are used to indicate major processing paths and most decisions accomplished by a program or routine; this representation is called a flow chart. The preparation of flow charts using standard flow chart symbols comprises the effort in this phase of operational program development.

Program flow charting is the second phase of program development. Its inputs are a Coding Specification and Data Designs (if a compiler is to be used), and its output is a flow chart or a set of flow charts, depending upon the complexity of the problems.

2. Flow Charting. Flow charting is the preparation of a diagram, called a flow chart, showing the flow of data and the sequence of operations to be performed on the data in a program or program module.

3. Reasons for Flow Charting. There are several reasons for flow charting, especially for flow charting before coding is begun. The four basic reasons are:

a) Work Division.

1) Each programer knows which computations he has to program.

2) Each programer knows which data or variables he is responsible for and which ones he can assume are computed by someone else's program.

3) The person in charge of the programing effort knows just who is responsible for each section of the program and can determine how the coding is progressing.

4) After each module is written, the flow chart is the guide for incorporating the program modules into the program.

b) Coding Guide.

1) The programer can refer to the flow chart to make sure he does the computations in the proper order.

2) The flow chart shows where the tests and branches in the program are located.

3) The programer can compare the flow chart to the coding to make sure the re-entries to the program are in the proper places.

c) Debugging aid.

1) The flow chart can help the programer locate errors in computations.

2) If addresses or tags are placed on a working model of the flow chart, the programer can tell at a glance just where computation is located in the computer storage.

d) Future Reference.

1) The flow chart is a very important reference for persons who are going to be working with the program in the future. If the program has to be modified, updated, or corrected in the future, the flow chart is one indispensable reference. Even if this work is done by the original programer, he cannot remember all he needs to know about the program. The person using the program can tell from the flow chart what the program will do and how it accomplishes the task.

2) The flow chart is a handy reference for someone who may want to use the program. If the flow chart is properly drawn, the potential user can usually tell in a very short time if he can use the program. The potential user can determine what modifications, if any, are necessary for his purpose.

3) If someone wants the same routine for another computer, he can use the flow chart, along with other documentation, to assist with program design. This could save valuable time.

4. Sizes. Each flow chart should be prepared on the standard 8-1/2 x 11 document form. If the flow chart requires more than a single 8-1/2 x 11 sheet, it should be prepared on two or more 8-1/2 x 11 sheets. If it is impossible to effectively split a flow chart into understandable 8-1/2 x 11 portions, the flow chart may be prepared on one or more sheets of standard 11 x 17 document form. The use of other than the 8-1/2 x 11 form results in difficult-to-handle documentation; so such use should be kept to a minimum.

5. Standard Symbols. The standard flow chart symbols in MIL-STD-682A shall be used in the preparation of all flow charts. These symbols shall not be redefined, and other symbols shall not be substituted for the defined functions. If additional symbols, for functions not defined, are required for a specific application, such additional symbols may be used providing that they have pre-use approval of the Manager of Defense Systems Programing and that they are fully explained in this document.

6. Symbol Template. All of the standard symbols can be drawn with the aid of the UNIVAC symbol template labeled MIL-STD-682A, see figure 1.4-1. The template should be used in the preparation of all flow charts.

7. Symbol Sizes. The size of each symbol shall be as specified in MX-3983/G and as provided for in the MIL-STD-682A template. The sizes are considered appropriate for the full-scale preparation and reproduction of any flow chart, regardless of the sheet size.

8. Symbol Orientation. All of the symbols shall be oriented as illustrated in paragraph 1.4-4b11.

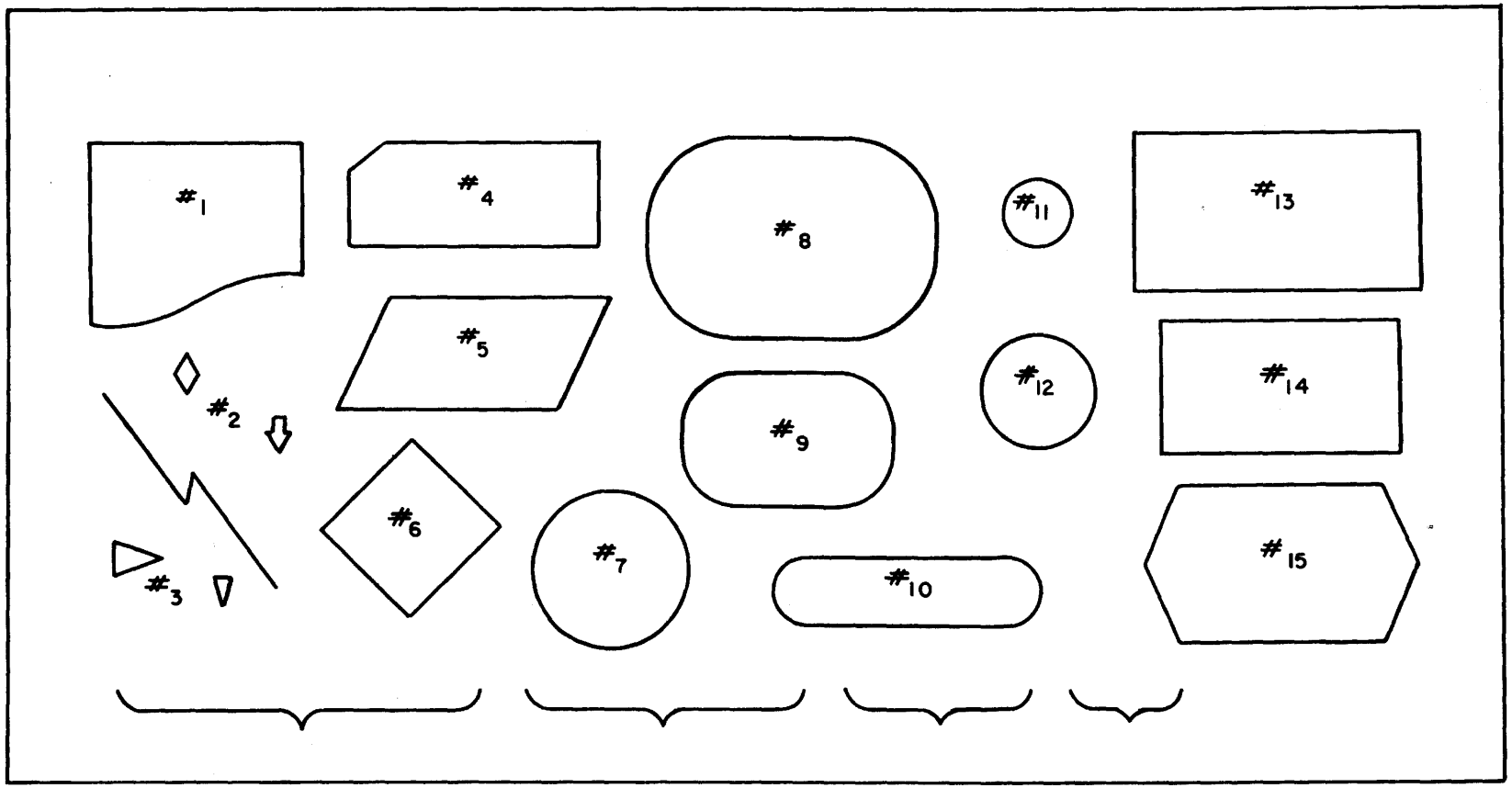


Figure 1.4-1. Template for Flow Chart Symbols

9. Lines of Flow. Straight lines, vertical or horizontal, shall be used to show the flow of control or data between the symbols on a flow chart. In general, the lines shall be aligned in the direction of data flow, i.e., if the flow is from top to bottom, the lines shall appear vertical with input at the top and output at the bottom; if the flow is from left to right, the lines shall appear horizontal with input at the left and output at the right.

Symbols requiring two output lines shall normally have one output line, in-line with the input line and the other output line in a direction perpendicular to the input line.

None of the lines -- input or output -- is considered to be inherently the part of any symbol.

10. Direction of Flow. Vertical flow charting, in which successive symbols are placed one below the other, is preferable to horizontal flow charting.

All direction of flow of inputs and outputs is from top to bottom or from left to right unless otherwise indicated. If the flow is from bottom to top or from right to left, a properly oriented direction arrowhead shall be placed on the line with the tip of the arrowhead touching the symbol to which the flow is directed.

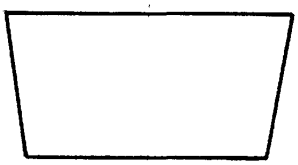
When increased clarity is desired, direction arrowheads may be placed anywhere along a flow line.

11. Flow Chart Symbols. Symbols are used on a flow chart to represent the functions of a data processing program or system. The basic functions are input/output, processing, and annotation.

A basic symbol is established for each basic function and can always be used to represent the function on a system flow chart. Specialized symbols are established for detailed functions and shall be used to represent those functions on a program flow chart.

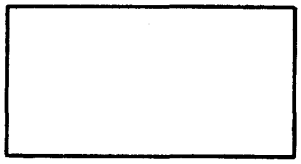
a) Basic Symbols. (For use only on system flow charts.)

1) Input/Output Symbol. The symbol shown below represents the basic input/output function (I/O), i.e., the making available of data for processing (input), or the recording of processed data (output).



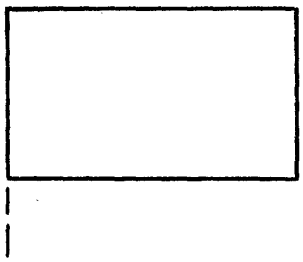
Dimensional ratio
width: height = 1:2/3
(Combination of symbols #13 & #14)

2) Processing Symbol. The symbol shown represents the basic processing function, i.e., the process of executing a defined operation or a group of operations resulting in a change in value, form, or location of data, or in the determination of which output path is to be followed.



Dimensional ratio width:
height = 1:2/3 (Symbol #1)

3) Annotation Symbol. The symbol shown below represents the basic annotation function, i.e., the addition of descriptive comments or explanatory notes as clarification. The dotted line may be drawn either on the left and down (as shown), on the right and down, on the top and left, or on the top and right. It is connected to the flowline, at a point where the annotation is meaningful, by extending the broken line in whatever manner is appropriate.



Dimensional ratio
width: height = 1:2/3
(Symbol #13)

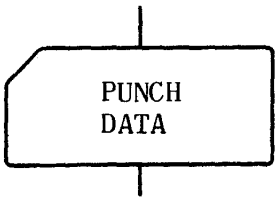
b) Specialized Symbols. (For use expressly on program flow charts.)

1) Input/Output Symbols. The specialized input/output symbols represent the I/O function and, in addition, denote the medium on which the data is recorded or the manner of handling data, or both.

a. Data Card Symbol. The symbol shown below represents an I/O function in which the medium is punch cards, or the like. The symbol also represents a card file.

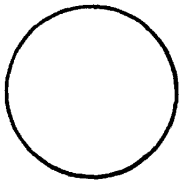
Symbol #4

Typical Use

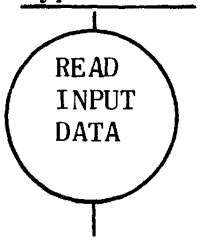


b. Magnetic Tape Symbol. The symbol shown below represents an I/O function in which the medium is magnetic tape.

Symbol #7

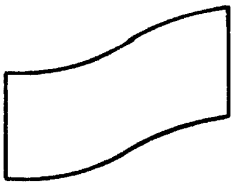


Typical Use

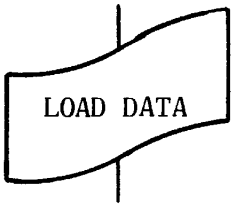


c. Punch Tape Symbol. The symbol shown below represents an I/O function in which the medium is punch tape.

Symbol #1

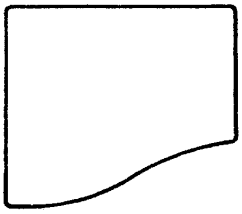


Typical Use

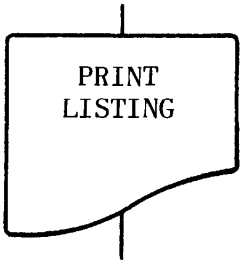


d. Printed Document Symbol. The symbol shown below represents an I/O function in which the medium is a (high-speed printer) printed document.

Symbol #1



Typical Use



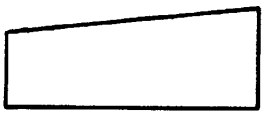
e. Manual Input Symbol.

NOTE

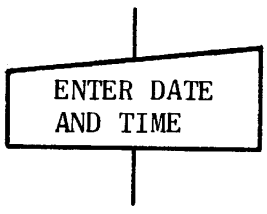
This symbol can be drawn with the template, but it is not defined as a Military Standard symbol. It may be used as an approved symbol. (See paragraph 1.4-4b5.)

The symbol shown below represents an I/O function in which the data is entered manually at the time of processing, by means of on-line keyboards, switch settings, pushbutton, etc.

Symbol #4

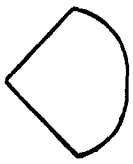


Typical Use

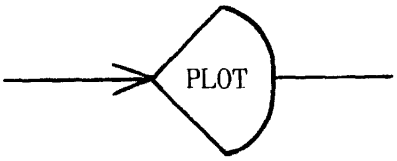


f. Display Symbol. The symbol shown below represents an I/O function in which the data is displayed for human use at the time of processing, by means of on-line printers, console printers, video devices, plotters, etc.

Symbol #6 & #8

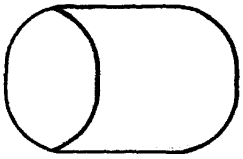


Typical Use



g. Mass Storage Symbol. The symbol shown below represents an I/O function utilizing auxiliary mass storage of data that can be accessed on-line, e.g., magnetic drums, magnetic discs, etc.

Symbol #9

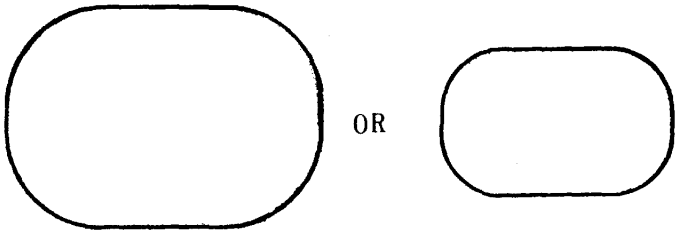


Typical Use

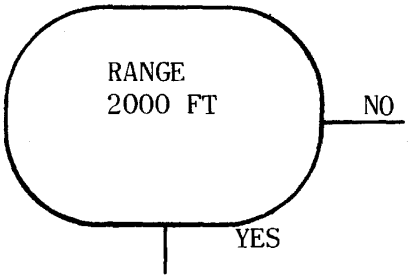
2) Processing Symbols. The specialized processing symbols represent the processing function, and, in addition, identify the specific type of operation to be performed on the data.

a. Decision Symbol. The symbol shown below represents a decision type operation that determines which of two or more alternate paths is to be followed.

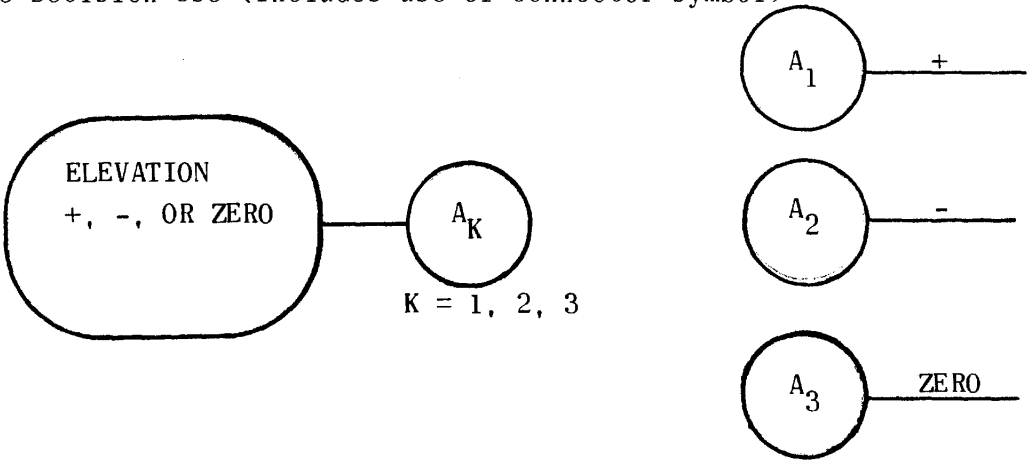
Symbol #8 or #9



Normal Decision Use

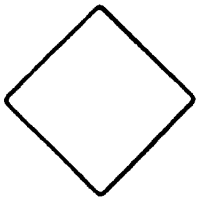


Multiple Decision Use (includes use of connector symbol)

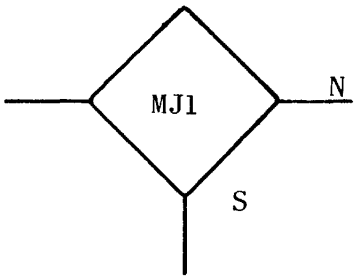


b. Switch & I Branch Symbol. The symbol shown below represents a switching type of operation wherein (input) data is routed to either of two (output) paths depending upon the setting of the switch, either "set" (S) or "normal" (N).

Symbol #6

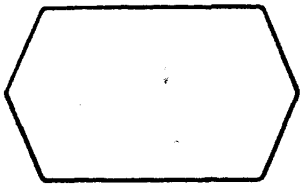


Typical Use

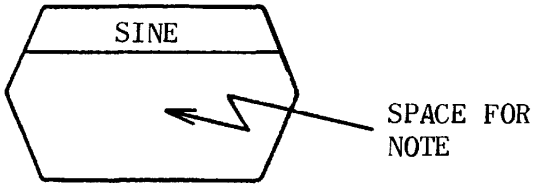


c. Subroutine Symbol. The symbol shown below represents a named process consisting of one or more operations or program steps that are defined elsewhere, i.e., a closed subroutine or a library subroutine.

Symbol #15



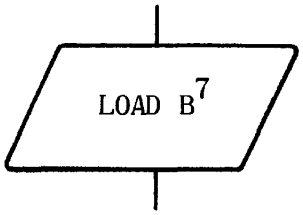
Typical Use



d. Manual Operation Symbol. The symbol shown below represents any off-line process geared to the speed of a human being.

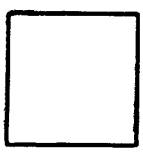
Symbol #5

Typical Use



e. Auxiliary Equipment Operation Symbol. The symbol shown below represents any off-line operation performed on equipment not under direct control of the central processing unit.

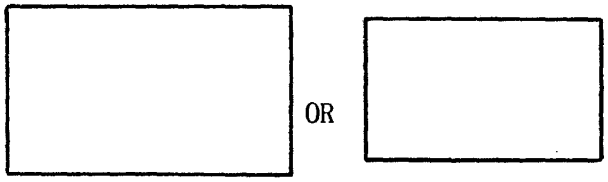
Symbol #6



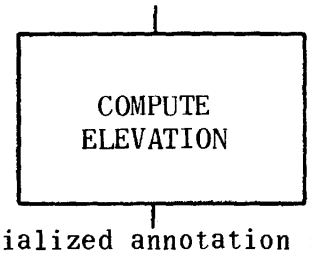
Typical Use

f. Non-Auxiliary Equipment Operation Symbol. The symbol shown below represents an on-line operation performed on equipment under direct control of the central processing unit.

Symbol #13 or #14



Typical Use



3) Annotation Symbols. The specialized annotation symbols are used to identify specific types of connections or terminations.

a. Terminal Symbol. The symbol shown below represents a terminal point in a system or communication network at which data can enter or leave, i.e., start, stop, halt, delay, or interrupt.

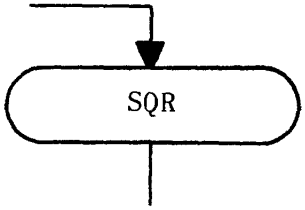
Symbol #10

ENTER PROGRAM LABEL IN A "START" SYMBOL

OR

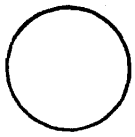
ENTER SPECIAL INFORMATION, IF ANY, IN A "STOP" SYMBOL

Typical Use

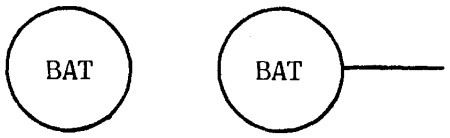


b. Connector Symbol. The symbol shown below represents a point at which a computational path conveniently starts or ends. Connectors are used for identifying common points in separate areas of a single sheet of a flow chart, primarily to avoid crossover of flow lines. Connectors are also used for identifying the end of flow lines that leave one sheet of a flow chart and resume on another sheet.

Symbol #12



Typical Uses



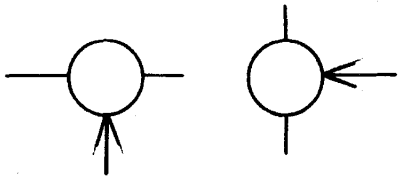
NOTE

This symbol is included in the template, but it is not defined as a Military Standard symbol. It may be used as an approved symbol (see paragraph 1.4-4b5.

The symbol shown below represents the point at which re-entry into the main path of a program is made.

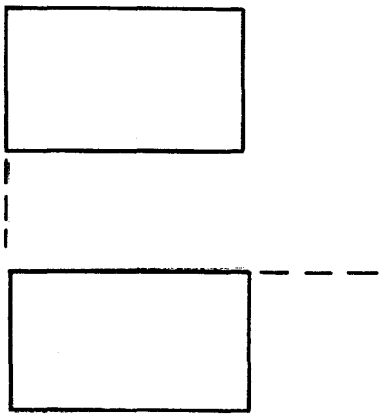
Symbol #11

Typical Uses

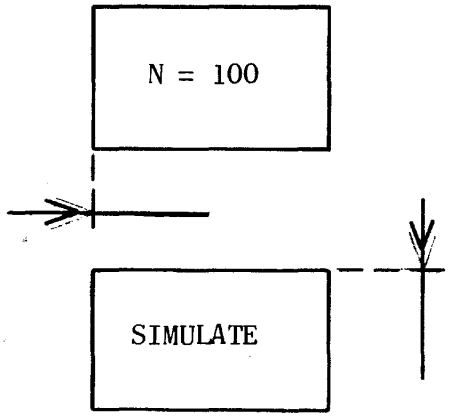


d. Assertion Symbol. The symbol shown below indicates a special condition existing at a certain point along a flow line.

Symbol #13 or #14



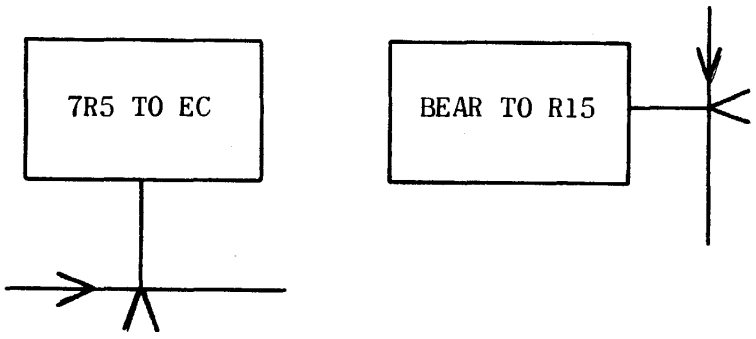
Typical Uses



e. Insertion Symbol. The symbol shown below indicates the insertion (without redrawing the flow chart) of a processing symbol into a flow line; the single line drawn to the symbol represents both the input line and the output line.

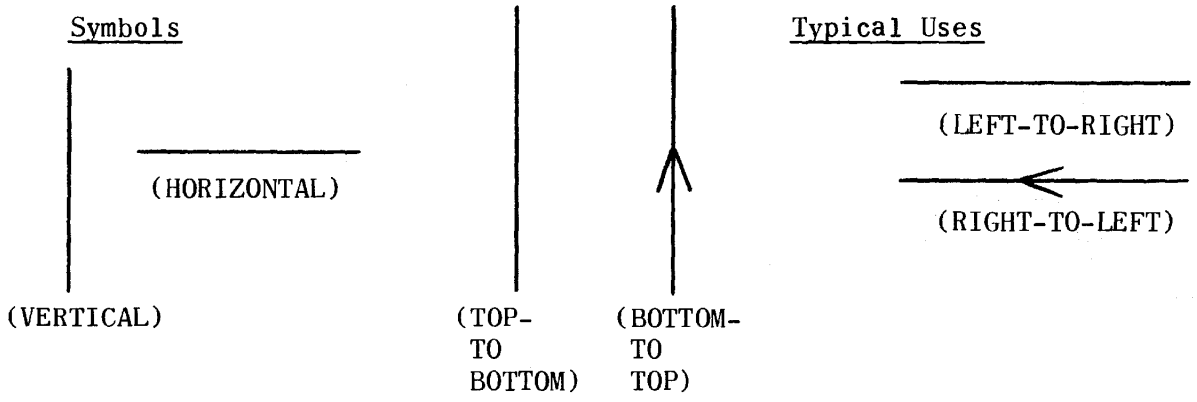
Symbol #3

Typical Uses



f. Communication Link Symbol. The symbol shown below represents an I/O function in which data are transmitted automatically from one location to another. The symbol shall always be drawn with superimposed arrowhead to denote the direction of data flow.

g. Flow Lines. Vertical or horizontal lines between symbols to show the flow of data or control. Arrowheads shall be used for other than top-to-bottom or left-to-right flow, and they may be used on any line for clarity.



13. Flow Chart (Fictitious). Figure 1.4-2 illustrates the application of some of the flow chart rules.

c. Coding.

1. Introduction. Translation of the terms of a problem from the symbology of the flow chart to the symbolic language of a compiler or assembler program defines the coding effort of a programing task. The translation is accomplished manually on coding forms or sheets.

Coding is the third phase of program development. Its inputs are a Coding Specification and either a flow chart or a set of flow charts, and its output is a manually-prepared list of program instructions.

2. Coding. Coding is the translation of the details of problem solution from the symbology of a flow chart to a sequence of instructions in a language (or, in some cases, in a combination of languages) acceptable to and understandable by a specific computer.

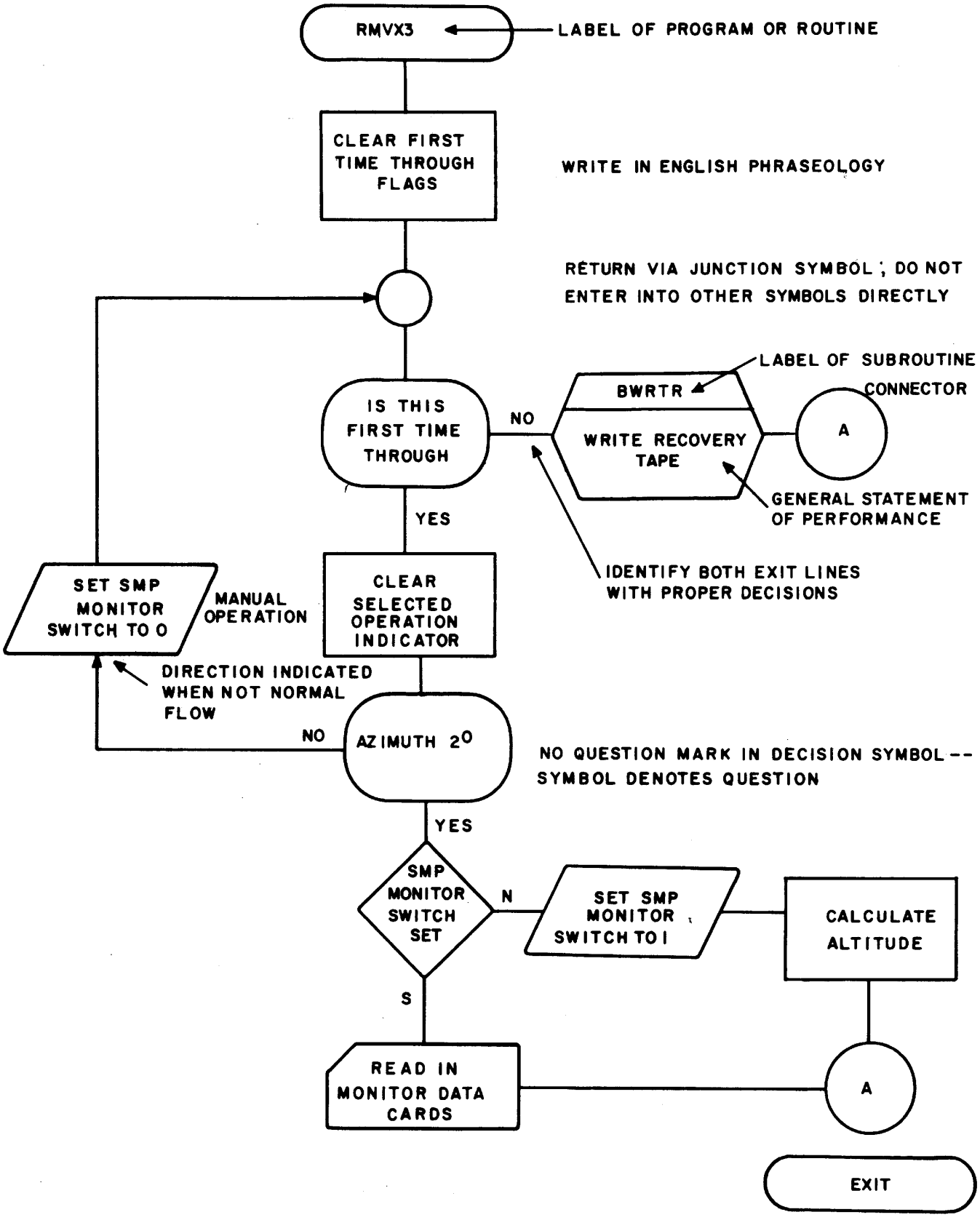


Figure 1.4-2. Sample Flow Chart.

3. Coding Languages. Two types or levels of program languages are associated with program coding. One can be used directly by the computer; the other must be translated before use. The languages are as follows:

a) Machine language (use directly).

b) Symbolic languages (translate before use). Consist of assembler language (machine oriented) and compiler language (problem, procedure or business oriented).

In machine-language coding the instructions are prepared in absolute code. In symbolic-language coding the instructions are prepared in either a mnemonic monocode or a mnemonic polycode, depending upon whether the object-program instructions are to be computer-prepared by an assembler (program) or by a compiler (program), respectively.

4. Machine Language Coding. Machine language coding refers to the use of machine language in the preparation of the computer instructions by the programer. Such preparation is usually referred to simply as absolute coding.

Absolute coding is the only method of coding available to programers before assembler and/or compiler programs have been produced for a computer.

The significant features of absolute coding are:

a) It requires an experienced programer.

b) It is usually a one-man operation.

c) It requires considerable (length of) time.

d) It offers great latitude and flexibility in coding.

5. Symbolic Language Coding. Symbolic language coding refers to the use of a symbolic language - either assembler or compiler language - in the preparation of the computer instructions by the programer. Such preparation is usually referred to simply as symbolic coding.

Symbolic coding can be used only if and when assembler and/or compiler programs have been developed for a computer.

Some of the significant features of symbolic coding and the associated use of an assembler or compiler are:

a) The number of trained programers capable of coding more complex problems in absolute code is far too small to meet the demands of the rapidly-growing applications of computers in industry.

b) Writing programs in absolute code requires more time than in symbolic code. As problems become more complex, the time difference becomes greater. Since computer programers are expensive, costs increase with problem complexity.

c) Symbolic languages, because they are written in (familiar) English or mathematical statements, are learned more easily and quickly then machine languages.

d) Errors are less frequent in symbolic coding, thereby reducing the quantity and hours of expensive computer reruns.

e) One of the more tedious tasks of coding, that of housekeeping, becomes much simpler and easier to handle with assemblers and/or compilers. Housekeeping involves keeping records of memory locations to avoid conflict of assignment, keeping records of controls and indexes, constant checking and referencing of memory assignements to avoid error, and other minute details, all of which are time-consuming, tedious, and subject to error. By assigning meaningful names to indexes, flags, memory locations, starting addresses, and other housekeeping functions, assemblers and compilers can make unique assignments and maintain records of such assignments without error.

f) The use of significant names instead of numbers makes instructions more meaningful and easier for another programer or for the user of the program to follow. The program becomes much more self-explanatory.

g) The use of mnemonic names for memory locations, counters and other controls makes it possible for many programers to work on subroutines of the same program independently. Names of similar memory locations and controls can be interchanged or equated subsequent to the writing of the subroutines.

h) The integration of independently produced subroutines into a program becomes much easier because of the ability to use the same names or to equate names in assemblers and compilers.

i) Since the computer can be used to perform the functions of equating and integration, much time and expense is saved in these operations.

6. Selecting the Coding Language. Normally the philosophy and policy is to use the highest level of coding language available and applicable to the project concerned. The absolute language for each different computer is explained in detail in the hardware documentation for the computer. The various symbolic languages, i.e., assemblers and compilers, are explained in special manuals prepared for each of the languages.*

7. Preparing the Coding Form. Once the language has been selected, i.e., absolute, assembler, or compiler, the programer completes the appropriate coding form with successive instructions written on successive lines of the form. Examples of typical coding forms are shown in figures 1.4-3 through 1.4-5.

Completing the coding form is the most critical step in coding. The completed form is used by keypunch operators or Flexowriter operators to prepare punch cards or punch tape, respectively, of the instructions written on the form; and errors in completing the form invariably result in errors which must be later debugged.

UNIVAC

PATRAC ASSEMBLER CODING FORM

PROBLEM _____ CODED BY _____ DATE _____ PAGE _____ OF _____

ITEM NUMBER	TAG	OPERATION	OPERAND	INDEX	SCALING	COMMENTS

Figure 1.4-3. PATRAC Assembler Coding Form

PAGE____ OF ____

PROBLEM _____

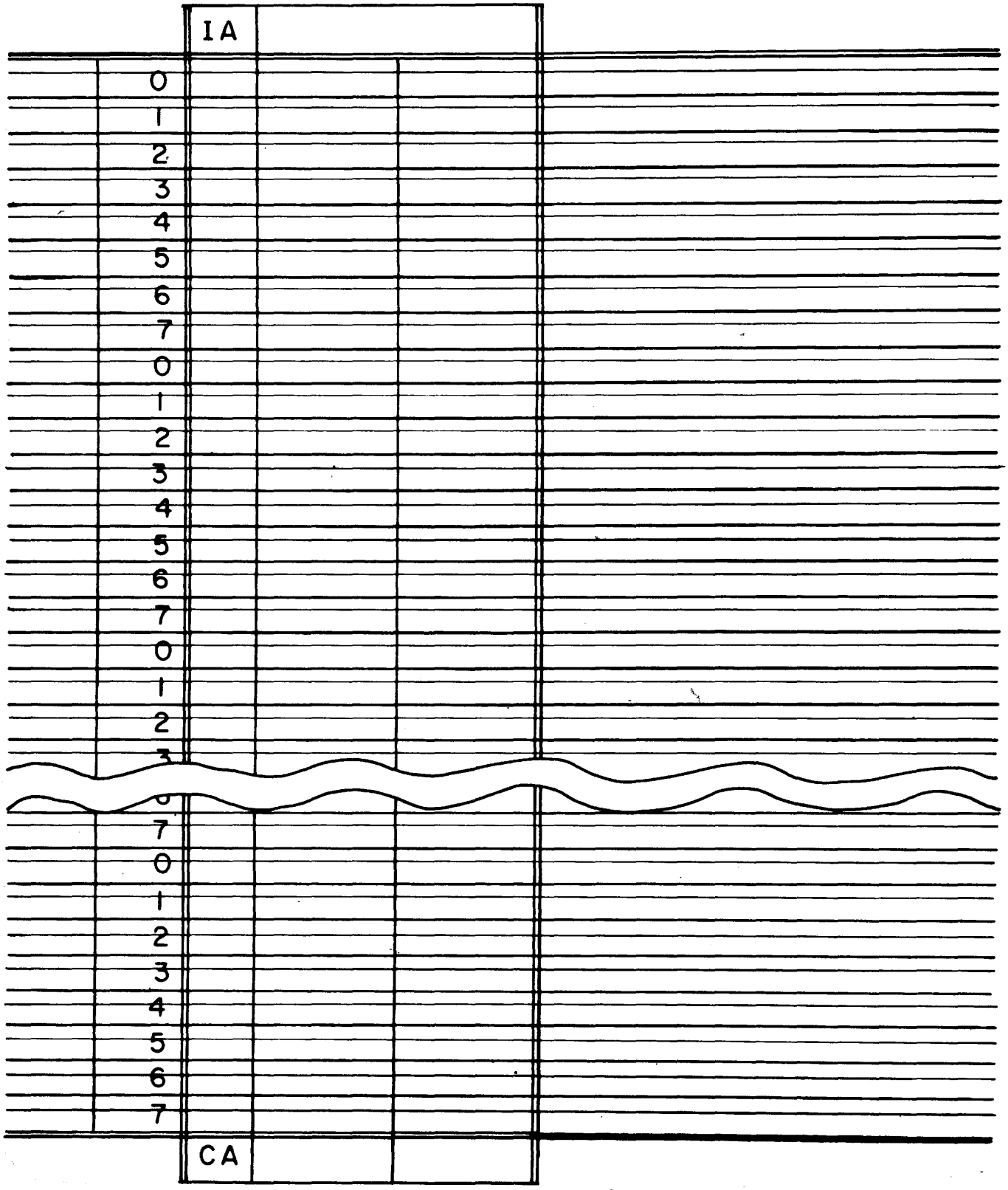


Figure 1.4-4. UNIVAC 1103 Absolute Coding Form

TITLE _____ UNIVAC CODING FORM PROGRAMMER _____

PAGE _____ of _____ PLT. ____ EXT. _____ MS _____

DATE _____

LABEL	OPERATOR	OPERANDS AND NOTES
	HEADER TYPE	

Figure 1.4-5. UNIVAC Coding Form

8. Preparing Program and User Documents and Publications. In addition to completing the coding form in the desired program language, the coding phase often includes the preparation of documentation in the form of program and user documents and publications; such preparations depend upon the nature of the program being developed.

The documents that may be prepared are:

a) Program synopses.

b) Program application guides.

c) Program operating procedures.

The publications that may be prepared are:

a) Handbooks.

b) Manuals.

c) Brochures.

In many cases, programers prepare the documents, but seldom, if ever, do they prepare the publications. The publications are normally prepared by technical writers, who sometimes also prepare the documents.

d. Compiling and Assembly. Conversion and translation of a symbolic language used to code a problem into a form of absolute machine language recognizable and understandable to a computer is accomplished during the compiling or assembly process. A computer is used to perform the process, and it is called compiling or assembly dependent upon the type of translation program used. The compiling or assembly program automatically produces a physical representation of a program that can be loaded into the memory of a computer. Another result of the process is the automatic production of a copy of the program on a document called the "Program Listing."

Compiling and assembly is the fourth phase of program development. Its input is an annotated sequence of instructions manually recorded on appropriate coding sheets, and its outputs are a Program Listing and a computer-prepared sequence of instructions recorded on magnetic tape.

Compiling is the conversion and translation of a manually-prerecorded sequence of polycode program-language instructions into a computer-recorded sequence of object-language instructions, with the use of a compiler (program).

Assembly is the conversion and translation of a manually-recorded sequence of monocode program-language instructions into a computer-recorded sequence of object-language instructions, with the use of an assembler (program).

1. Data Preparation & Processing. Several steps of data preparation and processing are involved in the compiling and assembly phase, as shown in figure 1.4-6.

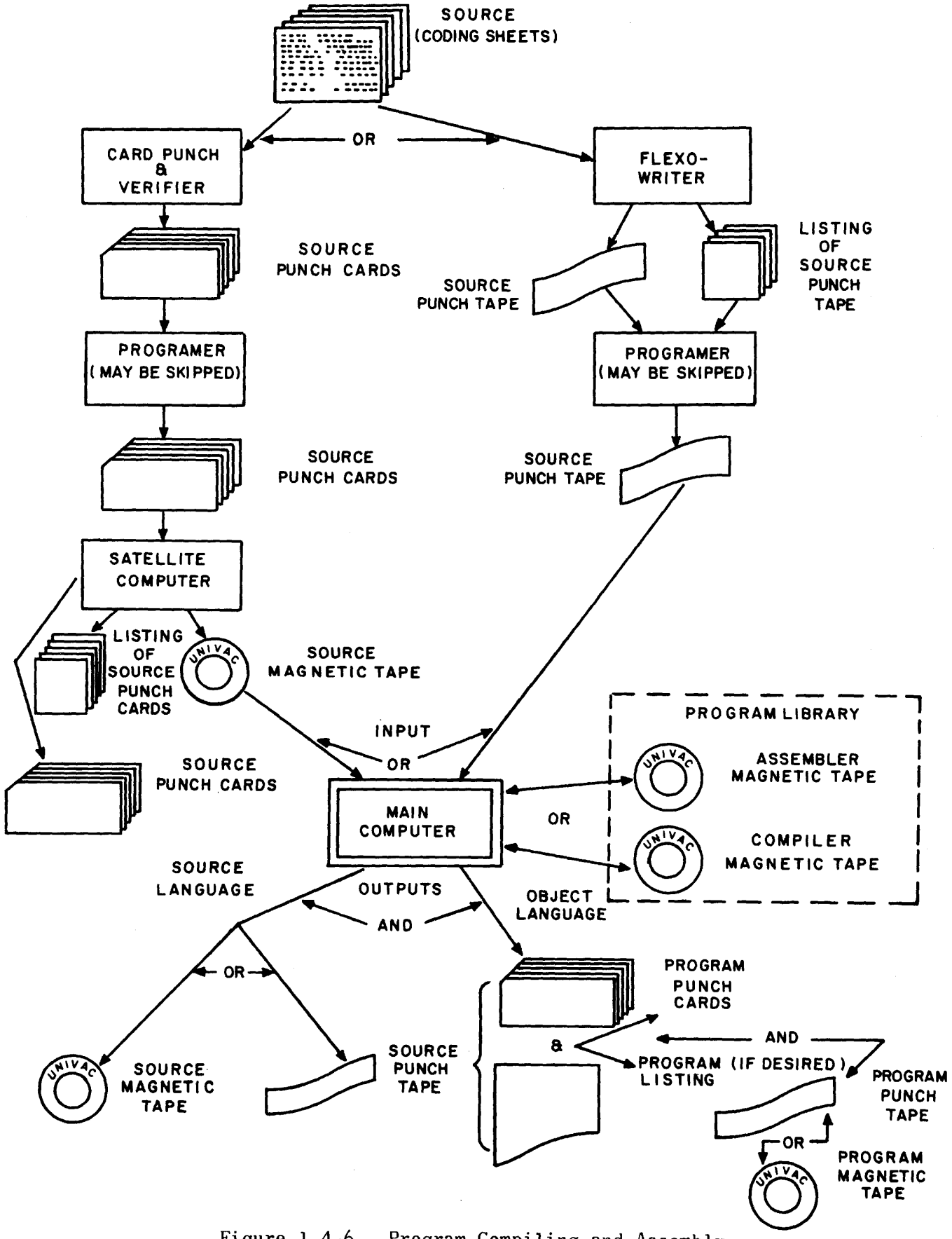


Figure 1.4-6. Program Compiling and Assembly

The coding sheets, which have been prepared specifically for either card punching or tape punching, are submitted by the programer to the computer center for appropriate preparation and processing. After processing is completed, the normal items delivered to the programer are the deck of source punch cards (or the roll of source punch tape), the listing of the source punch cards (or of the source punch tape), the listing of the source punch cards (or of the source punch tape), and the Program Listing. The normal items retained by the center are the source language and object language magnetic tapes. Unless theprogramer has specifically requested that these magnetic tapes be saved, the tapes are reused in other jobs.

2. Program Listing. A Program Listing is a document containing a complete list of computer instructions and constants as they are (to be) used in performing the tasks involved in solving the problem for which the program is designed.

The Program Listing is the document from which the programer works while debugging a program. Also, since the Program Listing is a permanent, readable copy of the program, it provides the starting point for making alterations or modifications to a program as the programing requisites change.

3. Content of Program Listing. The exact content of a Program Listing depends upon the method used in preparing the program and the specific program in computer storage at the time the Program Listing is produced. As a result, the instructions may be listed in any one or more of the following languages:

a) Compiler.

b) Assembler.

c) Machine.

If a compiler or assembler is used, the Program Listing may or may not include provisions for including one or more lines of explanatory comments or remarks to aid in the understanding of the program. If the absolute machine language instructions are included in the Program Listing as the result of a storage dump from the computer, any explanatory comments must be added by a manual operation. During the development of a program, the programer may be satisfied with a listing that does not contain the absolute machine language instructions, that only includes the first lines of multiline comments, or that is prepared from a storage dump.

The final Program Listing shall include the following information arranged in three columns:

1) Absolute machine language instructions.

2) Original symbolic statements.

3) Explanatory comments and remarks.

e. Program Checkout.

1. Introduction. Initial program checkout, e.g., debugging, parameter testing, etc., consists of testing each section of the program by itself with test parameters that generate predetermined results only if the programed section (of program) operates correctly. Locating errors that prevent successful generation

of known results requires extensive re-evaluation of the effort expended in the preceding phases of the program development.

Program checkout (debugging) is the fifth phase of program development. Debugging is the isolation and removal of mistakes from a newly assembled or compiled program or program module.

2. Sources of Mistakes. There are several sources of mistakes in newly coded programs. Knowing the sources should prompt all programers and program supervisors to maintain constant vigilance in critical areas to minimize the mistakes. To avoid all mistakes is humanly impossible; to ignore them is foolhardy; to minimize them is important.

Common sources of mistakes are:

a) Ignoring of program goals.

b) Ignoring of program restrictions.

c) Poor flow charting.

d) Poor coding sheet preparation (careless hardwriting).

e) Incorrect coding sheet interpretation (difficult reading).

f) Poor card, or tape, punching.

3. Isolation of Mistakes. The isolation of mistakes is normally accomplished by using one or more of the following methods.

a) Visual examination of the source punch-card (or source punch-tape) listing.

b) Visual review of the object program listing.

c) Running the program with a driver.

d) Running the program with a tracing routine.

e) Using debugging aids.

The selection of method(s) depends primarily upon the input medium and length and complexity of the newly coded program.

a) Using the Source Punch-Card Listing. When punch-cards are to be used as the input medium, the instructions from the programer's coding sheet(s) are punched on cards with a card punch. As each character is punched in code on the card, the character itself is printed (interpreted) near the top of the same column of the card. Thus, successive instructions are printed on successive cards.

The interpreted cards can be reviewed to verify the correctness of the punched cards and, occasionally, to reveal coding mistakes or keypunch operator errors. Or, it may be easier to use the listing of the source punch-cards rather than the

punch-cards themselves; the listing can be requested by the programer during the compiling and assembly phase.

b) Using the Source Punch Tape Listing. When punch tape is to be used as the input medium, the instructions from the programer's coding sheet(s) are punched on tape with a Flexowriter. As each character is punched in flex code on the tape, the character itself is typed on paper. Thus, a typewritten copy of the coding sheet is automatically prepared as the tape is punched.

A visual review of the typewritten copy can be made to verify the correctness of the punched tape, and occasionally, to reveal coding mistakes or Flexowriter operator errors.

c) Using the Program Listing. For debugging short programs, a review of the source punch-card (or source punch-tape) listing may be satisfactory. However, for long routines it is usually better to examine the program listing, for it contains not only the information from the coding sheets but also any error flags introduced by the computer when assembling or compiling the program.

d) Using a Driver. A driver is a special program used to test an operational program by creating conditions expected to be encountered under normal run conditions of the operational program. Drivers normally operate by creating inputs and producing conclusions that would be expected to affect the program in a predetermined manner. Normally, if the actual run results compare with the expected results, the tested program is assumed to be operating properly. It is very important, however, to be certain that the new program is tested to its designed limits before drawing any such conclusion.

e) Using a Tracing Routine. A tracing routine is a routine that monitors a program as it is being run. For any selected address or group of addresses, it extracts, examines, and executes the instructions of the tested program in the same order as would normally occur. After the instruction is executed, data is sent to a magnetic tape unit for off-line listing on the high-speed printer according to the input parameters to the tracing routine. The output listing normally consists of the operand of the instruction and the contents of the P, AU, AL, F and B registers. This listing is an invaluable aid to debugging when a mistake is known to exist but cannot be isolated by examining the program listing.

f) Use of Debugging Aids. Compilers usually contain built-in debugging routines with special capabilities. These include:

1) Contents-of-registers printouts.

2) Dumps of specified computer storage areas.

3) Tracing routines.

4) Records of changes to program instructions.

Individual projects also frequently develop special debugging techniques.

Programers shall use these special debugging features as prescribed for the project or as otherwise needed.

f. Program Integration.

1. Introduction. The purpose of the integration phase is to discover and eliminate interface problems between program modules. Special tests are used to provide predetermined results when two or more checkout program modules are operated together. These tests are continued until all modules are integrated into an operating unit called the program. Errors discovered in this phase of program development are more difficult to find and isolate than errors discovered during the checkout of program modules. Successful simulation runs indicate the completion of program integration.

Program integration is the integrating or uniting of two or more routines or program modules into a single routine or program.

The purpose of the integration phase is to discover and eliminate interface problems between routines or program modules.

2. Isolation of Mistakes. Mistakes discovered in this phase of program development are much more difficult to isolate than are the mistakes discovered in the checkout phase.

The value of good data reduction and utility programs becomes apparent in this phase of development. Simulation efforts should be started to aid in the integration. Successful simulation runs indicate completion of integration, even though this phase usually merges with the next phase - the installation phase.

3. Use of Compilers and Assemblers. Compilers and assemblers promote the ease of integration. Assemblers make use of symbolic addressing. The symbolic addresses are then assigned to appropriate computer storage locations during integration. This avoids duplication of assignment.

Most compilers have built-in methods of accounting for and assigning data to storage locations. In addition, they provide for the equating or substituting of data names. Routines can thus be integrated into programs by proper manipulation of data names.

If an assembler or a compiler is used, the programer should become thoroughly acquainted with the integrating capability of the system.

g. Program Installation.

1. Introduction. The programing effort associated with system integration and checkout is the primary activity of the phase of program development. The effort consists of running both simulated and real checks to determine the correctness of the program interface with all other elements of the operational system.

Program installation is the successful running of the object program on the object hardware in accordance with the customer's stated requirements.

Unfortunately, the stated requirements do not always present a full and correct picture of actual operating conditions, so it may be possible to satisfy the stated requirements but fail to perform the on-site operation.

2. Installation Tasks. The programer performs three distinct tasks in installing the program:

a) Program evaluation.

b) Program revision.

c) Documentation revision.

Program evaluation is the study of the operation of an integrated program to ensure that it performs all required tasks under all specified conditions. Essentially, program evaluation has the same function as the program checkout except that the program consists of a number of integrated routines rather than one routine.

After evaluation, the program is revised as needed, then reassembled and re-evaluated. Where minor program revisions are required, it is often convenient to change the input directly without going through reassembly. When this is done, however, no revised program listing is produced. This omission must be corrected as soon as possible by reassembling the program. Until the reassembly is accomplished, handwritten changes shall be recorded in the documents being used.

During the installation phase, the documentation for all portions of an integrated program are revised to reflect changes made to the program, and final copies are prepared for all activities and persons concerned.

To maintain orderly records and to ensure full distribution of final documentation, all changes to final documents and publications shall only be accomplished by following the formal change procedure of a Program Change Order.

NAME: _____

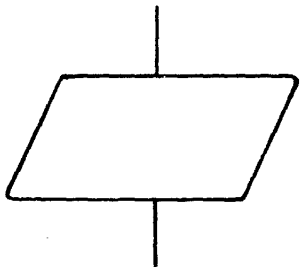
1.4-5. STUDY QUESTIONS

a. What two types of information should be available to programer(s)?

b. Name the typical topics included in a Program Design Report.

c. Define Flow Charting and the reasons for its use.

d. What does the following symbol represent?



e. What are the three levels or types of program languages associated with program coding?

f. What is the major difference between compilers and assemblers?

g. What is the program listing used for?

h. What are some common errors or mistakes in newly coded programs?

NAME:_____

i. Explain the difference between program checkout and program integration.

j. What are some of the problems that may arise in program installation?

SECTION 1 - 1219 COMPUTER FUNCTIONAL DESCRIPTION

1.5. DETAILED BLOCK DIAGRAM OF THE 1219 COMPUTER

1.5-1. OBJECTIVES

To provide the student with specific areas of study and self-check questions to give you a better understanding of the 1219 Computer through the block diagram.

1.5-2. INTRODUCTION

The 1219 Computer is made up of four sections, the memory section, the arithmetic section, the control section and the input-output section. In order to be able to understand how the computer processes data and instructions you should have a basic understanding of the sections of the block diagram.

1.5-3. REFERENCES

a. PX 3288, Programers Reference Manual

b. PX 3316, Vol. I & II

1.5-4. INFORMATION

a. General. The UNIVAC 1219 Military Computer is a medium-scale, general-purpose digital computer specifically designed to comply with the environmental specifications of MIL-E-16400. It is a faster version of the widely-used UNIVAC 1218 Computer and is functionally compatible with that machine.

To meet the extreme requirements of real-time and concurrent batch processing operations, the UNIVAC 1219 is equipped with a 2-microsecond, internal, random access core memory in sizes of 8,192, 16,384, 32,768 or 65,536 18-bit words with a read access time of 0.9 microseconds and a 500-nanosecond control memory. In addition to this, other random-access storage devices connected to input/output channels provide unlimited memory capacities. A portion (32 word locations) of the core memory addresses has been used for the nondestructive memory in which constants and instructions are stored for automatic recovery from fault situations and for an initial load of routines.

With its high internal operating speed, core memory, and 500-nanosecond control memory, the UNIVAC 1219 Computer is capable of transferring 500,000 words per second. Arithmetic and input/output 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 100

instructions allows complete programing freedom in mathematical and logical computations and operations. The computer features 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 UNIVAC 1219 Computer with a 32,768 word core memory, power supply, and all logic and control circuitry is contained in a single cabinet which occupies less than 32 cubic feet and requires less than five square feet of floor area.

A simplified block diagram of the computer appears in figure 1.5-1. Abbreviations on the diagram are explained as the operation of the various computer sections are discussed.

b. Memory. The computer memory consists of up to 65,536 18-bit words of addressable storage locations divided into three distinct sections in a continuous addressing structure.

1. Control Memory. An independent high-speed core memory, consisting of 128 18-bit words, is used for index registers, buffer control words, a real-time clock cells, real-time clock interrupts, and the fault interrupt address. The fixed addresses for these functions are listed in table 1.5-1.

TABLE 1.5-1. CONTROL MEMORY FIXED ADDRESSES

Address	Assignment
00000	Fault interrupt entrance register
00001-00010	8 index registers
00011	Inter-computer time-out interrupt register
00012	Real-time clock interrupt register
00013	Clock overflow interrupt register
00014	Real-time clock monitor word register
00015	Real-time clock incrementing register
00016	Synchronizing interrupt register
00017	Scale factor shift count
00020-00037	Continuous data mode (channels 0-7)
00040-00057	Output buffer control registers (channels 0-7)
00060-00077	Input buffer control registers (channels 0-7)
FOR UNIVAC 1219 COMPUTERS EQUIPPED WITH 16 I/O CHANNELS:	
00200-00217	Unassigned
00220-00237*	Continuous data mode (channels 8-15)
00240-00257*	Output buffer control registers (channels 8-15)
00160-00277*	Input buffer control registers (channels 8-15)

*When not assigned for these functions, the locations may be used for data storage.

An additional 128 words of control memory are optional.

2. 32-Word NDRO Memory (Bootstrap). The computer is provided with 32_{10} nondestructive readout memory locations (00500_8-00537_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 in that they are transformer cores which operate in a special type of nondestructive read-out mode. They are not accessible to the programer for store-type instructions. The bootstrap program is assigned to memory locations 00500_8 to 00537_8 inclusive.

MAJOR COMMAND SEQUENCES— I SEQ-READ INSTRUCTION FROM MEMORY & INDEX MODIFICATION
R SEQ-READ OPERAND FROM MEMORY
W SEQ-WRITE OPERAND INTO MEMORY

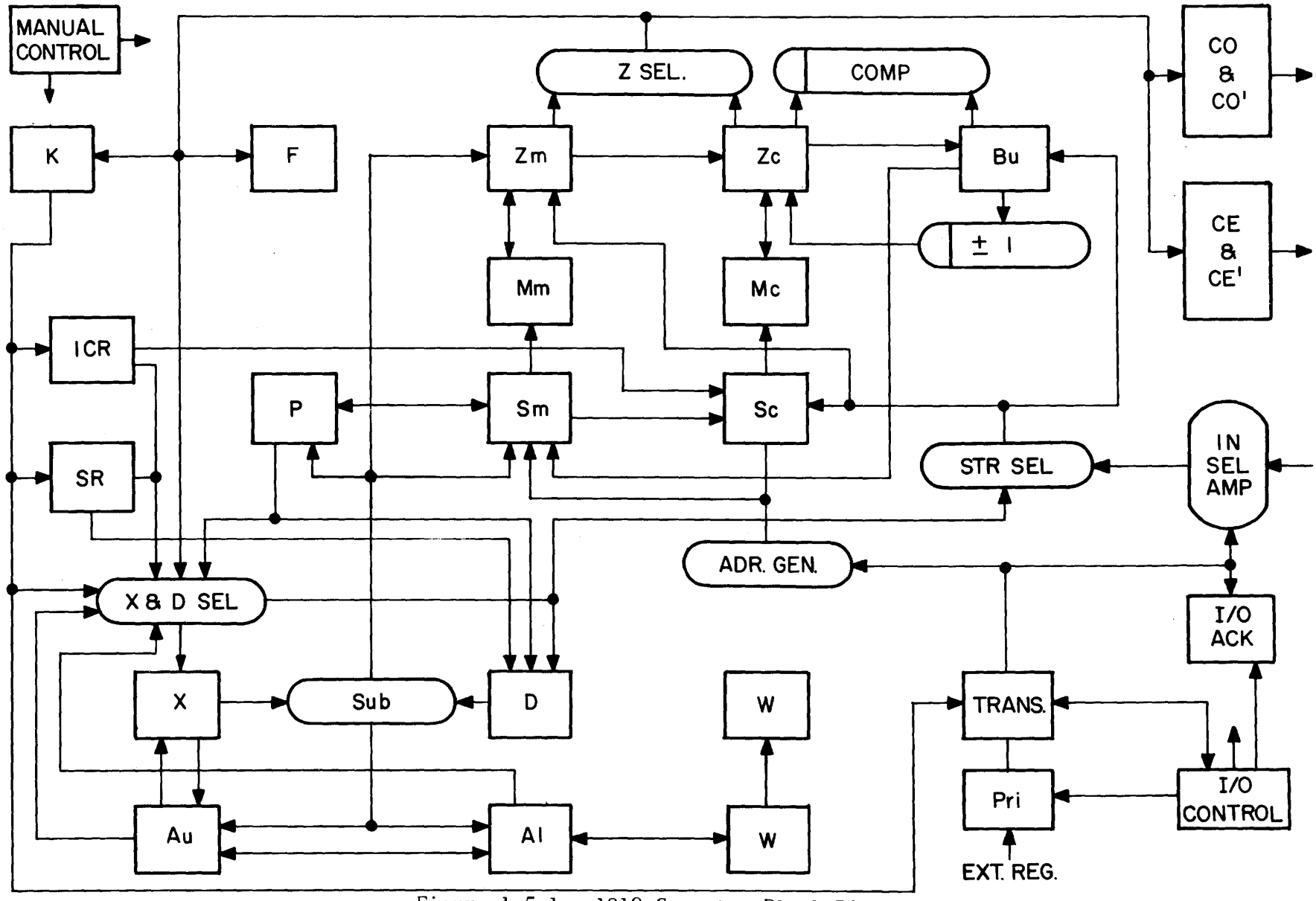


Figure 1.5-1. 1219 Computer Block Diagram

3. Main Memory. The main memory consists of a 2-microsecond core storage that is used for program, constants, and data storage. All locations are accessible to the programer at random and to all sections of the computer on a time-shared basis. Some locations are given special assignments which the programer must respect and provide for their contents. The fixed addresses assigned to main memory are listed in table 1.5-2.

TABLE 1.5-2. MAIN MEMORY FIXED ADDRESSES

Address	Assignment
000100-000117	External interrupt registers (channels 0-7)
000120-000137	Unassigned
000140-000157	Output monitor interrupt registers (channels 0-7)
000300-000317	External interrupt registers (channels 8-15)
000320-000337	Unassigned
000340-000357	Output monitor interrupt registers (channels 8-15)
000360-000377	Input monitor interrupt registers (channels 8-15)
000400-000477	Unassigned*
000540-177777	Unassigned (for computers equipped with 65K memory)
000540-077777	Unassigned (for computers equipped with 32K memory)
000600-000677	Unassigned*

* These addresses are in control memory when 256 words of control memory are used.

4. Main Memory and Control Memory Concurrent Operation. The master clock in the UNIVAC 1219 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 2 microseconds. All control and timing sequences for the various functions the computer performs are based on this 2-microsecond cycle. Four 500-nanosecond control memory read-write cycles occur during one main memory read-write cycle. An instruction from main memory storage can be transferred to the control section for execution in approximately 0.9 microseconds. Any modification to this instruction and complete trnaslation is completed before the end of that main memory cycle since the modifiers are extracted from control memory in less than 250 nanoseconds. The input/output section has independent access to control memory for its control words, clocks, etc., during instruction sequences.

c. Arithmetic Section. The arithmetic section performs numeric and logical calculations. Though greatly simplified, figure 1.5-2 shows the important components of the arithmetic section: AU-, D-, AL-, X-, and W-registers and the adder network (subtractive type).

The AL-register (18 bits) may be thought of, for programing purposes, as a conventional accumulator. Because of the logic employed, however, the AL-register is actually only the main rank of the accumulator; the D-register serves as a second rank.

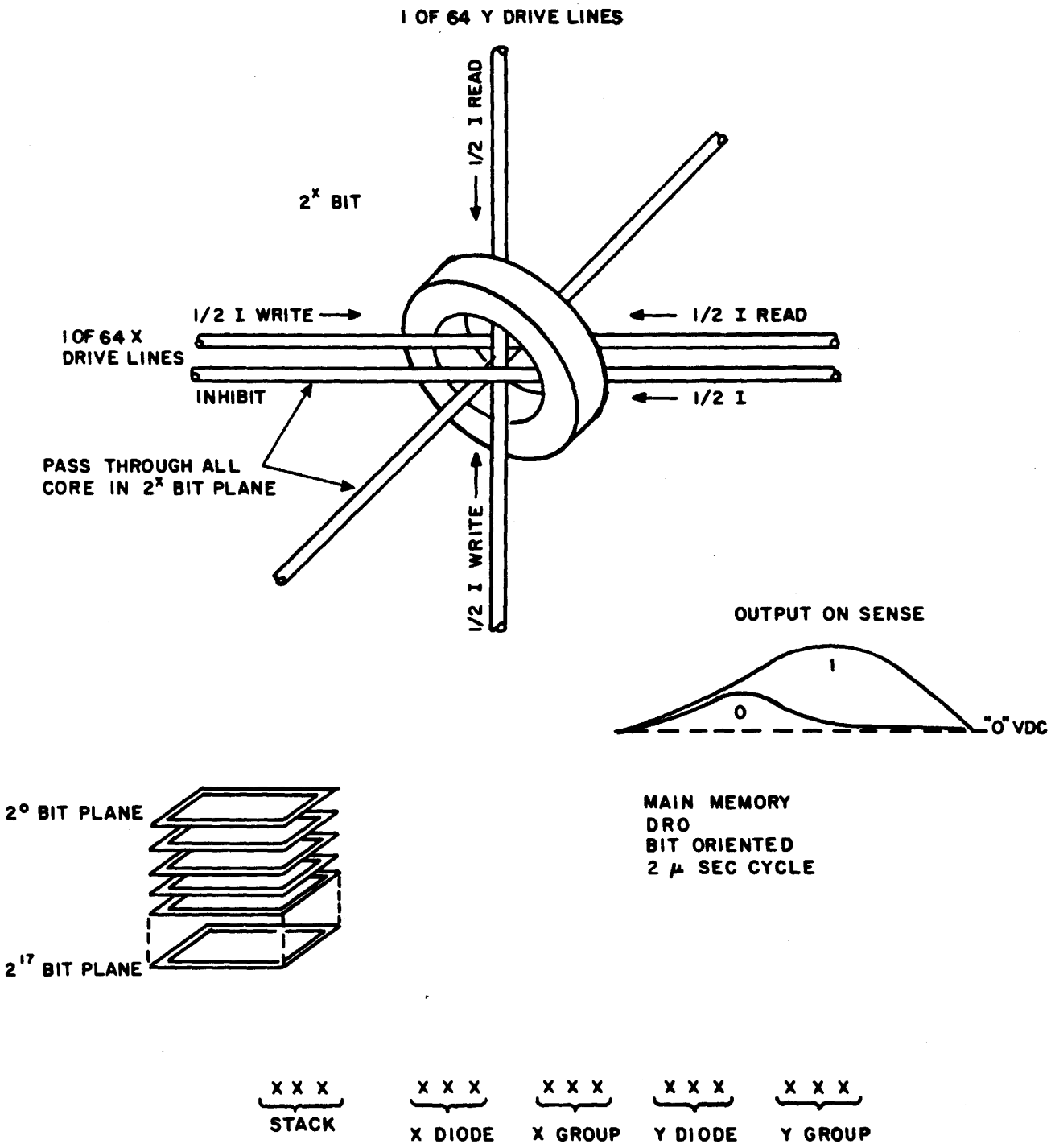


Figure 1.5-2. Core Memory Example

The ADDAL operation is typical of the relationship between the AL and D registers: the augend and addend are initially contained in AL and D. Before the addition is performed, the contents of the AL register are transmitted to the X register. The values of X and D are combined by the adder network (subtractive type) to form the sum of the two numbers in a parallel manner and are placed in the AL register.

The AU register (18 bits) is used principally during multiply and divide operations. The contents of both AL and AU may be shifted left or right, either individually or as one double-length 36-bit word. The AU and AL registers may also be combined to do 36-bit arithmetic operations.

The X-, D-, and W-, registers are 18-bit, nonaddressable registers. These registers are used primarily either for the exchange of data within the arithmetic section or for communicating with the remaining sections of the computer. The X-, D-, and W-registers are not displayed on the control panel of the computer; the AU-, and AL-registers do have indicators which allow the operator to inspect the contents of these registers during debugging operations.

The arithmetic section is used by the control section of 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.

d. 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 synchronous, electronic master clock. This section controls all arithmetic, logical, and sequential operations of the computer except those assigned to the input/output section. It has facilities to permit an interruption of the running program when certain real-time events require such interventions.

Timing commands within the computer are a function of the master clock circuit and the main timing cycle circuits. The master clock circuit produces the four basic timing pulses used to establish the operation of the main timing cycle circuits and provides the command enable pulses through gating by the main timing chain.

The master clock generates and distributes four basic timing pulses for each 500 nanoseconds of operation. These four pulses, called phases 1 through 4, constitute one complete clock cycle. Four clock cycles represent one computer memory cycle of 2 microseconds. This provides the basis for the major command sequences which are outlined in table 1.5-3.

The control section is made up of the P-, SR-, K, ICR-, and F- registers.

e. Input/Output.

1. Introduction. The input/output section includes those data paths and control circuits used by the computer for communicating with external equipment.

All communication between the computer and the external equipment is accomplished via 16 input and 16 output channels and their associated control circuits. The channels, both input and output are numbered from 0 through 17_8 with each channel consisting of 18 information lines plus control lines; channel priority ranks from 17 through 0, with the high numbered channels given preference over the lower numbered channels. Input and output communication alternates if both types of requests exist simultaneously.

The input/output section uses particular control memory addresses which dicate the memory area affected by this input or output operation (buffer control words) and particular core memory addresses which contain instructions executed when particular input/output conditions occur (interrupts).

TABLE 1.5-3. COMMAND SEQUENCES

TYPE	SEQUENCE	FUNCTION
CONTROL	1	Reads the instruction word from memory.
	R1	Reads the normal 18-bit operands from memory and performs the data manipulations as required.
	R2	Reads the second 18 bits of a 36-bit operand and performs the data manipulations as required.
	W	Writes an 18-bit word in memory and performs data manipulations as required.
INPUT/ OUTPUT	INTERRUPT	Reads the contents of the interrupt entrance registers from memory.
	B1	Reads the terminal address control word and stores it in the appropriate control memory address.
	B2	Reads the initial address control word and stores it in the appropriate control memory address.
	I/O 1	Reads from or writes into the memory, either an 18-bit word or the first 18 bits of a 36-bit word.
	I/O 2	Reads from or writes into the memory, the second 18 bits of a 36-bit word.
WAIT	WAIT	Inhibits the performance of all control sequences but permits input/output operations to continue until an interrupt occurs.

2. Input Channels. The input channels are used to receive two types of information from external equipment: input data and external interrupt information. External interrupt information originates at the external equipment and usually informs the computer of an abnormal condition such as tape breakage or incorrect parity. Input data information transfers are controlled by buffer control words in core memory.

3. Output Channels. The output channels are used to transmit data and external function information from the computer to external equipment. External function information, transmitted through the data lines, is used for external equipment control such as turn on reader, rewind tape, or turn off typewriter. Both data and external function information transfers are controlled by buffer control words in core memory.

Either an output buffer or an external function mode may be active. Both use the same assigned locations in control memory for buffer control words. Thus, initiating an output buffer will cancel any external function buffer on the same channel and initiating an external function mode will cancel any active output buffer on the same channel.

4. Priority. The higher numbered channel operation is given highest priority. Then the I/O function priority circuits provide automatic selection of the higher priority operation when two or more operations are requested by peripheral equipment or by the computer at the same time. Some real-time events as well as certain information transfers require special handling or main program intervention. These operations or interrupts are processed by the input/output section according to a prearranged priority scheme.

5. I/O Instructions. To initiate input/output buffers, 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; the format is shown in figure 1.5-3.

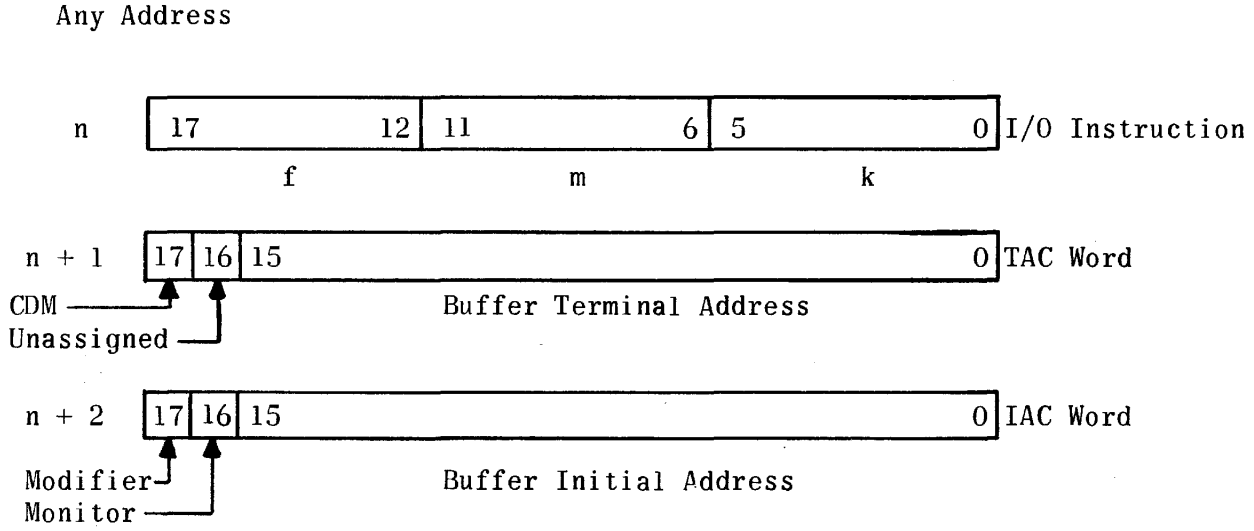


Figure 1.5-3. Buffer Initial Address

The following paragraphs apply to figure 1.5-3.

a) Bit 17. CDM of n + 1 (terminal address control word) is the continuous data mode identifier. If equal to one the computer I/O section operates in the continuous data mode. If equal to zero a normal buffer is executed.

b) Bit 17. Modifier of n + 2 (initial address control word); if equal to one the buffer current (initial) address is decremented for each word transferred in or out; if equal to zero the buffer current address is incremented.

c) Bit 16. Monitor of n + 2 (initial address control word); if equal to one the Monitor Interrupt occurs upon successful completion of the last transfer; if equal to zero no Monitor Interrupt will occur.

NOTE

Normal buffer termination occurs when the incremented/decremented buffer current address is equal to the buffer terminal address. A buffer is terminated when tests in the control section detect buffer control address equality.

6. I/O Buffer Initiating Instructions. During the execution of any instruction that initiates a buffer, three main memory references are involved. (5011-5013).

a) The I/O instruction is extracted from memory and interpreted by the control section.

b) The terminal address control word is transferred from the location following the I/O instruction to the control memory location assigned to that type buffer terminal address control word.

c) The initial address control word is transferred from the location following the TAC word in main memory to the control memory location assigned to that type buffer current address word and the specified channel is set active for that type transfer.

Computer control reads the next sequential instruction and continues the program leaving the input/output section with the task of handling the transfers. The input/output section generates the addresses in control memory to examine the control words placed there by the preceding steps, when it receives a request for word transfer from the device on the activated channel. For the actual word transfer the I/O section robs one main memory cycle from the program being executed.

The continuous data mode, requested when initiating a buffer on a channel, is a feature which provides an automatic reinitiation of the buffer upon completion. A new pair of buffer control words is transferred to the control memory buffer control addresses from the control memory CDM addresses for that channel. The monitor interrupt can be incorporated with the CDM and, if so, the interrupt will occur each time the buffer is terminated and reinitiated. The CDM is especially useful when a continuous, high rate stream of data must be transferred in or out of the computer.

7. Real-Time Processing (Interrupts). The ability of the UNIVAC 1219 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.

8. Special I/O Modes.

a) Externally Specified Index. This outstanding feature provides peripheral devices with a means of specifying core storage areas in the computer's memory for any input or output transfers they may request. The externally specified index (ESI) mode of operation is useful as multiplexing device for a number of slow transfer peripheral units occupying one dual channel. The buffer control words governing the transfers are located at the Index addresss. If input is desired an input request is presented with the index on one channel of the pair and the data on the other channel. If Output is desired, an Output Request is presented with the Index address.

b) Externally Specified Addressing (ESA). The ESA feature provides peripheral devices with a means of specifying an absolute core memory location for storage or retrieval of data. An active dual-channel mode of operation is required for computer response to this function. The address is presented on one channel and the data transmission path on the other. If input is desired the external device presents an Input Request with the address and data. If output is desired an Output Request is presented with the address.

c) Dual Channel. The Dual Channel mode provides for consecutive (even/odd numbered) channels may be "paired" to form a single 36-bit parallel channel.

d) Intercomputer. The Intercomputer mode provides for 18 or 36-bit parallel data transfers with other UNIVAC computers. No interface adapters are required for intercomputer communication.

e. 1219 Computer Physical Description. The computer was built to conform to specification MIL-E-16400 (i.e., for ruggedized equipment). The single cabinet contains power supply, logic circuits, core memory, (up to 32,768 words) control panel on front of cabinet and cooling system. See table 1.5-4.

TABLE 1.5-4. PHYSICAL CHARACTERISTICS

Physical Size and Weight: (32K, 8 I/O Channels)

Height:	71.75 inches	Depth:	30.03
Width:	25.88 inches	Weight:	900-1000 pounds

Environment:

Operating temperatures $0^{o}C$ to $50^{o}C$
Non operating temperatures -- $62^{o}C$ to $+75^{o}C$
Humidity -- Relative Humidity to 95%

Power Requirements:

115 volts $\pm$5 percent, 3-phase, 400 cps, 2000 watts maximum, air-cooled.
115 volts $\pm$5 percent, 3-phase, 400 cps, 3000 watts maximum, water-cooled.

f. Operational Characteristics. See table 1.5-5.

TABLE 1.5-5. OPERATIONAL CHARACTERISTICS

MEMORY	
Control Memory	
Cycle Time:	500 nanoseconds
Capacity:	128 18-bit words
Type:	Word organized, magnetic core
Purpose:	Index registers, clock cells, I/O buffer control registers; operates in the "shadow" of the main memory at a 4:1 ratio.
Main Memory	
Cycle Time:	2 microseconds
Capacity:	8,192, 16,384, or 32,768 18-bit words (standard options) 65,536 word memory (additional option)
Type:	Coincident current, magnetic core
Purpose:	I/O interrupt registers, program and data storage.
NDRO Memory	
Cycle time:	2 microseconds
Capacity:	32 18-bit words Word organized, transformer, unalterable
Purpose:	Bootstrap (initial load) program storage. Programs available for paper tape and magnetic tape load.
INPUT/OUTPUT	
Channels	
Type:	Simplex, 18-bit parallel
Number:	32 maximum; 16 input plus 16 output
Transfer Rate:	One channel--166,000 18-bit words/second (maximum) Multichannel--500,000 18-bit words/second (maximum)
Operation:	Each channel fully buffered and once activated operates without program attention, asynchronous, at the rate of the peripheral unit.

TABLE 1.5-5. OPERATIONAL CHARACTERISTICS (CONT.)

Information Transfers	
Input Channels:	Input data, interrupt data
Output Channels:	Output data, external command data
Processing Time Required:	2 microseconds/word transferred 0 microseconds during extended sequence instructions
Delay due to Program:	2 microseconds (maximum)
Operating Modes (Standard)	
Normal Single Channel:	18-bit parallel transfers
Normal Dual Channel:	Consecutive (even/odd numbered) channels may be "paired" to form a single 36-bit parallel channel.
Externally "Specified Index (Dual Channel)	18-bit parallel data transfers with storage address indirectly specified by external device; useful for multiplexing decommutating data to/from computer.
Externally Specified Address (Dual Channel):	18-bit parallel data transfers with storage address directly specified by external device.
Continuous Data Mode:	Program controlled automatic reinitiation of previously established buffers. Program controlled termination of CDM. 18-bit parallel or 36-bit parallel input/output word transfers.
Intercomputer Single Channel:	Direct 18-bit parallel data transfers with other Univac computers; no interface adapters required for intercomputer communication.
Intercomputer Dual Channel:	Direct 36-bit parallel data transfer with other Univac computers. No interface adapters required for intercomputer communication.
Interrupts	
Input Channels:	16 external interrupts plus 16 internal interrupts (programer option).
Output Channels:	16 internal interrupts (programer option).

TABLE 1.5-5. OPERATIONAL CHARACTERISTICS (CONT.)

CONTROL	
Instructions	
Type:	Single address.
Address Modification	8 Control-memory-contained index registers.
Repertoire:	102 instructions.
Clock	
Type:	Automatic, additive, under program control.
Location:	Control memory.
Duration:	Established under program control.
Granularity:	LSB represents 1/1024 second.
Interrupt:	Interrupt occurs when program preset value is reached.
Synchronizer	
Interrupt:	Interrupt occurs whenever the non-I/O synchronizing control line is set to logical one by an external device.
Purpose:	To allow a variable-granularity clock function or to provide a high priority alarm recognition capability.
ARITHMETIC	
Organization:	18-bit parallel, one's complement, integer.
Execution Times:	Typical execution times, including instruction and data fetch plus indexing. Add, subtract (single-length) 4 usec. Multiply/divide 14 usec. Add, subtract (double-length) 6 usec. Compare/masked compare and branch 6 usec. Register shifts: right, left, single, double 2 + .5n usec (n = shift count).

NAME: _____

1.5-5. STUDY QUESTIONS:

a. What are the four major sections of the computer?

b. How long is the core memory cycle time?

c. Why are the S and P registers 16 bits long?

d. When is the SR-register used? What precautions will the programer have to take regarding the use of the SR register?

e. When is the P-register advanced? What happens to the P-register during interrupt?

f. What is the purpose of the B-registers? When are they used and under what conditions?

g. What are the possible faults in the computer?

h. When an instruction word is read into the Z-register, to what register(s) is this information transferred?

i. What is the purpose of the CE and CO registers?

j. How is an input/output operation initiated?

NAME: _____

k. What is the meaning of the word "monitor" when used with input/output?

l. Dual option has been selected and you receive an interrupt on an even channel. How will the computer respond?

SECTION 1 - 1219 COMPUTER FUNCTIONAL DESCRIPTION

1.6. FUNCTIONAL INPUT/OUTPUT

1.6-1. OBJECTIVES

To acquaint the students with the characteristics of the input/output hardware of the 1219 computer.

To acquaint the students with the necessary program consideration for input/output operations.

1.6-2. INTRODUCTION

The input/output section of the 1219 computer is a major portion of the computer. The I/O section includes those data paths and control circuits used by the computer for communicating with external equipment. The I/O section uses particular control memory addresses which dictate the memory area affected by this I/O operation (buffer control words) and particular core memory addresses which contain instructions executed when particular I/O conditions occur (interrupts).

1.6-3. REFERENCES

a. PX 3316, Vol. I, Vol. II

b. PX 3288, Programers Reference Manual, Sections, I-B and I-E

1.6-4. INFORMATION

a. General Operation.

1. Introduction. Communication with the UNIVAC 1219 Military Computer is carried on in an 18-bit parallel mode. The computer is provided with up to sixteen input and sixteen output channels, each logically independent of the others and each brought to its own cable connector (32 connectors in all). Each channel contains 18 information lines plus control signals. If it is desired to communicate with an external device requiring more than 18 bits of parallel data, a dual channel option may be selected by one of four switches on a control panel. The selected option logically combines a pair of sequentially numbered even and odd channels to form a single channel having 36 bits of parallel data plus one set of control lines. All signals on information lines and control lines are at two d-c levels which may be changed upon interchange of information. These may be held stable for microseconds or days, depending on the nature of the particular task.

The computer is scanning for input/output or interrupts during the time it is transferring input/output data, performing an instruction, or at any time when the timing

chain is running. If it finds an input or output request it will not look for interrupt since I/O scan is performed ahead of interrupt scan. If both Input and Output requests are present, the data scan will alternate service--if not both present it will honor either.

All references to input or output in the documentation are made from the standpoint of the computer; that is, "input" is always input to the computer and "output" is always output from the computer.

Input/output is carried on in a request acknowledge basis. The request is a control signal sent by the peripheral device to the computer, telling the computer that peripheral device is ready to communicate with the computer in some given mode. The acknowledge is a control signal that originates in the computer and serves to inform the peripheral device that the computer has performed the requested transfer.

2. Control Signals. There are four basic types of transfers between the computer and peripheral equipment. They are data transfers or command code transfers. On the output cable the computer can either send information (output data) to be handled by the peripheral device or a command code (external function) which serves to tell the peripheral device to perform some operation. These two types of transfers are differentiated from one another in the peripheral device by observing which control signal (acknowledge) accompanied the transfer.

On the input cable the computer can receive from the peripheral information (input data) or a command code (external interrupt), which serves to inform the computer of the status of the peripheral equipment. These transfers are differentiated from one another in the computer by observing which control signal (request) accompanied the transfer. See Table 1.6-1 for an explanation of specific control signals.

All input and output statements are made with reference to the computer; that is, "input" is always input to the computer and "output" is output from the computer.

Examples will clarify the uses of the control lines. Figure 1.6-1 shows the computer communicating with a peripheral equipment over both input and output cables.

Notice the direction of information flow. Request and interrupt signals always originate at the peripheral equipment. Acknowledge and external function signals always originate at the computer. Examples are given in the following paragraphs.

a) Normal Input Sequence For Data Transfer to the Computer From Peripheral Equipment.

1) Program control initiates input buffer for given channel.

2) Peripheral equipment places data word on information lines.

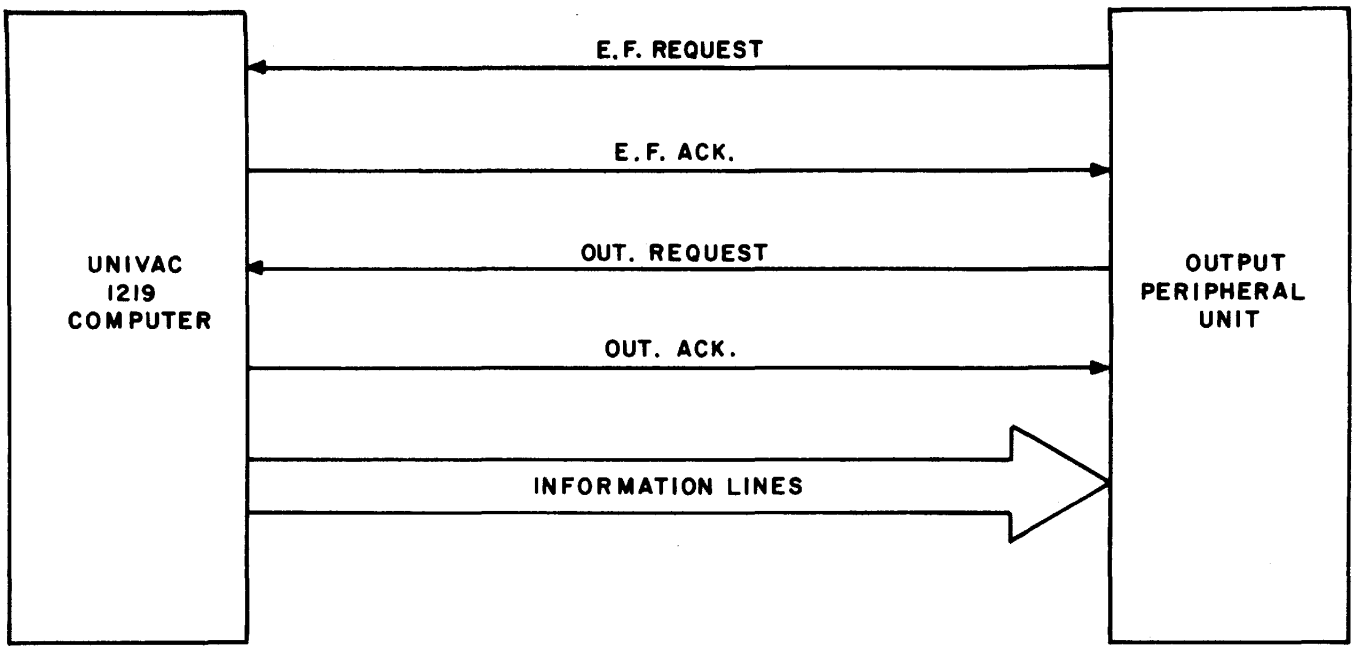
3) Peripheral equipment sets the input request line to indicate that it has data ready for transmission.

4) Computer detects the input request.

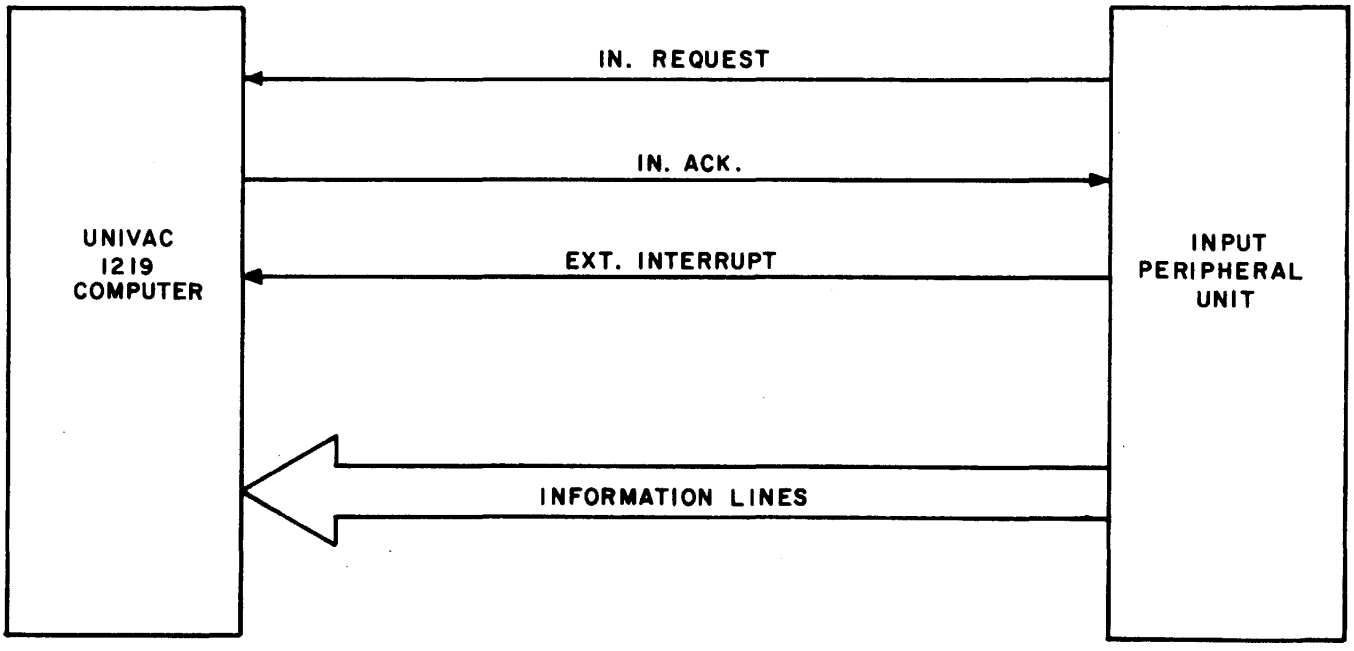
5) Computer samples the information lines at its own convenience.

TABLE 1.6-1. DESCRIPTION OF INPUT OUTPUT CONTROL SIGNALS

	SIGNAL NAME	ORIGIN	MEANING
Input Channel	Input Request (IR)	Peripheral Equipment	I have a data word on the input lines ready for you to accept.
	Input Acknowledge (1A)	Computer	I have sampled the word on the input lines.
	External Interrupt (EI) (Request)	Peripheral Equipment	I have an interrupt code word on the input lines ready for you to accept.
Output Channel	Output Request (OR)	Peripheral Equipment	I am in a condition to accept a word of data from you.
	Output Acknowledge (OA)	Computer	I have put a data word for you on the output lines; sample them now.
	External Function (EF) (Acknowledge)	Computer	I have put an external function message for you on the output lines; sample them now.
	External Function Request (EFR)	Peripheral Equipment	I am in a condition to accept an external function message from you.



ONE 1219 OUTPUT CHANNEL



ONE 1219 INPUT CHANNEL

Figure 1.6-1. 1219 Input/Output Connections

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

7) Peripheral equipment senses the input acknowledge line.

8) Peripheral equipment drops the input request line.

Steps 2 through 8 of this sequence are repeated for every data word until the number of words specified in the input buffer have been transferred.

b) Sequence For Peripheral Equipment When Transmitting an Interrupt Code to Computer.

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

2) Peripheral equipment sets the interrupt line.

3) Computer detects the interrupt.

4) Computer samples the input lines and stores interrupt code in memory location 101 plus twice the channel number (281 plus twice the channel number for channels 10_8 to 17_8).

5) Computer sets the input acknowledge lines, indicating that it has sampled the information and when no data request or interrupt lockout exist, it reads its next instruction from memory location 100 plus twice the channel number (280 plus twice the channel number).

6) Peripheral equipment senses the input acknowledge line.

7) Peripheral equipment drops the interrupt signal.

8) Peripheral equipment may change the data lines any time 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, IR may be dropped but data lines must remain stable for not less than four microseconds. If, during this four microsecond interval, an IA is received, the peripheral equipment may assume successful transfer of 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.

c) Normal Output For Data Transfer From Computer to Peripheral Equipment.

1) Program control initiates output buffer for given channel.

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

3) Computer detects output request.

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

5) Computer sets the output acknowledge line, indicating that data are ready for sampling.

6) Peripheral equipment detects the output acknowledge.

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

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

9) 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 has 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.

d) Sequence For Computer Transmitting External Function Messages To Peripheral Equipment.

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

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

3) Computer detects external function request.

4) Computer (at its convenience) places external function message on the output lines.

5) Computer sets the external function line indicating that an external function message is ready for sampling.

6) Peripheral equipment detects the external function.

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

8) Peripheral equipment samples the external function message on the output lines.

9) Computer drops the external function.

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 has 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 message regardless of the state of the EF 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.

3. Rules for Use of Control Signals.

a) Request Signal. A request signal (or an interrupt) once set, must remain set until it is acknowledged. This is necessary to maintain synchronism in the passing of data words back and forth between units. There is one exception to this rule, mentioned previously under paragraph 1.6-4a2. It is permissible for a peripheral device to drop its input request in order to interrupt when it needs to give the computer an urgent message and data loss is of secondary importance. This case is philosophically similar to the computer sending a forced external function, since the computer risks destroying data or an unexecuted command when it sends an urgent external function message.

b) Information Lines. Information lines must be stable at the time they are gated into storage elements. This self-evident axiom dictates the timing relationships between the information lines and the control signals. In the case of computer output, this rule requires the computer to provide a suitable delay between gating information into output registers and raising the output acknowledge or the external function signals, and similarly requires the computer to drop the output acknowledge or the external function a suitable interval before changing the information on the lines. Thus, the peripheral equipment is guaranteed that output lines will be stable for sampling any time the output acknowledge or external function is present. In any case of computer input, the computer detects the input request or the interrupt before the sampling of the data lines. The peripheral equipment cannot change the information lines after raising either its input request or interrupt until an acknowledge has been received, indicating that the data lines have sampled (except as stated in a) above).

c) Control Signals. All control signals will be resynchronized to ensure that the control line has been returned to a logical zero state between successive recognitions of control signals (being reset to the one state). Thie requirement guarantees that only a single recognition pulse will be generated each time the control signal is set to a one state, and also is a safeguard against false gating of data lines caused by noise or other spurious signals appearing on the control lines. There is no such restriction on the information lines. These need not be cleared to zeros between successive words.

4. Interface Option. The 1219 computer is available in two interface options, fast or slow. The fast interface option is characterized by interface circuits that operate with voltage levels of 0 vdc, a binary one, a -3 vdc, a binary zero. The slow interface is characterized by interface circuits that operate with voltage levels of 0 vdc, a binary one, and -15 vdc, a binary zero. In this mode, additional time is allowed for control signals. To operate in either option the correct option must be selected on the INTERFACE FAST/SLOW switch and the correct printed circuit cards must be inserted in the computer. (See figures 1.6-2 and 1.6-3.)

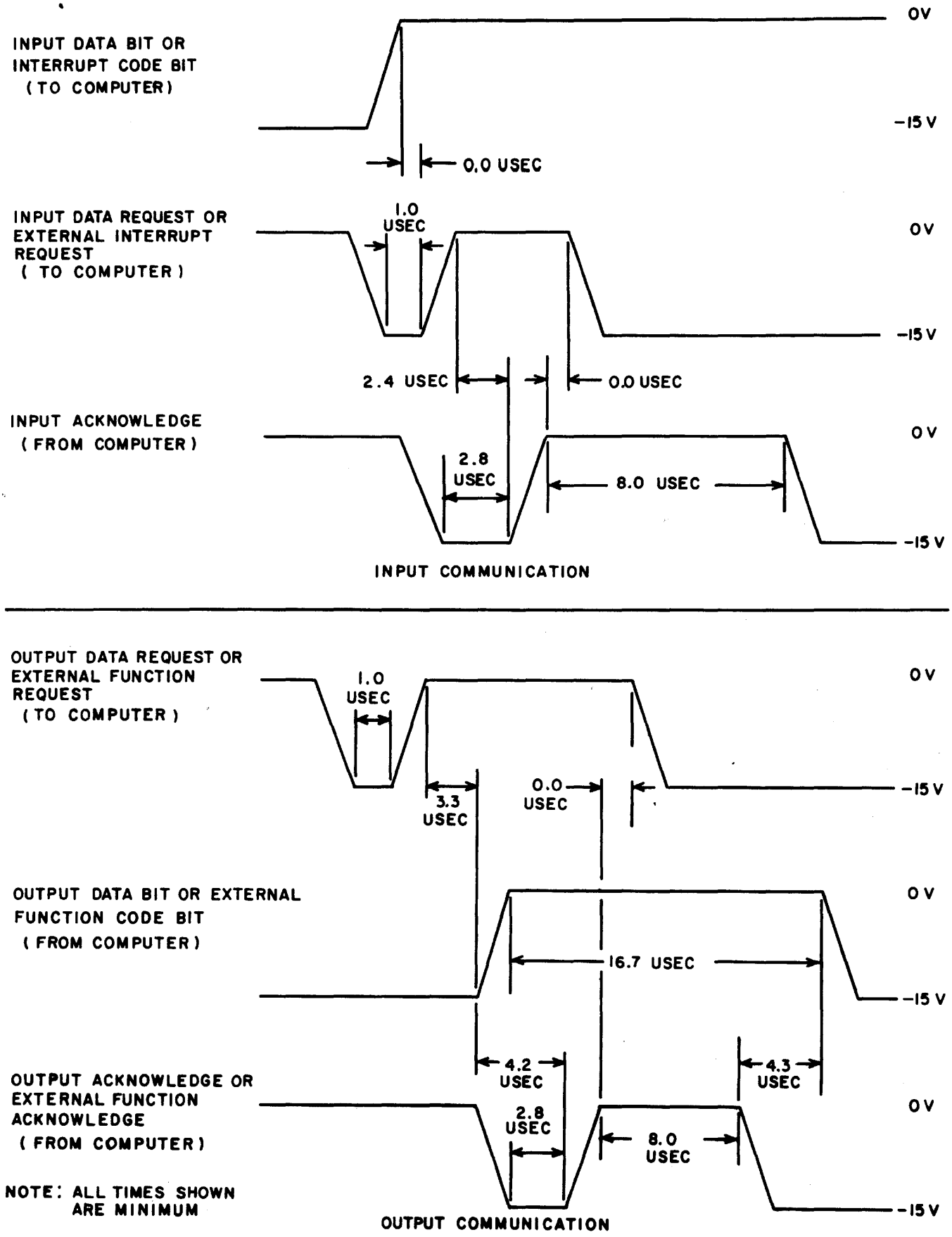


Figure 1.6-2. Slow Interface Timing

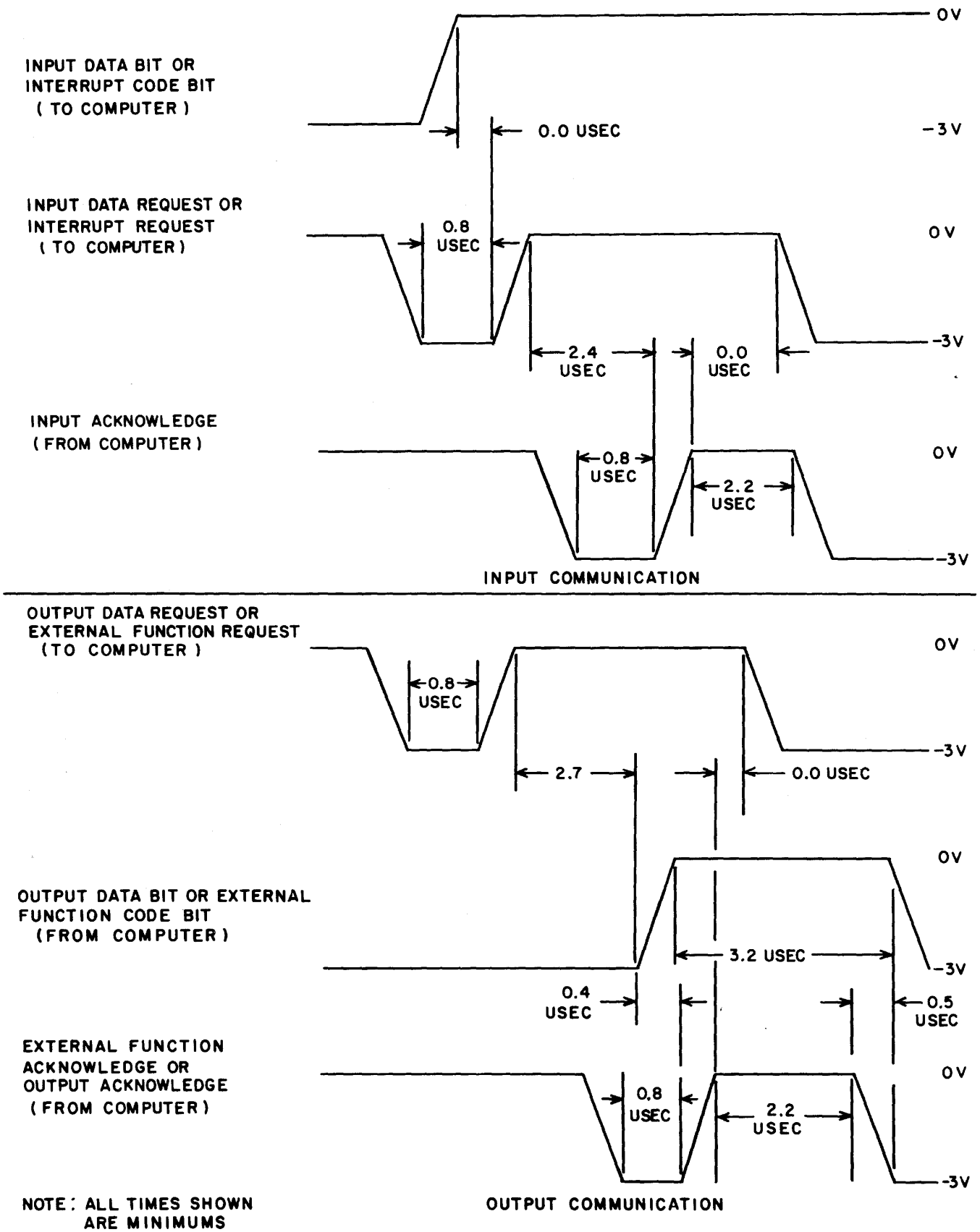


Figure 1.6-3. Fast Interface Timing

b. Interrupts and Assigned Interrupt Addresses.

1. General. Interrupts in the computer system may cause main program intervention. An instruction located in a core memory address, designated by the condition causing the interrupt, will be normally executed.

The instruction located in the designated memory location is chosen by the programer and is usually an indirect return jump. A return jump instruction stores the address of the next sequential instruction of the main program, so that computer control can return to the main program. External interrupts are processed after the interrupt code is stored in the next sequential address following the interrupt entrance register. The code is stored at the time the computer honors the interrupt from the peripheral device. When an interrupt is processed, the computer will generate the address required to call out the instruction from the assigned locations as well as the address for storage of the interrupt codes.

2. Classification of Interrupts. Interrupts can be classified as special or channel interrupts.

a) The Special Interrupts.

1) Fault Interrupts. A fault interrupt is a special case of internal interrupt caused by executing a meaningless function code, i.e., 00, 01, 77, or 5000, or 5077.

2) Intercomputer Time Out Interrupt. The intercomputer time-out interrupt is available during intercomputer operation. Any single bit of the RTC incrementing register may be wired to monitor the resume circuitry. When the RTC count reaches the specified bit, a designator is set. If no resume is received by the computer before the next time the count reaches that bit, the intercomputer time-out interrupt is activated and the next program instruction is taken for the IC interrupt entrance register (location 11).

3) Real-Time Clock Monitor. The real-time clock (RTC) monitor interrupt causes an intervention in the computer program when the value of the real-time clock monitor word stored at address 00014 equals the value of the real-time clock word stored at address 00015, provided instruction 50:14 (enable real-time clock monitor) has been performed. The equality of the two RTC words is checked in the BU $\neq$ ZO network. Once the interrupt has occurred, the computer is forced to address 00012 where it can enter a subroutine for RTC processing.

4) Real-Time Clock Overflow. The RTC overflow interrupt occurs only when an addition of one to the contents of the BU register (the RTC word) changes it from all binary ones to all binary zeros. Using the 1024-cps RTC, the time interval, measured from an all zero RTC word until it is all zeros again, is 256 seconds. When the overflow interrupt occurs the computer is forced to address 00016.

5) Synchronizing Interrupt. A synchronizing interrupt (not associated with any input or output channel) is provided on the computer via a single line. Whenever certain events occur at some external device which requests the computer to perform a given routine, this synchronizing input will be used to alert the computer. High priority is given to the interrupt and control is transferred to the instruction located in memory address 00016.

b) Channel Interrupts.

1) External Interrupts. External interrupts originate in peripheral equipment. The peripheral equipment places a code on the input data lines and sends an interrupt request to the computer. The code is stored in assigned memory locations in main memory, (101 + twice the channel number for channel 0-7)* during an I/O-sequence or sequences. At some later time the computer will generate the address required to call out the instruction from the appropriate assigned memory location (100 + twice the channel number for channel 0-7)*.

2) Monitor or Buffer Termination Interrupts. Monitor interrupts are generated by the input/output section of the computer whenever a buffer, which has been initiated with monitor imposed, terminates. This does not result in any code storage except in the case of ESI mode. In the ESI mode the sub-channel address is stored in the odd address. Use of monitor is a program option.

The assigned memory locations for output or external function monitor are 140 + twice channel number (channels 0-7).** The assigned memory locations for input monitor are 160 + twice channel number (channels 0-7).**

c. Priorities.

1. General. The input/output section of the 1219 handles two basic types of operations. These operations are the transfer of data or codes between peripheral equipment and the computer, and the processing of interrupts.

The 1219 scans for transfer requests or interrupt requests during the execution of instructions or the processing of I/O operations. These scan functions can be divided into two types. These two are the data scan and the interrupt scan.

Each scan is responsible for gating requests and selecting the order in which requests will be processed. These are accomplished according to definite priority schemes.

In the priority scheme the higher numbered channel activity is honored first. The scanning of functions for priority determination is based on the computer's 2-microsecond cycle time. During any major sequence (i.e., one that requires a memory reference) one data scan cycle transpires during the first microsecond and one interrupt scan follows in the next microsecond if some inhibiting condition does not exist.

---

* For computers with 16 channels the assigned memory location for the second 8 channels are 261 and 260 + twice the channel number for code storage and instruction.

** For computer with 16 channels the assigned memory locations for the second eight channels are 320 + twice the channel number for output or external function monitor, and 340 + twice the channel number for input monitor.

2. Data Scan Priorities. The data scan sequence, after determining channel priority, selects top function subpriority in the following order:

a) RTC update.

b) External interrupt status word storage.

c) Output request (or external function) if preceding I/O scan sequence honored input.*

d) Input request; if preceding I/O scan honored output or E.F. output or input if only one request exists.*

If none of the above functions are detected during the I/O scan sequence the interrupt scan is initiated to operate during the next microsecond of the computer cycle. If a request is detected, the I/O sequence is initiated. It processes the request being honored. Scan for interrupts is inhibited during extended sequences because interrupts cannot be honored during the execution of an instruction. The scanning sequence for interrupts is not effective when an interrupt lock-out exists.

3. Interrupt Scan Priorities. The interrupt scan sequence selects top interrupt priority as follows if no lock-out exists.

a) Fault.

b) Intercomputer time-out interrupt.

c) RTC monitor interrupt.

d) RTC overflow interrupt.

e) Synchronizing interrupt.

4. Conditions to Inhibit Scan Functions.

a) Data Scan.

1) I/O = 1 and dual mode.

2) I/O = 1 and ESI and terminate.

---

* The I/O section of the 1219 computer is constructed in four channel modules. These modules comprise one computer chassis. A given chassis would contain all of the odd or all of the even channels in either the high or low eight channel group, i.e., channels 0, 2, 4, and 6, would be on one chassis, channels 1, 3, 5, and 7, would be on a second chassis. Function priority is determined in each chassis. Therefore, the function priority applies for each set of 4 channels. For example, it may happen, assuming proper requests, that the computer would process an output request on channel 7, and then an output request on channel 6 (another chassis) before doing any input operations.

Dual mode processes the odd channels according to function priority.

3) Continuous data request.

During R-1 of function 20-23 and 57 scan runs but the setting of I/O-1 sequence is inhibited.

b) Interrupt Scan.

1) During any I/O sequence or instruction.

2) After accepting an interrupt and until the interrupt lockout is cleared.

3) During compare instructions.

d. Modes.

1. Single Channel Operation. Input and output data transfers are 18-bits parallel. Two 24-pair cables connect the peripheral unit to the input/output channel for two way communication. Input and output connections are physically separated for each channel.

2. Dual Channel Operation. Users of the computer have the option of communicating over the 18-bit parallel single channels or of logically combining sequential even and odd numbered channels into a 36-bit parallel dual channel. This option is selected by one of eight switches (16 channels) on the control panel. Selection of this dual channel option affects only that pair of channels selected, but both input and output channels of a given channel number are combined by a single switch setting. For example, if the first switch is activated, input and output channels 0 and 1 are combined to form a 36-bit input and a 36-bit output channel, but input and output channels 2 through 17 remain 18-bit, logically independent channels. If this dual option is selected, the set of control lines belonging to the odd-numbered option is selected, the set of control lines belonging to the odd-numbered channel will have control of the information transfers over all 36 lines.

With this dual option selected, energizing of the request lines or the interrupt line on the even-numbered channel will cause the computer to interpret this as a desire to communicate in a single channel mode and the computer will reply over the even-numbered set of control lines. Eighteen bits of information will be accepted from the peripheral equipment on the even-numbered set of lines, and, as in any single channel operation, identical information will appear to both sets of output lines.

3. Externally Specified Indexing (ESI). ESI is usable only in a dual channel mode. The I/O mode switch must also be set to ESI on the computer control panel. The index address word is always placed on the set of input lines for the odd-numbered channel. If the peripheral equipment desires to send an input word, it will place the input data word on the 18 lines for the odd-numbered channel. If the peripheral equipment desires to send an input word, it will place the input data word on the 18 lines of the even channel and raise the odd input request line. The computer will reply on the odd input acknowledge line. If the peripheral equipment should raise the even IR line instead of the odd, the computer will interpret this as normal single channel communication and ignore the index address, as explained above. The program must provide an active channel for the ESI operation (instruction 5001, 5002, 5011 or 5012).

If the peripheral equipment wants a word of output data, it will place the index address on the odd-numbered set of input lines and raise the odd output request. The computer will reply with 18-bits of data, duplicated on both sets of output lines, with the odd channel output acknowledge. Activation of the even-numbered output request will similarly cause the computer to ignore the ESI feature.

Because the I/O control words are located in control memory, a limitation is imposed on the ESI mode of operation. For this reason the control memory may, by option, be expanded to 256 words to gain additional control word storage. In the 1218 some delay arises when a subchannel buffer termination occurs as the computer program must search all subchannel buffer control words to determine which buffer terminated. This has been improved in the 1219 by automatically storing the subchannel address into a fixed location (the vacant address associated with the channel monitor interrupt entrance address) when termination occurs. In this manner the program can immediately determine which subchannel terminated.

A sequence of events for output from a computer using ESI is as follows:

a) The computer program provides an active output channel to the equipment.

b) The external device places a 6-bit control memory index address (128 word control memory) N/2 on the odd input channel. The subchannel address is left-shifted one place on input.

NOTE

Addresses n and n + 1 are treated as normal buffer control words with the same buffer control options available (refer to section on control words). When the monitor interrupt occurs during an ESI input/output sequence the index address N/2 is stored in the associated monitor interrupt status word location (odd address) and the channel is deactivated. The subchannel address is not left-shifted on input when stored in an assigned memory location.

c) The external device sets the output request control line on the odd output channel.

d) The computer detects the output request and at its convenience reads and compares the addresses stored in n and n + 1.

e) The computer transfers the word from the address located in n + 1 to both output channels.

f) The computer sets the output acknowledge line on the odd channel.

4. Externally Specified Addressing (ESA). ESA is a switch-selectable dual channel mode of operation that allows external devices random access to core memory for retrieval and storage of data as opposed to the continuous addressing scheme associated with normal buffers. Individual and/or random entries in a data pool lend themselves to an effective use of ESA. The external unit must be capable of presenting the absolute address on the odd input channel of the pair.

A sequence of events for output from the computer using ESA is as follows.

a) The computer program provides an active output channel to the equipment. (The 5002 and 5012 instructions are used to activate the channel.)

b) The external device places a 16-bit absolute address, n, on the odd channel input lines.

c) The external device sets the output request control line on the odd output channel.

d) The computer detects the output request and at its convenience transfers the address, n, to the S-register.

e) The computer places the contents of address n on both output channels of the pair.

f) The computer sets the output acknowledge on the odd channel.

The sequence of events for input is similar to output except that a data word is placed on the even input channel with the storage address on the odd channel and the input request line is raised. The computer responds with an input acknowledge. The computer channel is activated by using instruction codes 5001 or 5011.

NOTE

No additional equipment should be connected to the even-numbered channel of the dual, ESA, or ESI selected pair.

5. I/O Transfer Under Continuous Data Mode. Assume a buffer has been initiated via a 5011 03 (initiate input on channel 3) instruction, and the input transfer has been completed. If the CDM bit was set (bit 17 of terminal address buffer control word), the contents of control memory addresses 26 and 27 would be transferred to addresses 66 and 67 and the input buffer for channel 3 will have been re-initiated without program attention. Before the buffer, defined by the new BOW's, has been completed, the program has the option of storing another set of BOW's in addresses 26 and 27 with or without the CDM bit set; this cycle continues until the program clears the CDM bit in location 26. Similar action occurs for output and external function buffers.

6. Intercomputer Communication Mode. Switches on the control panel provide the option of intercomputer communication to any or all input/output channels. The selection of a given I/O channel as an intercomputer channel has no effect upon the modes of the unselected channels. An intercomputer channel can function in either the dual or ESI mode in addition to the normal 18-bit mode.

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 function messages 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 this 4-channel group can cause the data to be changed. However, a channel

sending data or external function messages to another computer must hold the information in the output registers until the receiving computer acknowledges receipt of those words. This acknowledge signal is received on what is known as the output request line when not in the intercomputer mode. This line, in the intercomputer mode, is known as the resume line.

In the case of UNIVAC 1219-to-1219 communication,this resume line is connected to the input acknowledge line of the receiving computer. Activation of the resume signal on the transmitting computer channel causes the setting of the resume 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 will be tied up until the intercomputer time-out interrupt branches to a remedial routine. During the interim no output buffers or external function buffers to other equipment on that channel group can proceed. To limit the possibility of this hang-up occurring, two instructions and the intercomputer time-out interrupt are provided by which the computer program can monitor the status of the resume flip-flop. These instructions are: Skip On No Resume (5057 k), and Set Resume (5020 k). The former allows examination of the resume flip-flop, and the later allows the program to correct the situation in which the hang-up exists. (See figure 1.6-4.)

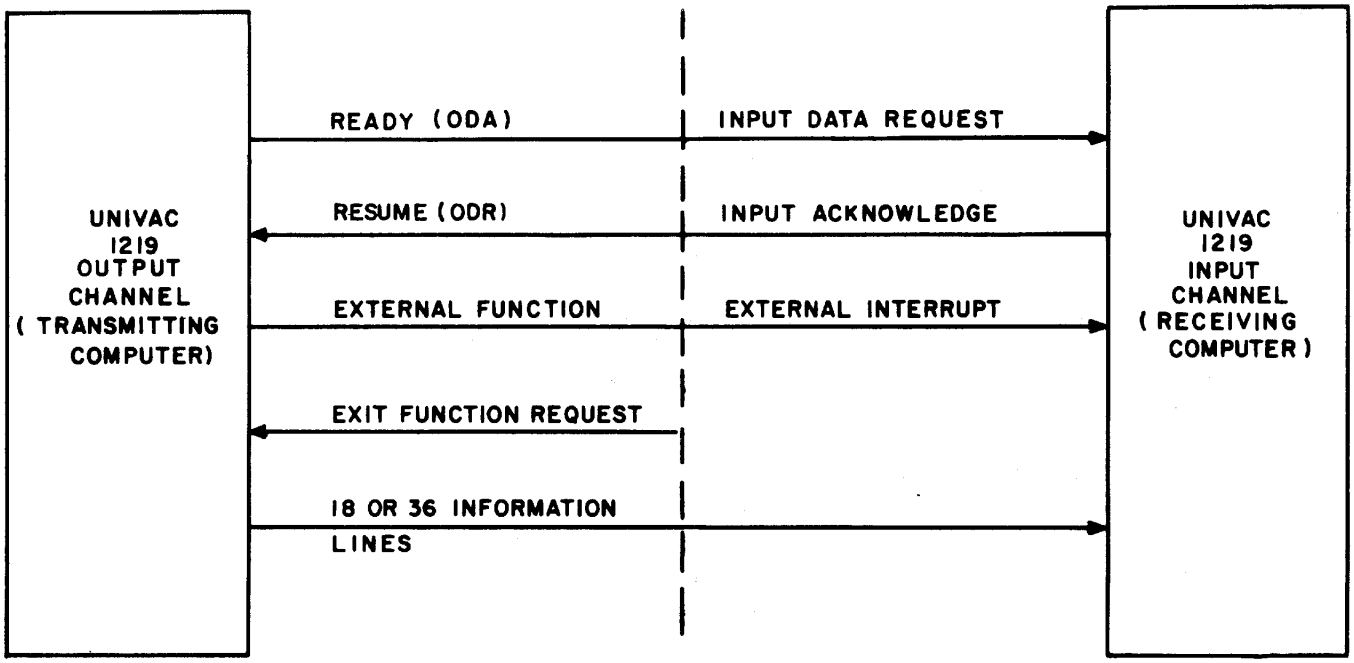


Figure 1.6-4. 1219-to-1219 Communication

e. I/O Timing.

1. General. The 1219 computer accomplishes input/output on a time-share basis with the running program. There are two basic types of operations accomplished by the I/O section, the transfer of data or codes and the processing of interrupts.

During the first microsecond of the execution of an instruction sequence the 1219 computer scans for data request, except in those cases when the scan is inhibited. If at this time the scan detects a request, the computer will hold up the execution of the instruction at the end of the first microsecond and go into an I/O sequence or sequences. The scan also normally runs during the last I/O sequence. If this scan detects a request the 1219 stays in the input/output mode and continues to inhibit the second microsecond of the instruction sequence. If this scan does not detect a request, control is transferred back to the program and the instruction sequence is completed. The data scan detects input/output, external function, external interrupt code storage and real time clock update requests. (See figure 1.6-5.)

The interrupt scan runs during second microsecond of the instruction sequence. This scan is inhibited during some conditions. If this scan detects an interrupt request, the instruction is completed and the computer performs an interrupt sequence. The interrupt sequence calls up an instruction from an appropriate assigned memory location, and sets the interrupt lockout. The instruction is generally a return or indirect return jump, or a clear the interrupt lockout instruction.

2. Input/Output Sequences.

a) Single Channel I/O Sequence (2 μsec). The I/O sequence includes three control memory (500-nsec) cycles. During this sequence the current address control word and the terminal address control words are read from control memory. The current address control word is compared to the terminal address control word. If during all this time they are found equal, the active flip-flop is cleared. The current address control word is updated and stored in control memory. The data is then transferred to or from memory at or from the contents of the address specified by the current address control word. This sequence is altered for real-time clock update.

b) Dual Channel I/O-1 - I/O-2. I/O-1 (see preceding paragraph). I/O-2. During this sequence the second half of a 36-bit word is transferred to or from the address specified by the current address control word plus one if the buffer is incrementing or minus one if the buffer is decrementing.

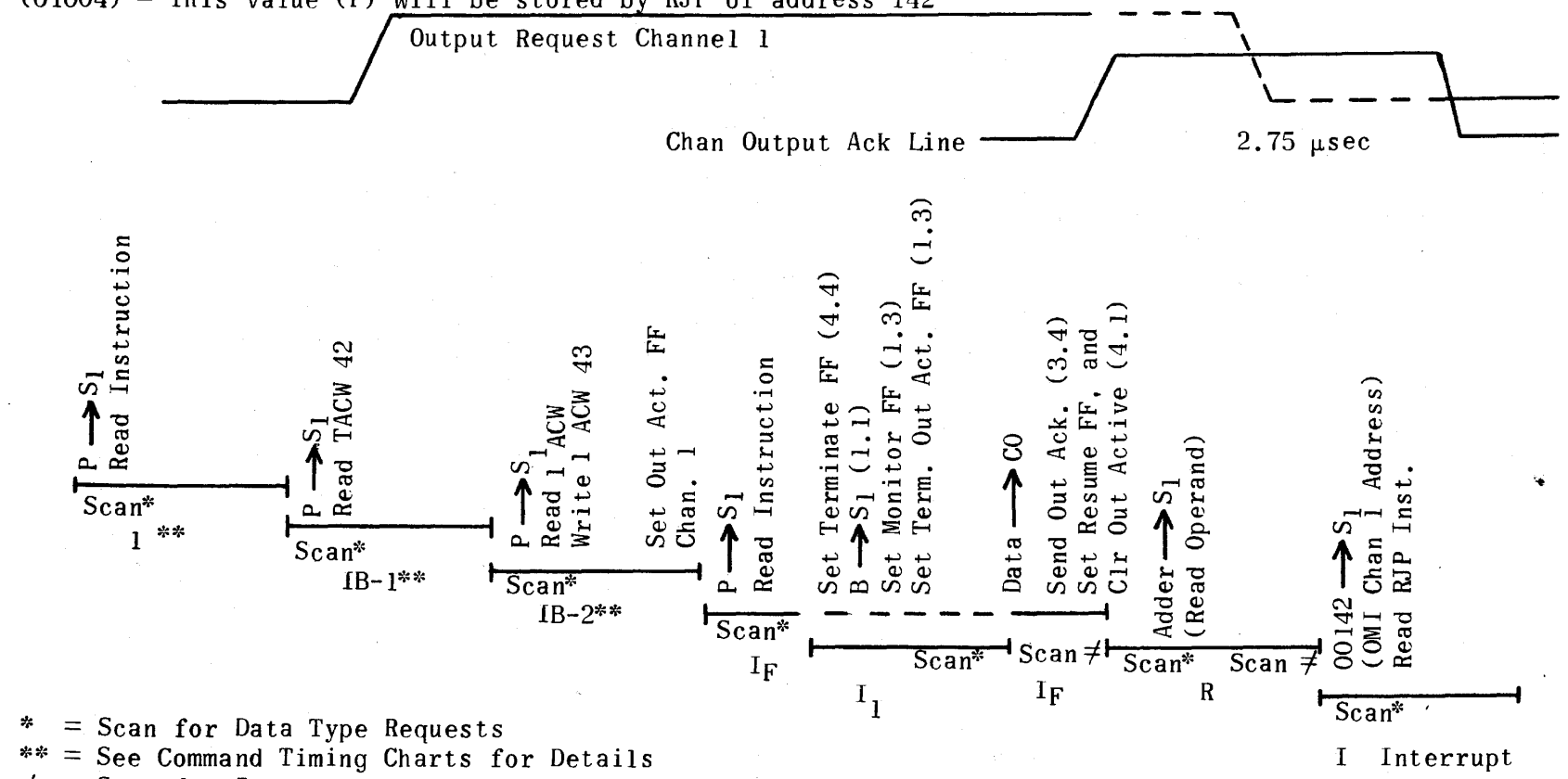
c) ESI I/O. The I/O-1 of the ESI is a normal I/O-1 sequence except that the location of the buffer control words in control memory is specified by the peripheral equipment and sent on the odd channel.

d) I/O-2. The I/O-2 sequence is run when in the ESI mode only if monitor has been requested and the buffer is terminated. During this sequence the subchannel address is stored in an appropriate assigned memory location.

e) ESA I/O-1. The address of the memory location to be transferred is input directly to the S register from the odd channel input selectors and the word is transferred to or from computer memory.

Assume the peripheral equipment has received an E.F. word to enable output, and responds with an output data request. Single channel mode has been selected. The buffer has been set up with monitor and is incremental, continuous data mode has not been requested, i.e. Bit 17 of the TACW = 0. The program continues as follows:

(01000) = 50 12 01
(01001) = 00 60 00
(01002) = 20 60 00
(01003) = 14 00 50
(01004) = This value (P) will be stored by RJP of address 142



* = Scan for Data Type Requests
** = See Command Timing Charts for Details
≠ = Scan for Interrupts
OMI = Output Monitor Int.

Figure 1.6-5. One Word Buffer With Monitor (No Higher Order Type Requests - Fast Interface)

f) Continuous Data Mode I/O Sequence. This is an additional I/O sequence that is run after an I/O-1 or I/O-2 (dual) during which the buffer control words were found to be equal. During this sequence the terminal and initial address control words are transferred from the CDM assigned memory locations for the given channel to the assigned memory for the buffer control words of the buffer that terminated. The active flip-flop is reset.

NAME: _____

1.6-5. STUDY QUESTIONS

a. General

1. What control signals originate in the computer?

2. What control signals originate in the peripheral equipment?

3. Discuss the use of each control signal.

4. When, in relation to the request, is data placed on the data lines for input? Output?

5. List in sequence all transfers of control signals and information between 1219 and a peripheral device (assume external function request) either output or input.

6. What must the programer do to enable the preceding events?

b. Interrupts

1. Define an interrupt.

2. What interrupts result in a code storage?

NAME: _____

3. How is the code word stored? Where?

4. List the interrupts that do not result in code storage and the corresponding interrupt entrance address.

c. Priorities

1. Why is a system of priorities necessary?

2. Why are there two scans?

3. Explain the purpose of each scan and why they occur when they do.

d. Modes

1. What programing considerations apply to the following? Mention the way in which control words are stored.

a) Single channel

b) Dual

c) ESI

NAME: _____

d) ESA

2. Write a program to transfer 5 words from one 1219 to another 1219. Program transmitting computer - Program receiving computer.

e. I/O Timing

1. How long does the 1219 delay the program in order to output one 18-bit word?

2. Do control and main memory cycles overlap? Explain.

3. How many control memory cycles are needed for a normal output transfer? Explain the job of each cycle.

4. Is the arithmetic section used during I/O? Why?

5. Assuming an output request, how long, in microseconds, after main memory is initiated to call up a 5012 k instruction until a word is available to the peripheral equipment? How long until the second word could be available?

6. Why does it take 4 microseconds to transfer 36 bits?

SECTION 2 - SOFTWARE

2.1. TRIM II ASSEMBLER

2.1-1. OBJECTIVES

To provide information and study questions for the student to analyze the use of TRIM II and to give the student practice in writing programs in TRIM II language.

2.1-2. INTRODUCTION

During the last week you probably have had many problems arise because you have been doing your coding in absolute machine language. You have probably come to the realization that machine coding in absolute language is difficult and very time consuming because of the necessity to keep track of all allocations of memory locations. The counting process becomes involved and human beings are likely to make mistakes in this counting so the question has arisen, why not let the machine do the paperwork and bookkeeping involved when making a program. The TRIM II assembler will do the necessary bookkeeping for the programer so that it allows him to bypass the meticulous task of machine coding. However the programer must learn the requirements of TRIM II in order to use it effectively.

2.1-3. REFERENCES

PX 3288, Programer Reference Manual, Section on TRIM II

2.1-4. INFORMATION

a. General. TRIM II is a set of instructions or program that will accept a predefined symbolic language in certain format and convert it into machine language. The assembler will make the necessary allocations for memory addresses and will also output a listing of these memory addresses and machine instructions in one of several available formats.

The method of using TRIM II is for the programer to prepare his program in the required format and then prepare a tape of this program on a standard Flexowriter using the required format. The programer has the option to request his output in one of several permissible outputs.

TRIM II is an assembler which operates on an 8K memory 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 dump of registers and memory contents wherever desired by the programer. The assembler can be instructed to ignore debugging operations if desired.

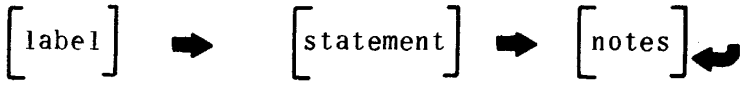
b. TRIM II Assembly System

1. Introduction. The TRIM II Assembly System for the UNIVAC 1219 Computer provides programing 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 UNIVAC 1219 computer via the 1219 utility packages.

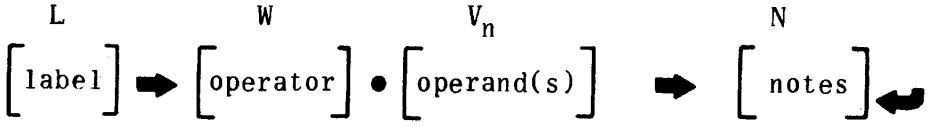
2. Description. TRIM II is a two-pass assembler designed for a minimum equipment configuration of one UNIVAC 1219 computer with 8,192 (decimal) words of core memory and one paper tape reader, paper tape punch, and console typewriter. The assembler accepts a source program expressed symbolically, absolutely, or in combination thereof and converts it into an ordered set of machine instructions suitable for loading via the 1219 utility packages.

The term two-pass means the source program tape(s) must be loaded into the computer twice. The first such loading, constituting pass one, assimilates and stores information needed for pass two. Refer to figure 2.1-1, Block Chart for TRIM II - Pass One. At the completion of pass one the source program tape(s) must again be loaded, and the desired output must be selected. Using the information accumulated during pass one, pass two reads, assembles, and punches on paper tape each source program instruction, statement by statement. This second loading constitutes pass two. Refer to figure 2.1-2, Block Chart for TRIM II - Pass Two. Subsequent outputs are achieved by repeating (reloading) pass two.

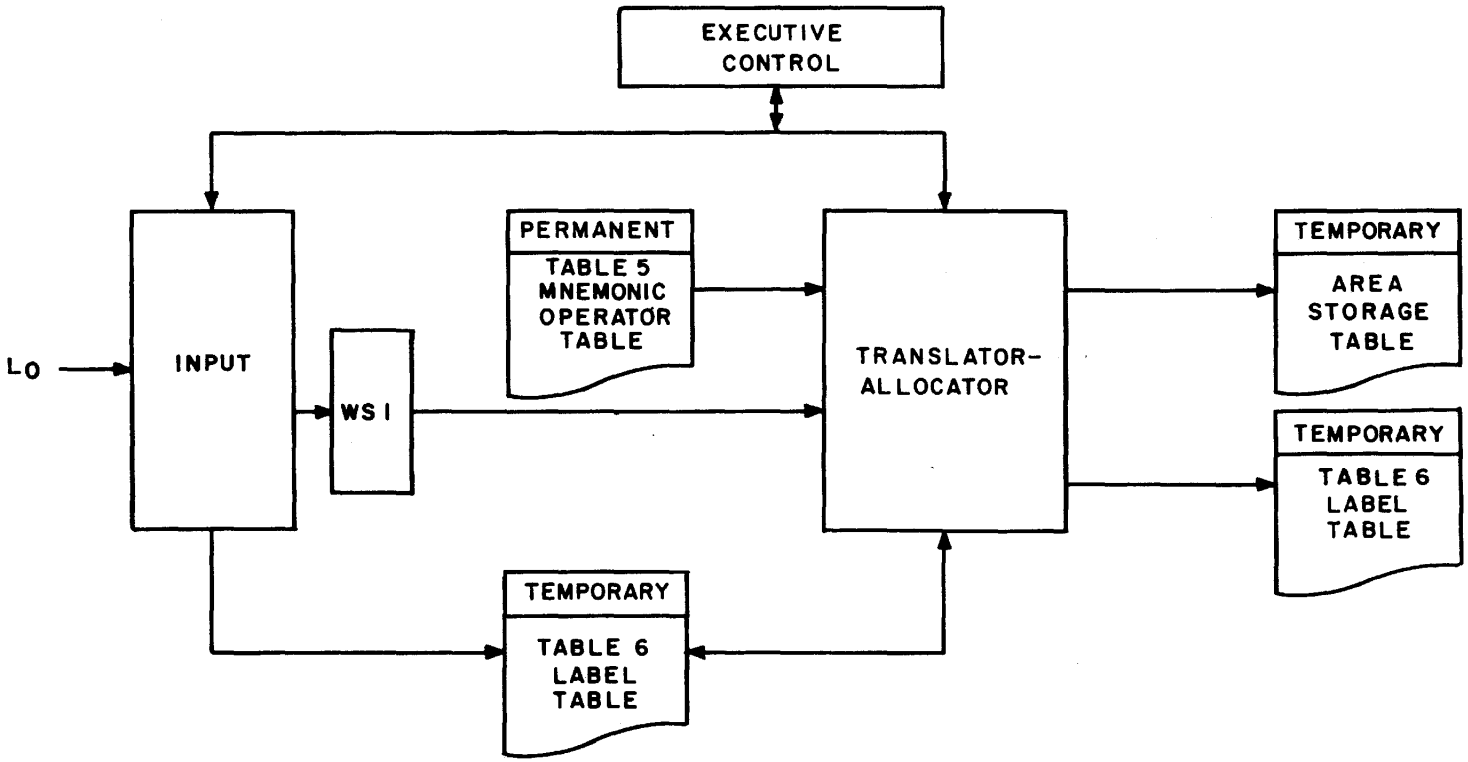
3. Source Language. A TRIM program as prepared by the programer is composed of a list of operations which perform the step-by-step processing of a problem. An operation has the following general format:



The general format may be further subdivided into:



a) 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.



INPUT:
a. READS ONE ITEM INTO WSI
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 2.1-1. Block Chart for TRIM II - Pass One

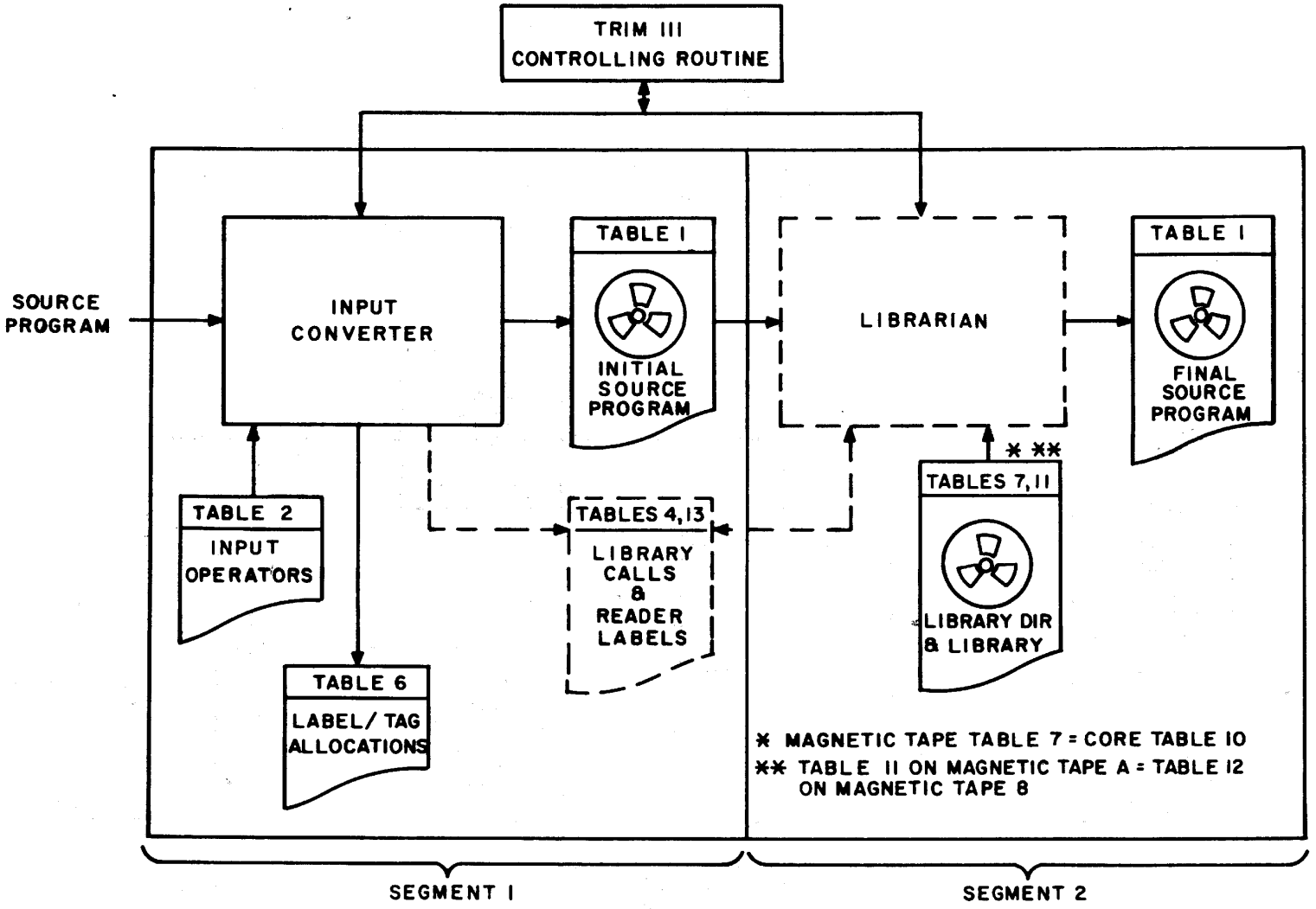


Figure 2.1-2. Block Chart for TRIM II - Pass Two

A label may consist of not more than six alphanumeric characters, but it never begins with the letter O or a number and 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 an 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.

b) Statement. The statement of an operation is made up of an operator and operand(s). The statement defines the operation. In the instruction word format it consists of the fu or fmk portions or as otherwise specified for header, declarative, debugging, or polyoperations.

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 and/or subfunction codes of the instruction word.

2) Operand(s). One of 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).

c) Notes. Descriptive notes may follow the statement; they are for the programer's use and in no way affect the instructions generated from the statement. Notes may not exceed 40 decimal characters.

d) Symbols. The programer uses a uniform set of symbols as separators in all coding. These symbols are depicted in tables 2.1-1 and 2.1-4.

4. Header and Declarative Operations. TRIM II recognizes two types of header operations.

L		W		V_0		V_1		N
POKER	➡	ALLOC	•	JONES	•	10 MAY 1963	➡	
POKER	➡	PROG	•	JONES	•	10 MAY 1963	➡	

TABLE 2.1-4. EQUIVALENT INPUT FORMAT CODES
(FLEXOWRITER vs. FIELDATA

SOFTWARE NAME	SOFTWARE SYMBOL	FLEXOWRITER CODES	FIELDATA SYMBOL SUBSTITUTION	FIELDATA CODES
Carriage Return	↵	4 5		0 4 0 3
Shift Up	↑	4 7		0 1
Shift Down	↓	5 7		0 2
Tab	→	5 1	Special ▭	7 6
Point Separator	•	4 4	Apostrophe '	7 2
Double Period	..	5 7 4 2 4 2		7 5 7 5
Space	△	0 4		0 5
Comma	,	5 7 4 6 4 7		5 6
Vertical Bar	|	57 50 4 7	Exclamation !	5 5
Plus	+	5 7 5 4 4 7		4 2
Minus	-	5 6		4 1

TABLE 2.1-3. FLEXOWRITER CODE CHART

TYPE LETTER UC	LC*	OCTAL**
A	a	30
B	b	23
C	c	16
D	d	22
E	e	20
F	f	26
G	g	13
H	h	05
I	i	14
J	j	32
K	k	36
L	l	11
M	m	07
N	n	06
O	o	03
P	p	15
Q	q	35
R	r	12
S	s	24
T	t	01
U	u	34
V	v	17
W	w	31
X	x	27
Y	y	25
Z	z	21

TYPE LETTER UC	LC	OCTAL
1	1	52
2	2	74
3	3	70
4	4	64
5	5	62
6	6	66
7	7	72
8	8	60
9	9	33
0	0	37

PERFORM TYPEWRITER OPERATION	OCTAL
Space	04
Shift up	47
Shift down	57
Back space	61
Car. return	45
Tabulator	51
Color shift	02
Color delete	77***
Stop	43***

TYPE SYMBOL UC	LC	OCTAL
- (Superscript Minus)	- (Hyphen or Minus)	56
• (Multiply)	= (Equals)	44
/ (Virgule)	+ (Plus)	54
((Open Parens)	, (Comma)	46
) (Close Parens)	. (Period)	42
_ (Underline)	| (Absolute)	50

* The upper case, UC, or lower case, LC, character is typed according to the position of the type bars.

** Codes not used are: 00, 10, 40, 41, 53, 55, 63, 65, 67, 71, 73, 75, 76.

*** Codes 43 and 77 are considered illegal codes in operations with the computer.

a) ALLOCation Header. 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 be preceded by a carriage return (and a shift to upper case for Flexcode oriented TRIM II). When the allocations are on a separate tape, the tape must terminate with a carriage return and two lower case periods. An ALLOC tape has the following format:

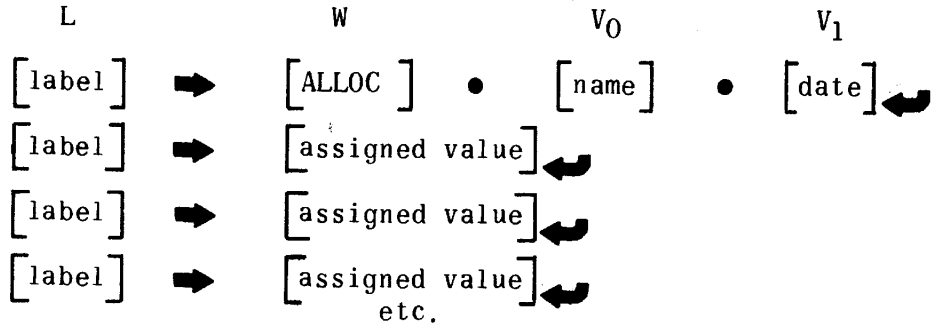


TABLE 2.1-1. TRIM II CODING SYMBOLS

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.

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, e.g.

CAT ➡ 01000↵

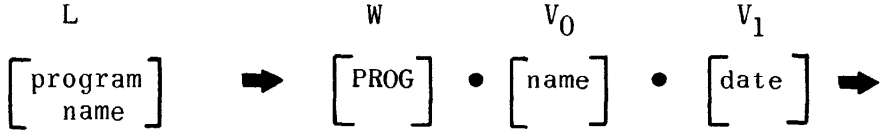
DOG ➡ 512D↵

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.

b) PROGram Header. 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 (and shift to upper case for the Flexcode oriented TRIM II). A program header has the following format:

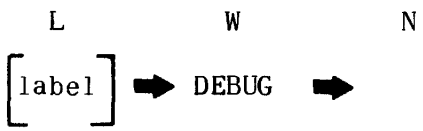


1) L. The label of the PROG header operation is optional; however, when present, it is considered to be the name of the program.

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.

c) DEBUG Declarative. TRIM II accepts the declarative operation,



The DUBUG operator informs TRIM II that generation is to be performed for debugging operations contained in the source program. If the DEBUG operator is absent, 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, e.g.,

L	W	V_0	V_1
POKER	ALLOC	JONES	10 MAY 1963
CHIPS	01234		
CHOPS	1245		
CHAPS	1388D		
	DEBUG		

5. Mono-Operations. Mono or one-to-one operations consist of mnemonic function codes and symbolic addresses, absolute machine codes, or constants.

Mono-operation statements may be in one of the following formats.

a) Format A.

W	V_0
[operator]	[operand]

1) W. The operator is the f, fb, or fm portion of the operation statement and is the mnemonic respresentation of the desired function code of the computer instruction repertoire.

2) V_0. Represents the u or k portion of the statement and may be a tag, ± 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

b) Format B. TRIM II also accepts programs coded with 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 six consecutive digits without a point separator.

Examples:

➡ 12•3505 ➡

➡ 63•CAT+6 ➡

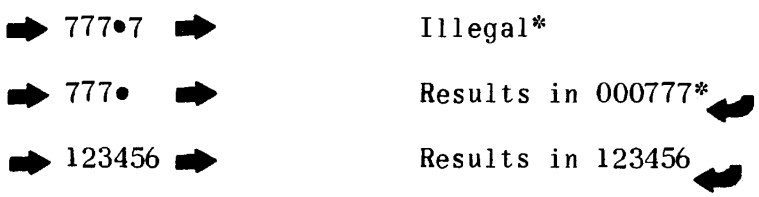
➡ 50•1306 ➡

➡ 50•6200 ➡

➡ 506200 ➡

c) 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 ↵



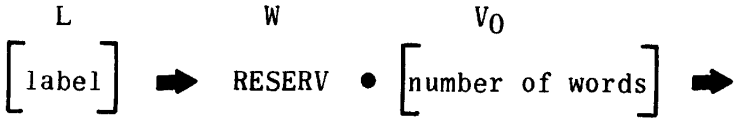
Two special mono-operations are available for the programer's use: STOP and SKP. If either of these operators i used with a k operand, TRIM II will automatically generate an unconditional instruction of 505640 or 505040 respectively.

If the programer 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 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 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.

6. 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 eleven 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 programer 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 programer's special register setting.

a) RESERVe Operation.



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.

* Whenever there is an expressed value following the point separator, only 1 or 2 digits are permitted in the operator position.

3) $\underline{V_0}$. Specifies by an octal or decimal integer the number of zero words to be generated. V_0 may never equal zero.

Examples:

CAT ➡ RESERV • 12 ➡

DOG ➡ RESERV • 10D ➡

b) CLEAR Operation.

L	W	V_0	V_1
[label] ➡	CLEAR •	[Number of words]	[starting address] ➡

The CLEAR operation clears to zero those memory addresses specified in the operation.

1) $\underline{L}$. The label for this operation is optional.

2) $\underline{W}$. CLEAR must always be present.

3) $\underline{V_0}$. 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) $\underline{V_1}$. 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, i.e., CAT-12D or CAT-14. All the words to be cleared must be wholly contained within one bank.

Examples:

[label] CLEAR • 18D • FLIP+12D ➡

[label] CLEAR • 22 • FLIP±14 ➡

[label] CLEAR • 100D • FLAP-5 ➡

[label] CLEAR • 4000 • 170000 ➡

c) MOVE Operation.

L	W	V_0	V_1	V_2
[label] ➡	MOVE •	[number of words] •	[from address] •	[to address] ➡

* TRIM II output number 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.

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 data are 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:

[label] ➡ MOVE●78D●CAT●DOG-7 ➡

[label] ➡ MOVE●10●HORSE+10●COW+8D ➡

[label] ➡ MOVE●4000●CAT●PIG ➡

[label] ➡ MOVE●100D●0●125000 ➡

d) INPUT/OUTPUT Operations.

L	W	V_0	V_1	V_2	V_3
[label] ➡	EXFCT ●	[channel number] ●	[AD CAD / MAD CMAD / BK CBK / MKB CMBK] ●	[number of buffer words] ●	[buffer starting address] ➡
[label] ➡	BUFIN ●	[channel number] ●	[AD CAD / MAD CMAD / BK CBK / MBK CMBK] ●	[number of buffer words] ●	[buffer starting address] ➡
[label] ➡	BUFOUT ●	[channel number] ●	[AD CAD / MAD CMAD / BK CBK / MBK CMBK] ●	[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

CAD - Advance without monitor, continuous data mode

CMAD - Advance with monitor, continuous data mode

CBK - Back without monitor, continuous data mode

CMBK - Back with monitor, continuous data mode

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:

[label] ➡ EXFCT●7●AC●1●CAT ➡

[label] ➡ BUFIN●6●MAD●10●CAT ➡

[label] ➡ BUFOUT●13D●BK●10●CAT+7 ➡

[label] ➡ BUFOUT●1●CMBK●100D●CAT+100D ➡

[label] ➡ BUFIN●CHAN●CAD●10●1250000 ➡

e) REMARK Operation.

L	W	V_0
[label] ➡	REMARK ●	[desired statement] ➡

The REMARK operation causes no object program generation. It is simply an aid to the programer in expanding normal program notes. REMARK operations may not exceed one line in length.

f) DATA Operation.

L	W	V_0
[label] ➡	DATA ●	[integer, binary point specification] ➡

The DATA operation allows the programer 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:

[label] ➡ DATA • 24D,9D ➡

or

[label] ➡ DATA • 30,11 ➡

would result in

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

[label] ➡ DATA • 123,4 ➡

would result in

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

g) PunCH Contents Operation.

L	W	V_0
[label] ➡	PCHC •	[information to be punched and/or typewriter commands] ➡

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 three typewriter control symbols to provide subsequent listing in the desired format.

Operand	Performance
•| CR |•	Carriage return
•△• or •| SP |•	Space
•| TAB |•	Move to next tabular stop (Flexowriter only)

The vertical bars indicate the information enclosed is a special symbol directing the typewriter. Each CR, SP, and TAB must begin and end with the vertical bar. Controls are separated from other operands by point separators.

V_0 - Specifies the operands in the order in which they are to be punched. Except for the typewriter commands, all operands imply that 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 • △ • 7070 • △ • DOG-11D • |CR| ➡

➡ PCHC • DOG • △ • B • |TAB| • AL • |TAB| • AL ➡

➡ PCHC • |CR| • |TAB| • TA ➡

➡ PCHC • DOG+10 • SP • CAT ➡

h) PunCH Text Operation.

L	W	V_0
[label]	➡ PCHT •	[text and/or typewriter commands]

The PCHT operation produces coding which, when run on the computer with PCHT* subroutine, causes the text(s) 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, SP, and TAB must be set off between two vertical bars

Operand	Performance
|CR|	Carriage
△ or |SP|	Space
|TAB|	Move to next tabular stop (Flexowriter only)

NOTE

Point separators are not required within V_0; they will be punched if present.

V_0 - Is the text to be punched interspersed with typewriter commands desired by the programer. If the text is too long for one PCHT operation, the programer can write successive operations.

Examples:

CAT ➡ PCHT • PROFIT △ AND △ LOSS △ FOR |TAB| ➡

➡ PCHT • JULY △ 10, △ 1964 |CR| ➡

i) TYPe Test Operation.

```
        L          W              V0
[label] ➡ TYPT • [text and/or line printer 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 line printer. The text may be interspersed with the following line printer control symbols:

Operand	Performance
|CR|	Carriage return, line feed
△ or |SP|	Space (may be used for formatting)

The vertical bars indicate the information enclosed is a special symbol directing the line printer. Each CR or SP must begin and end with a vertical bar.

V_0 - Is the text to be typed interspersed with line printer commands. If the text is too long for one TYPT operation, the programer may use successive operations to complete the text.

Examples:

```
CAT ➡TYPT • PROFIT |SP| AND △ LOSS △ FOR ➡
    ➡TYPT • JULY △ 10,△ 1963 |CR| ➡
```

NOTE

Point separators are not required within V_0; they will be typed if present.

j) TYPe Contents Operation.

```
        L          W              V0
[label] ➡ TYPC • [information to be typed and/or
                 line printer control symbols]
```

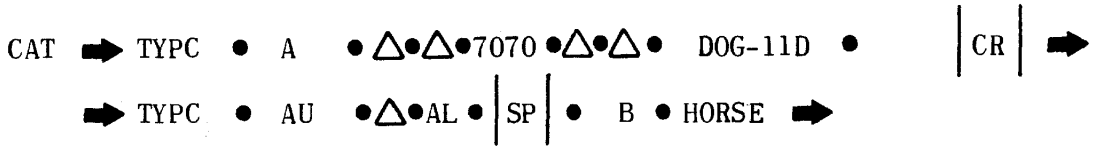
The TYPC operation results in generated coding which, when run on the computer with the TYPC* subroutine, causes the octal contents of A, AU, AL, current B, or any memory location to be typed on the line printer. The words to be typed may be interspersed with the following line printer control symbols.

Operand	Performance
•|CR|•	Carriage return, line feed
• △ • or•|SP|•	Space (may be used for formatting)

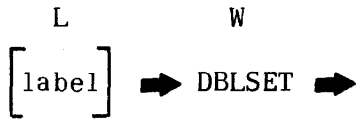
The vertical bars indicate the information enclosed is a special symbol directing the line printer. Each CR or SP must begin and end with a vertical bar.

V_0 - Specifies the operands in the order in which they are to be typed. Except for the line printer commands, all operands imply that their contents are to be typed. Such operands may be A, AU, AL, active B, a tag or a tag ± an absolute value or an absolute address.

Examples:

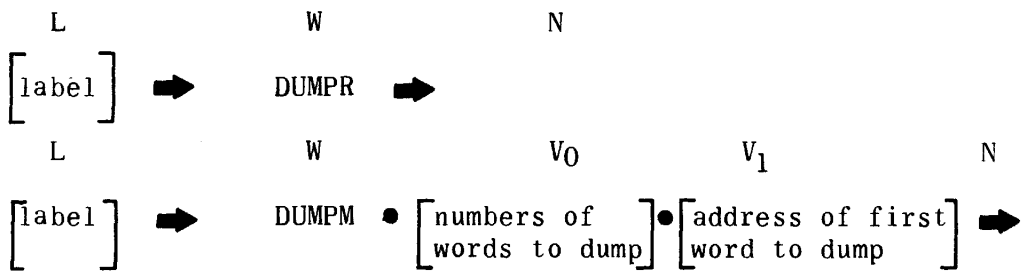


k) DouBLe SET Operation.



The DBLSET operation frees the programer from the responsibility of insuring that the Y of a double add or subtract instruction be located at an even address. The DBLSET operation may be followed only by the Y constant concerned. TRIM II examines the address to which the constant would normally be assigned. If the address is odd, a word of zeros is first generated to ensure that the constant will be assigned to an even address.

7. Debugging Operations. TRIM II provides two debugging operations for punching a paper tape output of either the contents of register AU, AL, and current B, or the contents of specified sequential memory locations. These operations generate a set of three of 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 programer recognition of repeated times through given coding paths. The debugging operations take the following form:



a) L. Label is optional.

b) W. DUMPR or DUMPM must always be present.

c) 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.

d) V_1. Applicable to the DUMPM operation only. Expresses the address of the first word whose contents are 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      ➡
```

Both DUMPR and DUMPM operations preserve existing values in AU, AL, and the current B-register.

8. TRIM II Outputs. TRIM II provides six optional punched paper tape outputs of the assembled program. All the outputs except output no. 6 are loadable via the 1219 utility packages. However, only Output No. 5, in relocatable bioctal format, can be loaded above address 100000 at those installations possessing a 65K memory computer.

The available outputs are as follows.

a) 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 A addresses.

b) 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.

c) No. 4. Absolute assembled program in bioctal format, consisting of a 76 code, inclusive area addresses followed by the instructions only, and a checksum.

d) 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.

e) 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 ensure a complete allocation tape, output 6 should not be the first requested output.

f) No. 11. Source program on paper tape in source code.

c. Programing Procedures.

1. Input Tape Format. Source program tapes will be punched in field-data code or standard flexcode, and the resulting punched paper tape serves as input to TRIM II.

2. Ground Rules.

a) TRIM II is available in two versions; one version accepts a source program prepared in fieldata code, the other version accepts a source program in standard Flexowriter code. Input to TRIM II is via punched paper tape. For the Fieldata-oriented TRIM II each source tape must begin with a carriage return and terminate with a carriage return and two periods. Shift up and shift down codes are ignored. For the Flexcode-oriented TRIM II each source tape must begin with a carriage return and shift to upper case since the assembler assumes all input is in upper case except for the comma, the vertical bar, +, and the period. Each source tape must terminate with a carriage return, shift to lower case, and two periods. The term "source code" used herein refers to the code in which the input tapes are prepared.

b) 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.

c) Each break in sequence of addressing constitutes a program area. A total of 27 such areas is permitted.

d) 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 1184_{10}.

e) TRIM II provides a limited amount of error detection with error typeouts. 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, etc. Typical errors are unconvertible numbers, illegal operators, no label first instruction, duplicate label, etc.

f) TRIM I operators SETADR and EQUALS are ignored by TRIM II. The ALLOCation operation replaces these two functions.

g) When specifying a decimal integer, the letter D occupies one digit position; therefore, the maximum decimal integer that can be expressed is 99999D.

h) Sample coding generated by the CLEAR poly-operation.

1) Assume CAT is in bank 3

➡ CLEAR ● 10 ● CAT ➡

```
ENTBK●7
STRSR●LOK + 4
ENTSR●13
CLB●CAT
BJP●LOK - 1
ENTSR●0  (User's SR stored here)
```

2) ➡ CLEAR ● 10 ● 115000

```
ENTBK●7
STRSR●LOK + 4
ENTSR●31
CLB●5000
BJP●LOK - 1
ENTSR●0  (User's SR stored here)
```

i) Sample coding generated by the MOVE poly-operation.

1) Assume MOW, COW, and the MOVE instruction are in bank 3

➡ MOVE ● 10 ● MOV ● COW ➡

```
ENTBK●7
STRSR●LOK + 6
ENTSR●0
ENTAUB●MOW
ENTSR●0
STRAUB●COW
BJP●LOK - 4
ENTSR●0  (User's SR stored here)
```

2) Assume CAT is in bank 4

```
➡ MOVE ● 12 ● CAT ● 135000 ➡
        ENTBK●11
        STRSR●LOK + 6
        ENTSR●14
        ENTAUB●CAT
        ENTSR●33
        STRAUB●5000
        BJP●LOK - 4
        ENTSR●0  (User's SR stored here)
```

j) 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 programer 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 programer must allocate the tag CHAN used by these source language subroutines to reference the paper tape input/output channel. If the programer 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.

TYPT	16400
TYPC	16560
PCHT	17000
PCHC	17200
DEBUG	17470

k) Keyboard correction methods for TRIM fieldata assembler and corrector input tapes: Typing error correction procedures have been incorporated in the fieldata 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 Type 1232 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 1232 I/O Console, we have designated the BACKSPACE code of the keyboard 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.

A single reject code (77) anywhere on an input tape to the TRIM assemblers or corrector informs that routine that the legal code that immediately preceded the 77 should be rejected. (Any code other than 01, 02, 03, or 57 is a legal code), e.g.,

□ ER ↑ NTE ↑ AL / ↑ CAG ↑ T

appearing on the console printer and punched as one of the statements on a program tape to be assembled will be interpreted as

□ ENTAL CAT

by that assembler.

Three consecutive reject codes in a row (77-77-77) 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, e.g.,

□ 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)

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)

Another example:

143'1□2 (Carriage Return - Line Feed)

GOOF□SLSUB'JUNK ↑↑↑ (Carriage Return - Line Feed)

RIGHT□JK ↑ P 'HON1 ↑ OR (Carriage Return - Line Feed)

on a correction tape will be interpreted by the TRIM Corrector as

143'2 (Carriage Return)

RIGHT □ JP'HONOR (Carriage Return)

l) 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 II Output No. 5 into computer memory:

1) If the program was assembled starting at an even address, it must be loaded starting at an even address.

2) If the program was assembled starting at an odd address, it must be loaded starting at an odd address.

m) If it is desired to assemble a program to load and run at or above address 100000, the user must obtain a No. 5 output.

n) (Applicable to 65K memory computers only). Any program to be assembled by TRIM II or 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 interbank references. The TRIM assemblers do not provide any alarm indications for this condition.

d. TRIM II Operating Procedures.

1. Basic Information. TRIM II is a two-pass assembler which accepts source programs written with absolute or mnemonic function codes and symbolic addressing and produces an output program on punched paper tape suitable for loading into the UNIVAC 1219 computer via the 1219 utility packages.

2. Loading the Assembler. To load TRIM II, a 1219 utility package must already have been loaded into the computer.

STEP 1. Master clear the console.

STEP 2. Mount the TRIM II Assembler tape in the reader.

STEP 3. Set P to the UPAK starting address for paper tape load.

STEP 4. Start.

3. Using the TRIM II Assembler.

1. Pass 1.

STEP 1. Master clear the console.

STEP 2. Set P to 1400.

STEP 3. Set Skip Keys 1 and 2.

STEP 4. Set Skip Key 3 for error typeout suppression during assembly.

STEP 5. Mount the program tape in the reader.

STEP 6. Start. The computer will stop after each program tape has been read. Repeat steps 5 and 6 until all tapes have been read in.,

2. Pass 2.

STEP 1. Release Skip Key 1.

STEP 2. Set AL to desired output number.

STEP 3. Mount the program tape in the reader (program tapes must be loaded in exactly the same order as for Pass 1.

STEP 4. Start. The assembler will assemble the input tape and punch the output tape. Repeat Steps 3 and 4 until all tapes have been read in and punched.

STEP 5. Release Skip Key 2.

STEP 6. Start. The assembler will finalize the output tape with a checksum and trailer.

STEP 7. Set Skip Key 2 and start with Step 2 to obtain additional outputs.

4. Error Detection and Display. TRIM II contains certain error detection capabilities. The majority of programer errors can be handled internally. However, since skip key settings are essential to the assembly process, assembly will always stop when 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 skip keys and start to continue assembly.

The programer has the option of requesting that the assembler stop and type out pertinent information for four basic programer errors, or requesting that the assembler handle these errors internally, thereby "forcing" an assembly.

If Skip Key 3 is set, the assembler will force the assembly. If Skip Key 3 is not set, the assembler will stop for the following errors:

a) Set Base Address in AL. This error stop occurs during Pass 1 and means that no starting address has been given. To correct,

STEP 1. Clear AL.

STEP 2. Set $AL_{(15-0)}$ to the desired address in octal.

STEP 3. Start. The assigned base address will then be typed out and assembly will continue.

If typeout suppression has been selected (Skip Key 3 set), assembly will not stop and TRIM II will arbitrarily assign the program to the base address 01200.

b) Illegal Output Reselect in AL. This error stop indicates an illegal output has been selected at the start of Pass 2. To recover, start. When the computer stops, reselect the output in AL, and start.

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.

c) Unalloc Tags. This error stop occurs during Pass 2 and indicates an unallocated tag. The typeout occurs for the first such tag. Thereafter only the sequential line identifier, the tag name, and, after manual allocation, the value to which the tag was equated are typed.

The recovery procedure is:

STEP 1. Set AL to the desired value.

STEP 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.

STEP 3. If the user wishes any later unallocated tags allocated to the same address, set AU $\neq$ 0.

STEP 4. Start. TRIM II will type the manual allocation and use it to continue assembly.

d) 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.

5. Assembler Outputs (Loadable Via the 1219 Utility Packages).

a) Output No. 2. A side-by-side listing of the absolute assembled program in source code, sequential line identifiers, the source program and alarm codes. Output 2 is preceded by a program summary consisting of memory locations used and inclusive addresses.

b) Ouput No. 3. Absolute assembled program in typewriter code.

c) Output No. 4. Absolute assembled program in bioctal code.

d) Output No. 5. Relocatable assembled program in bioctal code.

e) Output No. 6. ALLOC output consisting of all labels and addresses. Output 6 is not loadable via the 1219 utility packages, but may be used as an ALLOCation tape for input to TRIM II. To ensure a complete list of all labels and tags, output 6 should not be the first output selected.

f) Output No. 11. Source program on paper tape in typewriter code. Output No. 11 is not loadable via the 1219 utility package, but may be used as source input to TRIM II.

6. Assembler Addresses.

01400 Start assembly, Pass 1.

01401 Restart outputs, Pass 2.

TABLE 2.1-2. FIELDATA CODE (6 BITS) UNIVAC 1232 KEYBOARD AND LINE PRINTER

	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	*	(	"	:	?	;	'	S STOP
6	0	1	2	3	4	5	6	7
7	8	9	,	;	/	.	▭ SPEC	↑ IDLE

△ = SPACE

NOTE: Master space indicates an absence of information.

NAME: _____

2.1-5. STUDY QUESTIONS

a. What is the difference between a label and a tag? Why is it that a tag can be incremented or decremented and a label not?

b. In the space below show the form you would use, 1) to make an allocation header, and 2) to make a program header.

c. What is the use of the allocation tape? When is it used?

d. List the nine poly-operations of Trim II.

e. What is the procedure to correct an unallocated tag?

f. Write a program in Trim II language to count the number of 1's in A and have the result displayed on the printer of the I/O Console. The input in A will be made manually.

g. Write a program to convert flex characters to fieldata characters. Expand the program so that the flex characters may be inputted on a paper tape from the 1232 I/O Console and the fieldata characters will be punched on a paper tape to the 1232 I/O Console.

SECTION 2 - SOFTWARE

2.2. TRIM III ASSEMBLER

2.2-1. OBJECTIVES

To provide areas of study for the student and to provide study questions for the student to analyze the use of Trim III.

2.2-2. INTRODUCTION

Trim III in many respects is like Trim II but Trim III is more powerful and will allow the programer to assemble his program on magnetic tape instead of paper tape. The format in many cases is the same in both assemblers but in ways it is different. The student should be aware of the many differences between the two.

2.2-3. REFERENCES

PX 3288 Programers Reference Manual, Section on Trim III

2.2-4. INFORMATION

a. General. The use of Trim III is very similar to the Trim II assembler with the exception that Trim III is a one-pass assembler and gives more poly-operations and outputs. The method of study should probably follow the same format as you followed by analyzing Trim II.

Trim III is an assembler which operates on a 16K memory computer with a magnetic tape system, paper tape reader-punch unit, a console typewriter, and UNIVAC 1004 Card Processor.

This assembler has a source language librarian for aiding the programer in selecting subroutines for incorporation into the program during the assembly process. The programer uses CALL operations in his source program to implement retrieval from the source library.

The source programing 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 programer during a run. This works with the TRIM DEBUGGING PAK 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 typewriter. These features cut TRIM III assembly time to a minimum.

The assembler possesses source language correction capability in conjunction with an assembly run.

b. TRIM III Assembly System.

1. Introduction. The TRIM III Assembly System for the UNIVAC 1219 computer provides programing assistance through the use of its symbolic shorthand. 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 the high-speed printer, punched paper tape, punched cards, or magnetic tape.

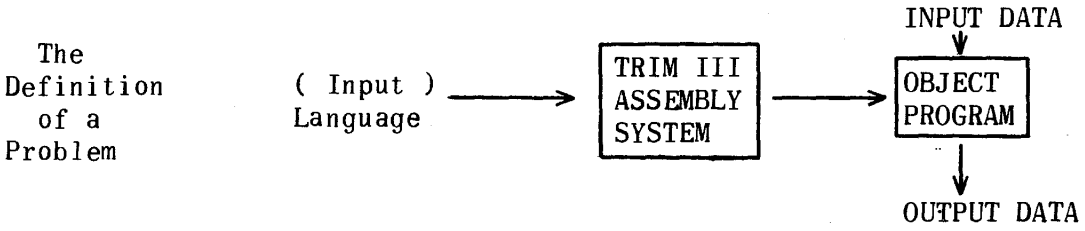


Figure 2.2-1. TRIM III Solution of a Problem

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 the following minimum equipment configuration listing:

a) UNIVAC 1219 computer with 16,384 words of core memory

b) Magnetic Tape System with a minimum of two tape transports

c) Input-Output Console consisting of a perforated tape reader, tape perforator, keyboard, and printer.

Optional equipment is an on-line UNIVAC 1004 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 2.2-2 and 2.2-3. 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.

3. Header and Declarative Operations. TRIM III recognizes four types of header operations,

L		W		V_0		V_1		N
POKER	➡	CONTR	•	JONES	•	10 DEC 1964	➡	
POKER	➡	ALLOC	•	JONES	•	10 DEC 1964	➡	
POKER	➡	PROG	•	JONES	•	10 DEC 1964	➡	
POKER	➡	CORREC	•	JONES	•	10 DEC 1964	➡	

a) CONTRol Header. The CONTR header operation is a convenience for the user enabling 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.

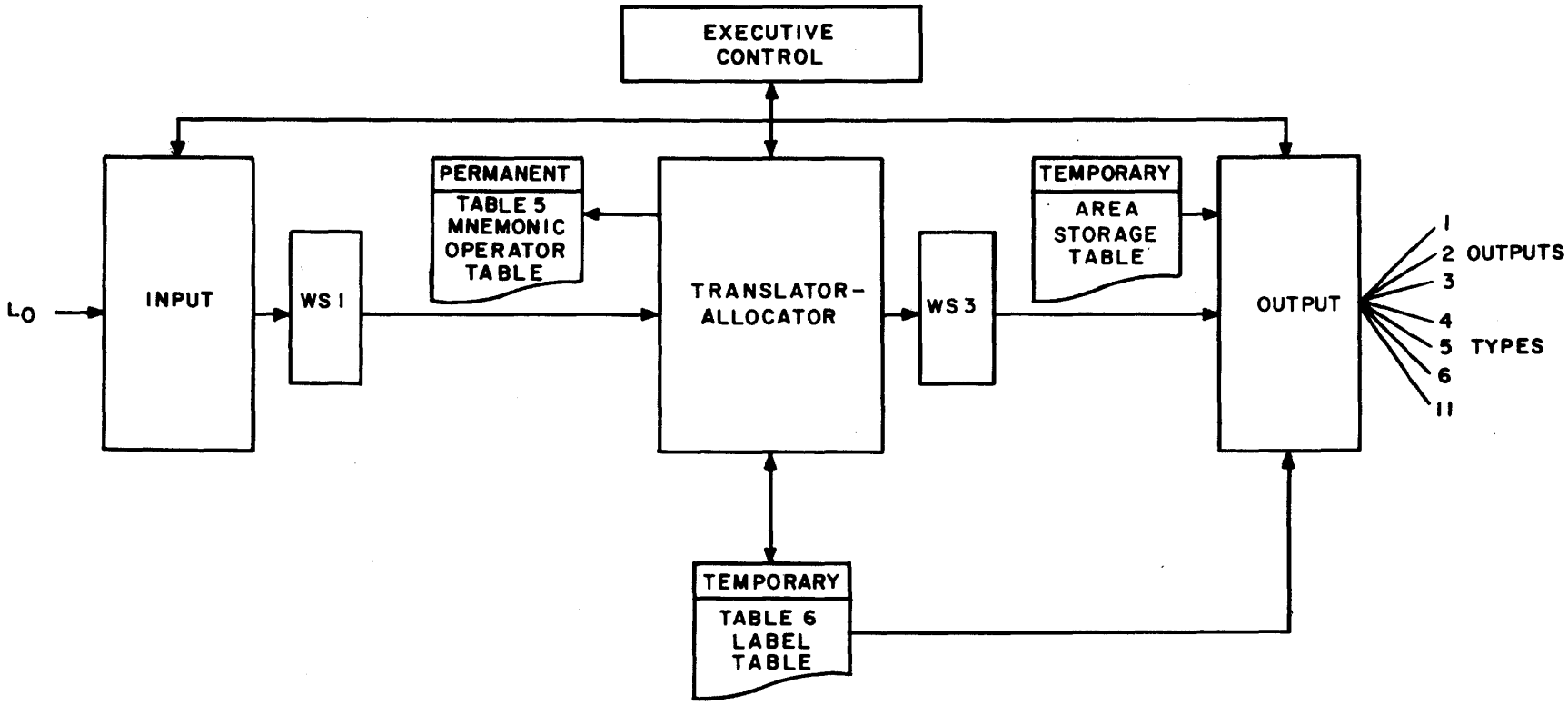
L		W		V_0		V_1	
[label]	➡	[CONTR]	•	[name]	•	[date]	➡

Operations which may follow a CONTR header are ALLOC, DEBUG, OUTPUT, REMARK, DECKID and CALL. CALL operations may also follow a PROG header.

b) ALLOCation Header. (See figure 2.2-4). 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. 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:

L		W		V_0		V_1	
[label]	➡	[ALLOC]	•	[name]	•	[date]	↵
[label]	➡	[assigned value]	↵				
[label]	➡	[assigned value]	↵				
[label]	➡	[assigned value]	↵				

etc.



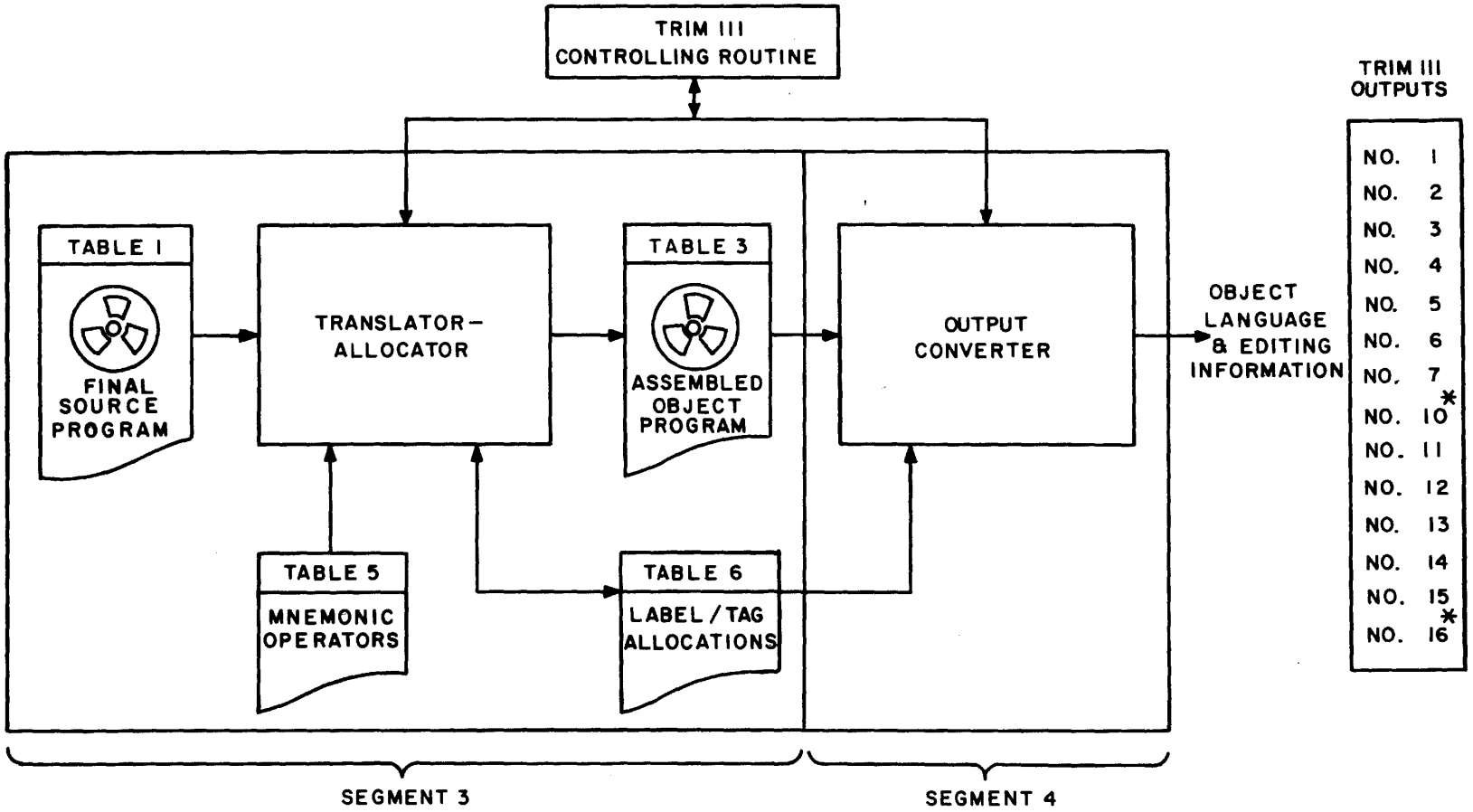
EXECUTIVE:
a. MONITORS KEY SETTINGS
b. PERFORMS INITIALIZATIONS
c. EXECUTES SECONDARY ROUTINES

INPUT:
a. READS ONE ITEM INTO WSI

TRAN/ALLOC:
a. PERFORMS GENERATION INTO WS3 ONE ITEM AT A TIME
b. ADDS UNALLOCATED TAGS INTO TABLE 6

OUTPUT:
a. PUNCHES ONE WS3 ITEM AT A TIME IN ACCORDANCE WITH THE NUMBERED OUTPUT REQUEST (FOR OUTPUT 2 ALSO PUNCHES THE WSI 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 FOR OTHER OUTPUTS
c. OUTPUT 1 IS NOT A SELECTABLE OUTPUT, BUT IS AUTOMATICALLY PRODUCED WITH OUTPUT 2

Figure 2.2-2. TRIM III Segments 1 and 2



* OUTPUT NO. 10 IS TABLE 3, OUTPUT NO. 16 IS TABLE 15

Figure 2.2-3. TRIM III Segments 3 and 4

TITLE _____ PAGE _____ of _____

UNIVAC CODING FORM

PROGRAMMER _____ PLT _____ EXT _____ MS _____ DATE _____

LABEL	OPERATOR	OPERANDS AND NOTES
	HEADER TYPE	
POKER	CONTR	JONES•16 NOVEMBER1963
POKER	ALLOC	JONES•16JUNE1963
POKER	05000	
CHIPS	05500	
DEBUG	13000	
TYPT	12700	
	OUTPUT	1•5•2•5•6•11
	DEBUG	
	CALL	SINE•TODEC • TYPT
	REMARK	CONTR TAPE FOR DATAX REVISION 3.
	DECKID	SINE

Figure 2.2-4. Sample CONTR Tape

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; e.g.,

CAT → 0100↵

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.

c) PROGram Header. The PROG header informs TRIM III that the operations to follow 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. A program header has the following format:

L	W	V_0	V_1
[Program name]	→ PROG	• [name]	• [date]

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.

d) CORRECtion Header. 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

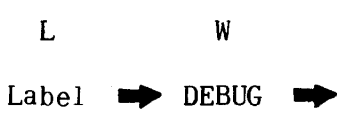
3) Deletions

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.

Corrections are always made on the basis of the sequential line identifier associated with each source program statement. This sequential identifier appears on a TRIM III Output No. 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 same preceding one(s) with the same identifiers, permitting a programer to correct a correction. Only the last such correction will count towards the 192 maximum.

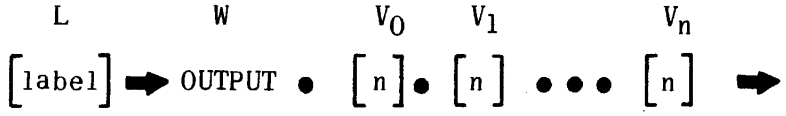
The format of correction operations is similar to that required by the TRIM CORRECTOR. Figure 2.2-5 shows a sample of correction coding.

e) DEBUG Declarative. TRIM III accepts the declarative operaton.

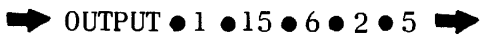


The DEBUG operator informs TRIM III that generation is to be performed for debugging operations contained in the source program. If the DEBUG operator is absent, no generation will occur for such debugging operations. The DEBUG operation when used must be loaded prior to the first PROG header. It normally appears on the CONTRol tape.

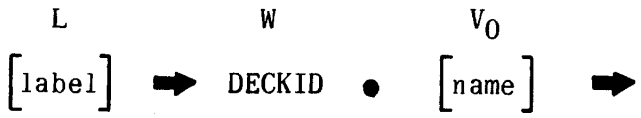
f) OUTPUT Declarative. TRIM III accepts the declarative operation.



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.



g) DECKID Declarative. TRIM III accepts the declarative operation.



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 four-digit sequential octal number beginning with 0001, will be 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

TITLE MANL (CORRECTIONS)
PAGE 1 of 1

UNIVAC
TAPE CORRECTION FORM

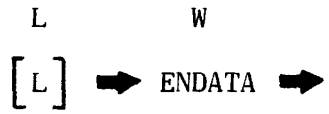
PROGRAMMER W. C. Roos
PLT. 1 EXT 2341 MS 1091
DATE 8 November 1964

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 2.2-5. Sample Correction Coding

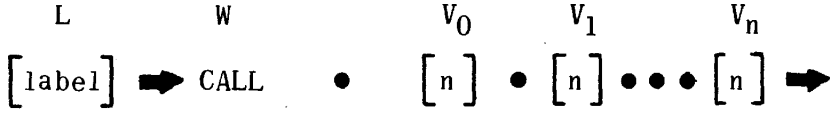
(output 12) and the punched card output in source language (output 15). The new card identification and numbering pre-empts that which might be present if the input source program is on cards. Any number of DECKID statements may be inserted anywhere in the source program; however, each DECKID operation affects only those statements following that DECKID statement, and the card numbering will always begin with 0001.

h) 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.



4. Poly-Operations (different from TRIM II).

a) Library CALL Operation.



The CALL operation permits the programer 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 never 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.

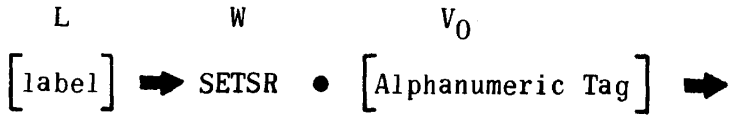
The CALL operation causes no object program generation.

Examples:

CALL • TYPT • GLP • SINE • TYPC

CALL • PCHC

b) SETSR Operation.



The SETSR operation enables the programer 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 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 170460 and COW is allocated to 01000, then

➡ SETSR•CAT ➡ Generates 507313

➡ SETSR•DOG ➡ Generates 507337

➡ SETSR•COW ➡ Generates 507310

c) 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. These operations generate a set of threee 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 programer 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 words to dump] • [address of first 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. Express the address of the first word whose contents are 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 • 10000 ➡

label ➡ DUMPM • 10D • CAT+28D ➡

label ➡ DUMPM • 12 • CAT-15 ➡

label ➡ DUMPM • 64D • CAT ➡

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 programer use: DSTOP and DTYPT.

L W

[label] ➡ DSTOP ➡

The DSTOP operation permits the programer to intersperse strategic debugging stops within his program. If the DEBUG operator is used, the DSTOP will generate an unconditional stop(505640); otherwise, TRIM III will ignore the operation.

L W V_0

[label] ➡ DTYPT • [text and/or line printer commands] ➡

The DTYPT operation performs in the same way as the TYPT operation. If the DEBUG operator is used, TRIM III will perform the generation. Otherwise, the operation will be ignored.

5. TRIM III Outputs. TRIM III provides thirteen 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 in the following paragraphs.

a) Monitoring Typewriter:

No. 1. Program summary consisting of the number of memory locations used and inclusive addresses.

b) Paper Tape. Except for outputs 6 and 11, all paper tapes are loadable via the 1219 utility packages. Outputs 6 and 11 may be used as input to TRIM III.

1) 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.

2) 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.

3) No. 4. Absolute assembled program in bioctal format, consisting of a 76 code, inclusive area addresses followed by the instructions only, and a checksum.

4) 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.

5) No. 6. Allocation output in source code consisting of an ALLOC header, followed by all program tags and labels and addresses in allocation format.

6) No. 11. The source program only, produced on punched paper tape in source code.

c) High-Speed Printer.

1) 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.

2) 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 wise by 8 1/2 inches long.

d) Punched Card.

1) 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. The initial load card may be removed and a load address set in A upper.

2) 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.

e) Magnetic Tape.

1) 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 1219 UPAK II or UPAK III utility programs.

2) 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.

f) 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.

6. Programing Procedures.

a) Paper Tape Input Format. Two versions of TRIM III are available, one version accepts a source program paper tape prepared in fieldata code, the other version accepts a source program paper tape prepared in standard Flexowriter code (see tables

For the fieldata-oriented TRIM III each source tape must begin with a carriage return and terminate with a carriage return and two periods. Shift-up and shift-down codes are ignored.

For the Flexcode-oriented TRIM III each source tape must begin with a carriage return and shift to upper case since the assembler assumes all input is in upper case except for the comma, the vertical bar, the plus, and the period. Each source tape must terminate with a carriage return, a shift to lower case, and two periods.

Typing-error correction procedures have been incorporated in the fieldata 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 Type 1232 I/O Console.

Since it is impossible to back up the paper tape 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 1232 I/O Console, we have designated the BACKSPACE code of the keyboard 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.

A single reject code (77) anywhere on an input tape to the TRIM Assemblers or Corrector informs that routine that the legal code that immediately preceded the 77 should be rejected. (Any code other than 01, 02, 03, or 57, is a legal code), e.g.,

□ER ↑ NTE ↑ AL ↑ CAG ↑ T

appearing on the console printer and punched as one of the statements on a program tape to be assembled will be interpreted as

□ENTAL CAT

by that assembler.

Three rejects in a row (77-77-77) 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, e.g.,

□ENTAL CAT (Carriage Return-Line Feed)

MOOSE □STRAL D ↑↑↑ JP TIGER (Carriage Return-Line Feed)

□ADDALK 63 (Carriage Reutrn-Line Feed)

MICE □STRAL DOG (Carriage Return-Line Feed)

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)

Another example:

143'1↑2 (Carriage Return - Line Feed

GOOF □ SLSUB'JUNK ↑↑↑ (Carriage Return - Line Feed)

RIGHT □ JK ↑ P'HON1 ↑ OR (Carriage Return - Line Feed)

on a correction tape will be interpreted by the TRIM Corrector as

143'2 (Carriage Return)

RIGHT □ JP'HONOR (Carriage Return)

b) 80-Column Card Input Format. For those installations whose peripheral equipment configuration includes a UNIVAC 1004 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 input. Interpretation of coding separator symbols for card input is given in table 2.2-1.

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 represented by the "*" key in all card input.

TABLE 2.2-1. CODING FORMAT SYMBOLS

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

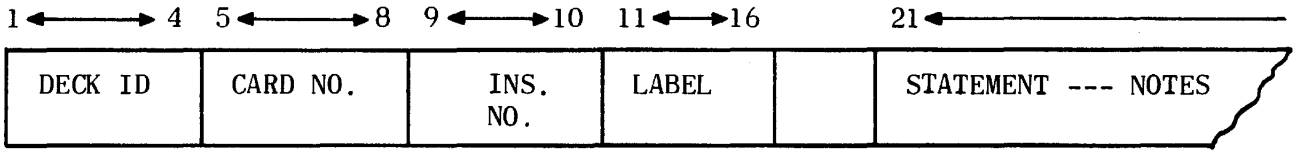
The ENDATA operation card followed by one blank card denotes the end of a card input deck.

The following examples illustrate the basic coding format for card input:

```
DECK   CARD      INS.
 ID     NO.      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 ↵
```

See figure 2.2-6. Typical Code Program for Card Input.

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.



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.

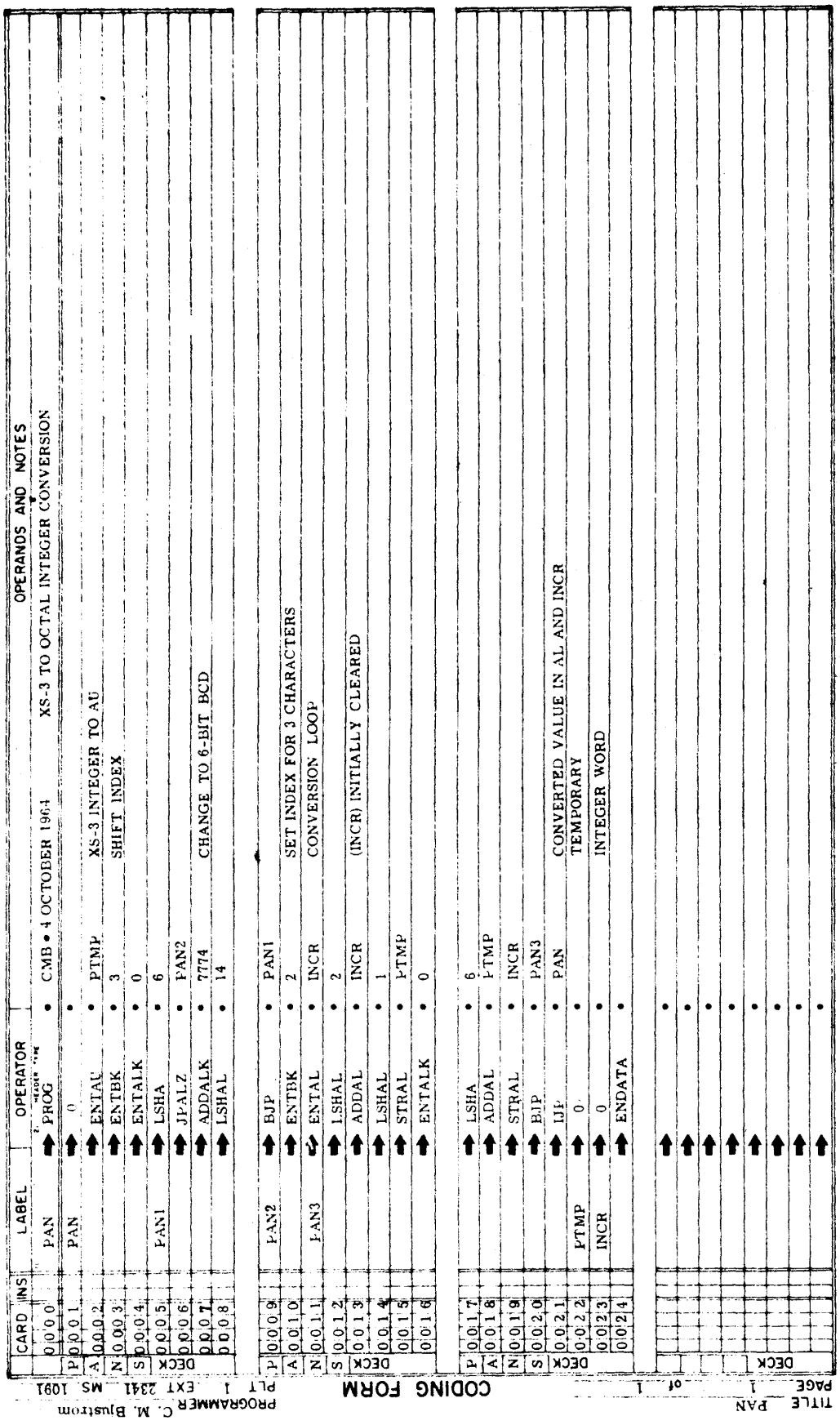
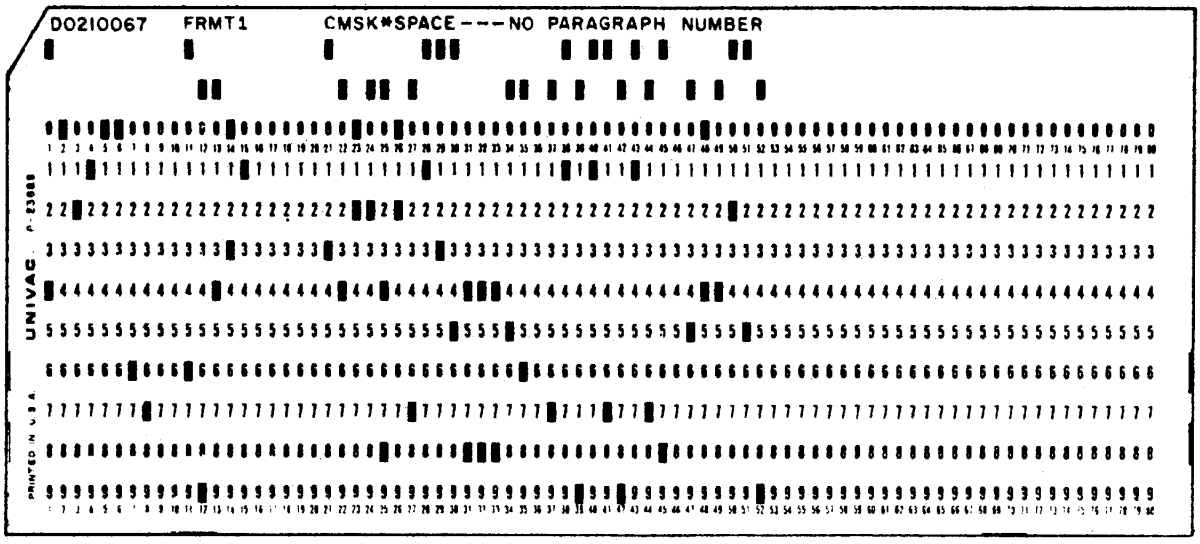


Figure 2.2-6. Typical Coded Programer Card Input

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 hits the SKIP key and the card is automatically positioned at column 21. The statement always begins at column 21. Three dashes separate the statement from the notes even though no notes may be present. The REL key terminates each card.



See figure 2.2-7 for an example of TRIM III Output No. 12 resulting from card input.

c) Source Program Correction Formats. 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) Corrections must be loaded prior to the loading of the source program(s) 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) Where two or more correction operations bear the same integral and fractional value, the last such operation overrides permitting programers to correct an erroneous correction.

6) Corrections are based on the assembler-assigned sequential line identifier for each source statement.

OUTPUT 12

```
MEM. STRG. USED  11061
00240  THRU  00404
01000  THRU  07743
20000  THRU  21747

LOK    INSTR      L1ID  DECK CARD  L0
                  0     LIBX 0001  BITSUM   PROG*JRS*6NOV64      FLEX
00240  34 0364    1     LIBX 0002  BITSUM   JP*SE0K              LIBBLD BOOTSTRAP LOAD
00241  76 0355    2     LIBX 0003  UPAKX    RJP*ERP                                    100
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 2.2-7. TRIM III Output 12 from Card Input

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 six digits; the fraction (which indicates insertion or addition) is limited to three digits. The fraction is a straight binary magnitude, (e.g., 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.

d) Paper Tape Correction Format. The format of correction operations prepared on paper tape is identical to that required by the TRIM CORRECTOR (see paragraph 2.3-4e of this manual) with the following exceptions:

1) Corrections must follow a CORREC header.

2) A maximum of six integral digits is permitted.

e) 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, this second card may also contain card identification in its first ten columns which will be included in the outputs 12 and 15 of the assembled corrected program unless a DECKID operator has been 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 conventional 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 in figure 2.2-6.

Column:		11	21
Card 9			
Card 8			ENDATA ---
Card 7	PANS00181		SUBAL * INCR ---
Card 6		22	1 ---
Card 5			DELETE * 3
Card 4		14	0 ---
Card 3	PANS0011	PAN3	ENTALK * 53 --- CONSTANT TO AL
Card 2		13	0 ---
Card 1		PAN	CORREC ---

f) Ground Rules. Regardless of the input format, there are certain conventions which the programer 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 the 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 are permitted.

4) TRIM III provides a limited amount of error detection with error typeouts.

5) TRIM I operators SETADR and EQUALS are ignored by TRIM III. The ALLOCation operation replaces these two functions.

6) When specifying a decimal integer, the letter D occupies one digit position; therefore, the maximum decimal integer that can be expressed is 99999D.

7) Sample coding generated by the CLEAR poly-operation.

a. Assume CAT is in bank 3.

```
➡ CLEAR●10●CAT ➡
        ENTBK●7
        STRSR●LOK + 4
        ENTSR●13
        CLB●CAT
        BJP●LOK - 1
        ENTSR●0          (User's SR stored here)
```

b. ➡ CLEAR●10●115000 ➡

```
        ENTBK●7
        STRSR●LOK + 4
        ENTSR●31
        CLB●5000
        BJP●LOK - 1
        ENTSR●0          (User's SR stored here)
```

8) Sample coding generated by the MOVE poly-operation.

a. Assume MOW, COW, and the MOVE instruction are in bank 3

➡ MOVE●10●MOW●COW ➡

```
ENTBK●7
STRSR●LOK + 6
ENTSR●Ø
ENTAUB●MOW
ENTSR●Ø
STRAUB●COW
BJP●LOK - 4
ENTSR●0              (User's SR stored here)
```

b. Assume CAT is in bank 4

➡ MOVE●12●CAT●135000 ➡

```
ENTBK●11
STRSR●LOK + 6
ENTSR●14
ENTAUB●CAT
ENTSR●33
STRAUB●5000
BJP●LOK - 4
ENTSR●0              (User's SR stored here)
```

9) Assembler support subroutines TYPT, TYPC, PCHT, PCHC, and the debugging package, DEBUG, are included in the TRIM III library of subroutines. The programer uses a CALL operation to retrieve them from the library. The programer may allocate these subroutines through normal ALLOCation operations. If he does not allocate them, TRIM III will assign them sequential addresses immediately following the principal program. If these subroutines are not assembled with the principal program and the programer 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 use the tag CHAN for all input/output instructions. It is the programer's responsibility to provide an ALLOCation operation equating CHAN to the appropriate I/O channel.

10) 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.

11) TRIM III informs the user of a duplicate label via a type-out 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.

12) (Applicable to 65K memory computers only). Any program to be assembled by TRIM II or 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 intersegment references which would be implemented exactly as for inter-bank references. The TRIM assemblers do not provide any alarm indications for this condition.

7. TRIM III Operating Procedures. See figure 2.2-8.

a) 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 UNIVAC 1219 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 various modes of input without skip key settings other than Skip Key 1. TRIM III has 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

TITLE TYPE TEXT ROUTINE
PAGE 1 of 2

UNIVAC CODING FORM

PROGRAMMER J. E. Scanlan
PLT 1 EXT 340 MS 1091
DATE 28 September 1964

LABEL	OPERATOR	OPERANDS AND NOTES
TYPT	PROG (HEADER TYPE)	. SCANLAN • SEPT 1964
TYPT	0	• ➡ TYPE TEXT SUBROUTINE
	STRSR	• TYPT20
	STRAU	• TYPT3
	STRAL	• TYPT4
	STRB	• TYPT5
	ENTALK	• 3
	RJP	• DOTYF ➡ ENABLE TT KEYBOARD
TYPT 1	ENTB	• TYPT ➡ ADVANCE EXIT ADDR
	ENTBKB	• 1
	STRB	• TYPT
	ENTSR	• 1Ø
	ENTAUB	• Ø ➡ PICK UP NEXT FIELD DATA WD
	ENTSR	• Ø ➡ CLEAR SR ACTIVE
	ENTBK	• 2
TYPT 2	ENTALK	• Ø
	LSHA	• 6 ➡ NEXT FIELD DATA CD TO AL
	CMAL	• TYPT 6
	JPEQ	• LOK + 4 ➡ ALL DONE IF 77
	RJP	• PIP ➡ TYPE IT
	BJP	• TYPT 2
	JP	• TYPT 1
	ENTALK	• 1
	RJP	• DOTYF ➡ DISABLE KEYBOARD
	ADDAL	• TYPT

Figure 2.2-8. Sample TRIM Coding Format

TITLE Type Text Routine
PAGE 2 of 2

UNIVAC CODING FORM

PROGRAMMER J. E. Scanlan
PLT 1 EXT 340 MS 1091
DATE 28 September 1964

LABEL	OPERATOR	OPERANDS AND NOTES
	HEADER TYPE	
	STRAL	TYPT ➡ SET REAL EXIT
	ENTAU	TYPT 3 ➡ RESTORE REGS
	ENTAL	TYPT 4
	ENTB	TYPT 5
TYPT 20	ENTSR	0
	IJP	TYPT
TYPT 3	O	
TYPT 4	O	
TYPT 5	O	
TYPT 6	77	
••		

Figure 2.2-8. Sample TRIM Coding Format (Cont.)

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.

b) Loading TRIM III.

1) For installations possessing a magnetic tape wired bootstrap,

STEP 1. 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.

STEP 2. Press the computer control panel MASTER CLEAR LOAD, and START switches. 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,

STEP 1. 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.

STEP 2. Mount the TRIM III paper tape loader in the paper tape reader.

STEP 3. Press the computer control panel MASTER CLEAR LOAD, and START switches. The TRIM III Executive will be loaded into memory and the computer will stop with P = 01404.

c) Initializing TRIM III. When the computer stops with P = 01404, it is necessary to identify the magnetic tape configuration. This identification need be made once only for all subsequent assemblies unless it is necessary to change the configuration. TRIM III expects tape information in the format 00CST where C is the channel number (bits 6 through 9), S is the tape synchronizer number (bits 3 through 5), and T is the transport number (bits 0 through 2). Thus, 001512 represents channel 15, synchronizer 1, transport 2, and 000112 represents channel 1, synchronizer 1, transport 2.

STEP 1. Set AU to the assembler tape (Tape A).

STEP 2. Set AL to a scratch tape (Tape B).

STEP 3. Start. TRIM III will record the information and stop with AU and AL cleared.

STEP 4. When using only two transports, Start. The computer will stop with P = 01400.

STEP 5. When using 4 transports,

a) If source input is on magnetic tape, set AU to the input tape.

b) If a source output on magnetic tape (TRIM III output 16) is being requested, set AL to the output tape. (TRIM III output no. 10 is always produced on scratch tape B.)

c) Start. The computer will stop with P = 01400.

d) Using TRIM III.

STEP 1. For paper tape input, mount source tape in paper tape reader.

STEP 2. For card input, initialize the UNIVAC 1004 Card Processor as follows:

a) Set Manual Alternation Switch 1 only.

b) Mount source card deck in card reader input hopper.

c) Press CLEAR, START, FEED, and RUN.

STEP 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 B transport. In this situation via the typeout, "REMOVE INPUT TAPE TO SAVE", TRIM III will instruct the operator to replace the tape upon completion of read in the event that the input tape is to be saved.

b) If three or four transports are being used, mount the input tape on the transport identified in AU. Since this transport is used only for magnetic tape input, no warning typeout occurs.

STEP 4. Master Clear.

STEP 5. Set P = 01400.

STEP 6. Set Skip Key 1.

STEP 7. For error typeout suppression, set Skip Key 3.

STEP 8. If initial source input is other than the normal mode, set AL to the proper code. (For normal mode AL remains equal to zero).

STEP 9. Start. On completion of source program read in, the computer will stop with AL equal to zero.

a) If input is complete, release Skip Key 1 and Start. Assembly will now proceed with informative type-outs occurring as required.

b) If there is addtional input, mount source input on the input device as for Steps 1, 2, or 3.

1) If the input mode is the same as before, start from Step 9.

2) If the input mode is to change, start from Step 8.

e) Assembly with Corrections. TRIM III includes the same correction features that the TRIM CORRECTOR possesses. 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. 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 have been read in, the source program must follow with 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).

f) Miscellaneous Information.

1) Tape Unit Re-identification. To re-identify tape units, set P = 01404 and execute the steps outlined in paragraph 2.2-4b7c).

2) TRIM III Output 10. To stack TRIM II output no. 10 on scratch tape B, execute the following steps.

STEP 1. All desired outputs (up to 8) must have been pre-selected either via an OUTPUT operation or manually following the first "SELECT OUTPUTS IN A" typeout. Once the stacking option has been exercised for a given assembly, additional output requests from that assembly will cause an error.

STEP 2. After the last preselected output has been produced and the "SELECT OUTPUTS IN A" typeout has occurred, the computer will stop.

STEP 3. Set Skip Key 2 and Start. TRIM III will adjust its table index and the computer will stop with P = 01400 in preparation for the next assembly.

STEP 4. Release Skip Key 2, then proceed normally with the next assembly.

3) Manual Output Selection. If outputs have not been selected via the OUTPUT operation, TRIM III will request manual selection at assembly time. When the source program has been read in, TRIM III will type "SELECT OUTPUTS IN A" and stop with AU and AL equal to zero.

STEP 1. Set the low order bits of AU and AL to a single output number each.

STEP 2. Start. This process may be repeated until up to 8 outputs have been selected or AU or AL equals zero, whichever occurs first.

4) Unallocated Tags. When the "UNALLOC TAGS" typeout occurs followed by the sequential (L1) identifier and the tag name, the user may:

STEP 1. Start. TRIM III will assign a zero value to the tag.

STEP 2. Set AL to a desired value (if the tag pertains to a missing label within the program, also set bit 17 of AL to a 1) and Start.

STEP 3. Set AL as for 1 or 2 above; set AU $\neq$ 0 and Start. TRIM III will assign the value in AL to all subsequent unallocated tags without further type-out.

5) Restart Outputs. If for any reaon it is desired to abort an output, perform the following steps.

STEP 1. Press STOP.

STEP 2. Master Clear.

STEP 3. Set P = 01401.

STEP 4. Start. TRIM III will either process the next requested output or request output selection if there are no more unprocessed outputs.

6) Output No. 2 Override. If during an assembly run, poly-code generation causes a bank overflow TRIM III will type "POLY-CODE BANK OFL, OUTPUT 2 ONLY", and automatically start to produce an Output No. 2. To override,

STEP 1. Stop the computer,

STEP 2. Inspect and change the contents of address 01245, clearing it to zero.

STEP 3. Set P = 01401 and Start.

7) TRIM III Outputs. TRIM III provides 14 different optional outputs. They are:

a. No. 1 - Program memory storage and areas used summary on the line printer.

b. No. 2 - Side-by-side listing in source code on paper tape consisting of the absolute assembled program, sequential identifiers, source program, and error alarms.

c. No. 3 - Absolute assembled program on paper tape in source code.

d. No. 4 - Absolute assembled program on paper tape in bioctal format.

e. No. 5 - Relocatable assembled program on paper tape in bioctal format.

f. No. 6 - ALLOCation output on paper tape in source code (labels and addresses).

g. No. 7 - Ignore all library CALL operations of the input source program.

h. No. 10 - Absolute or relocatable assembled program on magnetic tape.

i. No. 11 - Source program on paper tape in source code.

j. No. 12 - Side-by-side listing on the 1004 printer with same information as output no. 2 plus any card identification and labels and addresses.

k. No. 13 - Relocatable or absolute assembled program on 80-column cards via the 1004 card punch. Each data card contains up to 8 instructions or words.

l. No. 14 - Format side-by-side listing on 1004 printer containing same information as a no. 2 output. This output is designed to fit on 8-1/2 inch by 11 inch printer vellum.

m. No. 15 - Source program on 80-column cards via the 1004 punch.

n. No. 16 - Source program on magnetic tape. When selecting an output no. 16, the following rule applies; if no tape transport is selected as the output tape, TRIM III will use scratch tape B. However, it will alert the operator to change tapes (in the event that an output no. 10 is to be saved) via the typeout, "IF NECESSARY CHANGE SCRATCH TAPES FOR THIS OUTPUT". If an output no. 10 is not to be saved or only source language outputs were

requested (outputs 6, 11, 15, 16), press START switch to continue. If an output 10 is to be saved, change tapes before pressing START.

8) Magnetic Tape Input to TRIM III. There are two magnetic tape formats acceptable as input to TRIM III.

a. Assembler Output No. 16. This output consists only of the source program; therefore, any declarative operators (e.g., OUTPUT, ALLOCation operations, DECKID, etc) must be provided by the user via some input medium.

b. Card-to-tape Output From a Card-to-tape Operation of the CART Service Routine. Since CART has the ability to stack several source programs on one tape, TRIM III requests the number of the program to be assembled whenever CART input is detected. The typeout IDENTIFY TAPE JOB IN AL occurs and the computer stops with AL = 0.

STEP 1. Set AL to the relative program position on the tape (1 means the first stacked program, 2 the second, 5 the fifth, etc.).

STEP 2. Start. Assembly now proceeds for the selected program. If two or more programs from the same CART tape are to be assembled together this sequence is repeated for each selection.

g) Error Detection and Display. TRIM III contains limited error detection capability. The majority of programer 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. If he wishes to bypass some error indications, the user may set Skip Key 3.

1) SET KEY 1. This typeout will occur if the user has not set Skip Key 1 at the start of the assembly process. To correct, set Skip Key 1 and start. Skip Key 3 has no effect on this error.

2) IDENT. MTUS IN A. This typeout occurs when no magnetic tape assignment has been made prior to the first assembly. To correct this condition, perform the following steps:

STEP 1. Identify magnetic tape units exactly as outlined in paragraph 2.2-4b7c).

STEP 2. Start. Skip Key 3 has no effect on this error.

3) SELECT OUTPUTS IN A. This typeout occurs if the programer has neglected to select his outputs via a programed output operation. To correct this condition, perform the following steps:

STEP 1. Set AU_{5-0} and AL_{5-0} to desired outputs.

STEP 2. Start. The computer will accept and store the outputs and stop. Repeat Steps 1 and 2 until not more than eight outputs have been selected. If either AU or AL euqals zero, TRIM III assumes that all selections have been made and proceeds. Skip Key 3 has no effect on this typeout.

4) MTU ERROR CTXX IMPR. COND. This typeout indicates an improper (not capable of being corrected) condition on the XX tape unit. The user will have to correct the condition before attempting to proceed. Skip Key 3 has no effect on this error.

5) SET BASE ADDR. IN AL. This typeout indicates that the programer has neglected to allocate the first program label. To correct, set AL_{14-0} to the desired base address and start. If skip key 3 is set, TRIM III will arbitrarily allocate the program to address 01200.

6) 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 (See paragraph 2.3.4 of this section). TRIM III does not stop after the typeout. NNNNN is the sequential program line identifier number and XXXXXX is the duplicate label. If Skip Key 3 is set, the typeout does not occur.

7) UNALLOC TAGS NNNNN XXXXXX AAAAA. By far the most common programer 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 XXXXXX (sequential line identifier and tag name). To correct, perform the following steps:

STEP 1. Set AL to the desired value.

STEP 2. If the tag refers to an instruction contained within the program being assembled, set AL_{17} to a 1.

STEP 3. If the user wishes any later unallocated tags allocated to the same address, set AU $\neq$ 0.

STEP 4. Start. TRIM III will type the manual allocation and use 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 Skip Key 3 is set, TRIM III will arbitrarily allocate all unallocated tags to address 00000.

8) TCS ERR XX TBL XX. This typeout indicates the table control system 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 2.2-2 describes the errors which TCS will detect.

TABLE 2.2-2. TCS ERRORS

ERROR NUMBER	MEANING
1	Illegal table number*
2	Illegal media designation*
3	Illegal TCS function code*
4	Misused Q-replace
5	Illogical TCS function sequence
6	Table not found*
7	Table overflow
8	Too many tape units referenced or item length of zero
9	Unrecoverable tape error

* Incorrect table design or control parameters.

All TCS errors except the table overflow are the result of TRIM III internal trouble and are unrecoverable. TRIM III has been designed so that a table overflow error will seldom occur. If an error does occur, the programer 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 programer may have to reassemble his program in smaller segments. Skip Key 3 has no effect on errors of this type.

9) POLY-CODE BANK OFL OUTPUT 2 ONLY. 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 will not stop following this typeout but will produce one Output 2, suppressing all other output requests. Skip Key 3 has no effect on this typeout.

h) Useful Assembler Addresses.

01400 - Start assembly

01401 - Go on to next selected output

01404 - Identify magnetic tape units

01000 - Normal input mode key

01001 - Channel number for UNIVAC I/O Console

01002 - Channel number for UNIVAC 1004 Card Processor

01245 - Bank overflow indicator-clear for Output 2 override

NAME: _____

2.2-5. STUDY QUESTIONS

a. How is the source tape for TRIM III prepared?

b. What is the purpose of the control reader?

c. Must the ALLOC Header follow the control header?

d. What additional poly-operations will TRIM III provide compared to TRIM II?

e. What does the operator DUMPR do when you are using TRIM III?

f. What additional outputs are available from TRIM III compared to TRIM II?

g. What happens during assembly using TRIM III if you use the operator SETADR?

h. How many tape units are used when you are assembling using TRIM III?

i. If your installation possesses only a paper tape bootstrap and you want to use TRIM III, what procedure would you follow?

j. You have forgotten to allocate the first program label. What indication would you notice and how would you correct?

SECTION 2 - SOFTWARE

2.3. OPERATOR SERVICE ROUTINES

2.3-1. OBJECTIVES

To familiarize the student with the uses of the operator service routines.

2.3-2. INTRODUCTION

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.

2.3-3. REFERENCES

Programers Reference Manual, Section on Operator Service Routines.

2.3-4. INFORMATION

a. General. The following paragraphs list and describe the available operator service routines.

1. 1219 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 content, store memory, etc.

2. 1219 UPAK II. This is a utility package which loads assembled programs from paper tape or magnetic tape. The routine has the same capabilities as UPAK I except that it also loads assembler-produced magnetic tape programs.

3. 1219 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 parameter specified. The package has the following modules:

a) Paper-tape handling

b) Computer control panel operations such as inspect and change

c) Magnetic tape handler, UMTH (basic handler for UNIVAC 1240 Magnetic Tape System)

d) Magnetic tape duplicator

e) Loader for assembler-produced magnetic tape object programs

f) Memory dump on the 1004 printer

g) Card load-dump

h) UPAKM paper tape handler

i) Print image on magnetic tape and tape-to-printer

j) Magnetic tape handler, JOSH (complete handler for UNIVAC 1240 Magnetic Tape System)

k) UPAK III controlling routine

Since UPAK III is modular, additional utility functions may be added without changing the general characteristics of the package.

4. TRIM Corrector. This routine corrects source programs. It reads a correction tape(s) and erroneous source tapes into the computer, making the necessary corrections, and punching a corrected tape. The routine is a companion to the TRIM I and TRIM II assemblers.

5. TRIM Library Builder. This routine updates source magnetic tape libraries which are used with the TRIM III assembler. It also has editing capabilities.

6. Program Trace. 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.

7. TALK System. This system is a real-time data extracting program. Data is extracted on a time-share basis with the operational program. In order to extract a maximum amount of data in a minimum amount of time, the TALK system is divided into three major segments, the translator, the extractor, and the editor.

8. UPAKM. This utility package is similar to UPAK I with the additional capability of paper tape load and dump involving six-digit addresses (65K memory).

9. CART. This routine performs a punched-card-to-magnetic tape conversion, with optional correction and listing capability.

This study guide will go into detail only on several of the more frequently used word routines. For a complete description of all the routines, refer to the Programers Reference Manual.

b. UNIVAC 1219 Paper Tape Utility Package 1

1. General Information. UPAK I is a collection of seven subroutines on punched paper tape in relocatable bioctal format which provide paper tape input/output functions and examination and alteration of memory for program debugging. The seven subroutines are as follows.

a) Load absolute typewriter code

b) Load absolute bioctal code

c) Load relocatable bioctal code

d) Dump absolute typewriter code

e) Dump absolute bioctal code

f) Inspect and change

g) Store constant in memory

UPAK I 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. The paper tape load and dump routines are operable under program control or manually from the computer. However, the inspect-and-change and store-constant-in-memory functions are operable from the computer control panel only.

UPAK I has been designed for computers with a maximum of 32K storage locations. For this reason, the dump and load subroutines (with the exception of load relocatable bioctal code) cannot be used to either load or dump a tape above address 100000. Those installations possessing a larger than 32K memory computer should use UPAKM for operations involving addresses above 100000.

UPAK I occupies 1,031 octal memory locations exclusive of addresses 00540 through 00777 which are used by the UPAK I loader. Entrance addresses to the several subroutines are assigned relative to the UPAK I base address as will be subsequently shown. Increments to the base are in octal.

Address	Routine Entrance
Base + 0	Program entrance paper tape load
Base + 2	Program entrance dump typewriter code
Base + 4	Program entrance dump bioctal code
Base + 6	Manual entrance paper tape load
Base + 7	Manual entrance dump typewriter code
Base + 10	Manual entrance dump bioctal code
Base + 11	Manual entrance inspect and change
Base + 12	Manual entrance store constant in memory

UPAK I subroutines use the currently active B-register, but they store and restore its original value. Routines operating under program control also store and restore the user's special register setting.

2. Operations and Program Formats

a) Load Absolute Typewriter Code. A carriage return followed by either an 8 or an 88 and another carriage return activates the load routine. A single 8 indicates the input tape has no checksum; an 88 indicates the input tape

has a checksum. The routine ignores any data which may precede either of these two combinations. Each tape instruction is preceded by a five-digit address which need not be sequential; all five digits must be present. The next six digits constitute the instruction to be loaded; all six digits must be present. The load routine, in effect, accumulates the first 11 octal digits following a carriage return, ignoring all other character codes including notes. A carriage return signals the end of the instruction. If less than 11 octal digits are accumulated, the instruction will not be loaded. A final carriage return followed by a double period (..) terminates the load and initiates a checksum verification when required.

No checksum format	Checksum format
8	88
xxxxx xx xxxx	xxxxx xx xxxx
xxxxx xx xxxx	xxxxx xx xxxx
. . .	. . .
. . .	. . .
xxxxx xx xxxx	xxxxx xx xxxx
..	..
	xx xxxx (checksum)

When 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 Skip Key 1 is set, the load subroutine will perform a checksum verification without loading the tape into memory.

b) 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. Six-digit instructions follow without further addressing, and the tape is terminated by a six-digit checksum.

Absolute Bioctal Tape Format

76	Absolute bioctal code
II	
II	I - Initiate address
IF	
FF	F - Final address
FF	
XX	X - Instruction words
XX	

XX	
..	
..	
XX	
XX	
XX	
CC	C - Checksum
CC	
CC	

If Skip Key 1 is set, the absolute bioctal load subroutine will perform a checksum verification without loading the tape into memory.

c) 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 instruction is preceded by a one-digit code which tells the load routine how to modify the instruction for storage. The tape is terminated by a six-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.

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 Skip Key 1 is set, the relocatable bioctal load subroutine will perform a checksum verification without loading the tape into memory.

d) Dump Absolute Typewriter Code. The dump is initiated manually at the computer or under program control for output on punched paper tape. The 88 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.

e) Dump Absolute Bioctal Code. The absolute bioctal dump is initiated manually at the computer or under program control. The output on punched paper tape is the 76 code followed by:

1) The initial and final addresses of the area being dumped

2) The contents of the inclusive memory addresses

3) The checksum

If more than one program area is dumped successively on the same tape, the format for each such area is as just described.

f) 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 address other than the next sequential address, he may do so by setting the new address in AU before returning AL to memory.

g) 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. Operating Instructions.

a) Loading UPAK I. UPAK I is provided on punched paper tape. The tape is subdivided into two parts: the loader in absolute bioctal format and UPAK I in relocatable bioctal format. To load UPAK I, the following procedure is used:

STEP 1. Place tape in reader.

STEP 2. Master clear.

STEP 3. Press LOAD button to activate paper tape bootstrap.

STEP 4. Start. The computer will stop after the loader is in memory with AU and AL equal to zero.

STEP 5. Set AU to the desired UPAK I base address.

STEP 6. Start. The computer will stop, after UPAK I has been loaded, with AU and AL equal to zero.

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.

b) Error Detection. 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 will come to a normal stop with AU and AL equal to zero. If they do not agree, the computer will stop with (AU) = load checksum and (AL) = tape checksum.

The format of the UPAK I tape is such that only the basic bioctal load routine is loaded via paper tape bootstrap. Control is then given to a temporary checksum verification routine which verifies the bioctal load portion of the tape. If verification is correct, control is given to the bioctal load routine which reads in the UPAK I loader. If verification is incorrect, the computer will come to a normal stop with AU equal to the load checksum and AL equal to the tape checksum.

c) Paper Tape Load (All Formats).

1) Manual Operation.

STEP 1. Mount tape in reader.

STEP 2. Set P to UPAK I base address + 6.

STEP 3. Set Skip Key 1 if checksum verification only is desired.

STEP 4. For relocatable bioctal load only set starting address in AU. (Not required if Skip Key 1 is set.)

STEP 5. Start.

STEP 6. Successive tapes may be loaded without resetting P.

2) Program Operation.

STEP 1. Tape must be mounted in reader.

STEP 2. For relocatable bioctal load only, enter AU with starting address.

STEP 3. Execute a return jump or indirect return jump to UPAK I base address.

d) Dump Absolute Typewriter Code.

1) Manual Operation.

STEP 1. Set P to UPAK I base address + 7.

STEP 2. Set AU to first address to be dumped.

STEP 3. Set AL to last address to be dumped.

STEP 4. Start.

STEP 5. Successive dumps may be taken by starting from Step 2.

2) Program Operation.

STEP 1. Enter AU with first address to be dumped.

STEP 2. Enter AL with last address to be dumped.

STEP 3. Execute a return jump or indirect return jump to UPAK I base address + 2.

e) Dump Absolute Bioctal Code.

1) Manual Operation.

STEP 1. Set P to UPAK I base address + 10 (octal).

STEP 2. Set AU to first address to be dumped.

STEP 3. Set AL to last address to be dumped.

STEP 4. Start.

STEP 5. Successive dumps may be taken by starting from Step 2.

2) Program Operation.

STEP 1. Enter AU with first address to be dumped.

STEP 2. Enter AL with last address to be dumped.

STEP 3. Execute a return jump or indirect return jump to UPAK I base address + 4.

f) Inspect and Change.

1) Manual Operation Only.

STEP 1. Set P to UPAK I base address + 11 (octal).

STEP 2. Set AU to desired memory address.

STEP 3. Start. The computer will stop with the address in AU and its contents in AL.

STEP 4. The user may now change the address in AU and/or the contents in AL.

STEP 5. Start.

a) If contents only were altered, these will be stored at the original address and the computer will stop with the next sequential address in AU and its contents in AL, etc.

b) If the address only was changed, (AL) will be restored to its proper memory location, and the computer will stop with the new address in AU and its contents in AL, etc.

c) If both the address in AU and its contents in AL were changed, (AL) will be stored at the original address, and the computer will stop with the new address in AU and its contents in AL.

STEP 6. Any number of such sequences may be executed starting with Step 5.

g) Store Constant in Memory.

1) Manual Operation Only.

STEP 1. Set P to UPAK I base address + 12 (octal).

STEP 2. Set first storage address in AU.

STEP 3. Set last storage address in AL.

STEP 4. Start. UPAK I will record these addresses, and the computer will stop with AU cleared.

STEP 5. Set desired constant in AU.

STEP 6. Start. UPAK I will store (AU) at successive memory locations within the parameters established in Steps 2 and 3.

STEP 7. Additional entrances may be made starting from Step 2.

c. UNIVAC 1219 Utility Package II.

1. General Information. UPAK II is a collection of eight subroutines on punched paper tape in relocatable bioctal format which provide TRIM III output no. 10 load, paper tape input/output functions, and examination and alteration of memory for program debugging. The eight subroutines are as follows:

a) Load TRIM III output no. 10 from magnetic tape

b) Load absolute typewriter code

c) Load absolute bioctal code

d) Load relocatable bioctal code

e) Dump absolute typewriter code

f) Dump absolute bioctal code

g) Inspect and change

h) Store constant in memory

UPAK II 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 UPAK II functions operate either under program control or manually from the computer except for the inspect-and-change and store-constant-in memory functions, which are operable from the computer control panel only.

UPAK II has been designed for computers with a maximum of 32K storage locations. For this reason the dump and load subroutines, with the exception of Load TRIM III output no. 10 and load relocatable bioctal code, cannot be used to either load or dump a tape above address 100000. Those installations possessing larger than a 32K memory computer should use UPAKM for paper tape operations involving addresses above 100000.

UPAK II occupies 1633 octal memory locations, exclusive of addresses 00540 through 00777 which are used by the UPAK II loader. Entrance addresses to the several subroutines are assigned relative to the UPAK II base address as will be subsequently shown. Increments to the base are in octal.

Address	Routine Entrance
Base + 0	Program entrance TRIM III output no. 10 load
Base + 2	Program entrance paper tape load
Base + 4	Program entrance dump typewriter code
Base + 6	Program entrance dump bioctal code
Base + 10	Manual entrance TRIM III output no. 10 load
Base + 11	Manual entrance paper tape load

Base + 12	Manual entrance dump typewriter code
Base + 13	Manual entrance dump bioctal code
Base + 14	Manual entrance inspect and change
Base + 15	Manual entrance store constant in memory

UPAK II subroutines use the currently active B-register, but they store and restore its original value. Routines operating under program control also store and restore the user's special register setting.

2. Operations and Program Formats.

a) Load TRIM III Output No. 10 from Magnetic Tape. 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 the UPAK II magnetic tape load subroutine. The user has two load options: UPAK II will load the program absolutely with addressing assigned at assembly time, or UPAK II will load the program relative to any specified base address.

TRIM III writes output no. 10 on magnetic tape in 140_8 word records. Each record consists of up to 37_8 items consisting of three computer words. Each item has the following format:

Word 0	E	Sequential identifier	Used by TRIM III only
Word 1	M	Address	
Word 2		Instruction	

Word 0 contains a line error counter, E, and a sequential line identifier used by TRIM III but not by UPAK II.

M is the modification code which 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 requested base address for relocatable load)

Address is the address for this instruction assigned at assembly time.

Instruction is the machine instruction of constant to be stored.

Each output no. 10 begins with a sentinel record followed by the data records and terminated by one sentinel word corresponding to the sentinel record and a tape mark.

The no. 10 load subroutine indicates a successful load by terminating with (AL) = 0.

b) TRIM III No. 10 Output Load.

1) Manual Operation.

STEP 1. Mount magnetic tape holding the no. 10 output on a tape unit.

STEP 2. Set P to UPAK II base address + 10 (octal).

STEP 3. Start. The computer will stop for load parameters.

STEP 4. Set AU to NNCCST where NN is 1 through 40_8 indicating the file number of the output no. 10 to be loaded; CC is the tape unit channel number (0 - 17_8); S is the synchronizer (cabinet) number; and T is the tape transport number.

STEP 5. Set AL to zero for absolute load or to the desired base address for relocatable load.

STEP 6. Start. The computer will stop with one of the following indications in AL.

a) (AL) = 0 successful load.

b) (AL) = 1 tape status indicates improper condition. The user should correct the condition if possible and start, to continue with the load.

c) (AL) = 777777 indicates the load subroutine has attempted to read the same record seven times and failed.

2) Program Control.

STEP 1. Mount magnetic tape holding the no. 10 output on a tape unit.

STEP 2. Enter AU with NNCCST where NN is 1 through 40_8 indicating the file number of the output no. 10 to be loaded, CC is the tape unit channel number (0 - 17_8); S is the synchronizer (cabinet) number; and T is the tape transport number.

STEP 3. Enter AL with zeros for absolute load or with the desired base address for relocatable load.

STEP 4. Execute a return jump or indirect return jump to the UPAK II base address. The load routine will restore control to the using program with (AL) equal to one of the following conditions:

a) (AL) = 0 successful load

b) (AL) = 1 tape status indicates improper condition

c) (AL) - 777777 indicates the load routine has attempted to read the same record seven times and failed.

d. UNIVAC 1219 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 component modules. UPAK III presently has the following modules:

Module 1	Paper Tape Handler
Module 2	Magnetic Tape Handler (UMTH)
Module 3	Magnetic Tape Duplication
Module 4	TRIM III Output 10 Load
Module 5	Inspect and Change and Store Constant
Module 6	Print Memory Contents
Module 7	Data Card Handler
Module 8	Modified 1219 Paper Tape Handler
Module 9	Printer Line Image on Tape and Tape-To-Printer
Module 10	Magnetic Tape Handler (JOSH)

The modular framework of UPAK III, however, will permit the incorporation of additional modules with a minimum of effort.

UPAK III assumes a minimum equipment configuration of one UNIVAC 1219 Computer with 16,384D words of core memory, one Type 1240 Magnetic Tape Unit with two transports, a Type 1232A Input/Output Console, and a UNIVAC 1004 Card Processor. Presently, the 1004 is only used for modules 6, 7, and 9. 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 operate either under program control or manually from the computer control panel.

2. Control Program.

a) Program Description. The control program of UPAK III is basically a module loader. It may be loaded by magnetic tape bootstrap or paper tape load program 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 will load one or all UPAK III modules. If a specific (non-zero) module is requested, only one module may be loaded. If more than one module is desired at specific addresses, the user must repeat the load procedure for each particular module. If the module number is zero, UPAK III will load all of the modules at the base addresses denoted in table 2.3-1; the base address parameter will be ignored.

Module entrances are determined by standard increments to their load address as shown in table 2.3-2.

Care must be exercised in specifying a module base load address to the control program such that any module is loaded entirely within one memory bank.

b) Operating Procedure. To load the UPAK III control program via magnetic tape bootstrap, mount the UPAK III tape on transport 1 of cabinet 1. Then perform the following.

STEP 1. Master clear.

STEP 2. Push LOAD button to activate bootstrap.

STEP 3. Start. When the computer stops, if the parameter displayed in AU is not desired, enter the desired parameter in AU. The parameter format is:

15	12	10	6	3	0
S/D	Ø	Ø	C	S	T

where S/D $\neq$ Ø specifies single channel operation, = 0 specifies dual channel operation, and CST specifies the tape unit channel, cabinet and transport respectively.

NOTE

The logical selection of the tape transport is not restricted to cabinet 1, transport 1 at this time.

STEP 4. Enter a sixteen-bit control program load address in AL if desired. If AL is zero, the control program will be loaded at its assigned address.

STEP 5. Start. 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.

STEP 1. Mount the UPAK III magnetic tape on cabinet 1, transport 1.

STEP 2. Place the PTMTBS* No. 4 paper tape in the paper tape reader.

* Magnetic tape bootstrap on paper tape.

TABLE 2.3-1. ENTRANCE ADDRESSES AND ASSIGNED BASES

Module	No.	Base	Entrance Increments to Base
PTHAN	1	01100	+ Ø - ICH Entrance
Size: 1027_8			+ 1 - STC Entrance
			+ 2 - Programmed PT Load
			+ 4 - Manual PT Load
			+ 5 - Programmed Typewriter Code Dump
			+ 7 - Manual Typewriter Code Dump
			+1Ø - Programmed Bioctal Dump
			+12 - Manual Bioctal Dump
UMTH	2	02140	+ Ø - Programmed Entrance
Size: 600_8			+ 2 - Manual Entrance
MTDUP	3	03000	+ Ø - Programmed Entrance
Size: 712_8			+ 2 - Manual Entrance
LOAD1Ø	4	04000	+ Ø - Programmed Entrance
Size: 600_8			+ 2 - Manual Entrance
ICH-STC	5	04700	+ Ø - ICH Entrance
Size: 77_8			+ 1 - STC Entrance
PRINTC	6	05000	+ Ø - Programmed Entrance
Size: 450_8			+ 2 - Manual Entrance
DATCD	7	06000	+ Ø - Programmed Card Load
Size: 626_8			+ 2 - Manual Card Load
			+ 3 - Programmed Card Dump
			+ 5 - Manual Card Dump
UPAKM	8	06700	+ Ø - ICH Entrance
Size: 1060_8			+ 1 - STC Entrance
			+ 2 - Programmed PT Load

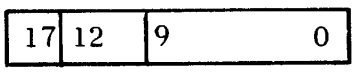
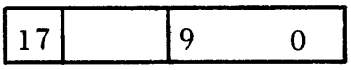
TABLE 2.3-1. ENTRANCE ADDRESSES AND ASSIGNED BASES (CONT.)

Module	No.	Base	Entrance Increments to Base
			+ 4 - Manual PT Load
			+ 5 - Programmed Typewriter Code Dump
			+ 7 - Manual Typewriter Code Dump
			+1Ø - Programmed Bioctal Dump
			+12 - Manual Bioctal Dump
POTPOP	9	10000	+ Ø - 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	10	10500	+ Ø - Programmed Entrance
Size: 640_8			+ 2 - Check Status Entrance
			+ 4 - Check Busy Entrance
			+10 - Manual Entrance

TABLE 2.2-2. CONDENSED PARAMETERS FOR UPAK III

		17 12 | 9 0	17 0
Control Program		M M Ø C S T	Load Address
Paper Tape Handler (PTHAN)	Absolute Typewriter Code Load	None	
	Absolute Bioctal Load	None	
		17 0	17 0
	Relative Bioctal Load	Load Address	Unused
	Absolute Mem. Dump Type-writer Code	Ø F I R S T	Ø F I N A L
	Absolute Mem. Dump Bioctal	Ø F I R S T	Ø F I N A L
		17 | 14 | 12 | 9 0	17 9 | 5 0
Magnetic Tape Handler (UMTH)		$^{0}/_{1}$ OP C S T	F D M #
		I A	T A
		Search Key 1	Search Key 2
		17 10 | 9 0	17 6 | 5 0
Magnetic Tape Duplicator (MTDUP)		Function Word C S T	(TO tape) S T
		FROM tape buffer IA	FROM tape buffer TA
		TO tape buffer IA	TO tape buffer TA

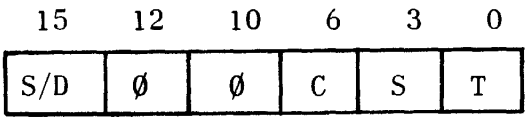
TABLE 2.2-2. CONDENSED PARAMETERS FOR UPAK III (CONT.)

		
Output 1Ø Load (LOAD1O)	N N Ø C S T	Load Address
Inspect and Change (ICH)	Address	Contents
Store Constant (STC)	Ø F I R S T	Ø F I N A L
	Constant	Unused
Print Memory Contents (PRINTC)	I A	T A
Card Load (DATCD-CLOD)	Load Address (If no Address Card)	Unused
Card Dump (DATCD-CDMP)	I A or T A	T A or I A
		
Line Image on Tape (POT)	A C S T	Buffer I A
Tape-To-Printer (POP)	C S T	Unused
		
Magnetic Tape Handler (JOSH)	$^{0}/_{1}$ OP C S T	M F P D #
	I A	T A
	Search Key 1	Search Key 2

UPAKM has the same A register parameters as PTHAN for the various load and dump routines.

STEP 3. Push the LOAD button on the 1219 control panel.

STEP 4. Start. When the computer stops, if the parameter displayed in AU is not desired, enter the desired parameter in AU. The parameter format is:



where S/D ≠ Ø specifies single channel operation, = 0 specifies dual channel operation, and CST specifies the tape unit channel, cabinet and transport respectively.

NOTE

The logical selection of the tape transport is not restricted to cabinet 1, transport 1 at this time.

STEP 5. Enter a sixteen-bit control program load address in AL if desired. If AL is zero, the control program will be loaded at its assigned address.

STEP 6. Start. When the computer stops the control routine is loaded.

To operate the UPAK III control routine,

STEP 1. Master clear.

STEP 2. Set P to the base address of the control routine.

STEP 3. Set AU to MMØCST* where MM is the module number; CST is the UPAK III tape address. If MM is zero, all modules will be loaded at their assigned addresses.*

STEP 4. Set a 16-bit load address for the requested module in AL, if desired. If AL is zero, the control program will assign a load address.

STEP 5. Start.

STEP 6. To load another module, repeat Steps 3 through 5 above.

c) 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 = ØØ. To validate this parameter it is necessary to specify the number of modules in the system by errata to the UPAK III control routine (word, base + 66_8) before specifying MM = ØØ. This word should contain 7000XX, where XX

* See module assignments in table 2.1-2.

is the number of modules minus one in octal. Without this errata, MM = ØØ results in loading the original UPAK III modules only.

To add modules to UPAK III,

STEP 1. Mount the TRIM III tape.

STEP 2. Mount the UPAK III tape as the TRIM III scratch tape.

STEP 3. Load TRIM III.

STEP 4. Change TRIM III address 01057 from zero to the number of files on the UPAK III tape (actual number of modules plus two).

STEP 5. Assemble the program to become the next UPAK III module.*

STEP 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" printout occurs, set Skip Key 2 and start.

STEP 7. The computer will stop at address 01400. Release Skip Key 2.

STEP 8. Repeat Steps 5 through 7 until all desired additional modules have been assembled.

STEP 9. Rewind and dismount the expanded UPAK III tape.

3. Paper Tape Handler Module.

a) Program Description. PTHAN is a collection of seven subroutines which provide paper tape input/output functions and examination and alteration 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

* Care should be exercised in specifying a base address to ensure compatibility with other modules.

** All outputs of any one program must be selected the first time TRIM III solicits outputs. Restarting an output from address Ø14Ø1 is not allowed.

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 below and in table 2.1-2. Increments to the base are octal.

Address	Entrance
Base + Ø	ICH entrance
Base + 1	STC entrance
Base + 2	Program entrance paper tape load
Base + 4	Manual entrance paper tape load
Base + 5	Program entrance dump typewriter code
Base + 7	Manual entrance dump typewriter code
Base + 10	Program entrance dump bioctal code
Base + 12	Manual entrance dump bioctal code

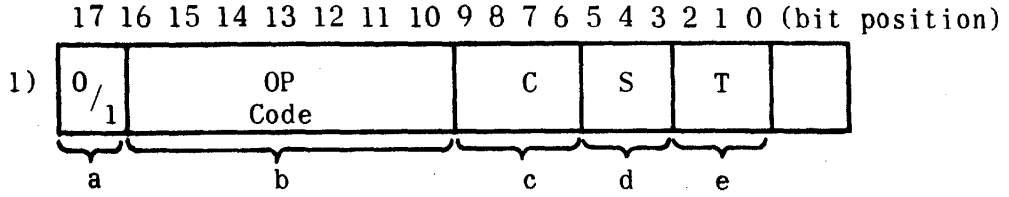
PTHAN subroutines use the currently active B register but store and restore its original value. The load routines when operating under program control also store and restore the user's special register setting.

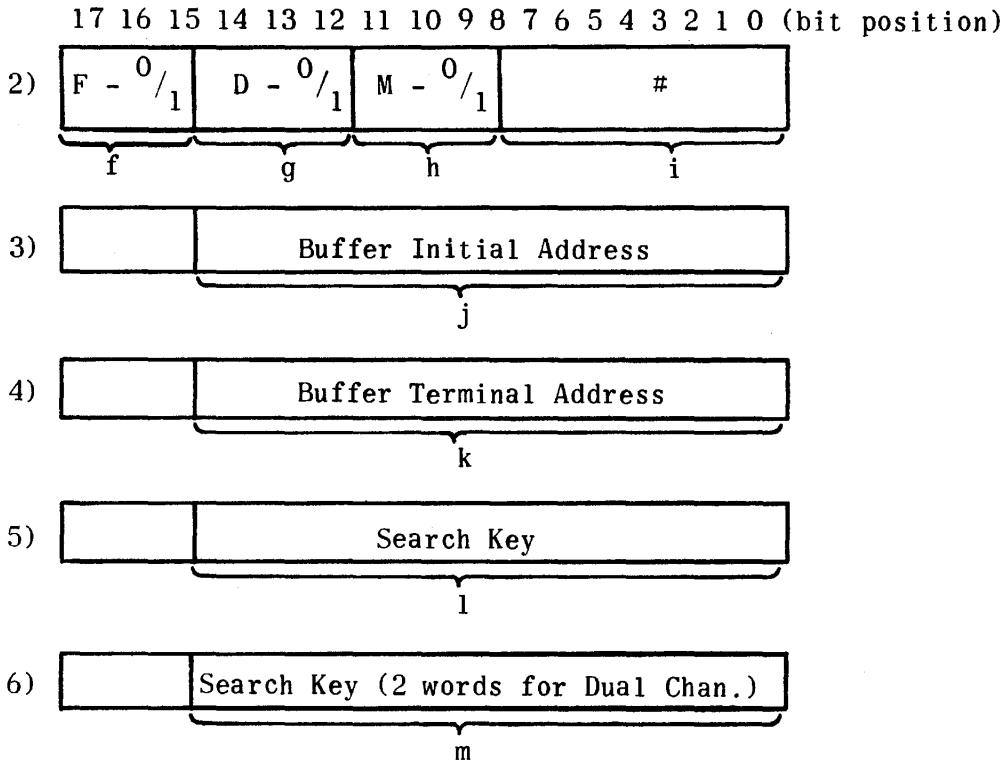
4. Magnetic Tape Handler Module.

a) 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 and rewind, with the added options of single or dual channel operation, high or low density, and bioctal or octal recording format. 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 programed entrances are provided to UMTH. Refer to table 2.1-2.

b) Input Parameters. From two to six parameter entries stored manually into the A register, or program stored in A and memory, govern the operation of the magnetic tape handler. They are as follows:





where:

a = Applicable to search, space file, and pass n records.

0 = forward

1 = backward

b = Operation Code

Code

1	Read variable length record
2	Write variable length record
3	Write tape mark
4	Search
5	Pass n records
6	Space file
7	Rewind

c = C - I/O Channel ($0\text{-}17_8$). For dual channel operation C must be the odd-numbered channel.

d = S - Magnetic Tape synchronizer (cabinet) number (1-4).

e = T - Magnetic Tape transport number (1-4).

f = F - Format of tape recording.

0 = bioctal

1 = octal (redundant)

g = D - density of tape recording.

0 = low

1 = high

h = M - mode of tape communication.

0 = single channel

1 = dual channel

i = Number of records to pass forward or backward or number of files to space forward or backward. Up to 77_8 such groups may be passed on 1 execution of UMTH. If more than 77_8 are required, it is necessary to re-execute UMTH.

j = Buffer initial address for read, write, or search operations.

k = Buffer terminal address for read, write, or search operations.

l = Search key for single channel search operations, or first word of search key for dual channel search operations.

m = Second word of search key for dual channel search operations.

c) Operating Procedure.

1) Operation Under Program Control. The calling program enters AU with parameter one, AL with parameter two, and, if applicable, enters the active B register with the first of a group of consecutive addresses. This group may consist of two, three, or four words depending upon the operation to be performed. Parameters three and four are needed for read or write operations; parameters three, four, and five are needed for single channel search operations; and parameters three, four, five, and six are required for dual channel search operations. The calling program then executes a return jump or indirect jump to UMTH base address.

Upon resumption of control, the calling program may check the contents of AU to verify a good operation.

To manually operate UMTH:

STEP 1. Set P to UMTH base address + 2.

STEP 2. Set parameter 1 in AU; set parameter 2 in AL.

STEP 3. Start. When the computer stops, set parameter 3 in AU; set parameter 4 in AL.

STEP 4. Start. If the operation to be performed is a search operation, the computer will stop again. Set parameter 5 in AU, and, if needed, set parameter 6 in AL. Start again.

2) Alarms and Status Indications. Under both manual and program operation, UMTH indicates the status of the operation attempted in AU.

If (AU) = zero, a successful operation is indicated.

If (AU) is non-zero, but positive, the job is incomplete. UMTH makes up to seven recovery tries before indicating job incomplete, and if the operation does terminate in an error, it terminates with the tape positioned to re-execute the incomplete operation. AU will contain the actual tape status word.

If (AU) is non-zero and negative, it indicates an improper condition of the tape unit requiring manual intervention. Again, AU will contain the tape status word, but the whole word will be normalized left (3 bits).

3) Special Considerations. UMTH assumes the following:

1) Search constants for a backward search must be reversed characterwise.

2) Whenever dual channel operation is selected, the buffer limits must begin with an even address and end with an odd address or begin with an odd address and end with an even address.

3) Address 00141 shall be reserved for use by UMTH as an indirect interrupt address.

4) The original contents of AU, AL, and B will not be restored upon exit.

5. Magnetic Tape Duplication Module.

a) Program Description. MTDUP is a module of UPAK III for the UNIVAC 1219 Computer when operating with Type 1240 magnetic tape units and a Type 1232A Input/Output 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:

STEP 1. Rewind FROM tape and TO tape.

STEP 2. Read one record from FROM tape into user-specified buffer.

STEP 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.

STEP 4. Rewind both tapes if verification option is selected.

STEP 5. Read one record from FROM tape into user-specified buffer and checksum it.

STEP 6. Read one record from TO tape into user-specified buffer and checksum it. Compare record checksums 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.

STEP 7. Rewind both tapes if rewind option was selected.

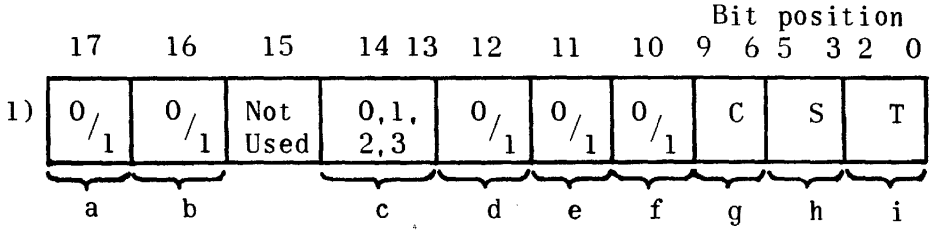
STEP 8. If error counter is zero, type COPY-OK; otherwise type ERR-XXX where XXX is the number of copy errors detected.

NOTE

The user may alter the number of successive tape marks terminating his FROM tape. To do this, he should store the exact number at MTDUP label CAT (base address + 626_8).

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 shall be loaded entirely within one memory bank. The module operates either under program control or manually from the computer control panel.

b) Input Parameters. Input parameters to MTDUP are shown below in the required order.



a = 0 - single channel mode

1 - dual channel mode

b = 0 - do NOT rewind tapes after duplication

1 - Rewind tapes after duplication

c = 0 - MOD 3 (18-bit transfer)

1 - MOD 4 (24-bit transfer)

2 - MOD 5 (30-bit transfer)

3 - MOD 6 (36-bit transfer)

d = 0 - bioctal character format

1 - octal character format

e = 0 - even parity

1 - odd parity

f = 0 - even parity

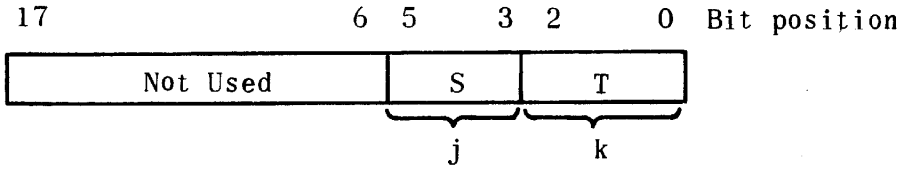
1 - high density

g = Channel number for both the FROM and TO tapes. If dual channel mode is selected, g must be set to the odd-numbered channel.

h = Cabinet number for FROM tape

i = Transport number for FROM tape

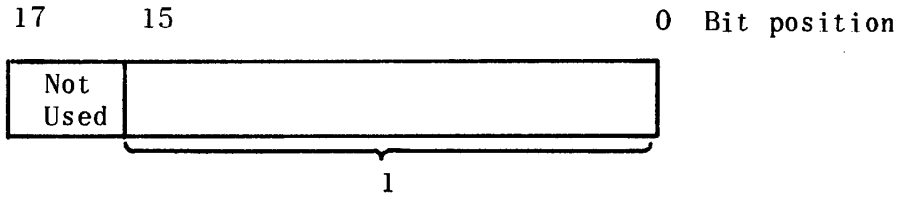
2)



j = Cabinet number for TO tape

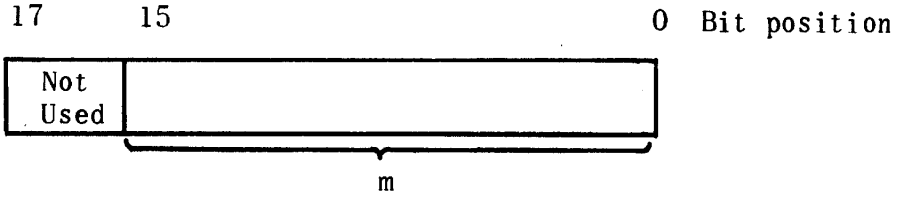
k = Transport number for TO tape

3)

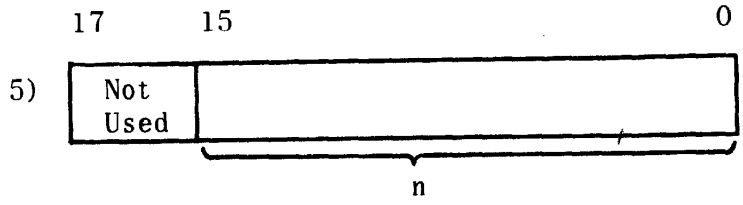


l = Memory address to be used as buffer initial address for FROM tape.

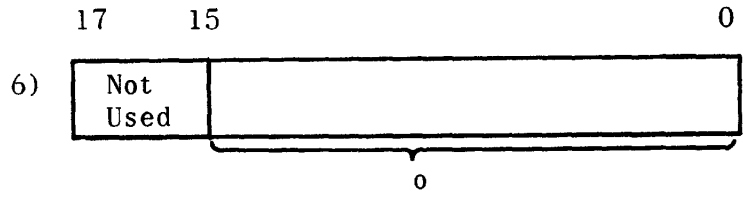
4)



m = Memory address to be used as buffer terminal address for FROM tape.



n = Memory address to be used as buffer initial address for TO tape. If copy verification is not desired, parameter 5 should be set to zero.



o = Memory address to be used as buffer terminal address for TO tape. If copy verification is not desired, parameter 6 should be set to zero.

NOTE

Parameters 4 and 6 need not be the exact buffer terminal address. However, the buffer limits defined must be adequate to accommodate the largest record on the FROM tape.

c) Operating Procedure.

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.

To manually operate MTDUP,

STEP 1. Set P to MTDUP base address + 2.

STEP 2. Set parameter 1 in AU; set parameter 2 in AL.

STEP 3. Start. When the computer stops; set parameter 3 in AU; set parameter 4 in AL.

STEP 4. Start. When the computer stops; set parameter 5 in AU; set parameter 6 in AL.

STEP 5. Start.

2) Alarms. An improper condition of a tape unit or a status word error after seven tries causes MTDUP to exit with (AU) = status word from tape system. This indicates the MTDUP did not complete the duplication. If the tape duplication is completed without tape system errors, (AU) = zero.

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) will be displayed in AL.

3) Special Considerations. MTDUP makes the following assumptions:

a That both the FROM and TO tapes are mounted on tape transports served by a common input/output channel.

b That the 1232A I/O Console is on-line with the 1219 Computer.

c That address 00141 shall be used by MTDUP as an indirect interrupt address.

d That MTDUP shall not restore the original contents of AU, AL, or B.

6. TRIM III Output 10 Load Module.

a) 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 will stack a maximum of 40_8 of these outputs on one tape. The user has two load options; a) that the program shall be loaded absolutely with addressing assigned at assembly time, or b) that the program shall be loaded relative to any specified base address.

TRIM III writes output no. 10 on magnetic tape in 140_8 word records. Each record consists of 37_8 items of 3 computer words each. An 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) M is the modification code which 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

Code	Meaning	Type of Instruction
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)

3) Address is the address for this instruction assigned at assembly time.

4) Instruction is the machine instruction or constant to be stored.

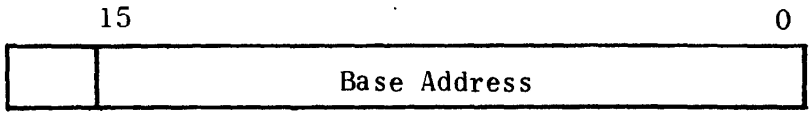
Each output no. 10 begins with a sentinel record followed by the data records and terminated by one sentinel word corresponding to the sentinel record and a tape mark.

The LOAD1Ø module indicates a successful load by terminating with (AL) = 0.

b) Input Parameters. Two parameters govern the operation of LOAD1Ø. They are as follows:

1)

2)



where:

NN is 1 through 40_8 indicating the file number of the output no. 10 to be loaded.

C is the tape unit channel ($0-17_8$)

S is the synchronizer (cabinet) number (1-4)

T is the tape transport number (1-4)

Base address is a 16-bit address; zero for absolute load.

c) Operating Procedure.

1) Program Operation of LOAD1Ø.

STEP 1. Mount the magnetic tape holding the output 10 on a tape unit.

STEP 2. Enter AU with parameter 1; AL with parameter 2.

STEP 3. Execute a return jump or indirect return jump to the LOAD1Ø base address.

To manually operate LOAD1Ø,

STEP 1. Mount the magnetic tape holding the output IO on a tape unit.

STEP 2. Set P to LOAD1Ø base address + 2.

STEP 3. Set parameter 1 in AU; set parameter 2 in AL.

STEP 4. Start.

2) Alarms. The load routine will restore control to the using program with (AL) containing one of the following status conditions:

a (AL) = 0 indicates a successful load.

b (AL) positive non-zero indicates the load routine has attempted to read the same record seven times and failed. AL contains the status word.

LOAD1Ø will always stop if an improper condition arises. AL will contain the tape status word normalized left (sign bit set).

3) Special Considerations.

a LOAD1Ø uses address 00141 as an indirect interrupt address.

b LOAD1Ø does not restore the original contents of AU, AL, or B.

7. Print Memory Contents.

a) Program Description. PRINTC is a module of UPAK III for the UNIVAC 1219 Computer when operating with a UNIVAC 1004 Card Processor. This module will list a specified area of 1219 memory on the 1004 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 five 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 two 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 an entire line's addresses are zeros, the address in column one is not printed either.

5) If the contents of two or more consecutive lines' addresses 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 non-zero 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.

b) Operating Procedure. For operation of print memory contents,

1) Make certain 1004 Card Processor is on and in a ready condition; printer should have sufficient paper.

2) AU must contain the initial dump address; AL must contain the final dump address.

Under program control:

3) Calling routine should do an RJP to PRINTC base address, or an IRJP to an address containing the PRINTC base address.

Under manual operation,

4) Set P = PRINTC base address + 2, then start. 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.

8. Data Card Handler.

a) 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 UNIVAC 1004 Card Processor. The two subroutines are:

CLOD - Load relocatable cards
CDMP - Dump absolute cards in relocatable format.

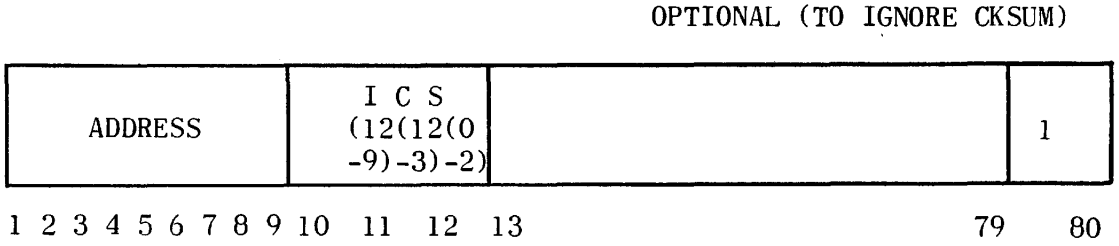
DATCD may be loaded anywhere in computer memory above address 01000 with the single restriction that the entire package must be entirely contained within a single memory bank. Both functions operate either manually or under program control.

CLOD is a subroutine of DATCD that will load into a specified area or areas of computer memory data words in relocatable format which it possesses from 80-column Hollerith coded cards read via the 1004 Card Processor.

1) Address Card. CLOD examines the eightieth 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 uses the contents of the upper half of the A register (AU) as the initial load address. Any other digit than a 1 or 4 in the 80th column of the first card causes CLOD to ignore that card data 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 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 data card format, he may cause CLOD to skip its checksum procedure by punching the three letters ICS in columns 10, 11, and 12 of address card (Ignore Checksum).

Hence, the address card is an optional card in the relocatable card load with an optional feature on the card itself.



ADDRESS CARD FORMAT

2) Data Card. Following the address card (if any), each card except the last card will have the following format:

a Eight (10 octal) 6-digit 1219 computer memory words or special load modifier words in columns 1 through 48 inclusive.

b Eight (10 octal) relocatable modification digits one for each data word) in columns 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 data 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, the memory dump on cards, in columns 67 through 72. For cards prepared on the key punch with ICS in columns 10 12 of the address card, these columns will be ignored. By cumulative, we mean that the preceding card's checksum is added to the present one and punched, etc.

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, so it is optional.

e A control digit of 4 in column 80. This is mandatory for all data cards to be loaded by CLOD; any other digit will cause the data of that card to be ignored and another card read.

1st Data word	2nd thru 8th data 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

3) Last Data Card. The last card of any relocatable or absolute card load will have almost exactly the same format as the other data cards. The number of loadable data words on the last card will vary from seven to none. The last card must always contain a data word of any value which has a corresponding modification digit of 7. CLOD ignores the data word and terminates.

CDMP is a subroutine of DATCD that will dump a specified area of core memory on 80 column cards. It first converts the octal words of the computer core memory area that is specified for output into excess-three code. It converts enough words to fill the card buffer or finish the specified area, and then it punches the data via the 1004 Card Processor onto 80-column Hollerith-coded cards. This process continues until the entire specified area has been punched.

b) Output Format.

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 eightieth column.

Address	(Blank)	1

Columns 1 2 3 4 5 6 7 79 80

ADDRESS CARD FORMAT

2) Data Card. Following the address card, each card except the last card will have the following format:

a Eight (10 octal) consecutive 6-digit 1219 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. As this is an absolute dump, all of these modification digits will be zero.

c One six-digit cumulative checksum in columns 67 through 72. By cumulative, we mean that the checksum from the previous card(s) is added to the total of the present card before being punched.

d One four-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 Data Card. The last card in any specific memory dump will have almost exactly the same format as the normal data cards. The difference is that the last card will contain anywhere from no memory words up to 7 of them 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) there will be punched a word of 6 zeros with a corresponding modification digit of 7.

c) 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 input parameters the address to begin the memory load in AU, but only if an address card does not precede the data cards.

d) Operating Procedure. To load 80-column cards in relocatable format:

STEP 1. Put cards in 1004 Card Processor's reader input hopper. The 1004 must be on and ready.

STEP 2. Press the CLEAR, START, FEED, and RUN buttons on the 1004 console in that order. This will position the first card at the 1004 read station ready for processing. If the first card is an address card, no parameters are required as input to DATCD. If no address card is present, AU must contain the base load address when DATCD is called or used.

Under program control:

STEP 3. The calling program must perform an RJP to DATCD base address, or an IRJP to an address containing DATCD base address.

Under manual operation:

STEP 3. Set P = DATCD base address + 2, then start.

To dump absolute data on 80-column cards in relocatable format:

STEP 1. Put the 1004 Card Processor, including the punch unit, in a ready condition.

STEP 2. Make certain the input hopper of the punch unit contains enough blank cards.

STEP 3. When DATCD is called or used, AU and AL must contain the first and last address of the memory area to be dumped on cards. Either limit may be in either register.

Under program control:

STEP 4. The calling program must perform an RJP to DATCD base address + 3, or an IRJP to an address containing DATCD base address + 3.

Under manual operation:

STEP 4. Set P = DATCD base address + 5, then start.

e) Alarms. The DATCD card load routine (CLOD) will always stop before loading any data words from a card if the load address is below 01000. By pressing start, data from the card will be loaded at the requested address. However, DATCD will stop before loading data words from the next card if the current load address is still below 01000. This cycle will continue until the load address exceeds 01000.

Unless the ignore checksum is being employed, DATCD's load routine will stop if it detects an error in the cumulative checksum. 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 will result in a checksum error.

9. UNIVAC 1219 Modified Paper Tape Utility Package.

a) General Information. UPAKM is a modified version of UPAK I which provides the same functions as UPAK I, but will facilitate 6-digit addressing when desired. UPAKM contains seven subroutines on punched paper tape in relocatable bioctal format. 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

UPAKM 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. The paper tape load and dump routines are operable under program control or manually from the computer. However, the inspect and change and store constant in memory functions are operable from the computer control panel only.

UPAKM occupies 1060 octal memory locations. Entrance addresses to the several subroutines are assigned relative to the UPAKM base address as will be subsequently shown. Increments to the base are in octal.

Address	Routine Entrance
Base + 0	ICH entrance
Base + 1	STC entrance
Base + 2	Program entrance paper tape load
Base + 4	Manual entrance paper tape load
Base + 5	Program entrance dump typewriter code
Base + 7	Manual entrance dump typewriter code
Base + 10	Program entrance dump bioctal code
Base + 12	Manual entrance dump bioctal code

UPAKM subroutines use the currently active B-register, but they store and restore its original value. Routines operating under program control also store and restore the user's special register setting.

b) Operations and Program Formats.

1) Load Absolute Typewriter Code. A carriage return, followed by either (1) an 8 or an 88, or (2) a 9 or a 99, activates the load routine. An 8 or an 88 following the carriage return indicates that the tape is prepared with 5-digit addressing (i.e., all addresses are below 100000), while a 9 or a 99 indicates 6-digit addressing. A single 8 or a single 9 indicates that the tape has no checksum (user prepared tapes). An 88 or a 99 indicates that the tape is terminated with a 6-digit checksum (TRIM outputs 2 and 3 and UPAK or UPAKM typewriter code dumps). A carriage return must follow the 8/88 or 9/99 code.

Once the load routine has been activated, it accumulates the first 5 or 6 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 or 12 octal digits are ignored until another carriage return is found. If less than 11 or 12 octal digits are accumulated, the instruction will not be loaded. Examples of tape formats are shown on the following page.

Each tape to be loaded must terminate with a carriage return and a double period (..), which will terminate the load and initiate 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 Skip Key 1 is set, the load subroutine will perform a checksum verification without loading the tape into memory. Examples of tape format:

5-digit Addresses No Checksum	5-digit Addresses With Checksum	6-digit Addresses No Checksum	6-digit Addresses With Checksum
8	88	9	99
XXXXX XX XXXX	XXXXX XX XXXX	XXXXXX XX XXXX	XXXXXX XX XXXX
XXXXX XX XXXX	XXXXX XX XXXX	XXXXXX XX XXXX	XXXXXX XX XXXX
. . .	. . .	. . .	. . .
. . .	. . .	. . .	. . .
XXXXX XX XXXX	XX XXXX (Checksum)	XXXXXX XX XXXX	XX XXXX (Checksum)
..	..	. .	. .

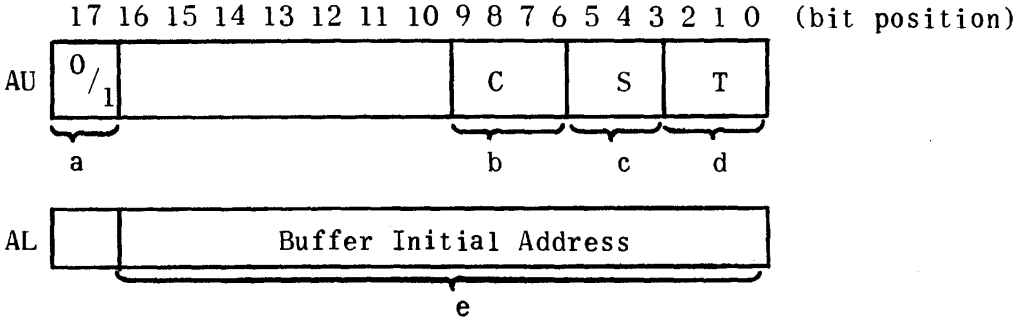
2) Dump Absolute Typewriter Code. The dump is initiated manually at the computer or under program control for output on punched paper tape. The 99 format (6-digit addressing with checksum at the end) is the only one dumped. The output tape includes both the address and the contents of the memory locations being dumped.

10. Printer Line Image on Tape and Tape-to-Printer Module.

a) Program Description. POTPOP is a module of UPAK III for the UNIVAC 1219 Computer when operating with Type 1240 Magnetic Tape Units and a UNIVAC 1004 Card Processor. One routine (POT) of the module writes a preformatted, excess-three coded, 132-decimal character buffer on magnetic tape for subsequent listing on the 1004 printer. POT will also write the tape mark which identifies the end of the listing for the listing routine.

The other routine (POP) will list a POT-generated magnetic tape on the 1004 printer. POP will continue to read and print line-records from the designated magnetic tape until it detects a tape mark.

b) Input Parameters. The printer line image on magnetic tape section of POTPOP (POT) requires as input parameters the following:



where:

a = Indicates whether to write line buffer or tape termination flag (tape mark).

0 = write the buffer indicated by AL address

1 = write a tape mark

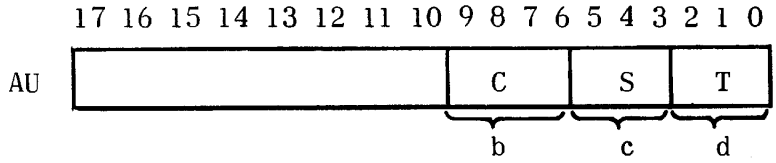
b = C - I/O Channel ($0 - 17_8$). The channel to which the magnetic tape unit is connected. POTPOP writes and reads only in single channel, high density, odd parity, and octal format. The computer must be set for the single channel mode.

c = S - Magnetic tape synchronizer (cabinet) number (1-4).

d = T - Magnetic tape transport number (1-4).

e = Initial address of a 54_8 word excess-three coded printer line buffer to be written on tape.

The tape-to-printer section of POTPOP (POP) requires as input parameters the following:



where b, c, and d correspond to the b, c, and d parameters of POT described above.

c) Operating Procedures. To write a printer line image or tape mark on magnetic tape:

Under program control:

STEP 1. Enter AU and AL with the desired parameters.

STEP 2. Do an RJP to POTPOP base address or an IRJP to an address containing the POTPOP base address.

Under manual operation:

STEP 1. Enter AU and AL with the desired parameters.

STEP 2. Set P = POTPOP base address + 2.

STEP 3. Start.

To list a POT-generated magnetic tape on the 1004 printer:

Under program control:

STEP 1. Enter AU with the desired parameter.

STEP 2. RJP to POTPOP base address + 3 or IRJP to an address containing POTPOP base address + 3.

Under manual operation:

STEP 1. Enter AU with the desired parameter.

STEP 2. Set P = POTPOP base address + 5.

STEP 3. Start.

d) Alarms. POT or POP will try one recovery each upon detecting an error in writing or reading a printer line record on magnetic tape. If an error still occurs on the second attempt, POT or POP will ignore it and proceed normally.

An improper condition status word from the tape unit will cause 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.

a) Program Description. The JOSH magnetic tape handler provides users with the ability to implement all of the hardware capabilities of UNIVAC 1240 Magnetic Tape Systems while on-line with a UNIVAC 1219 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.

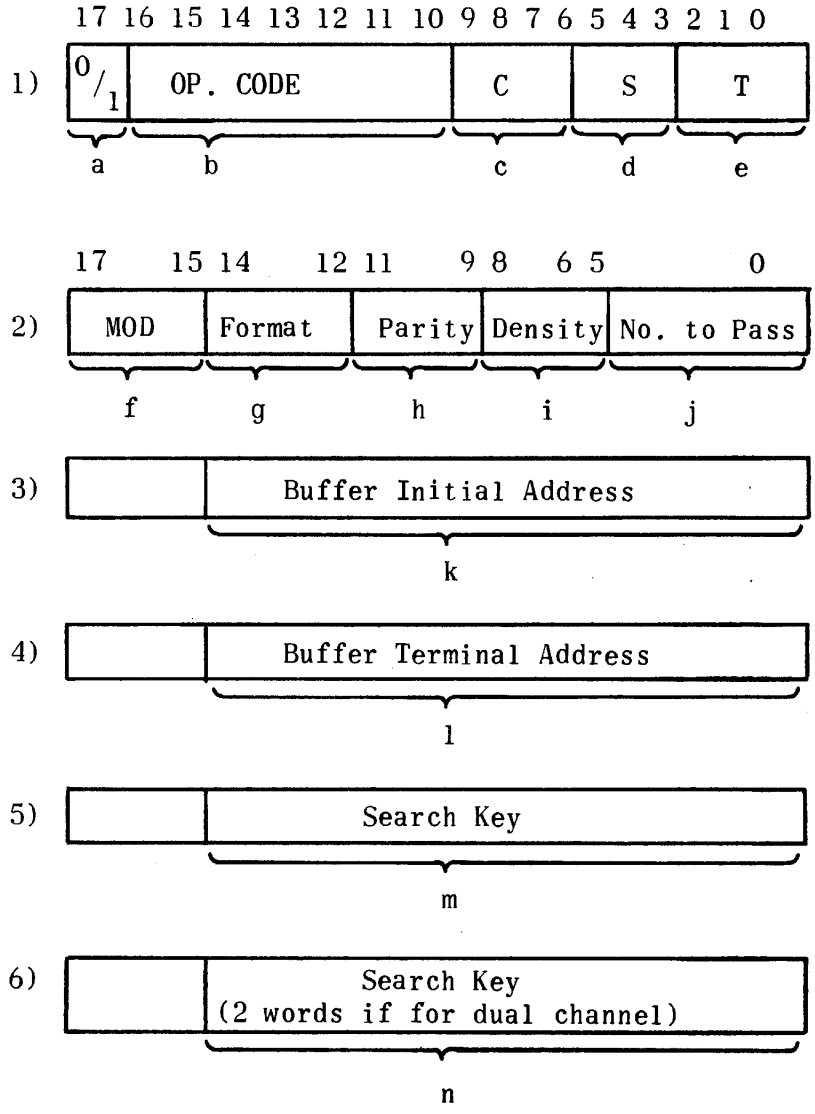
2) JOSH base address + 2 - Job completion status check. The tape handler interrupt routine stores a coded status indicator in the first word of the tape function request packet. The user may perform his own completion check by analyzing the contents of this word. More simply, he can enter JOSH at base + 2 and JOSH will perform the check for him and indicate completion status via a good or bad exit.

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 $\neq$ 0; if the function is complete (hardware interrupt status in), AL = 0.

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. JOSH occupies approximately 640_8 memory locations.

b) Input Parameters. From one to six parameter entries are manually entered into the A register or are program stored in memory to govern the operation of the tape handler. They are as follows:



where:

a = Mode of operation

Ø = dual channel
1 = single channel

b = Operation code - Ø - 33_8 to select particular tape function

Code

Ø	Read
1	Read Selective
2	Read - Ignore Error Halt

Code

3	Read - Retain Control (Tape Handler holds until function is complete)
4	Search - Type I
5	Search - Type II - Identical
6	Search File - Type I
7	Search File - Type II - Identical
10	Write
11	Write - Extended Inter-Record Gap
12	Write - Ignore Error Halt
13	Write - Extended Inter-Record Gap - Ignore Error Halt
14	Write Tape Mark
15	Write Tape Mark - Extended Inter-Record Gap
16	Pass "N" Files Forward
17	Pass "N" Records Back
20	Pass "N" Records Forward
21	Backspace
22	Backspace - Read
23	Pass "N" Files Back
24	Back Search - Type I
25	Back Search - Type II - Identical
26	Back Search File Type I
27	Back Search File Type II - Identical
30	Rewind
31	Rewind - Clear Write Enable
32	Rewind - Read
33	Rewind - Read - Clear Write Enable

c = I/O Channel 0 - 17_8

d = Magnetic tape synchronizer (cabinet) number (1-4)

e = Magnetic tape transport number (1-4)

f = Modulus selection - 3, 4, 5 or 6

g = Recording format

Ø = bioctal (non-redundant)
1 = octal (redundant)

h = Parity selection

Ø = even
1 = odd

i = Density selection

Ø = low (220 frames per inch)
1 = high (556 frames per inch)

j = Number of records or files to space forward or backward. Maximum of 77; if more are required it is necessary to re-execute the command to JOSH.

k = Buffer initial address for read, write, and search operations.

l = Buffer terminal address for read, write, and search operations.

m = Search key for single channel search operations or first word of search key for dual channel operation.

n = Second word of search key for dual channel operations.

c) Operating Procedure.

1) Operation Under Program Control.

STEP 1. Enter the active B-register with the first address of the input parameter packet. This packet may consist of 2-7 words depending on the operation to be performed. The first word of the packet must be a blank word where the coded status will be stored. The remaining words contain the input parameters.

STEP 2. Execute a return jump or indirect return jump to JOSH base address.

STEP 3. The user may sample a busy-bit by executing a return or indirect return jump to JOSH base address + 4. Upon return, one of the following indications will be in AL:

(AL) = Ø indicates the job is done
(AL) ≠ Ø indicates the job is still in progress.

STEP 4. The user may check the status of the tape functions by one of two methods.

a Execute a return or indirect return jump to JOSH base address + 2.

When the tape function is complete, control is returned to the user by the good or bad exit.

1 Bad exit - is via the normal exit with the following contents in the A register.

(AU) = tape unit status word
(AL) = one of the following codes

1 - tape mark
2 - improper condition
3 - unrecoverable tape error
4 - parameter error
5 - low tape
6 - end of tape

2 Good exit - (AU) and (AL) = Ø

b The user may check for the above-mentioned codes in the first word of the packet. (Ø word means good.)

2) Manual Operation.

STEP 1. Set P to JOSH base address + 10.

STEP 2. Set AU = parameter 1
AL = parameter 2

STEP 3. Start - when computer stops
Set AU - parameter 3
AL - parameter 4

STEP 4. Start - when the computer stops
Set AU - parameter 5
AL - parameter 6

STEP 5. Start again - tape function will be initiated. (For parameters not applicable to the requested operation - leave (AU) and (AL) = Ø and press start.)

3) Special Considerations.

a Search constants for a backward search must be reversed characterwise.

b Whenever the dual channel operation is selected the buffer limits must begin with an even address and end with an odd address or vice versa.

c Address 00141 shall be reserved for use by JOSH as an indirect interrupt address.

d The contents of AU, AL, and active B will not be restored.

e 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.

e. UNIVAC 1219 TRIM Corrector.

1. General Information. The UNIVAC 1219 TRIM Corrector is a companion to the UNIVAC TRIM I and II Assemblers using paper tape source program input. It has been designed to operate on a UNIVAC 1219 Computer with paper tape input/output and 8K core memory. The correction process, constituting a separate computer run, provides a convenient method for correcting source programs for subsequent assembly.*

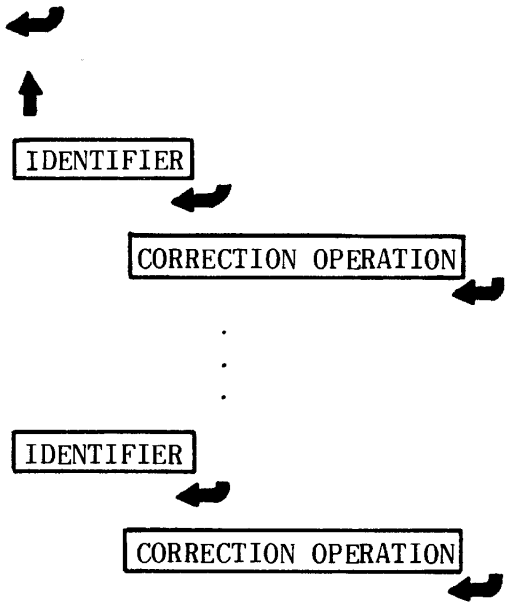
* A source program means any program prepared by the programer in TRIM format or output no. 2 of the TRIM Assembler (a side-by-side output of the assembled object program and the corresponding source program). The Corrector processes only the source program portion of an output no. 2.

Input to the Corrector consists of one or more correction tapes and the source program tape(s) to be corrected. The programer may make three types of corrections to his source program; 1) insertions, 2) replacements, and 3) deletions. Output from the Corrector is a single punched paper tape consisting of the corrected source program.

2. Input Formats. Two versions of the Corrector are available; one for standard Flexowriter prepared input, the other for fieldata coded input. In either format, a correction tape should not have a header of any type. A flex-coded correction tape begins with a carriage return and a shift to upper case, and terminates with a carriage return, shift to lower case, and two periods. A fieldata coded correction tape begins with a carriage return and line feed and terminates with a carriage return, line feed, and two periods.

To establish correspondence between corrections and the operation items of the uncorrected source program, the programer places an octal identifier (determined from a previous TRIM Assembler output no. 2) before each correction operation.

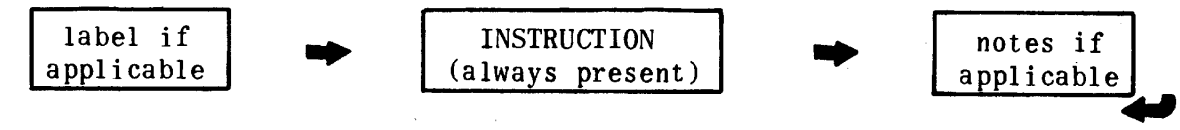
Coded corrections thus consist of alternate identifiers and the corresponding corrections as shown in the diagram that follows where ↵ means carriage return, ↑ means shift up, ↓ means shift down, and → means tab (Flex code 51 or FD code 76).



..

The identifier consists of an integer and fraction, separated by a point separator. A maximum of five octal integer digits are permitted. The fraction portion is a straight binary magnitude (from .001 through .777).

The correction operation format is shown below. A correction operation must not exceed 90 characters in length including control codes.



a) The Insert Correction. An insert correction requires the identifier of its preceding operation from the source program, the point separator, and additional octal insert digit(s). For example, an operation item to be inserted between identifiers 12 and 13 of the source program has the insertion identifier of 12·1.

Example:

```
140•31↵
WALTH ➡ ENTALK•0 ➡ CLEAR AL
```

b) The Delete Correction. A delete correction lists the identifier of the operation (or first operation of a group of sequential operations) to be deleted. 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 deleted, the word DELETE is followed by a point separator and the number (in octal) of operations to be deleted and a carriage return.

Example 1:

```
127•0↵
    ➡ DELETE↵
```

will delete item 127 from the source program.

Example 2:

```
127•0↵
    ➡ DELETE•5↵
```

will delete items 127 through 133 from the source program.

c) The Replace Correction. A replace correction lists the identifier of the source program operation to be replaced, a carriage return, and the replacing operation terminated by a carriage return.

Example:

```
105•0↵
    ➡ ENTBK•35    ➡ INITIALIZE INDEX
```

This set of operations replaces whatever operation previously existed for identifier 105 of the source program.

Additions are made to the end of a source program by means of either the insert or replace correction. For example, if the last identifier of a source program is 320, then either of the following examples will add the sample clear storages subroutine:

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. Performance. The TRIM Corrector occupies memory locations 01000 through 02505 plus buffer areas. The Corrector accepts a maximum of 212D corrections for any single correction run. As many correction tapes as desired are given to the Corrector, the only restrictions being that the total corrections do not exceed 212 and that each separate tape begin with the proper combination of control characters as prescribed by the input format.

At the start of a correction run, the correction tapes are read into memory, and the items are sorted by identifier in ascending numerical order. A control panel Skip Key selection indicates when the last correction tape has been read.

The Corrector next reads the source program tapes, one operation item at a time. An internal identifier is assigned to each such item, starting with zero and increasing by one for each subsequent item.

This identifier is compared with that of the next correction to determine processing action.

If the correction identifier is smaller, the correction item is punched on the corrected output tape.

If the correction and source program identifiers are equal, the correction item is inspected. When it contains a DELETE operator, the corresponding source item (or items) is bypassed (not punched on the corrected output tape). If the operator is not DELETE, the source program item is ignored and the correction item is punched on the corrected output tape, thereby replacing the original item.

If the correction identifier is greater, the source program item is punched without alteration on the corrected output tape.

The process just described continues until a Skip Key selection indicates the last source program tape has been processed. At this time the corrected output tape becomes finalized, and the computer stops.

If several source program tapes are involved, they must be read in the proper order. No restrictions are placed on the number of source program operation items; however, no single item should exceed 90 characters in length including control codes.

4. Operating Instructions. The TRIM Corrector may be loaded via the 1219 Utility Package or any acceptable bioctal load bootstrap routine or automatic recovery bootstrap. When the Corrector has been loaded into memory:

STEP 1. Master clear.

STEP 2. Set P = 01000.

STEP 3. Set Skip Keys 1 and 2 (set Skip Key 3 also if the source program to be corrected is a TRIM Assembler output no. 2).

STEP 4. Mount a correction tape in the reader.

STEP 5. Start. The computer will stop after each correction tape has been read. Repeat Steps 4 and 5 until all correction tapes have been read.

STEP 6. Release Skip Key 1.

STEP 7. Mount a source program tape in the reader.

STEP 8. Start. As the source program tape is read, the corrected output is produced. The computer will stop after each source program tape has been read. Repeat Steps 7 and 8 until all source program tapes have been read.

STEP 9. Release Skip Key 2 and (Skip Key 3 if it was previously set).

STEP 10. Start. The Corrector will finalize the output tape and the computer will stop. To process another correction run, begin from Step 3.

5. Error Detection. The TRIM Corrector recognizes two error conditions.

a) Computer Stops With (AL) = 000007. This error indicates that no correction tape has been loaded. Restart the correction run from Step 1 of the operating instructions.

b) Computer Stops With (AL) = 007777. This error indicates that more than 212D correction operations have been detected. The programer may prefer to terminate this correction run and replan his correction strategy. If he wishes to proceed on the basis of the first 212 corrections only, he should remove the correction tape from the reader and start from Step 6 of the operating instructions.

6. Corrected Tape Checksum Verification. The TRIM Corrector computes a cumulative checksum as it produces the corrected output tape. This checksum is punched as the last data on the tape preceded by double periods and a d. The user may checkread his output tape as follows:

STEP 1. Master clear.

STEP 2. Set P = 01000.

STEP 3. Set Skip Key 4.

STEP 4. Mount the corrected output tape in the reader.

STEP 5. Start. The computer will stop with P = 01000 and (A) = zero following an accurate checkread. If the computed checksum and the tape checksum do not agree, the computer will stop with (AU) = computed checksum and (AL) = tape checksum.

7. Expansion Capabilities. Installations possessing larger than an 8K memory computer may increase the maximum number of corrections allowed by adding 200_8 per additional available memory bank to the contents of the memory location RC277 (address 01063 in the Flex version and 01065 in the Fieldata version).

f. UPAKM - UNIVAC 1219 Modified Paper Tape Utility Package.

1. General Information. UPAKM is a modified version of UPAK I which provides the same functions as UPAK I, but will facilitate 6-digit addressing when desired. UPAKM contains seven subroutines on punched paper tape in relocatable bioctal format. The seven subroutines are:

a) Load absolute typewriter code

b) Load absolute bioctal code

c) Load relocatable bioctal code

d) Dump absolute typewriter code

e) Dump absolute bioctal code

f) Inspect and change

g) Store constant in memory

UPAKM 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. The paper tape load and dump routines are operable under program control or manually from the computer. However, the inspect and change and store-constant-in-memory functions are operable from the computer control panel only.

UPAKM occupies 1060 octal memory locations exclusive of addresses 00540 through 00777 which are used by the UPAKM loader. Entrance addresses to the several subroutines are assigned relative to the UPAKM base address as well as subsequently shown. Increments to the base are in octal.

Address	Routine Entrance
Base + 0	Program entrance paper tape load
Base + 2	Program entrance dump typewriter code
Base + 4	Program entrance dump bioctal code
Base + 6	Manual entrance paper tape load
Base + 7	Manual entrance dump typewriter code
Base + 10	Manual entrance dump bioctal code
Base + 11	Manual entrance inspect and change
Base + 12	Manual entrance store constant in memory

UPAKM subroutines use the currently active B-register, but they store and restore its original value. Routines operating under program control also store and restore the user's Special Register setting.

2. Operations and Program Formats. A carriage return, followed by either (1) an 8 or an 88, or (2) a 9 or a 99, activates the load absolute typewriter code load routine. An 8 or an 88 following the carriage return indicates that the tape is prepared with 5-digit addressing (i.e., all addresses are below 100000),

while a 9 or a 99 indicates 6-digit addressing. A single 8 or a single 9 indicates that the tape has no checksum (user prepared tapes). An 88 or a 99 indicates that the tape is terminated with a 6-digit checksum (TRIM outputs a 2 and 3 and UPAK or UPAKM typewriter code dumps). A carriage return must follow the 8/88 or 9/99 code.

Once the load routine has been activated, it accumulates the first 5 or 6 digits following each carriage return and assembles them at the accumulated address. All characters following first 11 or 12 octal digits are ignored until another carriage return is found. If less than 11 or 12 octal 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 will terminate the load and initiate 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.

g. TALK. TALK, meaning Take A LooK, is a debugging technique which aids substantially in debugging complex real-time programing systems by interrupting the user's program at desired points to extract previously specified data. The extracted data is later edited, listing the associated data with its high-level source language identification. The debugging process is:

The user's program is written and compiled normally with no additional instructions generated. An additional compiler output is also obtained which specifies and allocates the data definitions.

A separate translator program inputs statements that specify the data to be extracted, the points of extraction, and editing desired on the final output. This information is cross-referenced with the data definitions from the compiler to produce information in a tabular form for the extractor and editor programs.

An extractor program utilizes the tabled information provided by the translator to initialize the user's program by inserting jumps at the extraction points. As these extraction points are executed during program operation, the specified data units are extracted and written on external storage, or stored in core, for later editing.

NAME:_____

2.3-5. STUDY QUESTIONS:

a. List the seven subroutines of UPAK 1.

b. What is the difference between an absolute typewriter load and an absolute bioctal load? Show the two formats.

c. What is the difference between the absolute bioctal load and a relocatable bioctal load? How does the computer detect the difference between the two?

d. What is the use of the modification code in relocatable bioctal load?

e. What is the difference between an absolute typewriter dump and an absolute bioctal dump? What is the meaning of the word dump?

f. List the error detections performed by UPAK I. What is meant by the word "checkout?"

g. Is there any correlation between the Fieldata Code and the Flexowriter Code? If any, what?

h. List the types of corrections a programer may make to his source program when using the TRIM Corrector.

i. What is the input format for using the TRIM Corrector?

NAME:_____

j. How many corrections can you make on any correction run?

k. What is meant by identifier when using the corrector?

l. What is the prupose of the TRIM Library Builder?

m. What is a library director?

n. What is the purpose of the LIBLST operator?

o. What is the minimum equipment configuration for using the TRIM Library Builder?

p. What is the purpose of the Data Extraction System?

q. What is the purpose of the CART service routine?

r. What is the purpose of the Trace Diagnostic Routine?

s. Can TRACK be used on real-time problems? Why or why not?

NAME:_____

t. What is the purpose of the "JOSH" module of UPAK III?

u. What equipment configuration is required for UPAK III?

SECTION 3 - PERIPHERAL EQUIPMENT

3.1. PERIPHERAL EQUIPMENT

3.1-1. OBJECTIVES

a. To give the student additional information on the I/O Console.

b. To give the student additional information on the 1240 Magnetic Tape system.

c. To give the student additional information on the UNIVAC 1004 Card Processor.

3.1-2. INTRODUCTION

It is important that the programer be familiar with the operation of peripheral equipment that is connected to the 1219 computer system and of the formats required to make these peripheral units function as desired.

3.1-3. REFERENCES

PX 3640 - Magnetic Tape Unit, Type 1240, Section I, General Information.

3.1-4. INFORMATION

a. Input/Output Console.

1. General Information. The Input/Output Console is a peripheral device of the 1219 Computer. See table 3.1-1. As such, it communicates with the computer through input/output channels. The Input/Output Console function is to load programs and data into the computer and to monitor data from the computer. The console consists of keyboard, printer, perforated tape reader and punch, the logic circuitry necessary to control the above units, control panel, and power supply assembly. Normally, the console operates with only one input or output device at a time. However, during an output operation both printer and punch can be selected simultaneously. During an input operation either printer or punch, or both, can be selected to monitor data sent to the computer.

2. Operation. The Input/Output Console has two basic modes of operation as listed below:

a) On-line; for normal operation.
b) Off-line; for maintenance purposes and tape preparation.

In the on-line mode, two types of data transfers are involved: input data transfers, and output data transfers. An input data transfer refers to the transmission of information from the console to the computer. An output data transfer refers to the transmission of information from the computer to the console. In the off-line mode, these transfers are within the console; no data transmission between the

TABLE 3.1-1. GENERAL INFORMATION ON INPUT/OUTPUT CONSOLE, TYPE 1232A

External Function Codes								I/O Console Response
2^6	2^5	2^4	2^3	2^2	2^1	2^0	Octal	
0	0	0	0	0	1	1	03	Turn on printer, turn off punch, make Output Data Request.
0	0	0	0	1	0	1	05	Turn on punch, turn off printer, make Output Data Request.
0	0	0	0	1	1	1	07	Turn on printer and punch, make Output Data Request.
0	0	0	0	0	0	1	01	Turn off printer and punch.
0	0	1	1	0	0	0	30	Turn on keyboard. (Make Input Data Request after key is depressed).
1	1	0	1	0	0	0	150	Turn on reader and Start Read. Stop tape motion and make Input Data Request after frame is over photocells.
0	0	0	1	0	0	0	10	Turn off reader and keyboard.
0	0	0	1	0	0	1	11	Turn off all devices.

1. Input E.F. words and Output E.F. words are not interacting, i.e., any combination of legal codes, such as 33, 157, 15, etc., causes the same response as if the Input and Output codes were sent using two separate E.F. transmissions.

2. Any E.F. ACK signal from the computer causes the E.F. REQUEST to be dropped for a 70 millisecond period.

3. Octal 70 E.F. code (turn on keyboard and reader) is illegal.

4. When turning on an output device there is a 500 millisecond delay between when the E.F. code is received and when the first Output Data Request is made. (Motor turn-on delay)

5. Turning on the keyboard automatically turns off the reader, and turning on the reader automatically turns off the keyboard.

computer and console occurs. The operation of the console is explained by starting from the off condition. The off condition is defined as the condition when ac power is off and console is in the off-line mode.

a) On-Line Operations. In the on-line mode (ON-LINE/OFF-LINE switch is in the ON-LINE position), the console operates as an input/output device of the computer and normally the computer initiates and terminates an operation; however, the controls on the control panel also can be used.

1) Input/Output Options. The computer can select either printer or perforated tape punch, or both, for an output operation, and either keyboard or perforated tape reader, for an input operation. The console operator can obtain printed copy of the input data by pressing the COPY indicator-switch and the PRINT indicator-switch. Punched copy may be obtained by pressing the COPY indicator-switch and the PUNCH indicator-switch.

2) Input Operation. The computer selects an input operation by sending to the console an appropriate external function word. The console places data on the input data lines and sets the input request line, specifying that data is ready for the computer. The computer samples the data and sets the input acknowledge line, specifying that it is ready for more data. The input acknowledge clears the input register and other areas of console control, and re-initiates the cycle. The sequence is repeated until the input operation is completed.

3) Output Operation. The computer selects an output operation by sending to the console an appropriate external function word. The console control circuitry sets the output request line, specifying readiness to accept data. The computer places data on the output data lines and sets the output acknowledge line, specifying that the data is ready for transmission. The console detects the output acknowledge, samples the output data lines, and clears the output request line. Then, the control circuitry causes either the printer or punch, or both, to operate and record the data received from the computer. After the printing or punching cycle the output request line is set again. This sequence is repeated until the output operation is completed.

4) Interrupt Operation. The console may interrupt the computer from the keyboard. To accomplish this, the KEYBOARD and the INTERRUPT indicator-switches are pressed on the control panel. When a keyboard key is pressed, a corresponding data code is set on the input data lines and an interrupt signal is sent to the computer. The computer samples the data and sets the input acknowledge line. Console control circuitry detects the input acknowledge and clears the interrupt and input data lines.

b) Off-Line Operations. The console is independent of the computer in the off-line mode (ON-LINE/OFF-LINE switch is in the OFF-LINE position). Off-line operations are similar to on-line operations except that information and control signals are not transmitted between the computer and console, and the input and output devices are selected manually on the control panel. The ON-LINE/OFF-LINE switch on the control panel provides the gating which causes each output request to generate a synthetic input acknowledge, and each input request to generate a synthetic output acknowledge.

1) Off-Line Tape Preparation (Punch/Keyboard Operation).

STEP 1. Set ON-LINE toggle switch to OFF-LINE position.

STEP 2. Press MASTER CLEAR pushbutton-switch.

STEP 3. Press KEYBOARD indicator-switch.

STEP 4. Press PUNCH indicator-switch.

NOTE

COPY is automatically selected in off-line mode.

STEP 5. Run out desired amount of blank tape by pressing TAPE FEED indicator-switch.

STEP 6. Enter desired information via the keyboard.

STEP 7. Repeat Step 5 if necessary.

STEP 8. Press MASTER CLEAR pushbutton-switch when operation is completed.

2) Off-Line Tape Print-Out (Printer/Reader Operation).

STEP 1. Set ON-LINE/OFF-LINE toggle switch to OFF-LINE position.

STEP 2. Press MASTER CLEAR pushbutton-switch.

STEP 3. Press PRINT indicator-switch.

STEP 4. Press READ indicator-switch.

STEP 5. Load tape in reader.

STEP 6. Press START READ indicator-switch.

STEP 7. Press READ/READ-ONE switch to READ position.

STEP 8. Allow print-out to continue until entire tape or desired portion of the tape has been read.

STEP 9. Press MASTER CLEAR pushbutton-switch when operation is completed.

3) Off-Line Tape Reproduction (Punch/Reader Operation).

STEP 1. Set ON-LINE/OFF-LINE toggle switch to OFF-LINE position.

STEP 2. Press MASTER CLEAR pushbutton-switch.

STEP 3. Press PUNCH indicator-switch.

STEP 4. Press READ indicator-switch.

STEP 5. Load tape in reader.

STEP 6. Press START READ indicator-switch.

STEP 7. Press READ/READ-ONE switch to READ position.

STEP 8. Allow punch-out to continue until entire tape or desired portion of the tape has been reproduced.

STEP 9. Press MASTER CLEAR pushbutton-switch when operation is completed.

3. Console Input/Output Devices. The Input/Output Console utilizes the printer and paper tape punch as output devices, and the keyboard and paper tape reader as input devices. See Figure 3.1-1.

a) Printer. The printer uses 6-bit fieldata code for character representation. A total of 64 different characters and functions may be represented by this code. The printer is capable of printing 10 characters per second, 10 characters per inch horizontally, 72 characters per line, and six lines per inch vertically.

b) Paper Tape Punch. The paper tape punch utilizes 5, 6, 7, or 8-level code for character (frame) representation on 11/16, 7/8 or one inch tape. A maximum of 256 different code combinations are possible using this method of coding. The punch is capable of perforating 10 frames per inch at a tape speed of 11 inches per second.

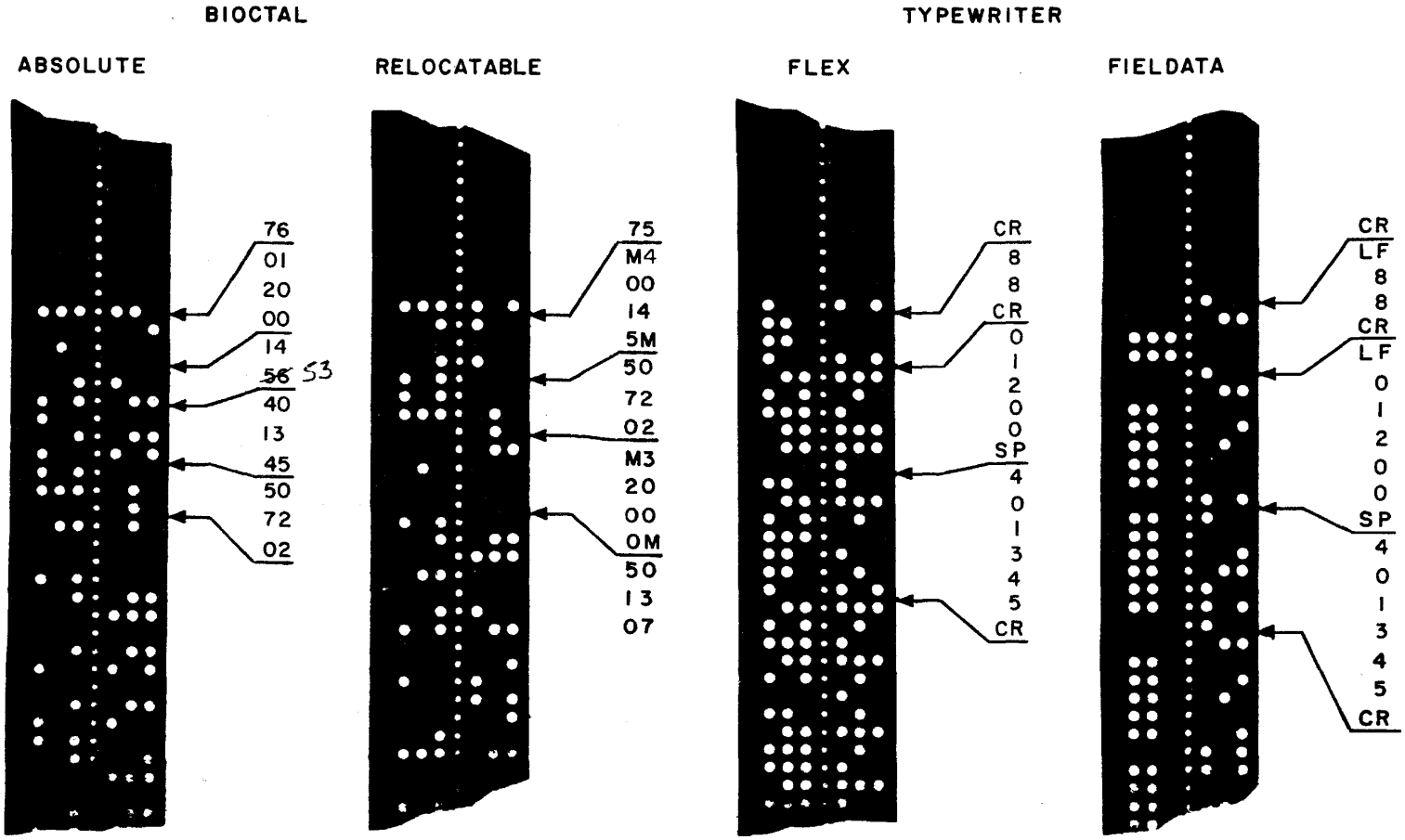
c) Paper Tape Reader. The paper tape reader is capable of reading 5, 6, 7, or 8-level code characters (frames) from 11/16, 7/8 or one inch tape, at a rate of 300 frames per second.

d) Keyboard. The keyboard generates data codes as specified in the fieldata code when the correspondingly labeled keys are pressed. Data entered into the keyboard can be simultaneously printed by the printer if the PRINT and COPY indicator-switches are selected on the control panel.

e) Use of Paper Tape Bin. The paper tape bin is used to prevent the paper tape from piling up on the floor and, for this reason should be slightly pulled out (approximately six inches) from the cabinet, during the paper tape punch and reader operations.

b. UNIVAC 1240 Magnetic Tape Unit.

1. General Information. The UNIVAC 1240 Magnetic Tape Unit is a large capacity, medium speed, auxiliary storage system. It may be operated on-line under complete computer program control as an input/output storage device or with a High-Speed Printer for off-line printing of tape recorded information. A flexible format allows recording and reading in four moduli (18, 24, 30 or 36-bit computer words)



2 OCTAL CHARACTERS PER FRAME PRECEDED BY A CODE ID AND, 15 BIT STARTING AND ENDING ADDRESSES.

ID CODE FOLLOWED BY 2 OCTAL CHARACTERS PER FRAME PRECEDED BY A MODIFICATION CODE FOR EACH INSTRUCTION.

ONE 6 BIT TYPEWRITER CODE PER FRAME STARTING WITH CARRIAGE RETURN 8 OR 88 CARRIAGE RETURN FOLLOWED BY CR 5 CHAR. OF ADDRESS AND 6 CHAR. OF INSTRUCTION.

Figure 3.1-1. Paper Tape Formats

and two densities, and provides recording and reading of magnetic tapes compatible in all respects with the IBM 727, IBM 729 II and IBM 729 IV system tapes. Either even or odd frame parity may be utilized and for added reliability, the redundant octal format is provided. A read after write feature is provided to check each frame for parity immediately after recording. Longitudinal parity recording and checking is automatic.

Records of data may be of variable lengths and are separated by 3/4 inch inter-record gaps (IRG) unless otherwise extended by suitable programing. Records may be lengthened if suitable inter-record gaps were provided in previous recordings.

The UNIVAC 1240 Magnetic Tape Unit is compatible with all UNIVAC Military Computers and may be supplied with an 18, 24, 30 or 36-bit parallel input/output interface with either of two sets of logic levels.

The Minimum Magnetic Tape Unit consists of one Magnetic Tape Control (MTC), one Tape Transport Control (TTC) in which are contained one Local Transport Control (LTC) for each of two or four Magnetic Tape Transports (MTT). The maximum configuration consists of one MTC, four TTC's and sixteen MTT's. Two or four MTT's are contained in one cabinet with one TTC. One MTC is used for any configuration on any one computer channel since its function is to communicate with the computer and with all TTC's. See Figure 3.1-2.

2. Electronics. The UNIVAC 1240 electronic control is a completely solid-state machine. The components are mounted on small, plug-in, printed-circuit cards which are assembled in a modular form for compactness. This building block technique, and the circuit cards utilized in the 1240, have a field-proven record of reliability in UNIVAC computers and peripheral equipment.

3. Maintenance. The solid-state, plug-in modules, in conjunction with the manual controls and register and logic indicators, make maintenance of the UNIVAC 1240 fast and easy, keeping repair time to a minimum. Manual controls permit off-line maintenance of one tape unit during on-line operation of other units in the system. The tape transports and all electronics are easily accessible from the front of the unit.

4. Tape Unit Specifications.

a) Tape.

Width - 1/2 inch
Type - "A" wound (oxide coating on inside of tape) - Mylar base
Length - 2400 feet
Reels - 10-1/2 inch, IBM compatible hub type
Tape Markers - Load Point and End of Tape reflection markers

b) Tape Speed.

Read/Write - 112.5 inches per second forward
Rewind - 225 inches per second
Start/Stop time - 1.5 to 2.5 milliseconds

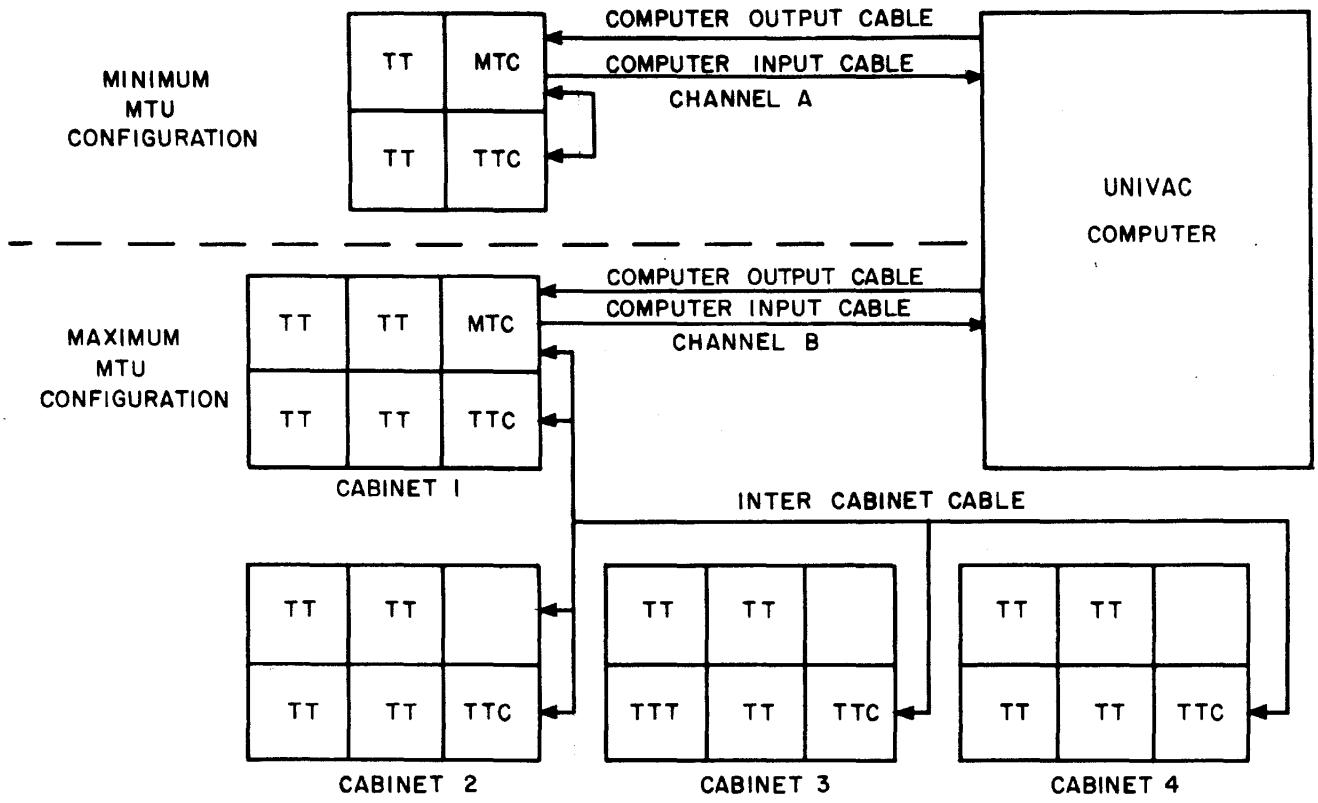


Figure 3.1-2. Magnetic Tape Unit and Computer Configurations

c) Recording Techniques.

Non-Return to Zero (change-on-one)
Seven tracks - six data bits, one parity bit
Density - 200 or 555.5 frames per inch (Program Controlled)
Inter-Record Gap - 3/4 inch
Extended-Inter-Record Gap - 3-1/2 inches
Record Length - Variable
Parity - Lateral and longitudinal
Tape Mark - End of File (Program Controlled)
Write Lockout - IBM Type; upper reel

d) Transfer Rate.

62,500 characters per second - High Density
22,500 characters per second - Low Density

e) Format (Program Controlled).

Moduli - 3, 4, 5 or 6
Frame Character - Bioctal or Redundant Octal
Parity - Odd or even

f) Tape Compatibility.

Transport to transport
IBM 727, IBM 729 II and IV
Other IBM-compatible systems

g) Physical.

Size: Height - 72 inches
Width - 37 inches (2 tape transports)
- 60 inches (4 tape transports)
Depth - 30 inches (air cooled)
- 34 inches (water cooled)

Weight: 1150 pounds (2 tape transports)
1900 pounds (4 tape transports)

Cooling: Ambient Air - 700 CFM (2 transports)
- 1000 CFM (4 transports)
- 4.8 Gallons/minute at 70° F (2 transports)
- 8.0 Gallons/minute at 70° F (4 transports)

h) Power.

Air Cooled:	2 Transports	4 Transports
115 VAC ± 10%, 1 ph, 60 cps	2.75 KW	5.5 KW
115 VAC ± 5%, 3 ph, 400 cps, Reg	0.6 KW	0.6 KW

Water Cooled:	2 Transports	4 Transports
115 VAC ± 10%, 1 ph, 60 cps	2.6 KW	5.3 KW
115 VAC ± 5%, 3 ph, 400 cps, Reg	0.6 KW	0.6 KW
115 VAC ± 5%, 3 ph, 400 cps, Unreg	1.8 KW	3.0 KW

NOTE

For detailed specifications for the Tape Transports, see UNIVAC Document DS 4613.

5. Magnetic Tape to Computer Specifications.

a) Performance of Function. The Magnetic Tape Unit communicates with the computer in the request-acknowledge mode (see figure 3.1-3). The computer issues commands to the MTU by means of the external function signal and function words; thus computers with an 18-bit word structure (e.g., 1219) can use the 1240 tape unit as well as computers with longer word lengths. Function Words may be in the Address Word format (see figure 3.1-4) or in the instruction word format (see figure 3.1-5). When the MTC inspects the word format and recognizes it as an Address Word, it selects the specified tape cabinet and tape transport.* When the MTC recognizes the function as an instruction word, it performs the specified operation on the transport selected by the last interpreted address word. Each tape transport control contains a 4-position address switch for each tape transport so that each can be assigned a logical number 1, 2, 3, or 4. Another 4-position

*See Paragraph 3.1-4b5d).

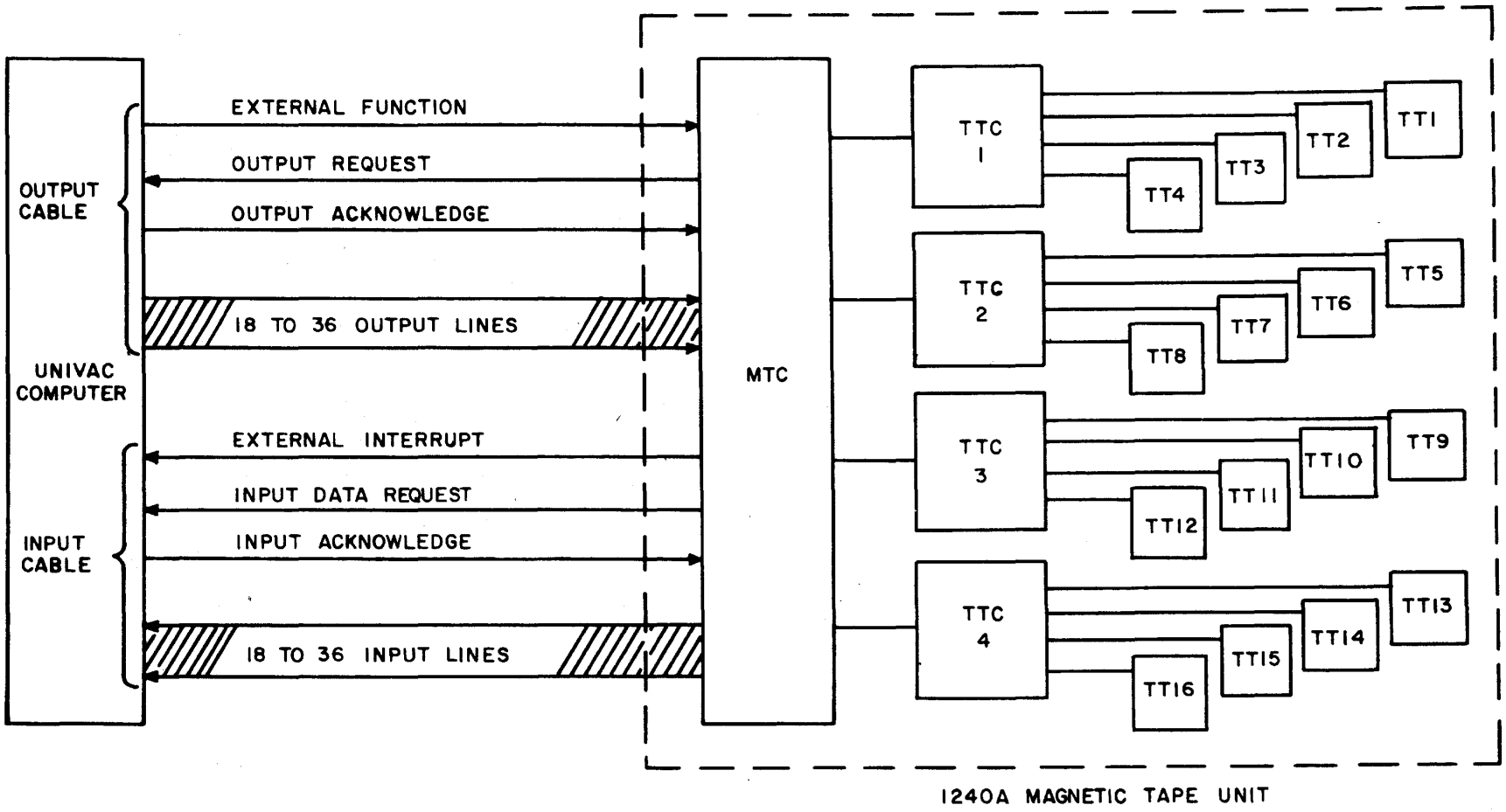


Figure 3.1-3. Magnetic Tape Unit - Computer Interface

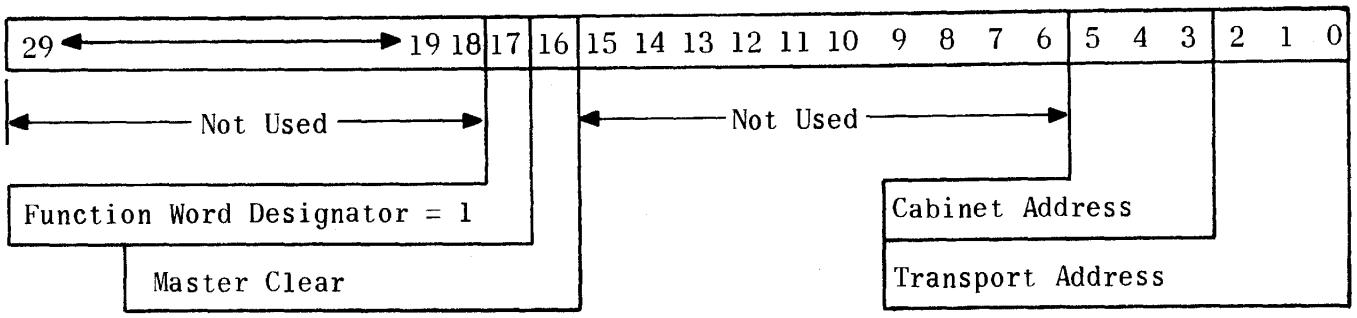


Figure 3.1-4. Function Word-Address Word Format

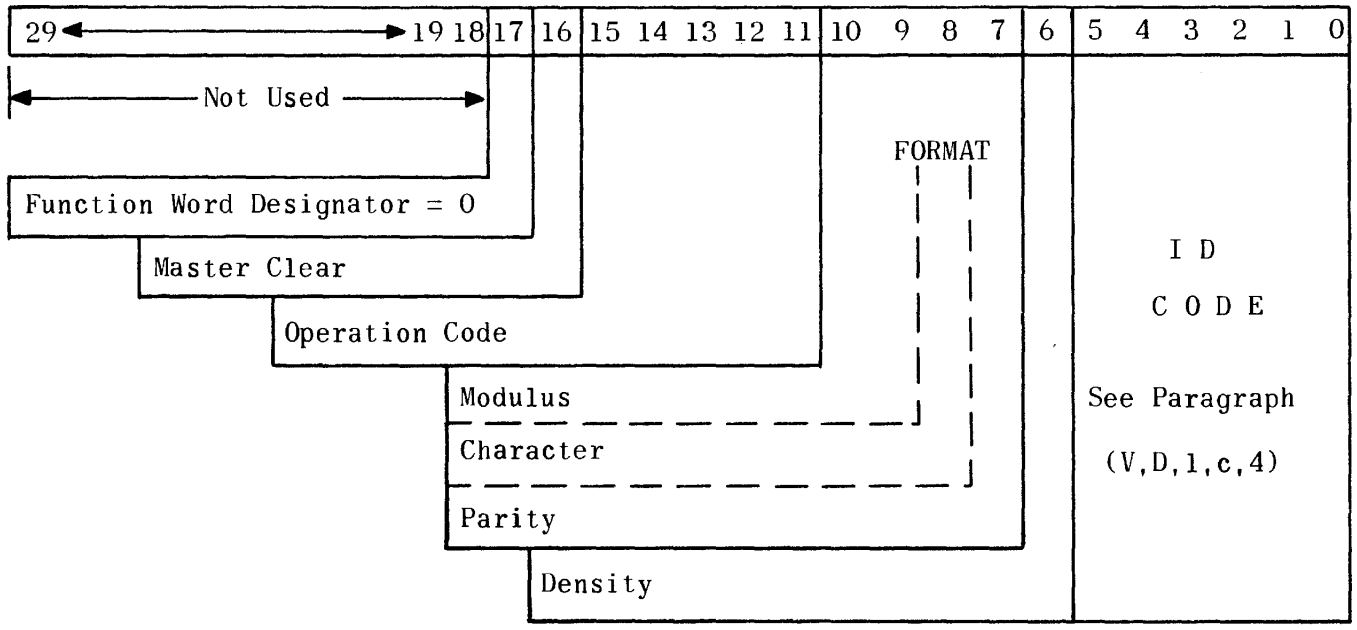


Figure 3.1-5. Function Word-Instruction Word Format

cabinet address switch assigns that cabinet a logical number 1, 2, 3, or 4. The address word selection bits are interpreted by the magnetic tape control according to physical position of these switches. No two transports in the same cabinet may be assigned the same logical number, i.e., to allow identical function on both units. If duplication does exist within the same cabinet, the lowest physically numbered tape transport is the one that will function. Example -- if transport 1 and 2 are both set on 2, only transport number 1 will respond to address word selection number 2. Similarly, no two cabinets on the same computer channel may be assigned the same logical number. One exception is permitted. Writing on two tapes simultaneously may be accomplished if two cabinets on the same channel are assigned the same logical number and the write-ignore-error-halt command is issued to the MTU. In no instance can any function other than Writing be accomplished simultaneously on two transports. The operator must determine the logical addresses required by each program and set the switches accordingly.

The instruction word will contain the code for one of five basic operations -- read, search, write, backspace, rewind -- or a combination of two basic operations. The master clear operation may be requested in either word format. This special operation has priority over all other operations and will terminate all others in process except rewind, which will go to completion.* To accept an external function command other than master clear, the magnetic tape unit must be in the IDLE state, i.e., operable but not performing a specific operation.

The completion of an operation or a master clear places the MTU in the IDLE state.

The general sequence of events for on-line operation with a computer (the MTC in Automatic mode and in the IDLE state) is as follows:

1) Computer issues an address word via the external function command.

2) The MTC selects the addressed tape cabinet and transport.

3) Computer issues an instruction word via the external function command.

4) MTC samples the instruction word and becomes BUSY.

5) The operations stated in the instruction word are initiated and carried to completion.

6) MTC sets a status word on the input lines.**

7) MTC interrupts the computer with external interrupt.

8) MTC issues STOP command to transport.

9) Computer samples status word and acknowledges interrupt whereby the MTU becomes IDLE.

(Items 8 and 9 may be interchanged, or take place simultaneously).

* See Paragraph 3.1-4b5d).
**See Paragraph 3.1-4b5c).

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 identifying signals on the external function line and the output acknowledge lines respectively. The output request line notifies the computer of the MTU ability to accept output data or function words. See figure 3.1-6.

b) 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. The Load Point marker is placed 1/32 inch from track 0 or outside 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 fourteen feet from the end of the tape. The markers are detected by reflective photoelectric sensors. See figure 3.1-7.

c) Status Word and Interrupt (Status Interrupt). The computer program is interrupted after completion of every operation, performed by the MTC, except master clear and transport address selection. The MTC 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 3.1-7) 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 MTU, 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 MTU 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 will be recognized by the MTC. An external function with a master clear or address word will be recognized at any time.

Successful completion of an operation will contain 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 is duplicated in the next high order bits beginning at bit 15. The following paragraphs present the detailed explanation of each bit of the status word.

1) Improper Condition (Bits 29 and 14 = 1).

a. An Improper Condition will occur whenever: selected tape transport is not in automatic condition. 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:

Power Off

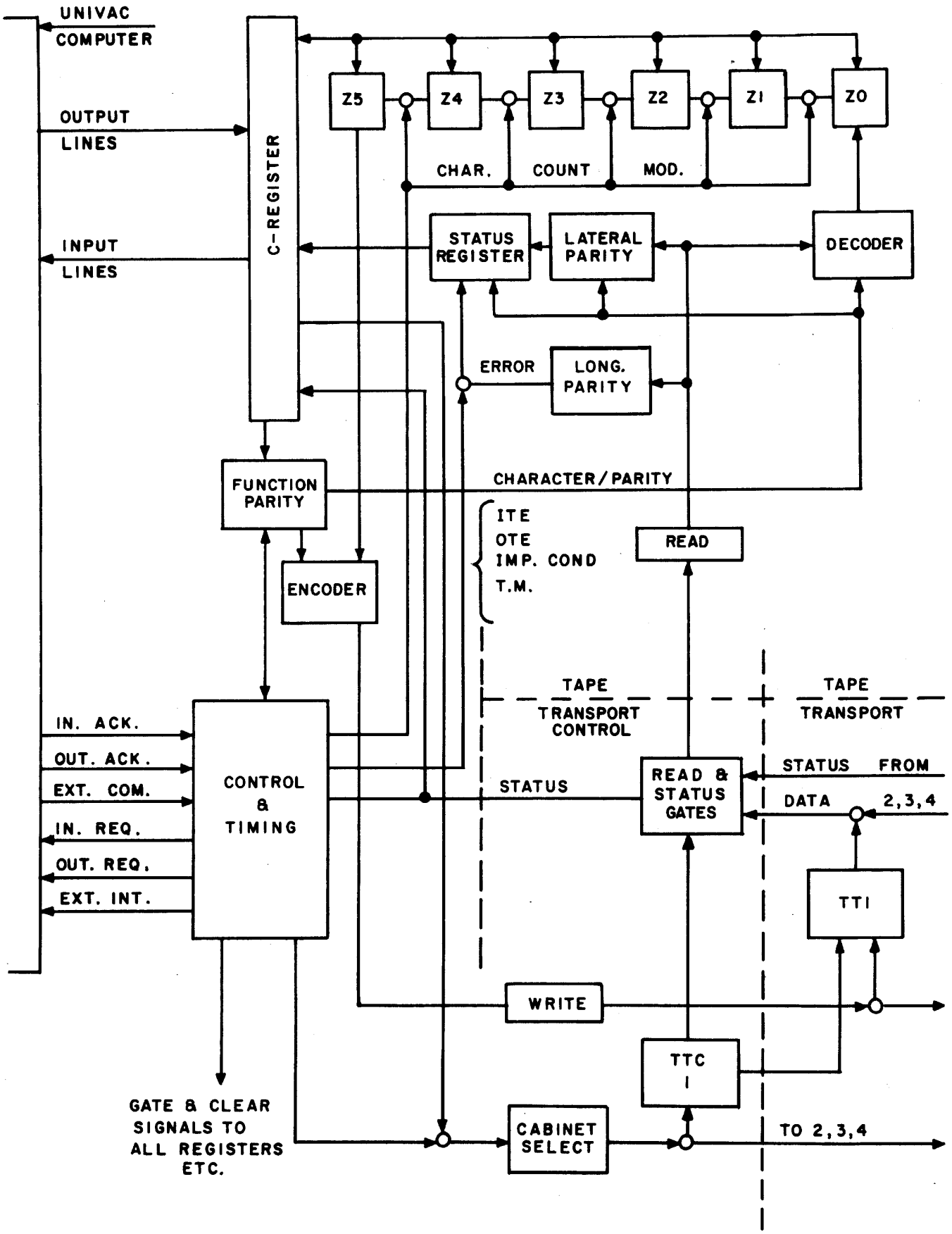


Figure 3.1-6 MTU, Simplified Block Diagram

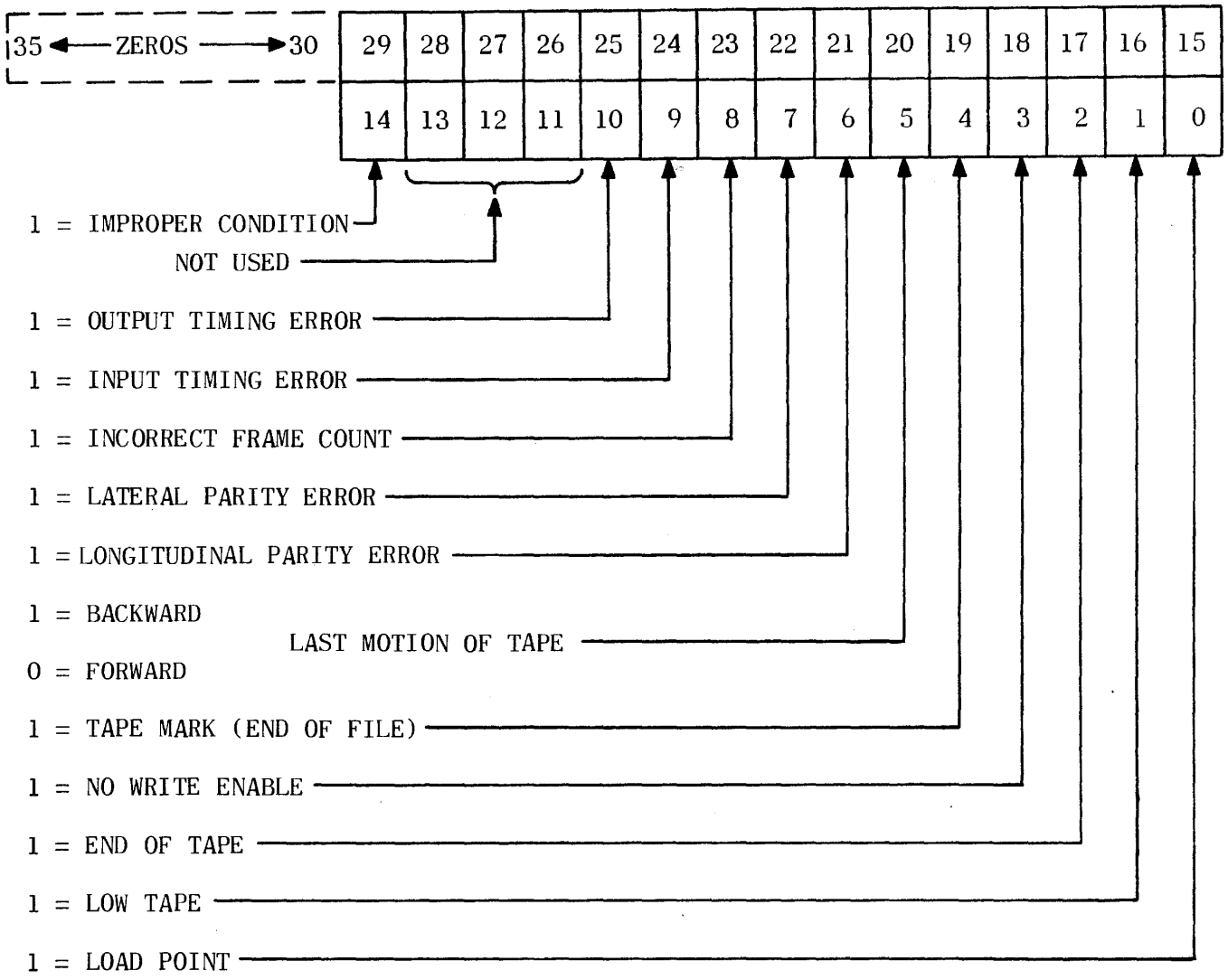


Figure 3.1-7. Magnetic Tape Unit - Status Word Format

Tape broken
Lamp burnout
Tape load was not accomplished when tape was mounted.

b. No tape transport is selcted when one is required.

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

d. A reverse command, other than a rewind operation, is sent to a tape transport whose tape is positioned at load point.

e. 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.) 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 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.

2) Output Timing Error (Bits 25 and 10 = 1). If the computer issued a write instruction to the MTC and did not transfer the first output data word or transferred 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 3.1-2. An output timing error can occur during search operations if the MTC does not receive a search key before the start of reading the record. The time requirement may be as short as 2 milliseconds from the time the instruction word is received by the MTC until the search key or the first data word must be received.

3) Input Timing Error (Bits 24 and 09 = 1). If the computer issued a read instruction and failed to accept a word placed on the input cable by the MTC before the next word was 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 MTC sends the status word indicating the error with an interrupt to the computer.

4) Incorrect Frame Count (Bits 23 and 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:

a. There were not enough frames in the record to complete an integral number of computer words.

b. One or more characters were not properly read or recorded.

TABLE 3.1-2. WORD ASSEMBLY TIME

FORMAT		WORD ASSEMBLY TIME	
Modulus	Character	Low Density	High Density
3	Bi-octal	133 usec	48 usec
4	Bi-octal	177 usec	64 usec
5	Bi-octal	222 usec	80 usec
6	Bi-octal	267 usec	96 usec
3	Octal	266 usec	96 usec
4	Octal	355 usec	128 usec
5	Octal	444 usec	160 usec
6	Octal	534 usec	192 usec

c. Bad spots on the tape caused characters to be lost.

d. 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 frome count error.

5) Lateral Parity Error (Bits 22 and 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 MTC 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.

6) Longitudinal Parity Error (Bits 21 and 06 = 1). During a writing process a longitudinal even parity bit is generated by the MTC for each tape channel and recorded after the last frame of the record. If the MTC 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.

7) Last Tape Motion (Bits 20 and 05; 1 = Backward, 0 = Forward). Any status word with interrupt sent to the computer at the completion of an operation will indicate the direction of the last tape motion. This bit indicator is especially useful to the program when a back-space-read operation results in a parity error detection. The program can determine whether the tape is positioned at the beginning or the end of the record.

8) Tape Mark (Bits 19 and 04 = 1). A recorded tape mark (see write tape mark) separates files of information on the tape. Any read, space file, search file or back space operation, that is limited to a file, and the post-write check of the write tape mark operation, will indicate a tape mark in the status word.

9) No Write Enable (Bits 18 and 03 = 1). When a write operation is attempted on a selected transport that has its write enable cleared or the write enable ring is inserted in the tape reel, the no write enable is indicated in the status word.

10) End of Tape (Bits 17 and 02 = 1). When the end of tape reflective marker is sensed by the MTU, 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 will appear 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.

11) Low Tape (Bits 16 and 01 = 1). A pressure-sensitive detector has sensed less than 100 feet of tape remaining on the selected transport reel. The MTC will indicate a low tape any time a status word is sent to the computer with the tape positioned within 100 feet of end of tape.

12) Load Point (Bits 15 and 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 about 3/4 inch beyond the load point marker (in the forward direction) before the first record may be written. The MTC will indicate load point in the status word whenever an operation requesting backward motion of tape is attempted with the selected tape positioned at load point.

d) External Function Commands - Function Words.

1) Master Clear. Operations and tape selections are requested by function words being sent to the MTU with an external function from the computer. A master clear of the magnetic tape unit is performed when bit 16 of the function word is a one. It differs from the other operations in these three respects:

a. It may be performed at any time, even when MTU is BUSY.

b. It has priority over all other operations in the Instruction Word, or Address Word. (see figures 3.1-4 and 3.1-5.

c. It does not result in a status interrupt.

The master clear stops all tape motion (except a rewinding tape) and sets the MTU in the IDLE state. At any time after a master clear the MTC will accept another external function. Since this function is not considered a normal operation, its use should be restricted to times when the MTU 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, the "rewind-clear write enable" instruction must be used.

2) Address Word. An individual tape transport is selected for operation by an address word and an external function. Figure 3.1-4 shows the format of the address word. When bit 17 of a function word is one and bit 16 is zero, it is sensed at the MTC as an address word. The address of the cabinet and tape transport to be selected is found in bits 05-00 of the address word. Refer to tables 3.1-3 and 3.1-4. An MTU consists of a maximum of four tape transport cabinets, with 2 or 4 transports each which must be assiged logical numbers by the operator. Within bits 05-00 of the address word, the cabinet is selected by bits 05-03 and the transport by bits 02-00 as shown in tables 3.1-3 and 3.1-4. Bits 15-06 of the address word are not used by the selection circuitry. The address word does not cause a status interrupt and the MTC will accept another external function command any time after the address word.

3) Instruction Word. Individual operations are performed by the MTU under the direction of an Instruction Word. Figure 3.1-5 shows the format of the instruction word. When bit 17 of a function word is zero and bit 16 is zero, it is sensed at the MTC as an instruction word. The operation to be performed by the MTU is stated in bits 15-11 of the instruction word. Format when required is stated in bits 10-7, high or low density in bit 6, and in selective read and write tape mark operations ,the identification key and tape mark first character respectively are stated in the identifier code bits 5-0.

a. 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 MTC in the assembly and disassembly of computer words from or to tape frames.

1. 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 3.1-8). 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 MTC when writing or reading octal characters. The redundant recording in octal format adds to the reliability (see figure 3.1-9) For compatible tapes, data must be recorded in bioctal format.

2. Modulus. The Modulus specifies the length of the computer word to be recorded on tape or read from the tape (see table 3.1-5).

(a) Modulus 3 (Designator Bits 10 and 09 = 00). An eighteen 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 eighteen bits for recording the MTC will record the lower order eighteen bits of the word on the tape and discard the remaining high order bits. During Mod 3 reading operations, three tape frames are assembled as an eighteen-bit computer word for bioctal character format, or six tape frames are assembled as an eighteen bit computer word in octal character format. If the

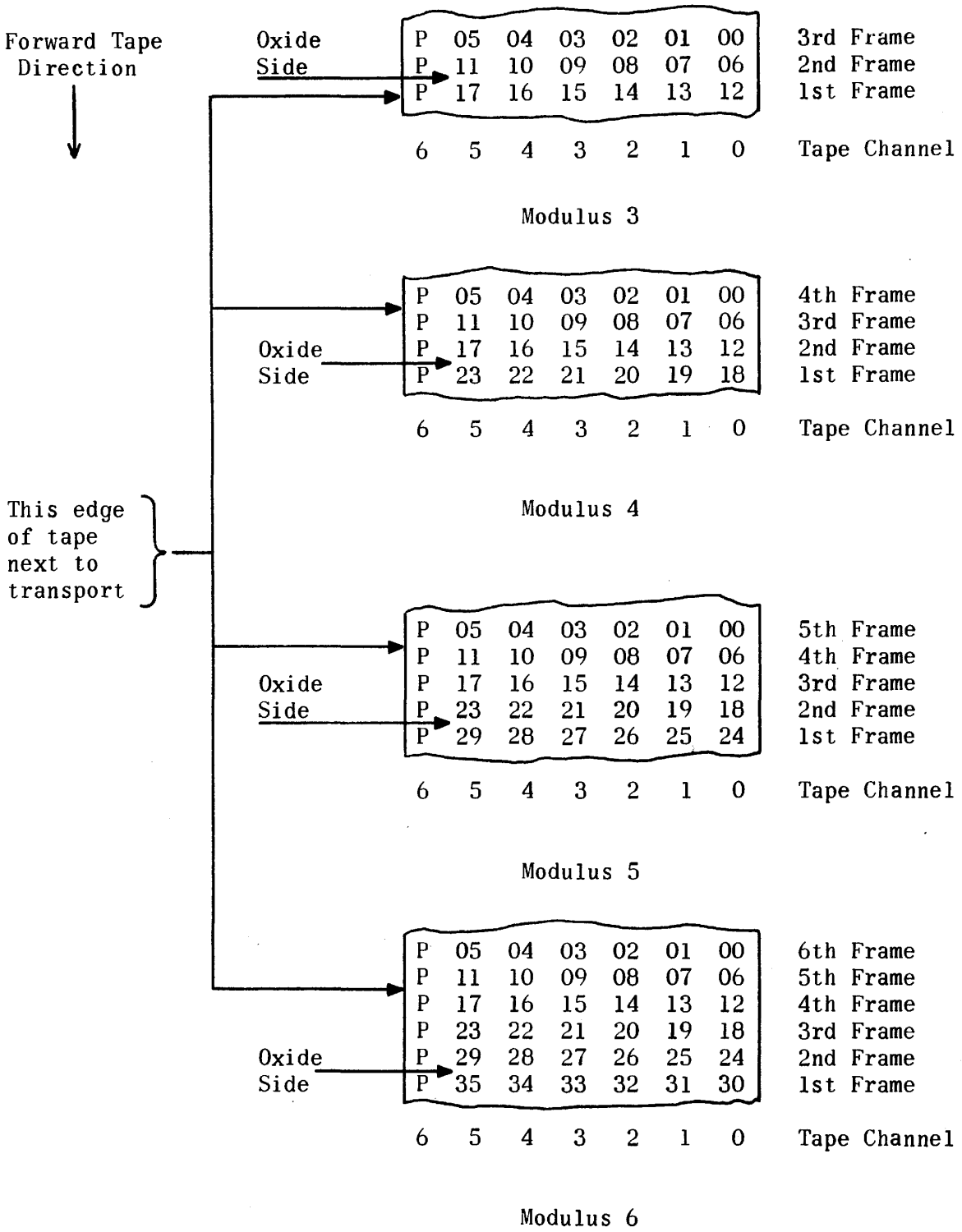


Figure 3.1-8. Bioctal Tape Format

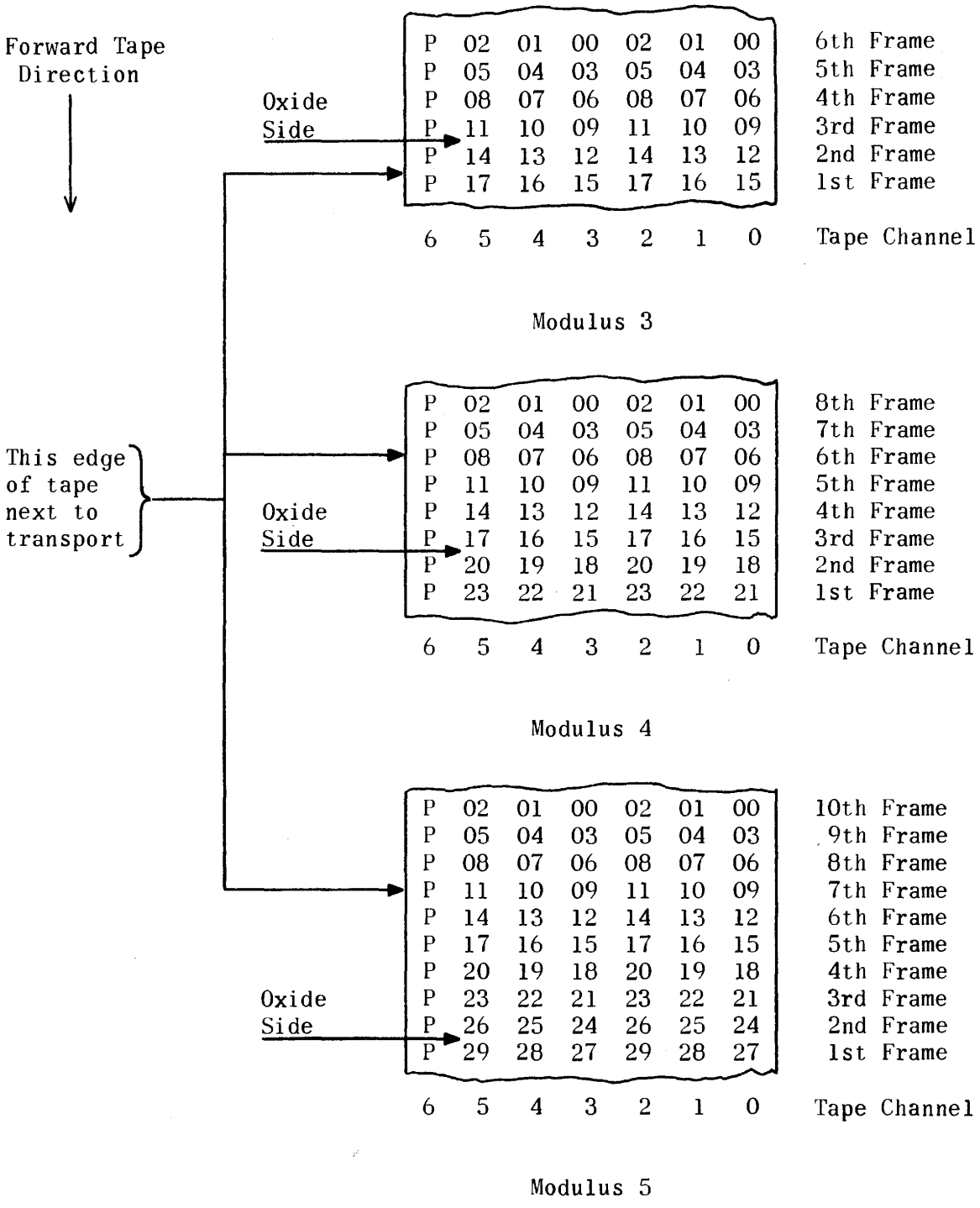
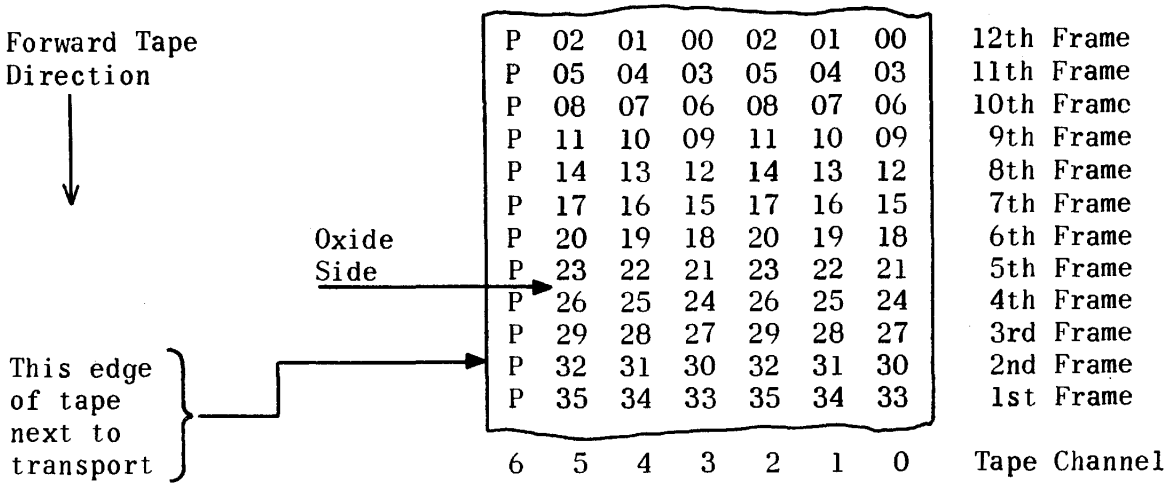


Figure 3.1-9. Octal Tape Format



Modulus 6

Figure 3.1-9. Octal Tape Format (Cont.)

TABLE 3.1-3. CABINET ADDRESS

Bits 05	04	03	Cabinet Selected
0	0	0	4
0	0	1	1
0	1	0	2
0	1	1	3
1	0	0	4
1	0	1	1
1	1	0	2
1	1	1	3

TABLE 3.1-4. TRANSPORT ADDRESS

Bits 02	01	00	Transport Selected
0	0	0	None
0	0	1	1
0	1	0	2
0	1	1	3
1	0	0	4
1	0	1	1
1	1	0	2
1	1	1	3

TABLE 3.1-5. EFFECTS OF VARIOUS UNIVAC COMPUTERS OPERATING WITH THE UNIVAC 1240 MAGNETIC TAPE SYSTEM

COMPUTER	MOD 3 BCD WRITE	MOD 4 BCD WRITE	MOD 5 BCD WRITE	MOD 6 BCD WRITE
UNIVAC 1218 Since Channel (One 18 bit word is output/request)	For each 18 bit word received, the 1240 writes 3 FRAMES on the tape, MOD 3 is RECOMMENDED for 1218 single channel operation.	For each 18 bit word received, the 1240 writes 4 FRAMES on the tape: 1 FRAME of zeros followed by 3 data FRAMES.	For each 18 bit word received, the 1240 writes 5 FRAMES on the tape: 2 FRAMES of zeros followed by 3 data FRAMES.	For each 18 bit word received, the 1240 writes 6 FRAMES on the tape: 3 FRAMES of zeros followed by 3 data FRAMES.
UNIVAC 1218 Dual Channel (One 36 bit word is output/request)	For each 36 bit word received, the 1240 writes 3 FRAMES on the tape. Since only the 18 LSB are written, the 18 MSB are "lost".	For each 36 bit word received, the 1240 writes 4 FRAMES on the tape. Since only the 24 LSB are written, the 12 MSB are "lost".	For each 36 bit word received, the 1240 writes 5 FRAMES on the tape. Since only the 30 LSB are written, The 6 MSB are "lost". MOD 5 is commonly used when preparing tapes for the 30 bit computers: CP-642B or 1206.	For each 36 bit word received, the 1240 writes 6 FRAMES on the tape. MOD 6 is RECOMMENDED for 1218 dual channel operation.
UNIVAC 1206 or UNIVAC 1212 or CP-642B (One 30 bit word is output/request)	For each 30 bit word received, the 1240 writes 3 FRAMES on the tape. Since only the 18 LSB are written, the 12 MSB are "lost".	For each 30 bit word received, the 1240 writes 4 FRAMES on the tape. Since only the 24 LSB are written, the 6 MSB are "lost".	For each 30 bit word received, the 1240 writes 5 FRAMES on the tape. MOD 5 is RECOMMENDED for operation with CP-642B or 1206 computers.	For each 30 bit word received, the 1240 writes 6 FRAMES on the tape; 1 FRAME of zeros followed by 5 data frames. MOD 6 is commonly used when preparing tapes for 36 bit computers.

.. LSB Least Significant Bits

.. MSB Most Significant Bits

computer word size is larger than eighteen bits, the frames are assembled in the lower order eighteen bits and zeros are placed in remaining high order bits (see figures 3.1-8 and 3.1-9).

(b) Modulus 4 Designator Bits 10 and 09 = 01). A twenty-four 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 twenty-four bits for recording, the MTC will record the lower order twenty-four bits of the word on the tape and discard the remaining high order bits. During Mod 4 reading operations, four tape frames are assembled as a twenty-four bit computer word for bioctal character format or eight tape frames are assembled as a twenty-four bit computer word for octal character format. If the computer word size is larger than twenty-four bits, the frames are assembled in the lower order twenty-four bits and zeros are placed in the remaining high order bits (see figures 3.1-8 and 3.1-9).

(c) Modulus 5 (Designator Bits 10 and 09 = 10). A thirty 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 thirty bits for recording, the MTC will record the lower order thirty bits of the word on the tape and discard the remaining high order bits. During Mod 5 reading operations, five tape frames are assembled as a thirty bit computer word for bioctal character format or ten tape frames are assembled as a thirty bit computer word for octal character format. If the computer word size is larger than thirty bits, the frames are assembled in the lower order thirty bits and zeros are placed in remaining high order bits (see figures 3.1-8 and 3.1-9).

(d) Modulus 6 (Designator Bits 10 and 09 = 11). A thirty-six 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 thirty-six bit computer word for bioctal character format or twelve tape frames are assembled as a thirty-six bit computer word for octal character format (see figures 3.1-8 and 3.1-9).

3. 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 MTC for the octal character writing and reading operations. For compatible tapes, odd parity is chosen for binary coded data and even parity is chosen for binary coded decimal (BCD) data.

b. Density Designator (Bit 6), 1 Selects High Density; 0 Selects Low Density. At low density data are recorded on the tape at 200 frames per inch (22.5 kc rate). At high density data are recorded on tape at 555.5 frames per inch (62.5 kc rate). Density must be specified in instruction words requesting reading or writing operations. See table 3.1-2 for word assembly and disassembly time.

c. Operation Code. The operation code is located in bits 15-11 of the instruction word. Legal operation codes exist for the five basic operations and for combinations of these operations. The five basic operations are read, search, write, backspace, and rewind. Operation codes using any basic oper-

ation (except rewind) must be supplemented by format and density codes placed in bits 10-07 and bit 06 respectively of the instruction word. Table 3.1-6 is a listing of the Operation codes and Figure 3.1-5 shows the structure of the entire instruction word.

1. Read Operations. The selected transport moves tape at 112.5 inches per second in the forward direction and transfer 7 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 MTC. The tape continues to move and new words are being assembled until the end-of-record (inter-record 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 4) 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 time of read operation, an input timing error will occur and the MTC will cease to transfer data to the computer for the remainder of the record. Following the detection of the end-of-record the MTC will set an input timing error status word on the input lines and interrupt the computer program by setting the external interrupt (EI) line on that channel. When the computer acknowledges the interrupt the MTU becomes IDLE.

In all types of read operations, format and density selections must be made in each instruction.

2. Normal Read. The selected transport will read one record according to the format stated and check each frame for parity. If a parity error is detected, the MTC will cease transferring data to the computer for the remainder of that record. After sensing the end-of-record, the MTC will send a status word to the computer with a signal on the external interrupt line. This status word will contain MTU status and any or all error indications encountered during the reading of the record.

3. Read-Ignore Error Halt. The selected transport will read one record accoring to the format stated and will check each frame for parity. If a parity error is detected the complete record will be sent to the computer after detection of end-of-record and will contain MTU status and any or all error indications including parity error.

4. Selective Read. The selected transport will read one record according to the format stated and check each frame for parity. The MTC will compare the least significant 6 bits (05-00) of each assembled computer word with the 6 bits of the identification (ID) code in the instruction word. If the comparison is affirmative the word will be transmitted to the computer. If the comparison is negative the word will be discarded. If a parity error is detected the MTC will cease transferring data to the computer for the remainder of the record. The status word sent to the computer after detection of the end-of-record will contain MTU status and any or all error indications.

5. Write Operation. When the MTC senses a write function, the selected transport moves the tape forward at 112.5 ips and records the inter-record gap (IRG). 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 MTC 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 MTC disassembles each word according to the modulus selected in the write function word, generates frame parity, and transfers the 7 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 MTC 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 (see Table 3.1-2). When the computer does not respond within the allotted time, the end of write is assumed by the MTC. Longitudinal parity is written and the recording process is terminated. Tape motion is stopped after a portion of the inter-record gap is written on the tape. The MTC removes the output request and places a status word, indicating successful completion of the write, on the input lines and sets the external interrupt line. When the computer acknowledges the interrupt the MTS becomes IDLE. If the computer acknowledges the output request after the allotted time but before the interrupt is sent, the MTC interprets the action as an output timing error and notifies the computer in the status word.

NOTE

The normal inter-record gap is approximately 3/4 inch in length and the extended inter-record gap is approximately 3-1/2 inches in length.

6. Write. The selected transport will write 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.

7. Write-Ignore Error Halt. The selected transport will write on the tape according to the format and density stated in the function word but the MTC will 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 will contain MTU status and any or all errors encountered.

8. Write-Extended Inter-Record Gap (XIRG). The selected transport will record an extended inter-record 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 are transferred from the computer for recording, the extended inter-record gap will be present on the tape and an output timing error will occur. The status word sent to the computer after completion will contain MTU status and any or all error indications detected as in a normal write operation.

NOTE

Under program control, inter-record 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 will be indicated in the interrupt status word following each write operation performed in this manner.

9. Write Tape Mark. The selected transport will write 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 MTC upon receiving the instruction word which must state even parity, bioctal character and 17_8 as the ID code. Modulus selection can be ignored. The ID code is used as the first frame of the tape mark record and longitudinal parity makes it the last. To be compatible with other tape systems, the tape mark must be exactly as specified above. However, the MTU will recognize tape marks which contain any first frame code other than zeros. A status word with interrupt is sent to the computer after completion of the write tape mark operation.

10. Back Space. The selected transport will move the tape in the reverse direction to the next IRG (back one record). The tape is properly positioned in the IRG for reading or writing. Format and density must be stated in the instruction word since parity will be checked during the backward motion. Any error detected and MTU status will be 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 instruction is given, an improper condition exists and will be noted in the status word.

11. Rewind. The selected transport will rewind the tape backward to the load point at 225 inches per seoncd. The status word with interrupt is sent to the computer after the MTC 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 motion or improper condition will result but the status word will indicate load point. This provides a method of testing for completion of the rewind operation.

12. Multi-Function Operations. Multi-function operations consist of combinations of basic operations of the MTU, which can be performed in response to 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 multi-function operations combine a read with a space or 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.

13. Search - Type I and Type II. The search operation combines the features of normal read and a search. The selected tape transport will read records from the tape either forward or backward and compare the first word* of each tape record with a search key (identifier word) which is transmitted

*In a forward Search the first word encountered in each record is the first word of the record. In a backward Search the last word encountered in each record is the first word of the record.

TABLE 3.1-6. 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 Mark, XIRG
1000X*	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.

from the computer to the MTU by an output buffer of one word. When a compare is affirmative, that find record is transmitted to the computer as in a normal read.

Tape motion is started upon receipt of the instruction word. If the MTU does not receive the search key from the computer before it starts reading the record, an output timing error will occur. This reading start time may be as short as 2 milliseconds. The search operation will be terminated by the MTU when a parity or timing error is detected by the MTC. The status word containing MTU status and any or all error indications will be 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 inter-record gap (IRG) 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 bad parity check on any frame of the tape being searched.

The ONES (Type I) compare is a "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 6-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

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.

14. Search File* Forward/Backward. The MTU will perform 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 will be terminated and the tape mark status code will be present in the status word sent to the computer after detecting end of record.

15. Space File Forward/Backward. The MTC will cause the selected transport to move the tape in the specified direction to the inter-record gap beyond the next tape mark and will place a tape mark code in the status word sent to the computer detecting the end of record (see figure 3.1-10). Space file forward will position a tape at A upon completion; space file backward will position the tape at B.

* A file is defines as one or more records separated by tape marks (see figure 3.1-10).

** A tape mark is a special record on a tape placed there by the operation "Write Tape Mark" See Paragraph 3.1-4b5d)3)c1 and figure 3.1-11.

16. Back Space Read. The selected transport will move the tape backward to the next inter-record gap (back one record) and then performs a normal read according to format and density stated in the instruction word. Parity will be checked while backspacing; if an error is detected during backward motion, the operation will be determined before the read operation is performed and the status word will contain the MTU status and the error indication. The computer program must determine if the error detection occurred during the back spacing operation or during the read operation by examining the 2^5 bit (last tape motion) of the status word received.

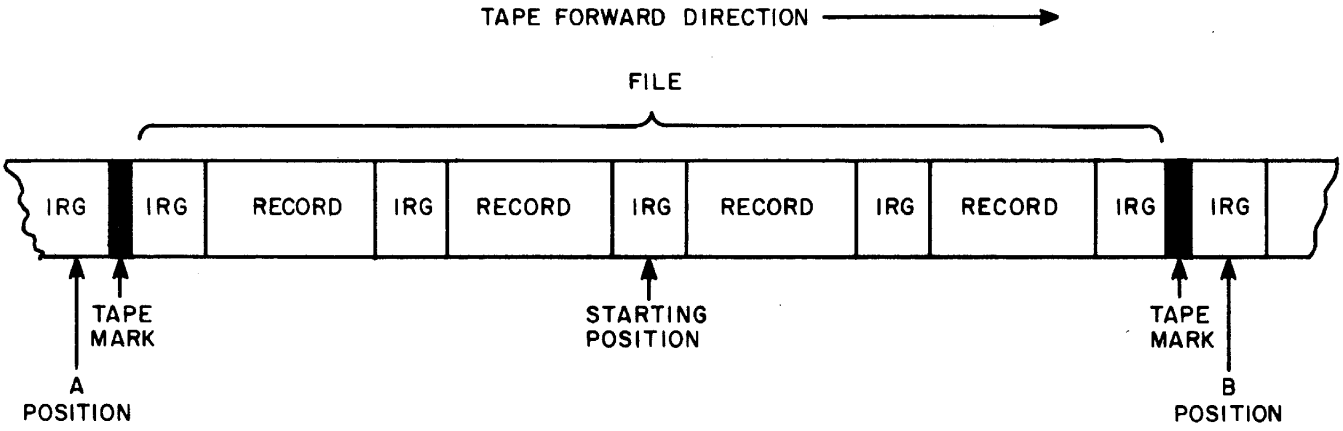
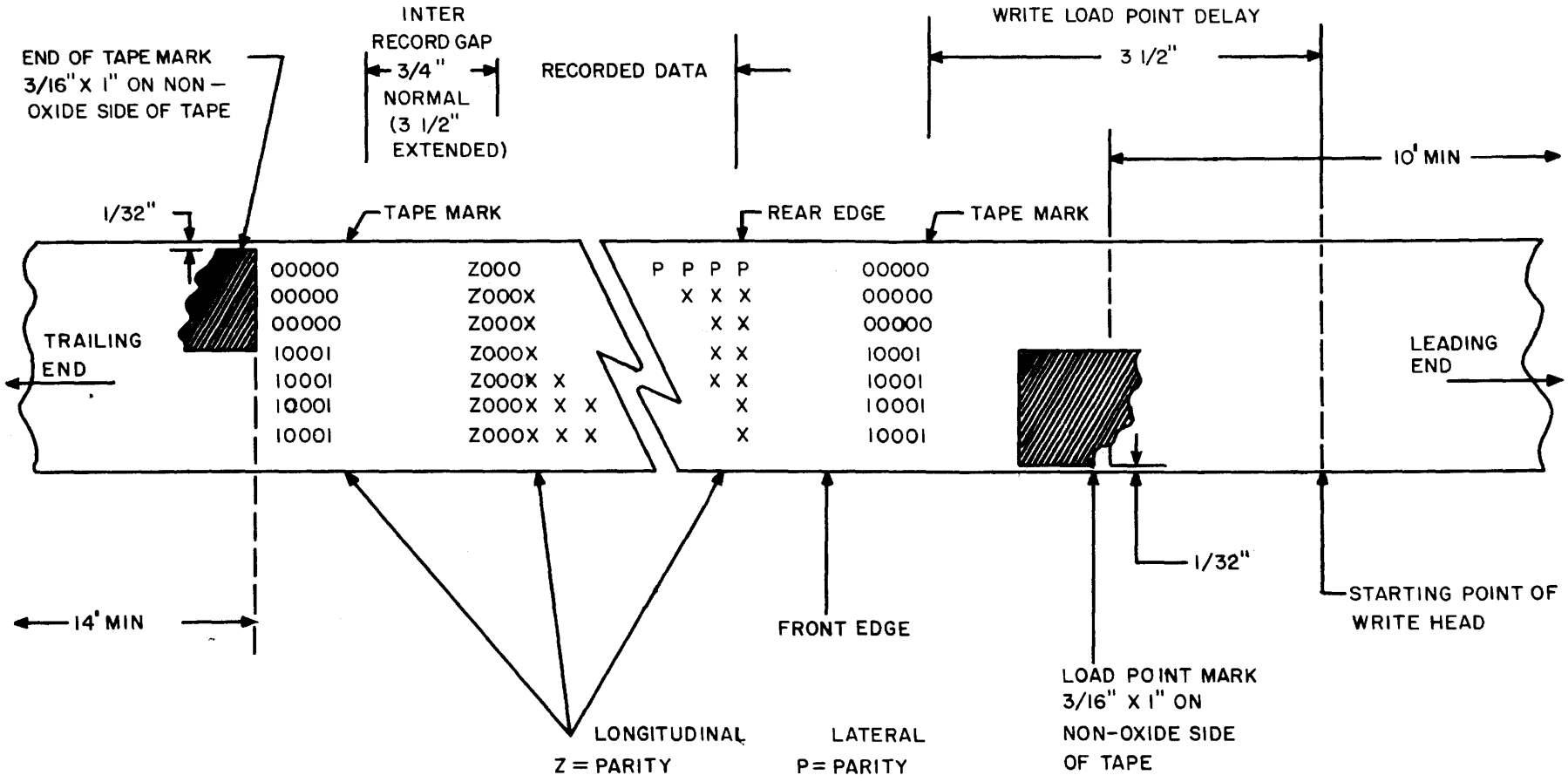


Figure 3.1-10. Magnetic Tape Unit - Tape File

17. Rewind-Read. The selected transport will rewind the tape to the load point at 225 inches per second and then perform a normal read of the first record according to the format and density stated in the Instruction Word. A status word containing MTU status and any or all errors will be sent to the computer with interrupt after detecting the end-of-record.

18. Rewind-Clear Write Enable. The selected transport will perform a normal rewind of the tape to load point and will clear the write enable. This selected transport will no longer perform a write function without manual intervention. The status interrupt will be presented upon initiation of the rewind and not upon completion.



FORWARD DIRECTION OXIDE DOWN

Figure 3.1-11. Magnetic Tape Format

6. Magnetic Tape Unit - High Speed Printer Off-Line Capability. The UNIVAC 1240 Magnetic Tape Unit is capable of communicating directly with the UNIVAC Type 1469 High Speed Printer for OFF-LINE operation. The MTU communicates with the HSP unit in the request acknowledge mode.

The Magnetic Tape - Printer Interface is shown in figure 3.1-12.

Using terms based on Computer - MTU communication and Computer - HSP communication in this discussion the output to the HSP interface is connected to the Input from the MTU inteface, i.e., the

HSP Output Request Line is connected to the
MTU Input Acknowledge line. The
MTU Input Request line is connected to the
HSP Output Acknowledge line. The
MTU Interrupt line is connected to the
HSP External Function line. The
MTU data lines are connected to the
HSP data lines.

The High Speed Printer exercises control of the OFF-LINE system after the MTU is switched to printer mode, the desired tape transport is selected and the tape positioned at load point. The HSP will initiate 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 fieldata character record lengths (120 characters per line on HSP). As each record is read from the tape and transmitted to the HSP, 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 HSP must contain 5 fieldata code (see table 3.1-7) characters. These are in turn disassembled into six-bit characters and stored in the character core memory of the HSP Control Unit. When the core memory character counter indicates 120 characters, the print cycle is initiated and the line is printed. A record of less than 24 thirty-bit words will indicate to the HSP to stop the print operation. A record of five space codes (05) will stop the print operation without printing a line.

a) Operating Instructions. To prepare the OFF-LINE MTU-HSP system for operation the operator must perform the following steps.

STEP 1. Select character, parity and density, of the recorded tape, at the Magnetic Tape Unit cabinet.

STEP 2. Switch the MTU to printer mode.

STEP 3. Select the desired tape transport.

STEP 4. Load and position the tape at load point.

STEP 5. Place the HSP in ON-LINE position.

b) Sequence of Events. The normal sequence of events for transfer of data to the HSP is as follows:

STEP 1. The HSP sets its output data request.

STEP 2. The MTS, in the IDLE state, recognizes this first ODR as a command to start the read operation.

STEP 3. The MTU places a word on the data lines and sets its input data request.

STEP 4. The HSP recognizes this IDR as an Output Acknowledge.

STEP 5. The HSP samples the data lines and clears its ODR.

STEP 6. The MTU recognizes the clearing of the ODR 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 reinitiated. The process continues until the end of file tape mark is read. The HSP recognizes the tape mark as a command to position the paper at top of form on the next page. The interrupt line of the MTU being connected to the external function line of the HSP permits the end-of-record and the tape mark codes to be sent to the HSP as commands to move paper one line space or top of form respectively.

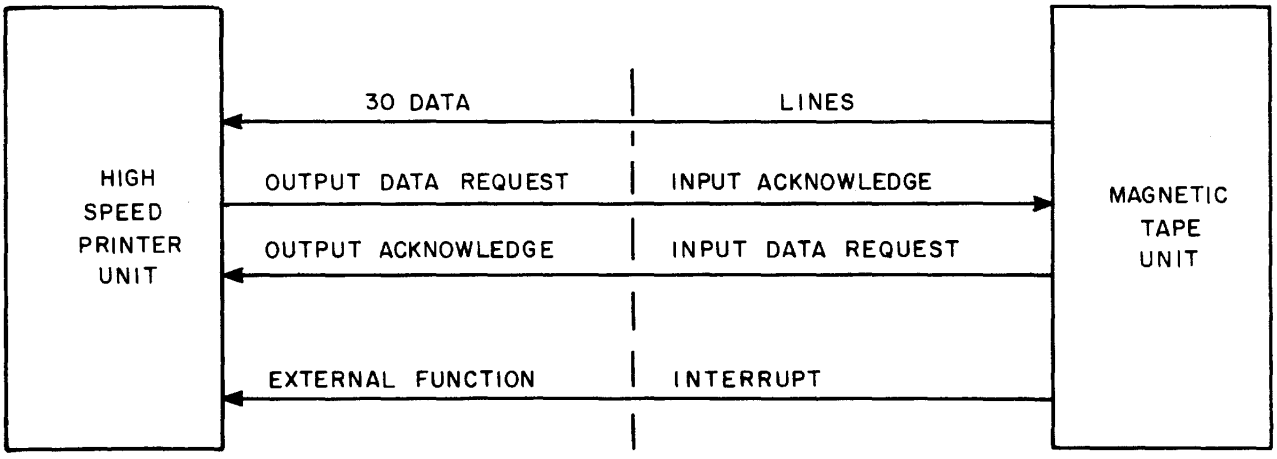


Figure 3.1-12. Magnetic Tape - Printer Interface

c. UNIVAC 1004 Card Processor.

1. Introduction. The UNIVAC 1004 Card Processor, 80-column edits and accumulates totals from data punched into 80-column cards and prints the results in any desired format. Built-in abilities to perform arithmetic, transfer, and compare operations, and reliable fast-access magnetic core storage provide a high degree of data-processing efficiency.

The 1004 processor consists of a card reader, a processor, and a printer, housed in a single compact unit. A card punch can be included as an optional output unit.

The functions of card reading, data-processing, printing, and punching are user-programed through wiring of a removable connection panel. To accomplish the desired procedures, the machine follows a series of instructions called STEPS. These steps are defined by connection panel wiring and can be executed in any sequence. The use of magnetic core storage allows the steps to proceed at microsecond speeds.

2. Read Section. The card reading is performed column by column at a basic speed of 300 cards per minute. During reading, the card image is transferred to a section of the core storage specifically assigned to card reading.

3. Processing Section. The processing section contains magnetic core storage and the control circuitry necessary to perform the machine operations required. Once input data has entered read storage, the control circuitry performs whatever operations have been programed for execution. When these are completed, the results are transferred to output storage for printing and/or punching. The capacity of core storage is 961 locations each of which is made up of six magnetic cores. Each core plane is 31-by-31.

The 1004 processor is a character oriented machine. Data is transferred between storage locations one character at a time. The machine code of the Processor is UNIVAC XS-3 code.

4. Printing Section. Printing speed is 300 lines of alphanumeric data per minute, with a maximum of 132 print positions per line. Character spacing is ten characters to the inch horizontally, with an operator option of 6 to 8 lines to the inch vertically.

5. Card Punch. The punch is directly connected to the 1004 Processor through an electrical cable. The speed of the card punch is 200 cards per minute.

6. Interfacing with 1219 Computer. The 1004 processor may be connected to any one of the I/O channels of the 1219 computer to permit medium speed printer and card punch output and card input capabilities for the 1219 Computer.

The particular input/output format required between computer and processor may vary with customer installation.

d. UNIVAC 1257, 1259, 1262 Teletypewriter Sets.

1. Introduction. The UNIVAC Teletypewriter Set is a convenient input/output and monitoring device for use with Univac computers when TELETYPE® and line communication is required. The set consists of a Teletype ASR-28 send-receive set,

® Registered Trademark of American Telephone and Telegraph Co.

a UNIVAC Adapter (type 1257, 1259, or 1262) and an auxiliary line relay. The adapter converts the serial nature code transmission characteristics of teletype to the parallel characteristics of the computer and vice versa and provides the logic necessary for interunit communication and buffer operations with the computer. The auxiliary line relay, under control of the adapter, routes data between the teletypewriter and external communication lines. Keyboard or paper tape entries of data may be made to the computer at the site or from some remote location. Printed copy or punched paper tape outputs of data from the computer may be available at the site or at some remote location. The components of the UNIVAC Teletypewriter set perform certain functions for the computer system. As a result, the following off-line and on-line operations are possible.

a) Off-Line Operations.

Printed and punched paper tape preparation
Keyboard transmission
Simultaneous keyboard transmission and paper tape preparation
Automatic tape transmission
Page copy of either incoming or outgoing messages

b) On-Line Operations.

Keyboard entries to the computer
Paper tape entries to the computer
Simultaneous keyboard entries to computer with paper tape copy and/or page copy and/or external line transmission
Simultaneous paper tape entries to the computer with page copy and/or external line transmission
Data outputs from the computer to page printer and/or paper tape copy and/or external line transmission.

2. Description. The teletype model 28 automatic send-receive set (28-ASR) consists of a page printer, a keyboard, a tape punch, and tape readers.

a) Keyboard. The keyboard consists of a set of manually operated keys which, when pressed, generate teletypewriter codes for transmission to the printer, the typing reperforator, the auxiliary reperforator, the computer, or the external communications line.

b) Page Printer. The page printer accepts teletype codes from the keyboard, tape reader, adapter, or communication line (auxiliary line relay) and converts them to print and printer control operations (carriage return, line feed, etc.). Up to 72 characters or spaces may be printed in a line on paper of 8-1/2 inches maximum width. Printing may be performed on page size forms or on a continuous roll of paper. Single thickness rolls of paper are fed by friction through the paper advance and printing mechanism while multi-copy forms are aligned and advanced by a pin feed mechanism.

c) Tape Reader (Transmitter/Distributor). The automatic tape reader/transmitter is a fixed-head, single-contact, serial read, tape reader that senses codes punched in the tape and generates teletype signals from a single-contact distributor. Manual controls permit starting and stopping on the tape reader, and a free-wheeling position of the start-stop switch disengages the tape feed wheel for easy threading and positioning of tape. An index line is marked on the tape guide,

six characters in front of the tape-reading station, to indicate the printed character on the tape that corresponds to the punched character in the reading station at any moment.

Automatic controls are incorporated for stopping the reader if the tape becomes tangled, taut, or the end-of-tape condition occurs.

d) Typing Reperforator. This unit is controlled by the keyboard code or by line signals and prepares a printed and punched, chadless, paper tape. Punched tapes can thus be identified and read directly, without the need for interpreting punch codes or running tapes through a reader. This is of particular value when tape is punched off-line and no simultaneous page record is being prepared; direct printing on the tape allows rapid verification and correction of punching errors.

Its ability to operate from both line signals and keyboard input makes the reperforator useful in combining and editing tapes. Portions to be reproduced from old tapes can be transmitted to the reperforator from the tape reader, while new portions can be inserted by manual operation of the keyboard. The typing reperforator prints a character for each code received, including machine function codes.

e) Auxiliary Typing Reperforator. A typing reperforator, identical with the one described above, is mounted in the upper left portion of the UNIVAC Teletypewriter set. This unit permits recording of all incoming messages on printed punched tape while the primary tape punch is being used for local tape preparation.

f) Teletypewriter Adapter. Teletypewriter (TTY) equipment cannot communicate directly with the computer because it operates serially whereas the computer operates in parallel. The adapter converts the parallel format of the computer word to the serial format of the teletypewriter, and vice versa. Each teletype character consists of seven pulses: five data pulses, a start space (binary 0), and a stop mark (binary 1) pulse. The space and mark serve as control signals between the teletypewriter and the adapter and indicate the beginning and end of a character code. It would be redundant to transmit these codes each time a character is transferred to and from the computer. Therefore, the teletypewriter adapter affixes the start and stop pulses to computer output words, and strips the pulses from the computer input words.

The computer, which must have some method of sending and receiving control information to and from each piece of peripheral equipment, sends external function commands and receives interrupt codes when communicating with external equipment. Teletypewriter equipment is not capable of generating, deciphering, and responding to these control codes. The adapter provides the necessary logical circuitry to perform these functions.

e. UNIVAC 1469 High Speed Printer.

1. General Information. The UNIVAC Type 1469 High Speed Printer is a 667 or 1000 line per minute printer for use on-line with any Univac military computer or off-line with the UNIVAC 1240 (or similar) Magnetic Tape Unit. It can print 120 alphanumeric characters per line. Sixty-three printable characters plus space,

selected by a 6-bit code, are available in each character position of the line. The unit, consisting of an Anelex 4-1000 printer, printer control unit (PCU), power supply, and cooling system, is packaged in a single cabinet, constructed of steel and fiberglass. The UNIVAC 1469 printer, having been designed for military applications, has a greater capability of operating reliably in adverse environmental conditions than the conventional type of printers normally used. Solid-state circuitry of advanced design and rigid Univac quality control procedures are combined in the fabrication process to provide the user with unexcelled, trouble-free, printing service.

The printer control unit (PCU) contains the electronic logic and control circuitry required to provide the communications necessary in the transmission of printable data and command signals between a computer or magnetic tape unit and the high-speed line printer. It incorporates a core memory for storage of the 120 characters that are used to print one line. The PCU disassembles the words transmitted to it for printing and provides the timing and commands required for the mechanical and electrical operations involved in the printing process and paper movement. Controls and switches easily accessible to the operator permit direct control of the characteristic features of the printer: loading paper, adjusting printing quality, printed page lengths, and for off-line operation and maintenance procedures.

In the on-line application, the computer program directs the format and printing of a line or page. Abnormal activities such as printer not ready, broken paper, and out of paper set corresponding indicators on the operator's panel and inform the computer so that remedial action may be taken.

The off-line printing characteristics of the UNIVAC 1469 High-Speed Printer contribute to more efficient computer processing systems since valuable computer time is not utilized for initiating and executing buffers and commands to the printer. During off-line printing operations with a magnetic tape unit, the PCU takes its commands and format control from the format of each record on the magnetic tape. Each line to be printed must appear in its desired format as one 120-character record. All lines desired on a single page must appear on the tape as one complete file. The tape mark (end-of-file) will be recognized by the high-speed printer as a top of form command, and each 120-character record will be recognized as a line of print.

2. Mechanical Description.

a) Printer Unit. The printer unit is composed of two main assemblies: printing assembly and paper feed assembly. In addition, there are fault sensing devices which monitor the operation of the printer unit.

1) Printing Assembly. The printing assembly consists of a print drum, print hammers, and pulse generators. The print drum is a cylindrical drum with characters engraved on its surface. The characters are arranged in 64 rows around the surface of the drum and 120 columns across the face of the drum. One of the rows is undercut. A row contains 120 identical characters. A column contains 63 characters and a blank space (undercut). Table 3.1-7 lists the 63 characters and the space along with the corresponding octal and binary codes associated with each of the characters on the drum. The drum revolves at a speed of either 1000 or 667 rpm.

Arranged in a row perpendicular to the surface of the drum are 120 hammers. Each hammer can be driven against the drum surface when its associated solenoid is energized. Paper and inked ribbon pass between the drum surface and the print hammers. The ribbon is next to the drum and extends across its width. Each hammer solenoid is connected to a triggering circuit in the printer. The triggering circuit is connected to a line going to the control unit and energizes its associated hammer solenoid, driving the printer hammeragainst the drum. A complete line is printed each drum revolution. Mounted on the drum shaft are two pulse generators composed of variable reluctance pickups and permanent magnets. One generator, called the character pulse generator, produces 64 pulses for one drum revolution. A character pulse is generated each time a drum character row is beneath the print hammers. The other generator, called the index pulse generator, produces one pulse for one drum revolution. The index pulse occurs between the character pulses generated when the character rows on the drum, represented by octal codes 77 and 00 in table 3.1-7, are beneath the print hammers. The index and character pulses are taken from their respective reluctance pickups, amplified, shaped, and presented to the control unit.

2) Paper Feed Assembly. The paper feed assembly consists of the engine, tractors, strobe generator, and vertical format tape reader. The printer paper is held by two pairs of sprocket type feed tractors, with one tractor pair above the print hammers and the other below the print hammers. The tractors are driven by a magnetic brake-clutch type engine. The clutch on the engine is engaged when an advance paper signal is received from the control unit. The clutch stays engaged as long as this signal is present, turning the tractors that move the paper. When the advance paper signal from the control unit is removed, the clutch disengages and the brake is applied, stopping the paper feed tractors.

Mounted on the paper feed tractor drive shaft is a paper feed strobe generator composed of a variable reluctance pickup and a permanent magnet. This strobe generator produces a strobe signal for each print line advanced. The strobe signal is amplified, shaped, and sent to the control unit and the vertical format tape reader.

The vertical format tape reader is a device to control the vertical movement of the printer paper. It monitors a 1-inch wide, 8-level, punched tape, which moves one frame each time the printer paper moves one line space. The reader consists of an insulated, sprocketed, metal cylinder on which a loop of punched tape is placed. The physical length of the form (page) dictates the length of the punched sequence on the loop. Riding on the tape are eight sensing brushes, each of which is connected to a line going to the printer unit interface. A ninth brush rides on the metal cylinder outside the tape area, and is connected to the paper feed strobe generator. When one of the tape brushes makes contact with the metal cylinder through a hole in the tape, coincident with the strobe signal, the strobe signal appears on the line of the associated tape track to the printer unit interface.

In this application of the tape reader, only three tape tracks are used. These tape tracks, numbers 1, 2 and 3, pertain to the functions top of form, load paper, and bottom of form respectively. The tape has a hole punched in track 1 corresponding to the top of the form on the printed paper. Track 2 has a hold punched corresponding to the load paper mark on the paper tractors. Track 3 has a hole punched to correspond to the bottom of the printer paper form.

TALBE 3.1-7. TYPE SYMBOLS AND CODES

OCTAL CODE	BINARY CODE	CHARACTER	OCTAL CODE	BINARY CODE	CHARACTER
00	000 000	absolute value	40	100 000	)
01	000 001	arrow (up)	41	100 001	-
02	000 010	8 subscript eight	42	100 010	+
03	000 011	bracket (open)	43	100 011	
04	000 100	bracket (close)	44	100 100	=
05	000 101	space (undercut)	45	100 101	
06	000 110	A	46	100 110	equal to or less than
07	000 111	B	47	100 111	left hand brace
10	001 000	C	50	101 000	star
11	001001	D	51	101 001	(
12	001 010	E	52	101 010	equal to or greater than
13	001 011	F	53	101 011	'
14	001 100	G	54	101 100	right hand brace
15	001 101	H	55	101 101	(or)
16	001 110	I	56	101 110	,
17	001 111	J	57	101 111	#
20	001 000	K	60	110 000	0
21	010 001	L	61	110 001	1
22	010 010	M	62	110 010	2
23	010 011	N	63	110 011	3
24	010 100	O	64	110 100	4
25	010 101	P	65	110 101	5
26	010 110	Q	66	110 110	6
27	010 111	R	67	110 111	7
30	011 000	S	70	111 000	8
31	011 001	T	71	111 001	9
32	001 010	U	72	111 010	and
33	011 011	V	73	111 011	;
34	011 100	W	74	111 100	/
35	011 101	X	75	111 101	.
36	011 110	Y	76	111 110	arrow right
37	011 111	Z	77	111 111	x multiply sign

3. Functional Description. Figure 3.1-13, a functional diagram of high-speed printer, shows the various registers used by the control unit to perform its buffer operation, to print a line, and to advance paper.

a) Transfer Mode. The output request signal is sent to the computer when the equipment is ready to receive data. The computer replies with 30 data bits accompanied by an output acknowledge signal. The output acknowledge signal removes the output request signal and gates the data into the printer control units. The equipment disassembles the 30-bit word and stores it in the memory as five 6-bit characters. The printer control unit then sends another output request signal to the computer, and a second 30-bit word is received accompanied by an output acknowledge signal. After the computer has sent the high-speed printer 24 30-bit words, the memory contains 120 6-bit characters. The character counter, having been advanced to a count of 119 (167_8), places the PCU into a print mode, and no output request is raised.

b) Print Mode. The 120-character memory is scanned completely to print all similar characters in the line. The print drum advances to the next character position, and memory is scanned again to print all similar characters in that position. The process continues until all 64 characters on the print drum are compared with those in memory, thereby printing the complete line during one revolution of the drum. When the print cycle is completed, the paper advance mechanism is activated and the PCU returns to the transfer cycle to reload memory with the characters for the next line. The printer control unit then sends another output request signal to the computer, and the process is repeated. Figures 3.1-14 and 3.1-15 describe the transfer and print cycle.

The interrupt signal and interrupt code are sent to the computer when the equipment is master cleared, or when there is a malfunction in the high-speed printer. An interrupt signal and a logical 1 on the interrupt code line inform the computer that the equipment was master cleared. An interrupt signal and a logical 0 on the interrupt code line inform the computer that there was a malfunction in the high-speed printer. The computer replies in both cases with an input acknowledge signal, which tells the PCU to remove the interrupt signal and interrupt code to the computer.

c) Paper Movement Mode. At the end of the print mode, the main timing sends an end-of-print mode signal to the paper feed control. This signal causes the paper feed control to send an advance paper signal to the printer unit, which causes the paper to start moving. When the paper has moved one line distance, the printer unit sends a strobe signal to the paper feed control. This signal removes the advance paper signal, stopping paper movement.

There are three exceptions to the normal movements of paper:

1) If the computer sends an external function signal accompanied by a logical 1 on the 2^0 data line, the paper is not allowed to advance after a line is printed. This condition is removed when the computer sends an external function signal accompanied by a 1 on the 2^1 data line;

2) An external function signal accompanied by a 1 on the 2^4 data line causes the paper to move to the top of the paper form after a line is printed;

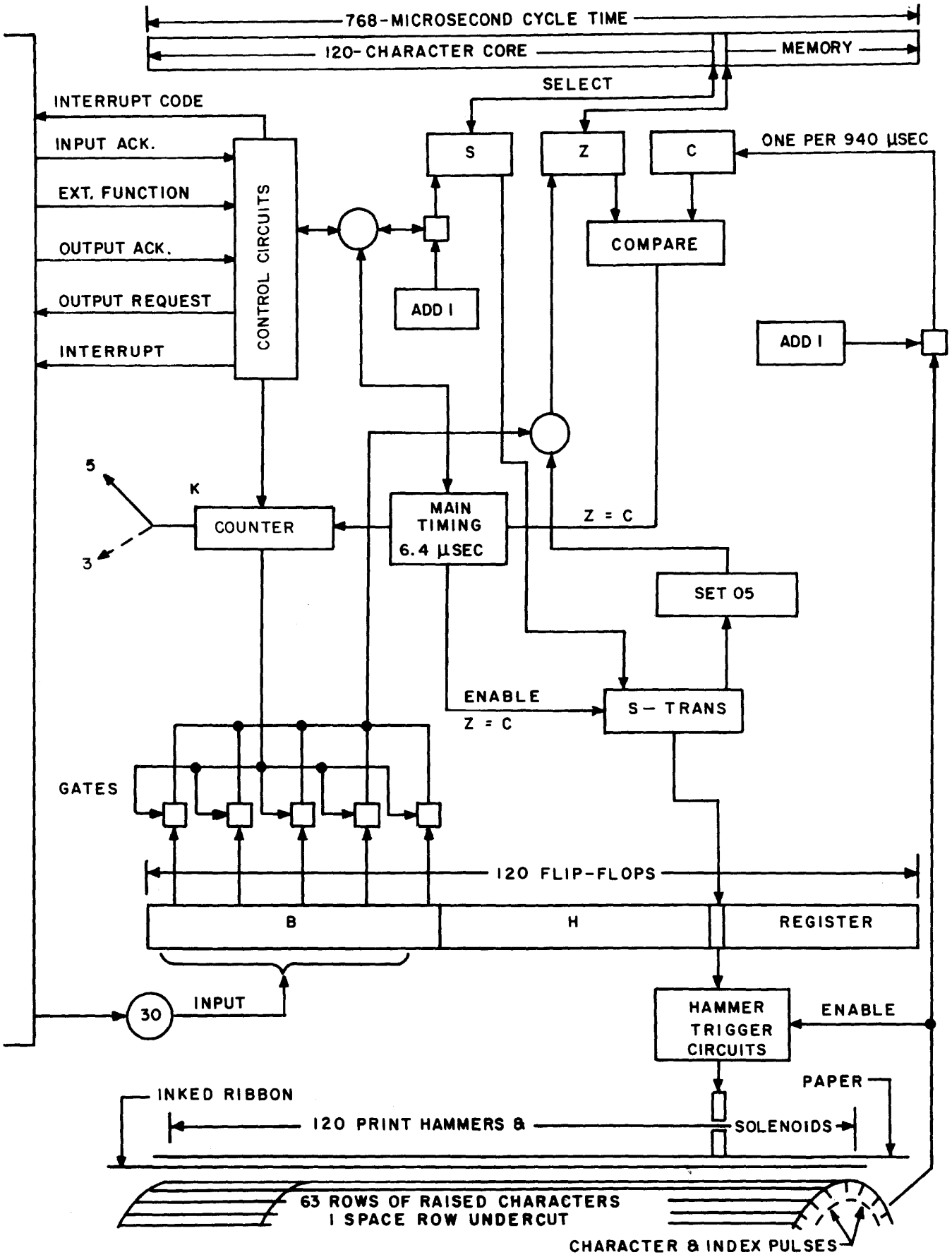


Figure 3.1-13. Functional Diagram, High Speed Printer

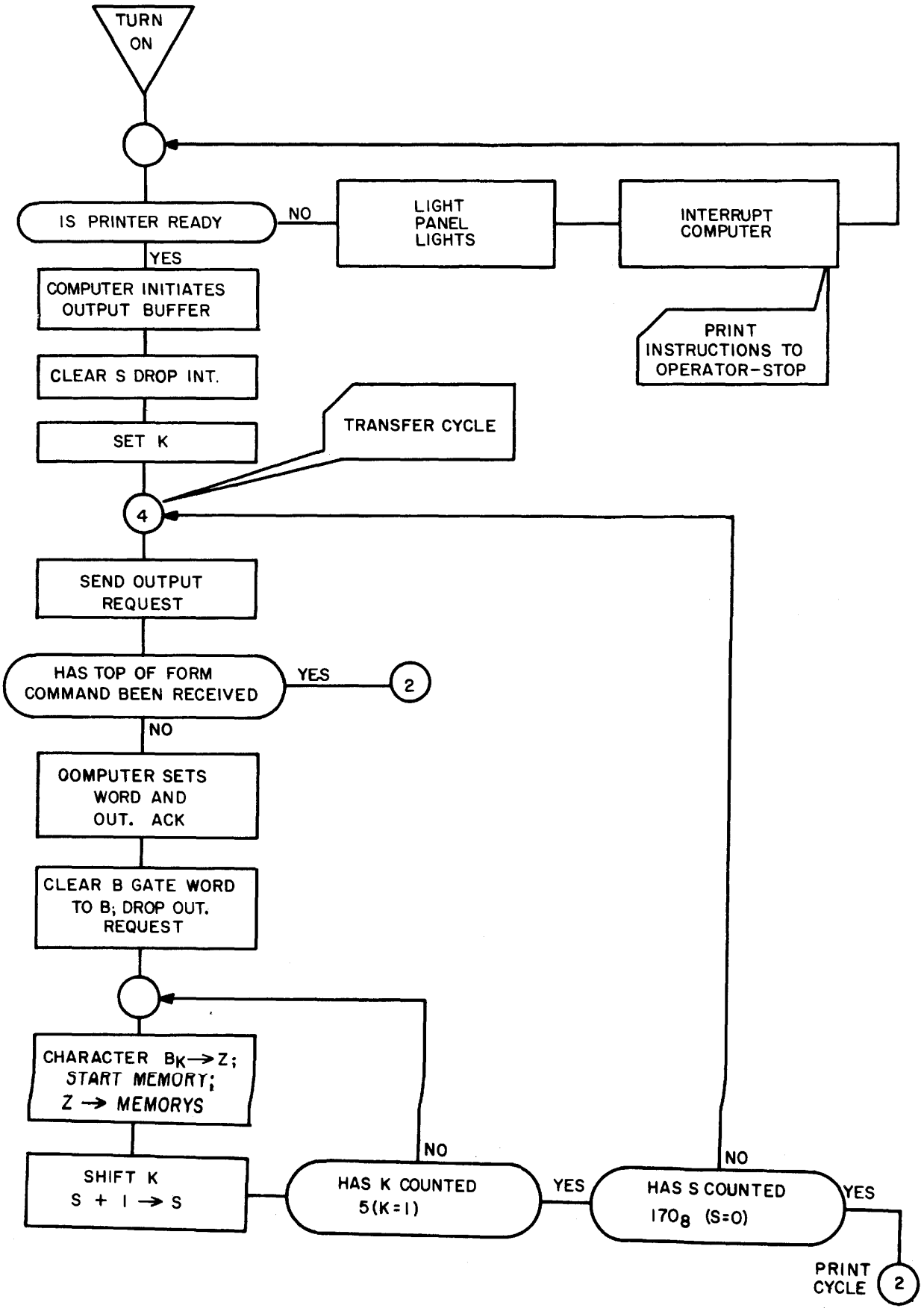


Figure 3.1-14. Transfer Cycle

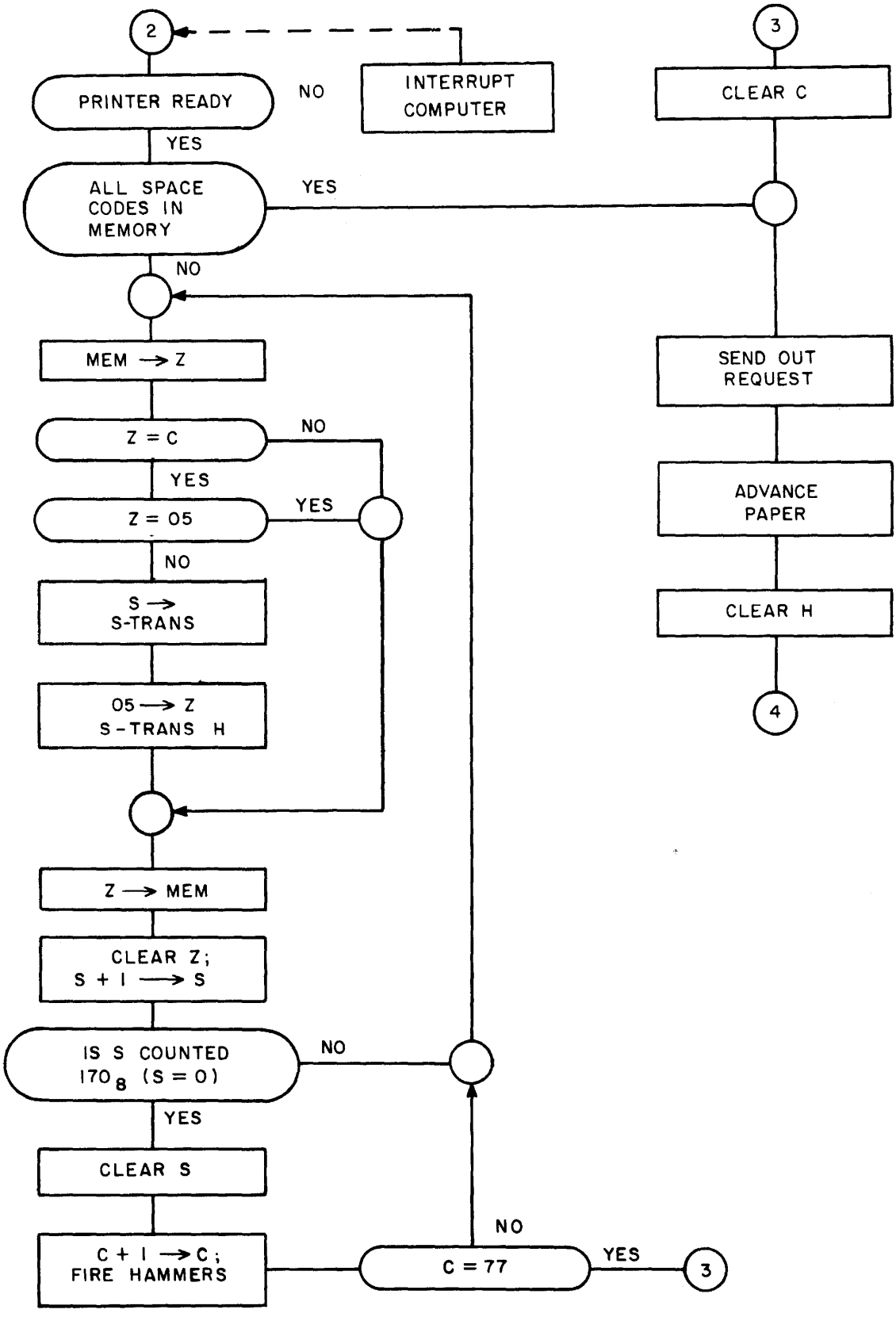
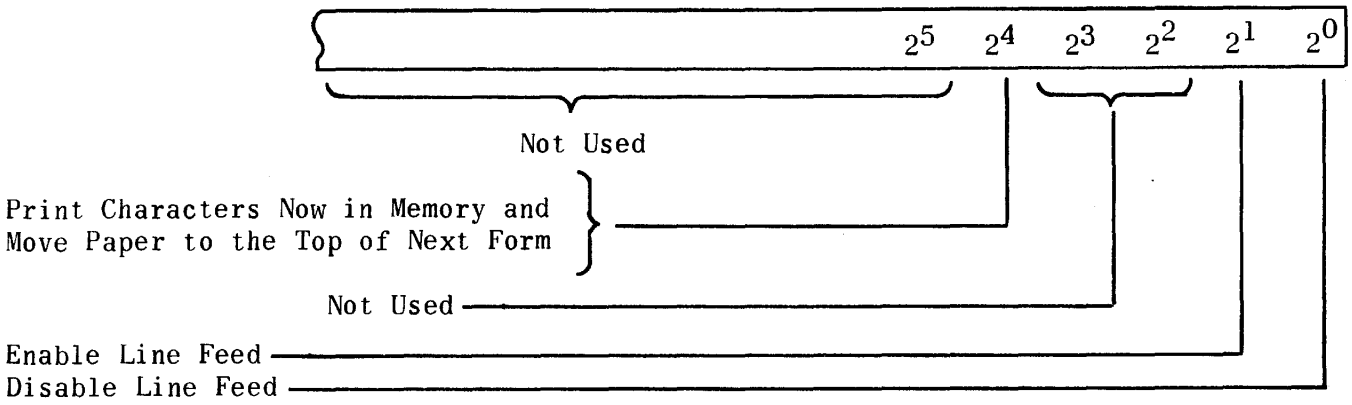


Figure 3.1-15. Print Cycle

3) If the computer sends a line of spaces, that is, 120 space code characters (octal 05's), the paper feed control disregards the strobe signal from the printer unit and keeps the paper moving another line feed.

d) Computer To Printer Commands. The high-speed printer will accept and respond to three external commands from the computer. Three bits of the external function word select the operations to be performed. If raised to the one state, the commands are as follows:



The printer control unit, responding to the disable line feed, inhibits the paper advance circuitry. Master clear sets the printer to the enable line feed condition. The top of form command positions the paper so that the type line is at the top of the next form. If this command is sent when the printer is in the disable line feed command, the paper will not move.

e) Registers and Their Functions. The memory is a 120 6-bit character core storage in which incoming data are stored during the transfer cycle. When 120 characters are stored, the control unit enables the print cycle which prints the 120 characters (described by the codes stored in memory) on one line. One memory cycle is defined as the loading or the reading of all 120 character locations in memory. Thirty-bit computer words are stored in the B-register and transferred to memory as shown in figure 3.1-16.

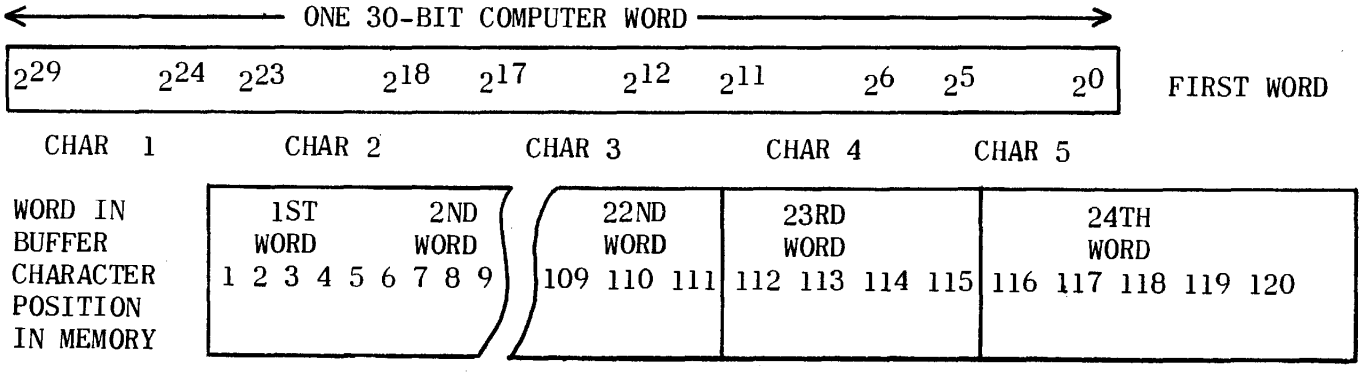


Figure 3.1-16. Printing Order

The Z register is a 6-bit transient register used as a memory buffer. All data entering or leaving memory must pass through this register.

The S register is a 6-bit counter and memory address register that is incremented by one each time a character position in memory is addressed. It selects the core location for storing the character in the Z register during the transfer cycle, and it selects the core location of the character in memory for transferring to the Z register during the print cycle.

The B, H register (B = 30 bits and H = 90 bits) together form the 120-bit printer hammer control register. During the print cycle, each set position enables a corresponding print hammer to strike the paper against the print drum. The B register has a twofold function. During transfer cycle, the output word (from the computer or magnetic tape unit) is gated into B. A 30-bit word is disassembled into five 6-bit characters each transferred to the Z register for sequential storage into memory as dictated by S.

The K counter is a 5-count distributor register. Each count allows six data bits to pass from the B register to the Z register. The 30-bit or 18-bit interface is controlled by a switch on a logic chassis at the rear of the cabinet. This switch enables or disables the most significant 12 bits of the B register and sets the K counter to a five or three count limit respectively.

The C register is a 6-bit counter that cycles through its count (000000_2 to 111111_2) once during a print cycle (one rotation of print drum). During each count (corresponding to the code of the next character on the drum to be printed), the S register is advanced through all memory locations (120_{10}). As each character designated by the address in the S register is transferred to the Z register, it is compared to the code in C.

The S translator, when enabled by main timing and upon C equals Z comparison, sets the flip-flop in the H register corresponding to the count in S. The H-bit fires the print hammer in the line position that corresponds to the character position in memory. The translator also sets the space code (05) in the Z register for the write portion of the memory read/restore cycle.

All locations which contain character codes equal to (C) will cause the S translator to set corresponding bit positions in the H register, thereby printing those characters. After the hammers are fired, the drum character pulse advances the C register, preparing it for the comparison with each memory cell during the next memory cycle. The index pulse from the drum surface resets the C register at the end of one revolution.

As each character from memory makes a positive comparison with (C), the Z register is forced to an 05 (space) code, and this information is stored back into the cell rather than its original code. If the comparison is negative, the Z register remains and the original code is stored back into the cell. At the end of each memory cycle (120 character positions read), all character positions will contain space codes.

4. Manual Controls and Adjustments.

a) Printer Adjustments. The following adjustment knobs are provided on the printer; Vertical Position, Character Phasing, Paper Tension, Penetration Control.

1) Vertical Position: controls the vertical positioning of the paper with relation to the printed line. The maximum adjustment is ± 0.25 inch.

2) Character Phasing: adjusts the index and character pulse time for firing the hammers on the character and compensates for print wheel speed. The maximum adjustment is 15 degrees.

3) Paper Tension: controls the paper tension by varying the distance between corresponding feed pins on the upper and lower feed tractors. The maximum adjustment is 0.125 inch.

4) Penetration Control: provides fine adjustment of the spacing between the print hammers and the print wheel in order to vary the density of print and to accommodate different paper thicknesses.

b) Control Panel Operations. Various operations are initiated from the control panel (see figure 3.1-17) by the following indicators and switches:

1) POWER ON indicator switch: initiates the power sequencing cycle and indicates completion of the cycle.

2) POWER OFF indicator switch: turns ac power off. The switch lamp is illuminated when ac power is available at the output of the main circuit breaker, and a power sequencing cycle has not been completed.

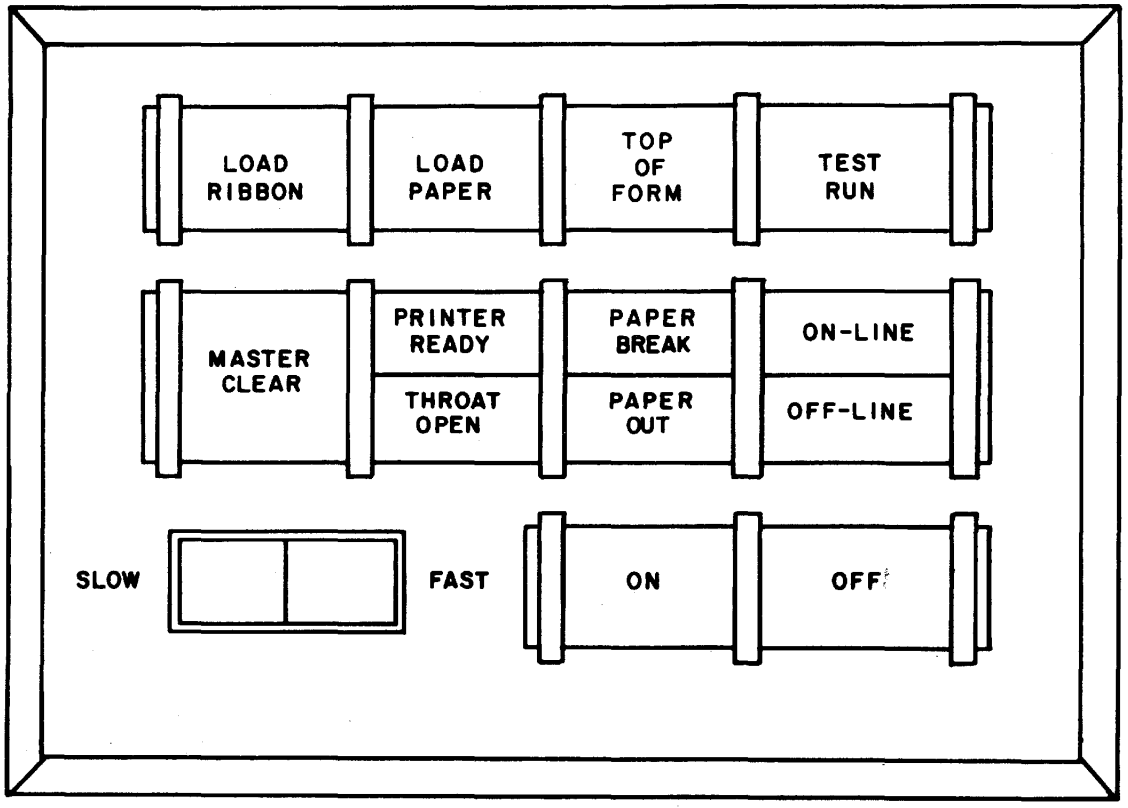


Figure 3.1-17. Operator's Control Panel

3) OFF-LINE/ON-LINE indicator switch: controls the logical connection of printer to computer. When the printer is master cleared and ON-LINE is selected, an interrupt is sent to the computer.

4) SLOW/FAST switch: a rocker type switch that changes the print wheel speed from 1000 to 667 rpm.

5) LOAD PAPER switch: positions the paper feed assembly so that paper can be loaded conveniently.*

6) LOAD RIBBON switch: re-spools the inked ribbon prior to its replacement. This operation is not available while the unit is printing.

7) TEST PRINT switch: actuates a test cycle under local control that utilizes a pattern which prints all characters in all column positions. When the PATTERN SELECT switch on the control chassis is in normal position, the test cycle causes all 63 characters to appear in all column positions. In any given line, one character will appear in all the 120 character positions, and after 63 lines of print, a line of each character will have been printed. When the PATTERN SELECT switch is in the M position, all M's are printed. When it is in the E position, all E's are printed. The paper advances one line after each line of print. The operator presses the TEST PRINT switch to terminate the test operation.*

8) TOP OF FORM switch: advances the paper so that the top of the next form is positioned at the type line.*

9) MASTER CLEAR switch: clears the circuit logic, sets enable line feed, sets the interrupt, and places the printer OFF-LINE.

10) PAPER OUT indicator: indicates that the paper supply is exhausted. This situation is detected 22 inches ahead of the type line when the throat cover is closed.

11) PAPER BREAK indicator: indicates absence of paper above the print station.

12) THROAT OPEN indicator: indicates that the printer throat cover is not in position. A switch is available for over-riding the throat open indication.

13) PRINTER READY indicator: indicates that the printer is ready for operation. The indicator goes out when any of the following conditions exist:

   a. The printer is out of paper;

   b. The paper is torn or incorrectly positioned in the feed tractors;

* This function is available for off-line operation only.

c. The throat cover is not in position;

d. There is no ac power input;

e. A main circuit breaker is open;

f. A fuse is blown.

c) Control Chassis Operations.

1) 1206/1218 Switch: provides interface and control logic change to either a 30-bit word or an 18-bit word.

2) Pattern Select Switch: selects one of three patterns that can be used in the test print operation:

a. Normal Position - prints all characters in the order they appear on the print wheel.

b. M Position - prints all M's

c. E Position - prints all E's

5. Programing Consideration. Certain communicating functions are of importance to the programer. Some functions inform him of the operational state of the printer, and others allow him full command of the printing operation.

a) Processing Interrupts. Before the printer is initially put on-line, the printer-handling routine should be ready to process either of two types of interrupts, or a fault condition will result in the computer. Processing interrupts requires the interpretation of interrupt codes.

A start interrupt (code line is set) informs the computer that the printer is operable. The start interrupt should be used to initiate the printer-handling program. A command (via external function) to the printer will not be honored when it is not operable. The first command to the printer is usually a top of form instruction, and if this is lost, data in an output buffer may be printed in the wrong position of the form.

Malfunction interrupts (code line is cleared) indicate that the printer is not operable. This interrupt is sent from the printer to the computer when any of the following malfunctions exist:

Printer is out of paper;
Paper is torn or incorrectly positioned in the feed racks;
Throat cover is not in position;
There is a loss of ac power input;
A main circuit breaker is open;
A fuse is blown.

The malfunction condition will destroy some data except in the case of the paper out condition. Such destruction of data will require a repetition of some or all of the previous output operation. When the printer is out of paper, printing will terminate

at the bottom of the last form. The program can continue its output without loss of data when a new supply has been loaded and the paper has been positioned at the top of a form.

The programer may desire to handle malfunction interrupts by providing a means for operator decisions concerning the continuation or repetition of output. Here again the programer should utilize the start interrupt to restart the printer-handling program.

b) Printing Format. Horizontal character format is controlled by the program by loading the data into the output buffer in the desired character positions. Space codes are placed in the buffer where blanks are desired in the printed line.

Vertical line spacing is controlled by commands to the printer via the computer external function instruction in conjunction with the paper movement control tape in the paper advance mechanism. Single line spacing is automatic in the printer upon completion of a print cycle. Extra line spacing is accomplished by initiating an output buffer of 120 space codes. Line overprint is accomplished by following the initial line print buffer (within 768 microseconds) with a command to disable line feed and initiating another output buffer of a line of overprint data. Overprint data buffers must contain space codes in positions previously printed as meaningful characters on the line. The command to enable line feed must be received by the printer control unit before or during the last overprint cycle to move the paper into position for the next line.

The top of form command forces the printer control into the print cycle if other than space codes are stored in its memory. The line is printed even though a portion of a record is transferred. If the line feed is enabled, the paper is advanced to the top of the next form. If the line feed is disabled, the paper does not move.

c) Assembly of Buffers. The printer will not print a line of data until 120 character codes have been received, or if less than 120 characters, a top of form command is received.

To print a line of less than 120 characters of intelligence and advance the paper one line space, the balance of the 120-character buffer must be filled with space codes. Figure 3.1-18 is a pictorial diagram of an assembled output buffer and its resulting printed line.

BUFFER WORD ORDER	CONTENTS OF CORE MEMORY IN BUFFER				
1	31	15	12	05	32
2	23	16	33	06	10
3	05	61	64	66	71
4	05	15	16	14	15
5	05	30	25	12	12
6	11	05	25	27	16
7	23	31	12	27	05
8	10	06	23	05	25
9	27	16	23	31	05
10	32	25	05	31	24
11	05	66	66	67	05
12	24	27	06	61	60

BUFFER WORD ORDER	CONTENTS OF CORE MEMORY IN BUFFER				
13	60	60	05	21	16
14	23	12	30	05	24
15	13	05	06	21	25
16	15	06	23	32	22
17	12	27	16	10	05
18	10	15	06	27	06
19	10	31	12	27	30
20	05	25	12	27	05
21	22	16	23	32	31
22	12	75	05	22	32
23	21	31	16	10	24
24	25	16	12	30	75

LINE WORD ORDER

1 2 3 4 5 6 7 8 9 10 11 12 13 14
THE UNIVAC 1469 HIGH-SPEED PRINTER CAN PRINT UP TO 667 or 1000 LINES OF
15 16 17 18 19 20 21 22 23 24
ALPHANUMERIC CHARACTERS PER MINUTE. MULTICOPIES.

Figure 3.1-18. Buffer and Printed Line