INTRODUCTION

A Programmer's Guide to the X-6 Assembly System is concerned with the preparation of a data processing program for the X-6 assembly on a USS 80 or 90 Tape System. For the most part, this consists of the coding of the object program according to X-6 symbolic and relative coding conventions and the preparation of the punched card input deck to be processed by the X-6 Assembly System program. Such preassembly preparations are covered in detail. An understanding of the reasons for these preparations, however, is only possible through a general knowledge of the processing steps during the actual assembly by the X-6 system. For this purpose, a general description of the X-6 processing has been included. The details of the processing can be found in the flow charts of the X-6 Assembly System.

Most of the examples used are applicable to both the USS 80, 80 Tape, and 90 Tape computers. Some, however, are inimical to one computer (for example, three part alphabetics and interlaces).

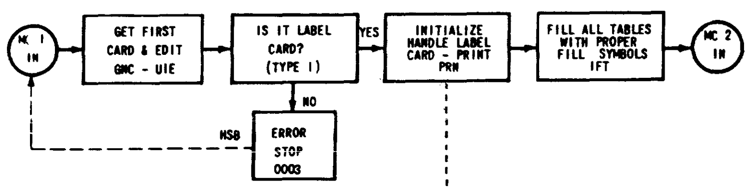
Much of the description and terminology used in this manual presupposes that the reader has a general knowledge of machine coding and operation of the USS 90/80 computers.

TABLE OF CONTENTS

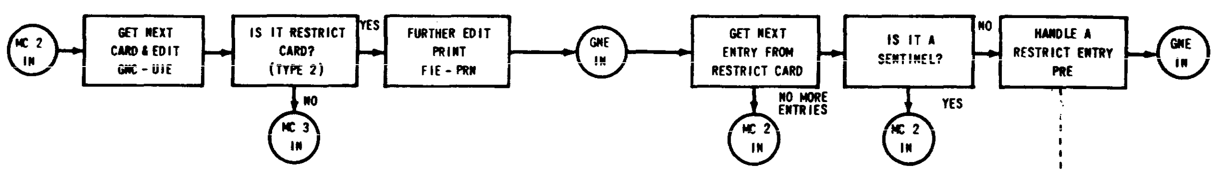
GENERAL DESCRIPTION

When the X-6 coding of a data processing program or operation has been completed, this coding, and any further information required by the X-6 Assembly System for the processing of the coding, is punched on appropriate input card types. These input cards are then placed in a specific order in the input deck and the actual assembly is begun.

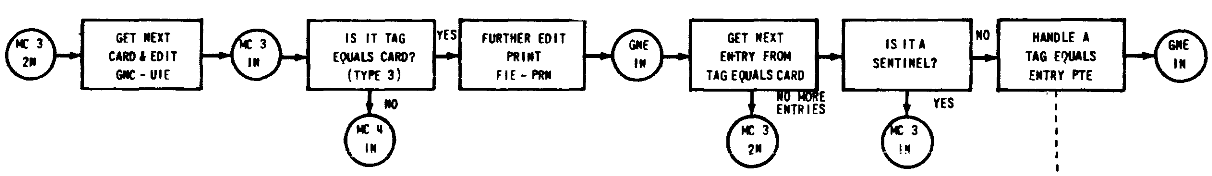
Each card type will be processed in a specific way:



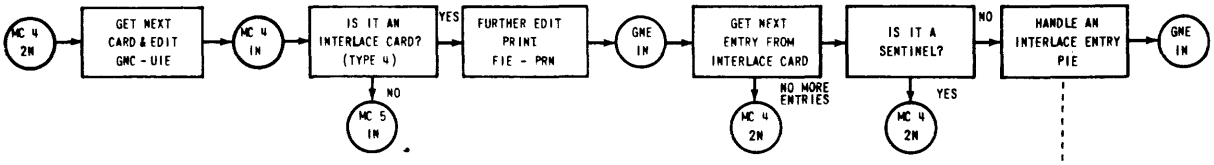
The fields of the label card are placed in the output interlaces without modification.



Restrict card entries are used to mark off locations in the storage availability table. No restricted location will be assigned when absolute addresses are generated.



Tag equals card entries are filed in internal tables equated to their absolute addresses. The absolute addresses are used to mark off locations in the storage availability table.



Interlace card entries are used to mark off interlace positions in the storage availability table. The origins are filed for future use.

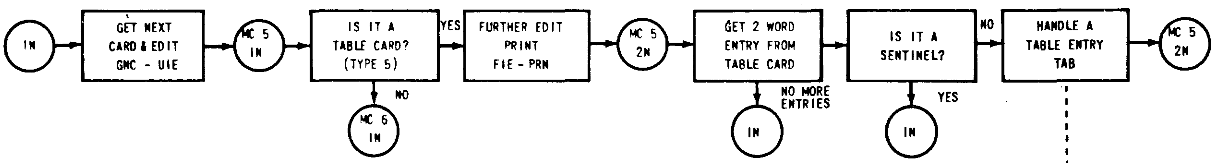
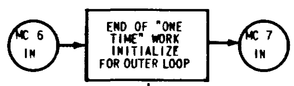
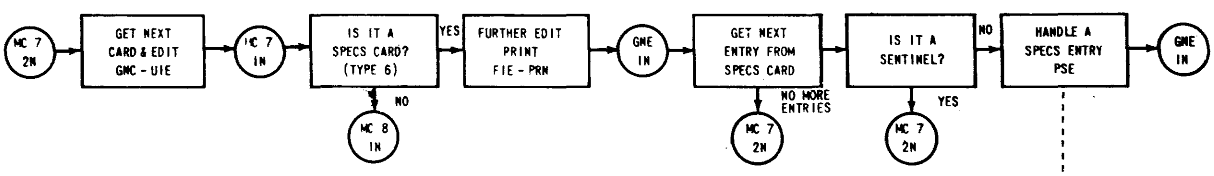


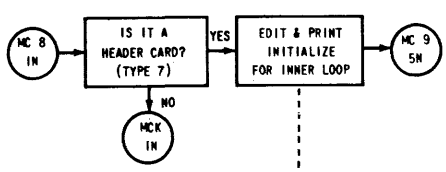
Table card entries are also used to mark off positions in the storage availability table. Increments and origins are filed for future use.



Card types 1 through 5 must be received by the X-6 Assembly System in order. After the Label card has been processed (MC 1 IN), if a card is received that is not a type 2, 3, 4, or 5, the assumption is made that all the above processing has been accomplished.



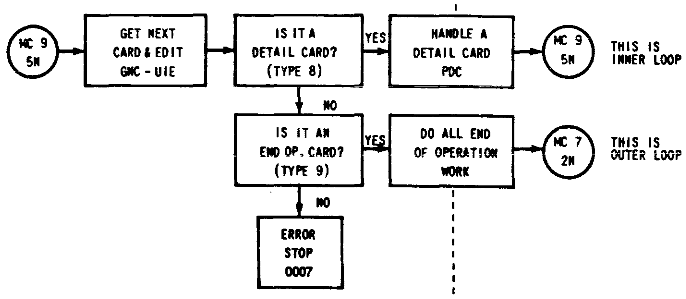
Specifications card entries are filed in tables for direct substitution later.



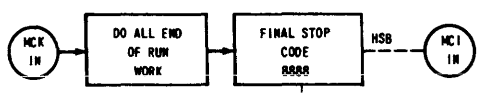
Operation Header card entries are placed in the output interlaces. Initial conditions are set for Detail card processing (MC 9 5N).

Detail cards contain the instruction lines and constants of a program. Only Detail card processing will produce output punching. The four basic steps in Detail card processing are:

1. Handle the a address.
2. Analyze the instruction code and separate instructions from constants. For instructions, obtain a code word to control further processing by the use of the necessary increment needed between the a, m, and c addresses and substitute the computer code equivalent of the mnemonic code. Determine if one or both of the m and c addresses are significant.
3. Handle the m address if necessary.
4. Handle the c address if necessary.



End operation card signals the end of a group of Detail cards.



End input card signals the last card of the program being assembled. It contains the instruction to be used by the loading routine to start execution of the assembled program.

X-6 INSTRUCTION CODES

X-6 Mnemonic Code			Computer Code	Minimum Word Times	Function
Arithmetic					
ADD	m	c	70	5	Add (m) to (rA). If overflow, next instruction is c+1.
SUB	m	c	75	5	Subtract (m) from (rA). If overflow, next instruction is c+1.
MUL	m	c	85	105	Multiply (rL) by (m).
DIV	m	c	55	115	Divide (m) by (rL). If overflow, next instruction is c+1.
Transfer					
LDA	m	c	25	4	Load rA: (m) ⟶ rA.
LDX	m	c	05	4	Load rX: (m) ⟶ rX.
LDL	m	c	30	4	Load rL: (m) ⟶ rL.
STA	m	c	60	4	Store rA: (rA) ⟶ m (m cannot be register address.)
STX	m	c	65	4	Store rX: (rX) ⟶ m (m cannot be register address.)
STL	m	c	50	4	Store rL: (rL) ⟶ m (m cannot be register address.)
ATL	-	c	77	3	(rA) ⟶ rL.
CTA	m	-	23	3	(rC) ⟶ rA.
CAA	m	-	36	3	Clear rA to zeros: Ø ⟶ rA. Original sign remains.
CLA	m	-	26	3	Clear rA to zeros: Ø ⟶ rA. Sign +.
CLX	m	-	06	3	Clear rX to zeros: Ø ⟶ rX. Sign +.
CLL	m	-	31	3	Clear rL to zeros: Ø ⟶ rL. Sign +.
CAX	m	-	86	14	Clear rA and rX to zeroes. Sign of rL goes to rA and rX.

X-6 Mnemonic Code	Computer Code	Minimum Word Times	Function
Translate			
CTM - c	12	3	Translate card to machine (computer) code: 80CC (rA, rL, rX)⟶MC-6 (rA, rX); Ø⟶rL.
MTC - c	17	3	Translate machine (computer) to card code: MC-6 (rA, rX)⟶80CC (rA, rL, rX).
TXM - c	C3	3	Translate XS-3 code to machine (computer) code: XS-3 (rA)⟶MC(rA).
TMX - c	C1	3	Translate machine (computer) code to XS-3 code: MC(rA)⟶XS-3 (rA).
Index Registers			
LIR m c Absolute Address	02	3	Load index register: m portion of instruction word⟶rBi.
IIR m c	07	4	Increment Index Register: m portion of instruction word +(rBi)⟶rBi and to m portion of rA; Ø⟶ balance of rA.

Note: When either an LIR or IIR instruction is used, the m address portion <u>must</u> be an absolute address.

Comparison			
TEQ m c	82	3	Test (rA) and (rL) for equality: If =, next instruction at m. If ≠, next instruction at c.

TGR m c	87	3	Test (rA) and (rL) for magnitude:
			If (rA) > (rL), next instruction at m.
			If (rA) ≤ (rL), next instruction at c.

X-6 Mnemonic Code			Computer Code	Minimum Word Times	Function
Logical					
BUF	m	c	20	4	Superimpose (m) on (rA)⟶rA.
ERS	m	c	35	4	Extract (m) from (rA)⟶rA.
SHR	ΔΔΔnn	c	32	3+nn	Shift right nn places: (rA)⟶(rX)⟶rA. nn is number of places to be shifted within range 00 through 10.
SHL	ΔΔΔnn	c	37	3+nn	Shift left nn places: (rA)⟵Ø. nn is number of places to be shifted within range 00 through 10.
ZUP	-	c	62	4	Zero suppress commas and zeros. MC-6 in rA, rX.
JMP	m	-	00	2	Jump to m.
STP	m	c	67	-	Stop. m or c is alternative. next instruction (requires manual intervention).
High-Speed Printer					
PBT	m	c	27	3 if c. 4 if m.	Printer test. If printer free, next instruction at m. If printer is not free, next instruction at c.
PFD	ΔΔΔnn	c	16	4	Advance nn lines. nn is within the range 00 through 79. If abnormal operation of HSP, next instruction is c+1.
PRN	Py0nn	c	11	592	Advance and print. y=Print interlace (0 through 9). nn=number of lines to advance (Δ0 through 79). If abnormal operation of HSP, next instruction at c+1.

X-6 Mnemonic Code	Computer Code	Minimum Word Times	Function
High-Speed Card Reader			
HBT m c	42	3 if c. 4 if m.	HSR buffer test: if buffer loaded, next instruction at m; if buffer not loaded, next instruction at c.
HBU Hn00d c	96	203 if d=0. 215 if d=1.	HSR buffer unload. n=HSR Interlace (0 through 9). d=0 if no automatic translation. 1 if automatic translation.
HCC m c	72	3 if c. 4 if m.	HSR card cycle. If HSR interlock, next instruction at m. If HSR not interlocked, next instruction at c. If abnormal operation of HSR, next instruction is c+1.
HSS ΔΔn00 c	47	3	HSR stacker selection. n=stacker 0, 1, or 2.
Read-Punch Unit			
RBT m c	22	3 if c. 4 if m.	RPU buffer test. If buffer loaded, next instruction at m. If buffer not loaded, next instruction at c.
RBU Rn00d c	46	203 if d=0. 215 if d=1.	RPU buffer unload. n=RPU input interlace (0 through 9). d=0 if no automatic translation. 1 if automatic translation.
RCC On00d c	81	203 if d=0. 215 if d=1.	RPU card cycle. n=RPU output interlace. (0 through 9). d=0 if no automatic translation. 1 if automatic translation.

Mnemonic Code	Computer Code	Minimum Word Times	Function
			If abnormal operation of RPU, next instruction is c+1.
RSS - c	57	3	RPU select Stacker 1.
Magnetic Tape			
TST m c	C2	3 if c. 4 if m.	Test servo availability. If servo free, next instruction at m. If servo not free, next instruction at c.
TBL xn000 c	C6	205	Tape buffer load. x=T or Z. n=Tape interlace (0 through 9).
TBT m c	C7	3 if c. 4 if m.	Test tape buffer. If buffer not available, next instruction at c. If available, next instruction at m.
TRW ΔΔxy0 c	F2	600 ms.	Rewind tape to first block condition. x=servo number (0 through 9). y=0 if rewind without interlock. 2 if rewind with interlock.
TBU xn000 c	F6	205	Tape buffer unload. x=T or Z. n=Tape interlace (0 through 9). If abnormal operation of tape, next instruction is c+1.
TRD ΔΔxyz c	G2	17	Read one block from servo x into tape buffer band. x=servo number (0 through 9). y=0 if USS mode. 5 if UNIVAC mode.

X-6 Mnemonic Code	Computer Code	Minimum Word Times	Function
			z=direction and gain: 0=forward normal. 1=forward low. 2=forward high. 5=backward normal. 6=backward low. 7=backward high.
TWR xy0 c	H2	17	Write one block from the tape buffer band onto the tape. x=servo number (0 through 9). y=mode and density. 0=USS 250 cpi. 5=UNIVAC 250 cpi. 6=UNIVAC 125 cpi.

PRINTED EQUIVALENTS FOR ALPHA-NUMERIC COMPUTER CODES

X-6 Mnemonic Code	Computer Code	Printed Equivalents
TST	C2	)2
TBL	C6	)6
TBT	C7	)7
TRW	F2	(2
TBU	F6	(6
TRD	G2	;2
TWR	H2	'2
TXM	C3	)3
TMX	C1	)1

ADDRESSING

The X-6 Assembly System will generate absolute a, m, and c addresses with optimal latency address development. In the assembly of a program, however, it may be necessary to establish certain relationships between data being assembled and data that has already been assembled or that will be assembled. The program is coded in small segments, termed "operations", with each of the operations coded by one or more programmers. To assemble these operations, X-6 instructions must be coded in such a way that the relation of each operation to any other is taken into account. It may also be that certain routines such as 90/80 HSR and RPU routines which already occupy fixed locations will be used with the program. Such routines must be referenced in absolute notation only and the assembly system must be restricted from assigning any of the fixed locations.

Various methods of addressing that relate lines and operations or that restrict the generation of addresses may be used. In a general sense, these methods come under the headings of Instruction Addressing and Data Addressing.

I. INSTRUCTION ADDRESSING

A. Space Addressing

Space addressing relates two successive lines of coding. It cannot relate one line of coding with another line separated from it by any intervening coded lines.

When the a, m, or c address of an X-6 instruction is filled with spaces, these spaces will have one of several meanings:

1. Following any instruction code that requires an m and c address, spaces in these portions will be interpreted:

Portion	Meaning
c	The next instruction to be executed is in the next line of coding. Therefore, the address generated for and assigned to this c will be identical to the a address assigned to the next line.
m	A computer operation is to be performed on the word in the next line of coding; or, the next instruction to be executed is in the next line of coding. Therefore, this m will be identical to the a address assigned to the next line.

2. When an instruction code requires only an m or only a c address, the portion not used may be filled with spaces or any other characters without affecting the program.

When using space addressing, certain restrictions must be observed:

1. Spaces cannot be used in both the m and c addresses of an instruction unless the instruction requires only an m or c address. If spaces are used when the instruction requires both an m and c address, the spaces in the m portion will be assumed to be in error and an error code will appear when the X-6 listing is printed out during assembly.

2. When an m or c address necessary to the instruction is space filled, the next line must contain spaces in the a address. If the a address in such a case does not contain spaces, it will be processed correctly but the line with spaces in the m or c address will not. When the X-6 listing is printed during assembly, an error code will be printed with the line containing the a address to indicate that the previous line must be recoded.

Examples of Space Addressing:

a	Op	m	c	Remarks
ΔΔ553	LDA	Δ4211	ΔΔΔΔΔ	This c and the next a address will be the same.
ΔΔΔΔΔ	LDX	Δ4216	ΔΔΔΔΔ	This c and the next a address will be the same.
ΔΔΔΔΔ	CTM	ΔΔΔΔΔ	ΔΔΔΔΔ	This m is ignored; this c and the next a address will be the same.
ΔΔΔΔΔ	LDA	ΔΔΔΔΔ	Δ4211	This m and the next a address will be the same; the contents of the next coded line will be loaded in rA. The next instruction is in the coded line with 4211 in the a address.
ΔΔΔΔΔ	ΔΔΔ	00000	Δ0001	
Δ4211	STA	Δ4215	ΔΔΔΔΔ	The contents of rA will be stored in 4215. This c and the next a address will be the same.

a	Op	m	c	Remarks
ΔΔΔΔΔ	JMP	ΔΔΔΔΔ	ΔΔΔΔΔ	This c is ignored; this m and the next a address will be the same.
ΔΔΔΔΔ	TEQ	ΔΔΔΔΔ	Δ4630	This m and the next a address will be the same. When the assembled program is used, if the result of the test is equality, the next instruction will be at the address generated for the m address; if inequality, the next instruction will be at location 4630.

B. Tag Addressing

A tag is a symbolic address that relates one non-successive line of coding with another and may be either a temporary or permanent tag. It may be used for an entrance to or an exit from common subroutines, to transfer control to a common line at the end of a branching chain of instructions, to transfer from one operation to another, or to reference lines that may be modified.

A temporary tag refers only to lines within the same operation in which it occurs. When a tag is referenced by more than one operation (that is, when it is referenced by lines within other operations than the one in which it occurs) it is a permanent tag.

To conserve the memory space used during an X-6 assembly, a table is kept of each type of tag. The tag identifier and the address assigned to it are entered in the appropriate table. When an operation has been processed, the temporary tag table is erased so that the temporary tags of the next operation to be assembled may be stored in those same table locations. The permanent tag table is not erased (thus permitting communication between operations).

1. Permanent Tags

A permanent tag is coded by using all five digits of the X-6 symbolic address:

Digits	12345
Symbolic Address	PPPPm

PPPP (Digits 1-4) identifies a permanent tag and may be composed of alphabetic and/or numeric characters. Since identification depends on the use of these digits (plus m), the first digit cannot be Δ or 0.

m (Digit 5) specifies the memory area the tagged line is to be assigned, or it may refer to an overflow or c+1 condition (see Overflow Addressing, below).

In either case, m must be one of the following:

N for Normal Access memory assignment.

F for Fast Access memory assignment.

0 or P for overflow condition.

When assigning permanent tags, the following should be observed:

a. No more than 300 permanent tags can be used in each program.

b. Permanent tags may be assigned to a specific memory location by the use of a Tag Equals Card, Card Type 3 (see Input Card Section, below).

c. The identifier of the tag (digits 1-4) is arbitrary. It is recommended that a meaningful tag coding scheme be developed for each program. This may be found useful after assembling the X-6 Instruction Deck in checking the X-6 listings.

d. An overflow line should be given a permanent tag if the overflow subroutines referenced are used by more than one operation.

Examples of Permanent Tag Coding:

Coding	Remarks
ΔΔΔΔΔ LDA ASINF ΔΔΔΔΔ	Load rA with the line whose a address is ASINF.
ΔΔΔΔΔ ADD K0015 STINF	The constant in K0015 is added to the contents of ASINF. Control is sent to the line whose a address is STINF.

		Remarks
STINF	STA ASINF A124F	Restore ASINF; transfer to line A124F.

2. Temporary Tags

A Temporary Tag is coded by using three of the five digits of the X-6 symbolic address:

Digits	12345
Symbolic Address	ΔΔttm

tt (Digits 3-4) identifies a temporary tag and may be composed of alphabetic and/or numeric characters. Digit 2 may also be used as part of the tag identifier; however, only digits 3-4 will be processed.

m (Digit 5) specifies the memory area the tagged line is to be assigned, or it may refer to an overflow condition (see Overflow Addressing, below). In either case, m must be one of the following:

N for Normal Access memory assignment.

F for Fast Access memory assignment.

O or P for overflow conditions.

When assigning temporary tags, the following should be observed:

a. No more than 50 temporary tags can be used in each operation.

b. It is not possible to assign absolute locations to temporary tags.

c. The identifier of the tag (digits 3-4) is arbitrary. However, to make certain that no more than 50 temporary tags are assigned in any operation, it is recommended that such tags be coded by numbers 01 through 50.

d. Temporary tags cannot be referenced within any operation except the one in which they occur.

Example of Temporary Tag Coding:

Coding	Remarks
ΔΔ11N LDA W0005 ΔΔΔΔΔ	Page/Line counter to rA.
ΔΔΔΔΔ LDL K0012 ΔΔΔΔΔ	Constant: 00 0000 0030
ΔΔΔΔΔ TEQ ΔΔ12N ΔΔΔ8N	Are they equal?
ΔΔ12N CLA ΔΔΔ8N ΔΔΔΔΔ	Zeros into rA.
ΔΔΔ8N STA W0005 ΔΔΔ1N	Zeros into Page/line counter; transfer to the beginning of this operation.

C. Overflow Addressing

Overflow, a c+1 condition, can result from either an arithmetic operation or an abnormal condition in an input or output unit. In an arithmetic operation, it is caused by the generation of a quantity beyond the capacity of the register which is to receive it. In an input or output unit, it may be due to any of a number of mechanical conditions (HSP out of paper, RPU card jam, for example). In either case, the instruction to be executed in the program is determined by the addition of 1 to the c portion of the instruction in which the overflow condition occurred.

There are eight X-6 instruction codes that can result in overflow conditions: ADD, SUB, DIV, RCC, HCC, PRN, PFD, TBU. Whenever one of these codes is used, a subroutine should be coded that will handle the possible overflow condition. In X-6 coding, this is accomplished by the use of temporary or permanent tags with an O̲ or P in the fifth digit position. The tag with the O̲ is placed in the c address of the instruction in which overflow may occur. If there is no overflow, control will be sent to the line with the O̲ tag in the a address portion. If overflow does occur, control will be sent to the line with the P tag in the a address portion. Thus, when the following instruction is assembled:

	Coding				Remarks
Digits	12345		12345	12345	
	a	Op	m	c	
	ΔΔΔΔΔ	DIV	KΔ295	ΔΔ18O̲	If overflow does not occur, control is to go to tag ΔΔ18O̲.
					If overflow does occur, control is to go to tag ΔΔ18P.

The address assigned to tag ΔΔ18P will be equal to the address assigned to tag ΔΔ18O̲ plus 1.

When coding for overflow conditions, it should be observed:

1. Neither the O̲ nor the P line has to follow the line from which the overflow may result.

2. If the subroutine coded to handle the overflow condition is common to more than one operation, a permanent tag must be used. If the subroutine is only entered from one operation, a temporary tag may be used. In either case, the tag must follow the correct format for its type (see Tag Addressing, above).

3. Overflow lines must be counted as part of the tag limits.

The O̲ and the P lines must each be counted once.

Coding	Remarks
ΔΔΔΔΔ LDA W0002 ΔΔΔΔΔ	Counter (original setting 99 9999 9975) to rA.
ΔΔΔΔΔ ADD K0109 ΔΔ42O̲	Update counter; if overflow, go to a address 42P; if no overflow, go to a address 42O̲.
ΔΔ19N LDA K0006 ΔΔ20N	
ΔΔ20N STA ΔΔΔ7N ΔΔΔ8N	
ΔΔ42O̲ STA W0002 ΔΔ19N	No overflow, store updated counter in W0002; go to a address ΔΔ19N.
ΔΔ42P LDA K0212 ΔΔΔΔΔ	Reset counter (99 9999 9975 to rA).
ΔΔΔΔΔ STA W0002 ΔΔ22N	Store reset counter in W0002; go to a address ΔΔ22N.

D. Absolute Addressing

When it is necessary in an operation to reference a fixed computer location or absolute address, it is coded by placing the specific numeric characters that designate that location in the X-6 symbolic address, digit positions 2-5. To refer to Fast Access memory location 4318, for example, the numbers 4318 would be placed in digit positions 2-5 of the appropriate X-6 symbolic address. Digit position 1 may be coded as a Δ or 0. Thus, digit positions 2-5 when used for absolute addressing must be in the range ΔΔΔ0 (or 0000) through Δ4999.[1]

An address coded in this manner will not be modified in any way. For example, if RPU04-8C01 is to be used with an X-6 coded program and it is necessary to enter the RPU04 Punch Section. The X-6 coded line that transfers control that section will contain the absolute address of the Punch Section entrance:

Coding				Remarks
a 12345	Op	m 12345	c 12345	
ΔΔΔΔΔ	LDA	ΔΔΔ1N	Δ3072	Bring the contents of tagged line 1N to rA, and go to location 3072 for the next instruction to be executed. (3072 is the entrance to the Punch Section of RPU04-8C01. Control will be returned to the X-6 assembled program at the line placed in rA.) The c address could also have been coded as 03072.

References to absolute addresses may be placed in the a, m, and c portions of an X-6 instruction.

To determine whether an address is absolute or not, during an X-6 assembly, a test is made to determine if the character in digit position 5 is alphabetic. If it is not, digit position 1 is checked. If this character is also not an alphabetic, the address is classed as an absolute address and is not modified in any way. If absolute addressing is to be used in a program, the specific locations must be restricted from assignment during the X-6 program assembly. This is done by specifying such locations, or even specific groups of locations (portions of the computer memory) on Restrict Cards, Card Type 2 (see Input Card Section, below).

E. Register Addressing

When it is necessary in an operation to address the contents of a register, the address is coded by using two of the five digits of the X-6 symbolic address:

---

[1]If the absolute address 0000 is to be assigned, it should be noted that at least one digit must be a zero. The other digits positions may be coded as spaces.

Digit	12345
Symbolic Address	ΔΔΔRi

R should be placed in digit position 4 though only digit 5 is processed.

i (Digit 5) must be:

A for register A.
X for register X.
L for register L.

The register contents should be added to the symbolic deck by use of a card with the register in the a address portion. This will allow the latency counter or Clock to be updated for correct address assignment of the next line to be assembled. For example:

Instruction Line

a	Op	m	c	Remarks
ΔΔΔΔΔ	LDA	K0005	ΔΔΔΔΔ	Contains JMP ASINF
ΔΔΔΔΔ	ADD	K0012	ΔΔΔRA	Add 00 0000 0010 to the contents of rA and go to rA for the next instruction. The next instruction is in line ASINF.
ΔΔΔRA	JMP	ASINF	ΔΔΔ10	

The card with rA in the a address portion will cause a print out on the listing. No corresponding output card will be produced.

II. DATA ADDRESSING

X-6 coding provides four basic types of data addressing:

Working Storage

Constants

Table Entry

Interlace

Working Storage and Constant addressing refer to data (or instructions treated as data). These are stored in locations related to the lines of the operations in which they are referenced but not to themselves. Table Entry and Interlace Addressing reference data stored in locations relative to themselves, the relation to their program references being of secondary importance.

A. Working Storage and Constant Addressing

Both constant and working storage data may be coded with spaces in the a symbolic addresses each time they are required by the program. Such coding would assure the best possible latency positions being assigned during an X-6 assembly. However, the data would have to be placed in a specific location for each reference and could not be referenced by any line of coding other than the line directly preceding it. When time alone is the prime consideration, this method can be used to advantage. The disadvantage, of course, is that more than one location is occupied by the same data word.

To conserve memory and assure at least minimal relative latency between a working storage or constant location and the lines of the operations that reference it, such data are assigned to pools. Working Storage data would be placed in the W-Storage pool and constant data in the K-Constant pool. When assigned to a pool, the addresses generated for a W-Storage or K-Constant by the X-6 Assembly System will depend upon the address assigned to the line in which it is first referenced. During the subsequent assembly process, the same address will be assigned whenever a particular W-Storage or K-Constant occurs.

To assure minimal relative latency to all the lines in which they are referenced, W-Storages and K-Constants will be assigned by the X-6 assembly system to the Fast Access memory until all such locations are exhausted. After that, they will be assigned to the normal access bands.

The most appropriate method of addressing W-Storages or K-Constants will depend upon the program to be assembled. Final determination will be made by considerations of program memory space and running time. Whatever the method, the decision must be made before the program is coded. For example, if the program flowchart indicates that the coding will take about a thousand lines, and computer running time is critical, space addressing would be the most logical method of coding. If the flowchart indicates that storage space may be critical, working storages and constants would be pooled, or a portion pooled (those most often referenced by various operations) and others space coded.

When data is placed in a pool, consideration should be given to when the first reference is to be made to it during the X-6 Assembly. For example, if an operation is to be executed repeatedly for each input item in a program, and working storage and/or constant data used in that operation is also referenced by other operations, the first references to the W-Storage and K-Constant data during the X-6 assembly should be made in the repeated operation. Thus, minimum latency would be obtained for the references in the repeated operation and minimal relative latency would be obtained for references in other operations by Fast Access memory assignment of the W-Storage and K-Constant data.

A maximum of 300 W-storages and 300 K-Constants are allowed in a program. Both W-Storage and K-Constant entries are addressed in X-6 coding by tags conforming to a particular format.

1. W-Storage and K-Constant Addressing

The W-Storage or K-Constant tag will most often occur in the m symbolic address portion of an X-6 instruction. When the contents of the W-Storage or K-Constant is given, the tag will occur in the a portion. If the contents should be an instruction to be performed, reference may be made in a c portion.

Coding

Digits	12345
Symbolic Address	y0xxx

y (Digit 1) Either W or K must be used in this location.
W=W-Storage pool.
K=K-Constant pool.

0 (Digit 2) This position is ignored during X-6 Assembly. It is usually coded with Δ or 0 but may be any character.

xxx (Digits 3-5) These must be a numeric in the range 000 to 299. Leading zeros may be coded as spaces (KΔΔΔ1=KΔ001). During X-6 assembly, these digits are extracted and used to form a table look up instruction when W and K tags are converted to absolute addresses.

When coding W-Storage or K-Constant addresses, the following should be observed:

a. The order of addressing is not important. For example, Δ299 may be referenced before Δ050.

b. All 300 numbers for each type of tag do not have to be used in a program.

c. An absolute address may be assigned to W-Storage or K-Constants by using a Tag Equals Card, Card Type 3 (see Input Card Section, below).

2. When the X-6 Symbolic deck is keypunched from the X-6 coding, for every W-Storage or K-Constant referenced in m or c addresses, there must be a card containing the W-Storage or K-Constant in the a address. For example, if in the coding there are m and/or c address references to W△000 through W△003 and K△015 through K△017, the following cards must be part of the symbolic deck:

a	Op	m	c
W△000			
W△001			
W△002			
W△003	CONTENTS		
K△015			
K△016			
K△017			

The contents of the constant addressed by the K-Constant tag will appear in the Op, m, and c address positions of the card. When W-Storage locations must be set to initial conditions, as with counters or limits, these initial conditions will be keypunched in the same manner as K-Constant contents. Whether the contents are for K-Constants or for W-Storages, they may be coded to be treated as absolutes, not to be modified in any way, or coded symbolically to be translated during the X-6 assembly.

3. If absolute coding is used, △△△ must be placed in the Op portion. The ten digits that are placed in the m and c portions may be alphabetic, numeric, or any combination of the two. For example, the contents of the following would be treated as absolute:

a	Op	m	c
W△074	△△△	99999	99975
K△284	△△△	00000	00000

In the case of data not to be translated into machine code, a Key of the card would also be punched. If, for example, the following K-Constants were to be used for punching and/or printing, the Key would be punched:

a	Key	Op	m	c	
KΔ025	U	ΔΔΔ	RUN01	EDIT	2 part alphabetic, USS 90 Card code. (U=Unprimed)
KΔ026	P	ΔΔΔ	RUN01	EDIT	(P=Primed)
KΔ015	U	ΔΔΔ	RUN01	EDIT	3 part alphabetic, USS 80 Card code.
KΔ016	P	ΔΔΔ	RUN01	EDITΔ	(U=Unprimed) (P=Primed)
KΔ017	D	ΔΔΔ	RUN01	EDITΔ	(D=Duoprimed)
KΔ050	N	ΔΔΔ	RUN01	EDITΔ	2 part alphabetic, USS 80/90 machine code. (N=Numeric)
KΔ051	Z	ΔΔΔ	RUN01	EDITΔ	(Z=Zone)

When X-6 symbolic coding is used, translation of the W-Storage or K-Constant data will be made during the X-6 assembly. The thirteen digit positions comprising the Op, m, c address portions must be used. For example, the contents of the following would be translated during assembly:

a	Op	m	c
KΔ008	LDA	KΔ004	ASINF

The processing of W-Storage and K-Constant data is determine by the presence or absence of spaces (ΔΔΔ) in the Op portion of the coding.

4. There are six non-numeric computer coded characters. The alphabetic designations for these are:

0101	A
0110	B
0111	C
1101	F
1110	G
1111	H

5. A Δ or a 2 in the control column will indicate a positive or negative value (see INPUT CARD FORMAT, Card Type 8).

6. During the assembly of the symbolic deck, it is advantageous to group the cards containing W-Storage data together under the same operation name and the cards containing K-Constant data under another operation name (usually, WWW and KKK are the operation names used). By using such an assembly, desk checking and program testing of an X-6 assembled program is simplified: When it is necessary to check the contents of a referenced W-Storage or K-Constant, it is easier to find if the location in the deck is a known relative position.

B. Table Entry Addressing

1. A table consists of data stored at regularly spaced intervals. The contents of any particular storage location in a table may be designated as an entry. Provision has been made in the X-6 Assembly System for as many as thirty tables of up to 1,000 words each in a program. A table entry reference will usually occur in the m symbolic address portion but may occur in the a or c portion. It is coded in the following manner:

Coding	
Digit	12345
Symbolic Address	tnxxx

tn (Digits 1-2) is the identifier of the table referenced: t (Digit 1) must be either S, U, or V. Thus allowing 30 possible table names.

n (Digit 2) must be a numeric in the range 0 through 9.

xxx (Digits 3-5) is the identifier of the table entry and must be a numeric in the range 000 through 999.

Thus, S3000 would reference the first entry of table S3, V4898 would reference the 899th entry of table V4.

The order in which tables are referenced is not important (the first table might be V8, the second S1, the third U9, etc.).

2. When the number of tables that will be used in a program has been determined, each table must be described on a Type 5 Card (see Input Card Section, below). The coding on the Type 5 Card will define the location of the first table entry, the number of entries (000-999) in the table, and the desired interval between entries. When this card is processed by the X-6 Assembly System, all locations required by the table will be restricted from other assignment.

Care must be taken during the X-6 coding of a program not to reference an entry that is not in a particular table. That is, if the number of entries in a particular table was defined as 25 on the Type 5 Card, only 25 locations were restricted to that table. Should a reference be made to an entry greater than 25 for that table, it will not be detected as a logical error during the X-6 assembly.

C. Interlace Addressing

1. Positions on the Input and Output Interlaces may be referenced as absolute addresses or in X-6 symbolic coding. When referenced symbolically, the coding, which may appear in the a, m, and c symbolic addresses, is:

Coding	
Digits	12345
Symbolic address	inxyz

in (Digits 1-2) is the identifier of the interlace.

i (Digit 1) specifies the I/O device and must be one of the following:

H the read interlace of the HSR.

R the read interlace of the RPU.

O the punch interlace of the RPU.

P the HSP interlace.

T
Z tape interlace.

n (Digit 2) specifies the number of the interlace and must be a numeric in the range 0 through 9.

Thus, the combination of the alphabetic specifying and I/O device and the numeric of 0 through 9 allows ten possible identifiers for each I/O device. Since two alphabetics may be used to specify a tape interlace, 20 tape interlace identifiers are possible. A program requiring the use of alternate input bands could be coded throughout with symbolic addresses. Alternate Cards, Type 4 (see Input Card Section, below) would be used to redefine each band.

xyz (Digits 3-5) depends upon the action desired by the reference.

2. To refer to an entire band:

a. xy (Digits 3-4) must be 00 when reference is made to an entire band of the HSR or RPU.

z (Digit 5) must be 0 if the contents of the band are not to be automatically translated; 1 if the contents of the band are to be automatically translated.

(For example, HBU H1000 would dump the HSR buffer into the first and second read interlace positions without automatic translation. For automatic translation, the instruction HBU H1001 would be used.)

b. When a reference is made to a complete HSP interlace band:

x (Digit 3) must be 0.

yz (Digits 4-5) will specify a number of lines and must be a numeric in the range 00 through 79.

(Thus, PRN P0000 would advance the paper zero lines before printing.

PRN P0030 would advance the paper thirty lines before printing.)

c. When an entire tape interlace is referenced, as in read and write instructions:

x (Digit 3) refers to the Uniservo number and must be a numeric in the range 0-9.

y (Digit 4) refers to mode and density and must be:

0 for USS, 250 cpi.

5 for UNIVAC, 250 cpi.

6 for UNIVAC, 125 cpi (used only with write instructions).

z (Digit 5), used only with read instructions, refers to direction and gain and must be:

0 forward normal.

1 forward low.

2 forward high.

5 backward normal.

6 backward low.

7 backward high.

When reference is to be made to a particular word of an interlace band, the above coding cannot be used.

3. To refer to a particular word of an interlace band:

a. x (Digit 3) relates to the translation mode and must be one of the following:

(1) For untranslated (Card Code) words of a band:

U=Unprimed.
P=Primed.
D=Duoprimed (applicable USS 80 only.)

(2) For the HSP Interlace and for translated (Machine Code) words:

N=Numeric
Z=zone.

b. yz (Digits 4-5) relate to the word in the interlace band. The coding varies for each I/O device:

(1) HSR and RPU Read Stations:

y (Digit 4) means the read station and must be 1 or 2.

z (Digit 5) means one of the eight words and must be a numeric in the range 0 through 7.

Thus, N11 specifies the numeric portion of the second word at the first read station.

Z20 would specify the zone portion of the first word at the second read station.

U25 would specify the unprimed portion of the sixth word at the second read station.

(2) RPU Punch Interlace:

y (Digit 4) must be 1.

z (Digit 5) indicates the word and must be a numeric in the range of 0 through 7.

Thus, U13 specifies the unprimed portion of the fourth word of the punch interlace.

Z10 would specify the zone portion of the first word of the punch interlace.

(3) HSP Interlace:

yz (Digits 4-5) must be a numeric in the range 01 through 13.

Thus, N12 would specify the numeric portion of the twelfth word of the HSP interlace.

(4) Tape Interlace:

x = N or Z

yz (Digits 4-5) when referring to a word of a tape interlace must be a numeric:

in the range 00-71 of an interlace in XS-3 Code,

in the range 00-99 of an interlace in USS Code.

4. As examples of interlace addressing from the foregoing:

H1Z10 HSR interlace #1, the zone portion of word zero at the first read station. H1Z20 would be the same word at the second read station.

P1N13 Printer interlace #1, numeric portion of word 13. P1Z13 would be the same word, zone portion.

T9Z11 The ninth tape interlace, zone portion of word 11. (TRDΔΔ800 would be, read one block from tape buffer band using Servo 8, USS mode, forward normal).

LATENCY MINIMIZATION

Latency minimization during a program or an operation assembly is achieved through use of a working storage location called a "Clock" in which the X-6 Assembly System stores the relative band level location. The value or setting of the clock is initially 00 0000 0000. At any subsequent time, the setting will always lie within the range 00 0000 0000 through 00 0000 0199. When an instruction line is analyzed by the X-6 Assembly System, the clock reading is used to obtain the tentative best address (TBA) for the next address to be assigned. The TBA is generated and assigned by using the value of the clock setting, incrementing the setting by the specific word increments associated with each instruction code, or by assigning a new setting to the clock and then incrementing the value of the new setting (these increments can be found in the Instruction Code Information Words Table, below). After the TBA is obtained, the available memory locations are searched. If a band location equivalent to the relative band level of the TBA is found, it is assigned. If no such band location is found, the TBA is incremented and another search is made. This process continues until an assignment is possible. When it is not possible to make an assignment because the memory is full, an arbitrary assignment to 9999 is made and the assembly continues. A printout indicating such an assignment is made in the listing. After an address assignment has been made, the absolute address is reduced to a relative band level value and is stored in the Clock.

INSTRUCTION CODE INFORMATION WORDS TABLE

If control column indicates Index Register modification, add one more word time before m.

	Digits 1-2	Digit 3 Action Code	Digits 5-7 Before m	Digits 8-10 Before c
ADD	70	0	002	003
BUF	20	0	002	002
DIV	55	0	002	113
ERS	35	0	002	002
LDA	25	0	002	002
LDL	30	0	002	002
LDX	05	0	002	002
MUL	85	0	002	103
STA	60	0	002	002
STL	50	0	002	002
STX	65	0	002	002
SUB	75	0	002	003
LIR	02	0	000	003
IIR	07	0	000	004
TRD	G2	1	000	017
TWR	H2	1	000	017
TRW	F2	1	000	150
TMX	C1	1	000	003
TXM	C3	1	000	003
ATL	77	1	000	003
CTM	12	1	000	003
MTC	17	1	000	003
ZUP	62	1	000	004
HSS	47	1	000	003
RSS	57	1	000	003
CLA	26	2	003	000
CLL	31	2	003	000
CLX	06	2	003	000
JMP	00	2	002	000
CAA	36	2	003	000
CAX	86	2	014	000
CTA	23	2	(002 / 003)	000
PFD	16	3	222	003
SHL	37	3	111	003
SHR	32	3	111	003

222 is a code not affecting timing; 111 means use amount of shift.

	Digits 1-2	Digit 3 Action Code	Digits 5-7 Before m	Digits 8-10 Before c
HBU	96	4	198	203
PRN	11	4	197	592
RBU	46	4	098	203
RCC	81	4	098	203
TBU	F6	4	048	103
TBL	C6	4	198	205
HBT	42	5	004	003
HCC	72	5	004	003
PBT	27	5	004	003
RBT	22	5	004	003
STP	67	5	003	003
TEQ	82	5	003	003
TGR	87	5	003	003
TBT	C7	5	005	003
TST	C2	5	004	003

CLOCK MODIFICATION

The purpose of the clock modification instructions is to allow relationships to be established between addresses when these relationships cannot be detected by the X-6 Assembly System. This is necessary because the X-6 Assembly System is a one pass program. Once an address has been assigned, therefore, it cannot be changed at any subsequent assembly point. Certain conditions may arise when the process by which the X-6 system assigns addresses will not result in the best latency from an overall program point of view. One example of this would be:

X-6 Coded Lines	Remarks
a Op m c	
ΔΔΔΔΔ TEQ ΔΔΔ1N ΔΔΔΔΔ	The address for temporary tag 1N would be assigned during the assembly of the TEQ line. This address would then be placed in the TGR line.
ΔΔΔΔΔ TGR ΔΔΔ1N ΔΔΔΔΔ	

X-6 Assembled Coding	
2145 82 2148 2348	Thus, if control is sent to 2348 by the equality test and then sent to 2148 by the magnitude test, a drum revolution would be lost.
2348 87 2148 2351	

In this case, it would be desirable to have the address assigned to 1N increased by the increment between the first reference to it in the TEQ line and the second reference to it in the TGR line so that the coding generated would be:

X-6 Assembled Coding	Remarks
2145 82 2151 2348	The process by which this is accomplished will be found in the Examples of Clock Modification at the end of this section.
2348 87 2151 2351	

The clock setting may be modified by any arbitrary increment, or the clock may be set to any arbitrary band relative reading. Such modification is programmed by the use of any of seven clock modification instructions. Each such instruction used is keypunched on a detail Card, Card Type 8 (see Input Card Section, below), and filed in the symbolic deck immediately preceding the instruction the new clock reading is to affect.[1] Each of the seven

[1]Clock modification cards do not require a card number in columns 6-8. Thus, they may be inserted at any time without breaking the detail card sequence and causing an entire operation to be renumbered.

clock modification instructions must have CLOCK in the a symbolic address portion of the coding.

The clock modifications may be divided into two basic types:

SE (Set) in which a new setting of the Clock is made before incrementation by a specified number of word times. An SE instruction may only directly modify one address in the succeeding instruction.

AD (Add) in which a specified increment is added to the normal band relative address which the X-6 Assembly System would normally assign. An AD instruction may directly modify two addresses in the succeeding instruction.

The clock modifications and their format are as follows:

A. ΔΔΔ Instruction:

a	Op	m	c	Remarks
CLOCK	ΔΔΔ	sssss	00xxx	The succeeding a address will be modified:
				sssss must be a legitimate X-6 symbolic address or an absolute memory location. This address will be converted to a band relative reading and placed in the clock.[2]
				xxx must be a numeric increment to be added to the new clock setting in addition to the normal incrementation. The result of this addition will be the TBA for the assignment of the succeeding a address.[3]

---

[2]If sssss is an X-6 symbolic address that has not already been processed, it will be assigned a permanent address when the clock modification instruction line is processed. Thus, it would be assigned in minimal latency to the line just preceding the clock modification in the assembly process. If this happens, it could result in a loss of word times when the object program instruction line that first references sssss is assembled.

[3]The word time increment of the clock modification instructions is always added to the clock setting. Since the clock setting will always lie within the range 000-199, the setting may, in effect, be decremented by subtracting the desired decrement from 200 and using the result as the specified increment.

This is the only clock modification that does not contain a mnemonic code in the Op portion of the instruction. The same modification may be accomplished by use of the SEA instruction (see below). It is also the only clock modification instruction that does not allow the clock to be reset to its premodification setting after the succeeding desired address portion has been assigned according to the modified clock setting.

B. SE Instructions:

For each of the succeeding SE instructions, the format of the a, m, and c address is the same:

1. The a address portion must always be:

   a
   CLOCK

2. The m address must always contain:

   m
   xxx0z

   xxx = The numeric increment to be added to the new clock reading that will be specified in the c portion of this instruction in addition to the normal incrementation. The new clock reading plus the increment will result in the TBA for the address to be assigned. (Spaces, Δ, cannot be used in place of zeros.)

   z = 0 if the clock setting is <u>not</u> to be restored to its premodification setting before obtaining the TBA for the address succeeding the address to be modified.

   z = 1 if the clock setting is to be reset to the premodification setting before obtaining the TBA for the address succeeding the address specified to be modified.

3. The c address must contain:

   c
   sssss

   sssss = A legitimate X-6 symbolic address or an absolute memory location. This address will be converted to a machine coded band relative reading and placed in the clock.

4. The mnemonic SE instructions and their format are:

CLOCK SEA xxx0z sssss	The succeeding a address TBA will be arrived at by using the band relative equivalent of sssss plus the increment xxx. The presence of 0 or 1 in the z digit position will determine whether the clock will be restored to its original setting when this modification has been accomplished or if the clock setting that results from this modification will be retained.
CLOCK SEM xxx0z sssss	The succeeding m address TBA will be arrived at by the above process.
CLOCK SEC xxx0z sssss	The succeeding c address TBA will be arrived at by the above process.

C. AD Instructions:

1. The a address portion must always be:

   a
CLOCK

2. The m and c address portions must always contain:

m	c		
xxx0	00yyy	yyy =	The numeric increment to be added to the present clock reading, in addition to the normal incrementation, to arrive at the TBA to be assigned to the next address specified in the operation code of the AD instruction.
		xxx =	The numeric increment to be added to the clock reading according to the numeral in the z digit. This addition is used to obtain the TBA for the address to be assigned after the address called for in the operation code of the AD instruction. If xxx=000, the address generated will be derived normally from the clock reading determined by the z digit.

(Space, Δ, cannot be used in place of zeros in the xxx and yyy portions.)

z = 0 if the clock setting is <u>not</u> to be restored to its pre yyy reading before incrementing by xxx.

z = 1 if the clock setting is to be restored to its pre yyy modification before incrementing by xxx.

3. The AD instruction Codes, and their format, are:

CLOCK ADA xxx0z 00yyy	The TBA for the succeeding a address will be arrived at by adding yyy to the clock reading. The succeeding m address will be arrived at by incrementing the new clock reading, if z=0; or, if z=1, by restoring the pre yyy incrementation clock reading before incrementing by xxx. The succeeding a address will be assigned normally.
CLOCK ADM xxx0z 00yyy	The succeeding m and c addresses will be arrived at by the above process.
CLOCK ADC xxx0z 00yyy	The succeeding a and m addresses will be assigned normally. The succeeding c and the a address following it will be arrived at by the above process.

4. When an absolute address on the Fast Access bands is specified in a clock modification instruction, the Fast Access address is reduced to a number in the range 00 through 49. This is placed in the clock in the form 000 through 049. Thus, if no further incrementation is specified, the absolute address derived from this reading will have to be on an even band level on the Normal Access bands. An odd numbered band assignment on the Normal Access bands is only possible when the clock seting, plus increment if called for, is in the range 100 through 199.

D. Examples of Clock Modification

The following examples of the use of the clock modification instruction are not intended to illustrate every possible condition that may arise. The application of these instructions will depend entirely on the nature of the object program to be assembled.

1. In the beginning of this section, the following example was given:

X-6 Symbolic Coding				X-6 Assembled Coding			
a	Op	m	c	a	Op	m	c
	TEQ	1N		2145	82	2148	2348
	TGR	1N		2348	87	2148	2351

It was noted that the address of temporary tag 1N was generated and assigned during the processing of the TEQ line. Thus, the same address was assigned when 1N was referenced in the TGR line. The result was that if during the object program execution control was sent to 2348 after the equality test and then to 2148 after the magnitude test a drum revolution would be lost. In such a case, a clock modification instruction should be used so that the address generated for tag 1N will be incremented by the word time interval between its first reference in the TEQ line and its second reference in the TGR line:

X-6 Symbolic Coding				X-6 Assembled Coding			
a	Op	m	c	a	Op	m	c
CLOCK	ADM	00001	00003				
	TEQ	1N		2145	82	2151	2148
	TGR	1N		2148	87	2151	2351

Thus, the address generated for 1N in the TEQ line would be incremented by 3 word times before assignment. The clock reading existing before the 1N address assignment would be used to obtain the c address in the TEQ line.

2. The X-6 Assembly System automatically increments the clock by 105 word times for every multiplication instruction: 2 word times between the a and m addresses and 103 between the m and c addresses. In those cases where the number of digits in the multiplier is known, this increment can be changed by use of a clock modification and insertion of a sentinel to the left of the most significant digit of the multiplier:[4]

[4]When the computer receives a multiplication order, the multiplier is placed in rX and a sentinel is automatically generated and placed in the least significant digit position of rA. As the multiplication process is carried out, this machine sentinel is shifted one position at a time toward the least significant digit position of rX, followed by the least significant digits of the product as they are developed. When the machine sentinel is shifted out of the least significant digit position of rX, the multiplication process stops. The product of the multiplication is in rA and rX with the least significant digits in rX. When a programmed sentinel is placed in rX with the multiplier, the machine sentinel is still placed in rA. When the programmed sentinel is shifted out of rX, the multiplication process stops. The machine sentinel is left in rX to the right of the least significant digits of the product.

X-6 Symbolic Coding

a	Op	m	c	Remarks
	LDL	W0012		
CLOCK	ADM	03000	00000	It is assumed that the sentinel has been positioned in the multiplier contained in K0001 and that thirty word times, plus the 2 word times between the a and m addresses, has been determined as the length of time needed for the multiplication to be completed.
	MUL	K0001		

Thus, the clock would be incremented by 000 before assignment of the address for K0001. The c address following would be generated and assigned with an incrementation of 30 word times instead of the usual 103.

3. An object program may contain a constant that is a variable instruction. This could be, ΔΔΔΔΔ SHR Δ0000 ΔΔΔ7N with the amount of shift ranging from 0000 to 0009. When assembling a shift instruction line, the X-6 Assembly System increments by the amount of shift specified by the m address plus three word times to obtain the c address. If the above line were assembled with the minimum shift value, the c address would be assigned three word times from the a address. As the instruction was executed during the object program, any incrementation of the shift value would result in the loss of a drum revolution. This can be corrected by the use of a clock modification instruction during assembly:

   X-6 Symbolic Coding

a	Op	m	c	Remarks
	LDA		6N	Load rA with constant.
CLOCK	ADC	00000	00009	Adjust c address of constant for maximum shift value.
	SHR	00000	7N	Constant.
6N	BUF	W 3	RA	Buff in amount of shift (already generated and stored in W-Storage 3) and go to rA for next instruction.

   It is assumed that the constant line in this case is only referenced in this operation and only at this point in the operation. Thus, it is not necessary to assign a K-Constant tag to it.

4. The principle used in example 3, above, can apply to any variable instruction line of a program to be assembled. For another example of this, an instruction line is to be modified by an index register before execution:

X-6 Symbolic Coding

a	Op	m	c	IR	Remarks
42N	STA	Δ1000	ASINN	2	For this example, assume the range for m to be 1000 through 1150 due to index register modification before execution.

Thus, the address to be assigned to ASINN should be relative to 1150 rather than 1000 which is the first executable value. To do this, the line could be preceded by:

CLOCK	ADM	00000	00150		The address generated for the m portion will be incremented by 150 (the upper limit of its range) before assignment. The c address will be derived normally from the resultant clock setting.
42N	STA	Δ1000	ASINN	2	

5. When an object program contains a subroutine which consists of operations of various word time lengths but with the same exit, it is usual practice to assemble the longest of these operations first. If this is not done, the first operation to be assembled should have its exit line preceded by a clock modification instruction which will increment the common exit address by the word time differential between the length of the operation being assembled and the length of the longest operation in the subroutine. For example, a subroutine contains the following three operations:

   a. Enter with tag 1N, process data (approximately 50 word times), and exit to tag ASINF.

   b. Enter with tag 2N, process data (approximately 100 word times), and exit to tag ASINF.

   c. Enter with tag 3N, process data (approximately 200 word times), and exit to tag ASINF.

   If operation a. is assembled first the exit line to tag ASINF would be preceded by:

   X-6 Symbolic Coding

a	Op	m	c	
CLOCK	ADC	00000	00150	The address generated for tag ASINF would be incremented by 150 word times, the difference between the length of the operation assembled and the length of the longest operation of the subroutine.
	STA	W 19	ASINF	

6. The same principle as in example 5 would be applied if the length of an operation is variable. For example, if the entrance to an operation were to be made from instructions entered in a table, the overall operation length set during assembly should allow for the longest possible length of the operation:

Given a table of five entries stored at intervals of twenty word times between each entry, the word time difference betweeen the first and the fifth entry would be 80.

X-6 Coding	Assigned Location	X-6 Coded Contents
S1000	2300	LDA W0001 ASINF
S1001	2320	LDA W0002 ASINF
S1002	2340	LDA W0003 ASINF
S1003	2360	LDA W0004 ASINF
S1004	2380	LDA W0005 ASINF

If the first assembled line is to be S1000 LDA W0001 ASINF, and this is the first assembly reference to ASINF, a clock modification instruction should be used to set the address assigned to ASINF so that when the last table entry line is assembled, minimal latency between addresses will result:

```
CLOCK ADC 00000 00080
S1000 LDA W0001 ASINF
```

In this way, the address generated for ASINF would be incremented by 80 word times before assignment. When, later in the assembly, S1004 LDA W0005 ASINF is assembled, the addresses would be in minimal latency. The amount of incrementation would depend on which table entry line is first assembled.

7. When a connector is to be set in an object program, it may be desirable to use a clock modification to relate the m address of the instruction to be placed in the connector with the address assigned to the connector. For example, the instruction lines that load the connector are:

X-6 Symbolic Coding

a	Op	m	c	Remarks
	LDA		5N	Load rA with connector setting.
	LDA	7N	9N	The connector setting.
5N	STA	ABC2N		Store setting in connector.

The clock modification used could be:

X-6 Symbolic Coding

```
  a    Op    m      c           Remarks

       LDA           5N    The address assigned to 7N will
CLOCK  SEM 00200 ABC2N     be equal to the band relative
       LDA    7N     9N    address assigned to ABC2N plus
   5N  STA ABC2N           an increment of 2 word times.
```

It is assumed, in this example that ABC2N has already been assigned an address during a previous portion of the assembly. If it has not and the ABC2N address is assigned during the assembly of the above lines, it may be necessary to use a clock modification during the assembly of the operation in which ABC2N is executed. This would insure minimal latency of the address generated for that operation in relation to the ABC2N address.

X-6 LIBRARY ROUTINES

Certain functions recur frequently as elements of an installation's programs. Such function are typically isolated and coded in the best possible manner for inclusion in an X-6 Library.

When an object program is to be assembled by the X-6 Assembly System, any X-6 library subroutine decks necessary are included with the main program deck. This allows the assembly system to generate the absolute addresses occupied by the subroutines.

When a subroutine is coded for inclusion in an X-6 library, input and output locations are characteristically assigned to registers in order to simplify access to the subroutine by the user. Provision is made, wherever possible, for the insertion of parameters which can tailor the subroutine to the needs of any object program. References to constants, working storages, interlaces, and tables which are used by such a subroutine but not contained within it are generalized by placing special tags to indicate parameters in the a, m, or c address portions where these references occur.

Twenty tags to indicate parameters are allowed in each operation within an X-6 library subroutine. The coding of this tag is in the form:

Digit	12345
Symbolic Address	X△△nn

X (Digit 1) must be X.
(Digits 2-3) may be △△ or 00.
nn (Digits 4-5) must be a numeric in the range △1 (or 01) through 20.

(Note: Should it ever happen that more than 20 parameters are necessary within a subroutine, all parameters beyond the X△△20 upper limit would be coded as permanent tags.)

When the X-6 library subroutine is assembled as part of an object program by the X-6 Assembly System, the parameters addressed within each operation of the subroutine are assigned specific locations related to the object program, by the insertion of Specifications Cards, Card Type 6 (see Input Card Format, below), before the operation to which they apply. The format of the entries on the Specification Card is:

Digits	1	2	3	4	5	6	7	8	9	10
Symbolic Coding	X	△	△	n	n	e	e	e	e	e

X△△nn (Digits 1-5) is the parameter to be redefined in relation to the program being assembled.

eeeee (Digits 6-10) is a legitimate X-6 address to be placed in the parameter designated by digits 1-5. This may be an absolute address or an X-6 Symbolic Address (that is, a permanent tag, an interlace or table reference, a K or W-Storage address, a register address, etc.).

The redefinitions contained on the Specifications cards are filed in a table and erased at the end of the assembly of the operation which they precede. This allows the table to be used again by any succeeding operation in which XΔΔnn parameters must be redefined.

The most advantageous method of building a library of X-6 subroutines is to file each subroutine under an operation name unique to itself with the cards in correct sequence. In some cases a library subroutine may contain a number of operations each of which has its own unique name. For library convenience, an overall operation name should be given to the subroutine. To avoid renumbering of the subroutine cards, before assembly, a library subroutine should be assembled as a separate object program operation, not as a part of an operation within the object program.

ASSEMBLY INPUT CARDS

After an object program has been coded according to the X-6 coding conventions, the symbolic deck used as input for an X-6 program assembly must be prepared. Besides those cards that will contain the coded lines, other cards must be prepared to set the limits within which the assembly is to take place and to signal the beginning or ending of certain assembly processing. That is, the beginning and the end of an object program must be signalled as must the beginning and end of operations within the program. Certain portions of computer memory must be restricted from assembly assignment: those locations that are used as absolute addresses in the coding and the locations that will be used by tables and interlaces, for example.

I. Symbolic Deck Organization

These are ten possible card types that may be keypunched for an X-6 program. Of these ten, there are five card types that must be used in any program to be assembled by the X-6 Assembly System:

Card Type	Title
1	Label Card
7	Operation Header Card
8	Symbolic Detail Card
9	Operation Sentinel Card
10	End of Run Sentinel Card

Every program must have only one Type 1 (Label Card) and only one Type 10 (End of Run Sentinel).

Each operation must have only one Type 7 and only one Type 9. The number of Type 8 cards must correspond to the number of lines of coding in the operation and the number of constants unique to that operation.

The other card types that may be used, depending on the needs of the program are:

Card Type	Title
2	Restrict Card
3	Tag Equals Card
4	Interlace Card
5	Tables Card
6	Specifications Card

Card Types 2 through 5 cause particular memory locations to be restricted from use by the X-6 Assembly System. Card Type 6 modifies coding within a library routine before it is assembled, thus allowing a redefinition of the library routine variables just before each operation is processed.

The Card Type number (in the form Δ1, Δ2, through 10) is keypunched in card columns 1-2.

When organizing the symbolic deck for a program, Card Type 1 must be the first card for input. All Types 2, 3, 4 and 5 cards must follow in numerical sequence. That is, all Type 2 cards must precede all Type 3 cards, etc. the grouping within the card type is unimportant. After Types 1 through 5, Card Types 6 through 9 are arranged by operation. That is, for each operation, the cards of that operation are grouped in sequential order: all type 6 cards for an operation will precede the Type 7 card. The type 7 card will be followed by all the Type 8 cards arranged in ascending sequence. The last card of each operation will be a Type 9. Usually, operations are grouped according to their relative importance in the program since the first assembled operation will receive the best possible X-6 latency minimization. The last card of the assembled deck must be the type 10 card.

II. Input Card Format

A. Label Card, Card Type 1

Function: To provide run identification for the edited listing. The information contained in this card will be printed as a header for each page of the listing.

80 Card Columns	90 Card Columns	Format	Name of Field
1-2	1-2	Δ1	Card Type
3-10	3-10	ΔΔΔΔΔΔΔΔ	Spaces
11-15	11-15	ppppp	Program Identification
16-20	16-20	ΔΔΔΔΔ	Spaces
21-26	21-26	ddddd	Date
27-30	27-30	ΔΔΔΔΔ	Spaces
	31-45	ΔΔΔΔ...ΔΔΔΔ	Spaces
31-80	46-85	zzzz...zzzz	Descriptive Comments
	86-90	ΔΔΔΔΔ	Spaces

Technical Notes:

1. Each run being assembled must have a Label Card as the first card of the symbolic deck. If the label card is missing, the computer will stop and display 67 0003 cccc.

2. Column 2 must contain a 1 punch.

3. Columns 3-10 are not examined by the system and can be used, if desired, to record additional descriptive information. This information is not printed in the output listing.

4. The program identification field is not altered by an X-6 assembly and can contain any combination of characters. However, the identification should be meaningful to the installation (for example, RUN01).

5. Columns 16-20 are never punched.

6. An X-6 assembly does not alter the date field; therefore, it may appear in any format desired.

7. Since the comments are not altered by an X-6 assembly, the comments field may contain any descriptive information.

B. Restrict Card, Card Type 2

Function: Specifies the absolute locations that will be used for some specific purpose and removes them from the Table of Availability before the Detail Cards, Card Type 8, are processed.

80 Card Columns	90 Card Columns	Format	Name of Field
1-2	1-2	Δ2	Card Type
3-10	3-10	ΔΔΔΔΔΔΔΔ	Spaces
11-20	11-20	iirrrraaaa	Entry 1
21-30	21-30	iirrrraaaa	Entry 2
31-40	31-40	iirrrraaaa	Entry 3
	41-45	ΔΔΔΔΔ	Spaces
41-50	46-55	iirrrraaaa	Entry 4
51-60	56-65	iirrrraaaa	Entry 5
61-70	66-75	iirrrraaaa	Entry 6
71-80	76-85	iirrrraaaa	Entry 7
	86-90	ΔΔΔΔΔ	Spaces

Technical Notes:

1. Column 2 must contain a 2 punch.
2. Columns 3-10 are not punched.
3. Entry contains ten digits in the following format:

   iirrrraaaa

   ii is the increment between elements.
   rrrr is the total number of locations to be restricted.
   aaaa is the beginning absolute address.

4. There is no limit to the number of Restrict Cards that may be used.
5. There is no limit upon the total number of addresses to be restricted by a single entry.
6. A particular restrict card may contain from one to seven entries. If there are less than seven entries the first invalid entry field must contain a sentinel word of nines (99 9999 9999).
7. The sentinel word stops the processing of a particular card, it does not signal the end of Type 2 Cards. That is, if the last Type 2 Card contains all seven entries, it is not necessary to prepare another card containing only the sentinel word. The end of Type 2 Cards will be detected by the punch in Column 2 of the next card.
8. During the actual assembly of the symbolic deck the interval of time during which the restrict card information is processed may be great enough to give the impression that the system has entered a closed loop. Actually, the length of time required is a function of the total number of locations to be restricted. In some cases, this might require up to seven or eight minutes.
9. All absolute addresses used in the X-6 coding of an object program that will not be specified on:

   a. A Tag Equals Card, Card Type 3
   b. An Interlace Card, Card Type 4
   c. A Tables Card, Card Type 5

   must be restricted from X-6 assembly assignment by an entry on a Restrict Card.
10. Usually the memory area required by a PTA routine (0000-0199) is restricted.

C. Tag Equals Card, Card Type 3

Function: Assigns a specific memory location to a permanent tag, K-Constant, or W-Storage.

80 Card Columns	90 Card Columns	Format	Name of Field
1-2	1-2	Δ3	Card Type
3-10	3-10	ΔΔΔΔΔΔΔΔ	Spaces
11-20	11-20	tttttΔaaaa	Entry 1
21-30	21-30	tttttΔaaaa	Entry 2
31-40	31-40	tttttΔaaaa	Entry 3
	41-45	ΔΔΔΔΔ	Spaces
41-50	46-55	tttttΔaaaa	Entry 4
51-60	56-65	tttttΔaaaa	Entry 5
61-70	66-75	tttttΔaaaa	Entry 6
71-80	76-85	tttttΔaaaa	Entry 7
	86-90	ΔΔΔΔΔ	Spaces

Technical Notes:

1. Column 2 must contain a 3 punch.

2. Each entry must contain ten digits coded in the following format:

   tttttΔaaaa

   ttttt is the name of the permanent tag, K-Constant, or W-Storage.

   aaaa is the absolute location to which ttttt is assigned.

3. There is no limit to the number of Tag Equals Cards that may be used.

4. Each Tag Equals Card may contain up to seven entries. Any Tag Equals Card containing less than seven entries must have a sentinel word (99 9999 9999) in the first invalid field to stop processing of the card.

D. Interlace Card, Card Type 4

Function: Provides automatic restriction of the input and output interlace positions. A single entry on this card restricts all interlace positions in the specified band for the unit desired. Information on the Interlace Card also permits the addressing of elements symbolically rather than in absolute notation.

80 Card Columns	90 Card Columns	Format	Name of Field
1-2	1-2	Δ4	Card Type
3-10	3-10	ΔΔΔΔΔΔΔΔ	Spaces
11-20	11-20	inΔΔΔxaa00	Entry 1
21-30	21-30	inΔΔΔxaa00	Entry 2
31-40	31-40	inΔΔΔxaa00	Entry 3
	41-45	ΔΔΔΔΔ	Spaces
41-50	46-55	inΔΔΔxaa00	Entry 4
51-60	56-65	inΔΔΔxaa00	Entry 5
61-70	66-75	inΔΔΔxaa00	Entry 6
71-80	76-85	inΔΔΔxaa00	Entry 7
	86-90	ΔΔΔΔΔ	Spaces

Technical Notes:

1. Column 2 must contain a 4 punch.
2. Columns 3-10 are not punched.
3. Each entry must contain ten digits coded in the following format:

inΔΔΔxaa0

i is the type of interlace and must be:

H for the HSR
R for the RPU read station
0 for the RPU punch station

P for the HSP
T or Z for tape

n is the interlace number (0-9).

x is the kind of interlace to be restricted:

0 for untranslated interlace
1 for translated interlace
2 for both
} For HSR and RPU interlaces

0 for HSP and Tape interlaces. Will always produce a two part interlace.

aa is the absolute address of the band and must be an even number.

00 is always coded as 00.

4. There is no limit to the number of Interlace Cards that may be used.

5. Each Interlace Card may contain up to seven entries. Any card containing less than seven entries must have a sentinel word (99 9999 9999) in the first invalid field to stop card processing.

6. The X-6 Assembly System does not distinguish between tape notations T and Z. The functions of these two symbols is to allow the use of up to twenty Tape interlaces by the use of T and Z plus digit n which ranges from 0 through 9.

E. Tables Card, Card Type 5

Function: Specifies the absolute locations to be used by a table or tables.

80 Card Columns	90 Card Columns	Format	Name of Field
1-2	1-2	Δ5	Card Type
3-10	3-10	ΔΔΔΔΔΔΔΔ	Spaces
11-20	11-20	tnΔΔΔΔaaaa	Word 1, Entry 1
21-30	21-30	iiiΔΔΔeeee	Word 2, Entry 1
31-40	31-40	tnΔΔΔΔaaaa	Word 1, Entry 2
	41-45	ΔΔΔΔΔΔ	Spaces
41-50	46-55	iiiΔΔΔeeee	Word 2, Entry 2
51-60	56-65	tnΔΔΔΔaaaa	Word 1, Entry 3
61-70	66-75	iiiΔΔΔeeee	Word 2, Entry 3
71-80	76-90	ΔΔΔΔ....ΔΔΔΔ	Spaces

Technical Notes:

1. Column 2 must contain a 5 punch.
2. Each entry must contain twenty digits coded in the following format:

Word 1	Word 2
tnΔΔΔΔaaaa	iiiΔΔΔeeee

t is the table identification (S, U, or V).

n is the table number (0-9).

aaaa is the absolute location of the first table element.

iii is the interval (or increment) between elements.

eeee is the total number of elements in the table.

3. There is no limit to the total number of Table Cards.
4. A particular Table Card may contain from one to three two-word entries. If it contains less than three entries, word 1 of the next invalid entry must contain a sentinel word (99 9999 9999).
5. Columns 71-80, on the 80 column card, and 76-90, on the 90 column card, are ignored by the X-6 Assembly System.

F. Specifications Card, Card Type 6

Function: Indicates that the next operation to be assembled contains parameters that will lie in the range X 01 through X 20 and specifies the X-6 symbolic address or the absolute address to be substituted for each parameter.

80 Card Columns	90 Card Columns	Format	Name of Field
1-2	1-2	Δ6	Card Type
3-5	3-5	www	Operation No. (or Name)
6-8	6-8	yyy	Card Number
9-10	9-10	ΔΔ	Spaces
11-20	11-20	xΔΔnnsssss	Entry 1
21-30	21-30	xΔΔnnsssss	Entry 2
31-40	31-40	xΔΔnnsssss	Entry 3
	41-45	ΔΔΔΔΔ	Spaces
41-50	46-55	xΔΔnnsssss	Entry 4
51-60	56-65	xΔΔnnsssss	Entry 5
61-70	66-75	xΔΔnnsssss	Entry 6
71-80	76-85	xΔΔnnsssss	Entry 7
	86-90	ΔΔΔΔΔ	Spaces

Technical Notes:

1. Column 2 must contain a 6 punch.

2. Each entry must contain ten digits coded in the following format:

   xΔΔnnsssss

   xΔΔnn is the generalized parameter.

   sssss is the address (symbolic or absolute) to be substituted.

3. Necessarily, sssss must be some kind of tag line or absolute memory address.

4. The total number of parameters allowed in the subroutine is twenty. However, there is no restriction upon how many Specifications Cards are used. For example, twenty cards with one entry each might be used or four cards with five entries each.

5. Each card may contain from one to seven entries. Any card containing less than seven, however, must contain a sentinel (99 9999 9999) in the first invalid entry field.

6. A new specifications card may be introduced only at the beginning of a new operation and must precede the Header Card.

7. Information provided on the Specifications Card is retained until the next operation begins.

G. Operation Header Card, Card Type 7

Function: Specifies the number or name of the operation to be assembled. Serves to set counter for processing of Type 8 Cards which will follow:

80 Card Columns	90 Card Columns	Format	Name of Field
1-2	1-2	Δ7	Card Type
3-5	3-5	www	Operation No. (or Name)
6-8	6-8	yyy	Card Number
9-30	9-45	ΔΔΔΔ...ΔΔΔΔ	Spaces
31-80	46-85	zzzz...zzzz	Descriptive Comments
	86-90	ΔΔΔΔΔ	Spaces

Technical Notes:

1. Column 2 must contain a 7 punch.

2. The card number is stored and becomes the base for the counter used when processing Type 8 Cards. Thus, the card number may be any three digit number; however, for the most flexibility as a counter base, it is usually 000 or 001.

3. The Descriptive Comments are printed without alteration.

4. An output card will not be produced by the Operation Header Card.

5. An Operation Header Card must precede each operation to be assembled.

H. Detail Card, Card Type 8

Function: Contains the object program coding that will be assembled by the X-6 Assembly System Program.

80 Card Columns	90 Card Columns	Format	Name of Field
1-2	1-2	Δ8	Card Type
3-5	3-5	www	Operation Number (or Name)
6-8	6-8	yyy	Card Number within Operation
9-10	9-10	ΔΔ	Spaces
11-15	11-15	aaaaa	Symbolic a Address
16	16	x	Control Code
17-19	17-19	000	Symbolic Operation Code
20	20	Δ	Space
21-25	21-25	mmmmm	Symbolic m Address
26-30	26-30	ccccc	Symbolic a Address
	31-45	ΔΔΔΔ...ΔΔΔΔ	Spaces
31-80	46-85	zzzz...zzzz	Descriptive Comments
	86-90	ΔΔΔΔΔ	Spaces

Technical Notes:

1. Column 2 must contain an 8 punch.

2. The Detail Cards must be numbered in sequence beginning one number higher than the card number appearing on the Header Card for the operation.

3. Only Columns 6-8 are extracted for the card number. Therefore, columns 9 and 10 should not be used as part of the card number, even though no other use is made of them.

4. The Control Code, column 16, signals that conditions are associated with the instruction. These conditions are of three categories: Index Registers, negative constants, and alphabetic constants.

   The code used may be one of the following:

   a. Δ if the instruction requires no specific control information.

   b. 2 for a negative constant.

   c. 1,2, or 3 if an Index Register is to be specified.

   d. U for the Unprimed portion of a two part alphabetic 90 column Card.

      P for the Primed portion of a two part alphabetic for 90 Column Card.

   e. U for the Unprimed portion of three part alphabetic for 80 Column Card.

P for the Primed portion of a three part alphabetic for 80 Column Card.

D for the Duoprimed portion of a three part alphabetic for 80 Column Card.

f. N for the Numeric portion of a two part alphabetic for 80 or 90 Column Card (machine code).

Z for the Zone portion of a two part alphabetic for 80 or 90 Column Card (machine code).

5. An alphabetic constant, to be properly entered, should be on two or three cards, depending on whether it is to be two or three part image. These cards would contain identical information, but the part of the image that was loaded would depend upon the control code in column 16. Each card would be numbered in ascending sequence.

6. Column 20 is not used.

7. Refer to the section on Coding for a discussion of the a, m, and c address possibilities.

8. The Descriptive Comments are printed without alteration.

9. Since the function of the X-6 Assembly System is to process Detail Cards, these cards must occur in any symbolic deck to be assembled.

I. Operation Sentinel Card, Card Type 9

Function: To advance the paper to the beginning of the next page so that the record of each operation is distinctly separated on the output listing, and to clear the storage tables containing temporary tags and specifications information.

80 Card Columns	90 Card Columns	Format	Name of Field
1-2	1-2	Δ9	Card Type
3-5	3-5	www	Operation Number (or Name).
6-8	6-8	yyy	Card Number within Operation
9-30	9-45	ΔΔΔΔ...ΔΔΔΔ	Spaces
31-80	46-85	zzzz...zzzz	Descriptive Comments
	86-90	ΔΔΔΔΔ	Spaces

Technical Notes:

1. Column 2 must contain a 9 punch.

2. The Operation Number or name must be the same as that given to the Type 8 cards of the operation.

3. The card number must be one more than the card number of the last Type 8 Card.

4. The Descriptive Comments are printed without alteration.

5. An Operation Sentinel Card must succeed the last Type 8 Card of each operation to be assembled.

J. End of Run Sentinel Card, Card Type 10

Function: Signals that all of an object program has been processed. The computer will be brought to an orderly halt.[5]

80 Card Columns	90 Card Columns	Format	Name of Field
1-2	1-2	10	Card Type
3-15	3-15	ΔΔΔΔ...ΔΔΔΔ	Spaces
16	16	x	Control Code
17-19	17-19	000	Symbolic Operation Code
20	20	Δ	Space
21-25	21-25	mmmmm	Symbolic m Address
26-30	26-30	ccccc	Symbolic c Address
	31-45	ΔΔΔΔ...ΔΔΔΔ	Spaces
31-80	46-85	zzzz...zzzz	Descriptive Comments
	86-90	ΔΔΔΔΔ	Spaces

Technical Notes:

1. Columns 1 and 2 must contain a 1 and 0 punch respectively.

2. All entries on the card from column 16 through the last column follow the same rules as the Detail Card, Card Type 8.

3. The symbolic instruction contained on the End of Run Sentinel Card will be translated and punched on an output sentinel card (it is assumed that the program deck produced by an X-6 assembly will be loaded by a PTA routine. These routines require the sentinel card to contain the first instruction of the object program).

4. The Descriptive Comments are printed without alteration.

5. Every object program assembled must contain an End of Run Sentinel Card.

---

[5] The final stop is 67 8888 cccc (cccc being the first a address of the X-6 Assembly System Program).

OUTPUT CARD FORMAT

The cards produced by the X-6 Assembly System are the machine code equivalent of the X-6 Symbolic input cards. This output format is acceptable to the loading routine. The differences between the X-6 produced card format and the exact PTA01 format are:

Card Columns	X-6 Produced Output Card Contents	Load Routine Input Card Contents
1-5	Five digit program identification from columns 11-15 of the X-6 Label Card, Type 1.	Program Name.
11-16	Operation and card number from columns 3-8 of the X-6 input card.	Page number, line number and suffix.
47-50	Card number in X-6 produced deck.	The PTA routines require a card count on the last card of the input deck only.

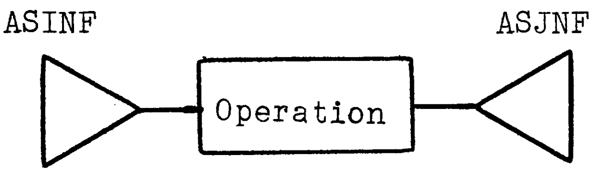
PROGRAMMING PROCEDURES

I. Flow-Charting

The only modifications to standard flow-charting procedures are:

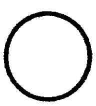
A. Operations should be kept short and well defined.

B. Designations for an operation are shown as:

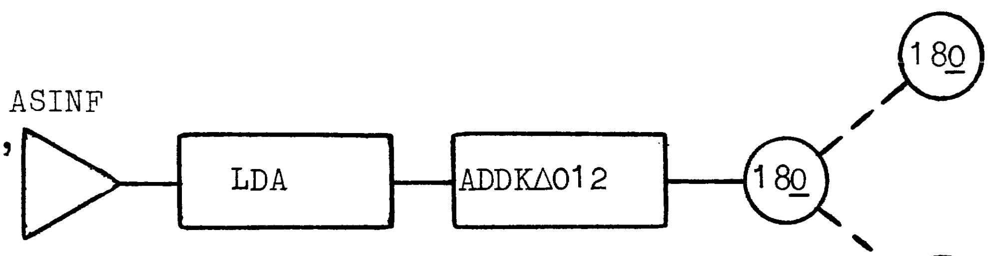


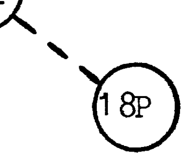
Permanent tags should be assigned to these triangles representing operation (or subroutine) entrances and exits.

C. Communications links within operations are shown as:



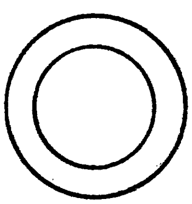
For example, in connectors:



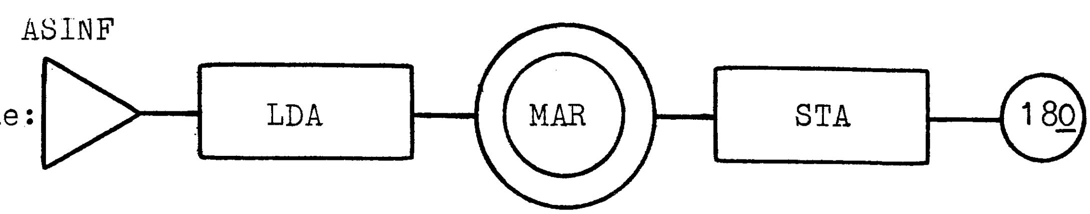


Temporary tags should be assigned to these.

D. Execution of one operation within another operation is shown by:



For example:



E. X-6 symbology should be used in the flow chart. Table and interlace symbols and working storage addresses should be assigned during flow-charting.

II. Coding

When coding, it must be kept in mind that buffer tests are not inserted by the X-6 Assembly System but must be inserted where required during the coding or after the object program is assembled. Accurate estimates for buffer test insertions can be made by consulting the Latency Minimization Section, above. Aside from this, the general rules for X-6 coding are:

A. Start each operation with a "Header" line (see Card Type 7 in Input Card Section, above) on a new sheet of coding paper.

B. Code the main chain of the object program first and then the lesser used branch paths. Since each address is assigned in order of reference during assembly, this technique will produce better minimization.

C. The comments columns should be used liberally since the X-6 produced edited listing will be more valuable for desk checking if full comments are appended. Comments should be limited to numeric and alphabetic characters.

D. A cross reference to the card number on which the instruction line is to be punched should be maintained in the box on the flow chart.

E. Each operation should end with an Operation Sentinel Card (see Input Card Format, above).

F. Initial conditions of all working storages should be coded.

The memory is usually filled with stop orders using PTA01.

PREPARATION FOR THE X-6 ASSEMBLY

1. Have all operations keypunched and verified.

2. Obtain any needed X-6 library routines and prepare specification cards.

3. Prepare card types 1, 2, 3, 4, 5, and 10 if this has not already been done. Be sure to restrict the area used by the standard loading routine.

4. Arrange the input deck in the desired order. If the program is very large, place the most important operations first; they will get better minimization.

5. Sight check the separate operations to make certain that card types 7, 8, and 9 within each operation are identically punched in columns 3-5 (operation number).

6. Either manually or by machine, check that card numbers are ascending within operations with <u>no</u> <u>omissions</u>.

OPERATING INSTRUCTIONS FOR THE X-6 ASSEMBLY

I. Loading and Assembling

1. Load X-6 Program Deck.[1] If the deck is in the three instruction per card format use a PLD routine. If it is in the one instruction per card format use a PTA routine.

2. After X-6 is loaded, or earlier:

   a. Feed blank cards through to all stations of the RPU.

   b. Advance paper in HSP so six free holes show above the paper holding clamps.

   c. Put X-6 input program deck in the HSR.

3. To assemble a program:

   a. Set on continuous, depress general clear, and depress Run button.

   b. Successful stop is 67 8888 cccc.

   c. Error stops are listed on the following pages along with error indications which do not stop the computer.

4. After assembly, the output program deck is complete in Stacker zero of the Read-Punch Unit. Any cards in Stacker one should be destroyed.

5. Check the edited listing carefully, all detected input data errors are coded and tabulated in print word 01 on the listing. These errors must be corrected before desk checking can begin.

6. Print the contents of the memory to preserve the information accumulated during the assembly which will be useful for desk checking.

   The X-6 Memory Layout, see below, can be used to interpret the contents of the memory.

7. The following routines might also be used, after one X-6 assembly, and prior to the next.

   a. An X6LNU routine produces a list of all storage locations not used by the assembled program. This routine should be used after printing the contents of the memory.

---

[1]See X6TLD for instructions to load X-6 instruction tape.

b. An X6LUR routine produces a listing of all storage locations with operation and card number of the program's contents.

II. Error Codes (These appear on listing)

Code	Originates In	Means
A	Permanent Tag Search Routine.	More than 300 permanent tags. Address 9999 has been assigned.
B	Temporary Tag Search Routine.	More than 50 temporary tags. Address 9999 has been assigned.
C	K/W Search Routine.	Address higher than K 299 or W 299 has been requested. 9999 has been assigned.
D	Memory Availability Routine.	No more storage. Have assigned 9999.
E	Memory Availability Routine.	No two consecutive addresses free. Have assigned 9999.
F	Specifications Table Search Routine.	Nothing in specifications table matches this "X" symbolic address. Absolute 9999 has been assigned.
G	Address Analysis Routine.	An incorrect "a" address. Previous instruction had blanks in m or c part. This a should have been blank. This a has been processed properly - the previous line must be fixed.
H	Process Action Code Routine.	Spaces in m and c. Spaces in m will be assumed to be in error.
I	Instruction Code Analysis Routine.	Invalid instruction code. The c address will be incremented by 3, a 67 instruction will be punched in the Op portion of the output card.
J	Interlace Availability Routine	Reference has been made to a word part in an interlace which was not properly restricted in summary card type 4. Address of 9999 has been assigned.

III. Stop Codes (in m part of STP order)

Code	Originates In	Means
0001	Get Next Card Routine.	The card being diverted to HSR Stacker 2 has failed to pass read check. Reposition cards and depress Run button to try again.
0002	Get Next Card Routine.	Malfunction in HSR has caused overflow. Fix trouble. Depress Run button to try again.
0003	Main Chain Routine.	No label card (Type 1). Prepare label card. Reposition input deck. Depress Run button to begin again.
0004	Process Specifications Entry.	Too many specifications for current library routine. Depress Run button to proceed. Error code F will appear later.
0005	Print Routine.	Malfunction in printer has caused overflow. Fix trouble. Depress Run button to print current line. (It was PRN order that caused it).
0006	Punch Routine	Malfunction in RPU. Fix trouble. Depress Run button to execute punch order.
0007	Main Chain Routine	Card type sequence error. Check last card read. If it is a type 7 card, depress Run button to get to next stop order. Go to c to process card. If it is type 8, go to m of next stop order.
0008	Process Detail Card Routine.	Operation number on detail card is incorrect. Depress Run button and machine will stop on 67 order. Go to m to process card. Go to c to get next card.
0009	Process Detail Card Routine.	Card number on detail card incorrect. Same action as 0008 Stop.
8888	Main Chain Routine.	Final successful stop. Reload last 100 cards of X-6 deck and follow normal operating instruction before depressing Run button if new assembly is wanted.

III. Stop Codes (in m part of STP order cont.)

Code	Originates In	Means
0010	Main Chain Routine.	Previous card was type 9, card now being processed is not a type 7 or 10 card. Depress Run button. If card last read is to be processed as type 10 card go to the c address of this order. If it is to be processed as a type 7 or 8 card, go to the m address. This will transfer control to another stop order. Now if the card to be processed is a type 7, go to the c address of this stop order. If it is to be processed as a type 8 card, to m address.

IV. X-6 Storage Layout

A listing of the memory at the end of a successful assembly is desirable for desk checking and patching of object program.

Location	Name	Use
0800	Table S8	Valid mnemonic codes stored 20 words apart.
0816	Table S9	Information words for each mnemonic code stored 20 words apart.
2110-2117	Table V3	Two or three part interlace word position for Q.
2118-2130	Table V4	Two part interlace word position for P.
2100-2109	Table S5	Interlace origins (from card type 4).
2200 Band	02 Interlace	Repunching of output cards which fail read check.
3250-3299	Table S3	Temporary tags with absolute addresses. Cleared after every operation. No value after complete assembly.

Location	Name	Use
2450-2465	Table V2	Two and three part interlace word positions for H and R.
2470-2479	Table S6	Interlace origins (from card type 4).
2480-2509	Table S7	Table origins and increments (from card type 5).
2520-2539	Table V1	X-6 equivalents for last set of specifications.
2540-2559	Table VO	Specifications. Cleared after every operation. No value after complete assembly.
2800-3099	Table S4	K and W addresses and absolute addresses are stored as follows: 2800 KO and WO as OKKKKOWWWW 2801 K1 and W1 as OKKKKOWWWW
3100-3249	Table S2	Address of permanent tags in same order as Table S1, stored as: OaaaaOaaaa. Left half-words used for first 150 tag-addresses, then right half-words are filled.
3300-3599	Table S1	Permanent tags. The 5 character alpha-numeric tag is stored as zzzzznnnnn. One tag per word.
3600-3799	Table SO	Storage availability. Each word of table represents a band relative address, 0-199. The 20 bits in the left half-word are zero for unused or 1 for used representing the 20 standard access bands. The 20 bits in the right half of words 3600-3649 represent high-speed access storage. Addresses 4000, 4050, 4100 and 4150 are included in first digit of right half-word. Right half of words 3650-3799 are unused.
3800 Band	PO Interlace	Header for X-6 listing.
4000 Band	HO Interlace	High-Speed Reader read-in area.
4200 Band	01 Interlace	Output punching area.

Location	Name	Use
4200 Band	R0 Interlace	Read-Punch Unit read in area.
4400 Band	P1 Interlace	Detail lines for X-6 listing.
0000-0199	Restricted	Used to load X-6 and later filled with memory print routine.

APPENDIX I

Operations and Subroutines within the X-6 Assembly System Program.

AAR - Address Analysis Routine - Analyzes the five character address in the a, m, or c portion of an instruction to determine which lower level subroutine should be used for processing.

ACO - Action Code Routine - After the PDC path has been completed, ACO continues the processing of instructions containing operation codes belonging to the Action Code O group.

AC1 - Action Code 1 Routine AC2 - Action Code 2 Routine AC3 - Action Code 3 Routine AC4 - Action Code 4 Routine AC5 - Action Code 5 Routine	Same as ACO except that processing is done for a different Action Code group in each case.

CAR - Clock Adjustment Routine - Updates the clock to the new relative band level after an address assignment.

CEP - Edit c for Print Routine - Edits the c address for printing.

CON - Process Constants Routine - Converts the mnemonic control indicators into computer code keys.

CPI - Clear Print Interlace Routine - Clears print interlace 1.

EDS - Edit a, m, or c routine - Edits the a, m, or c address prior to processing. EDS includes the subroutines: EDA, EDM, EDC.

EMP - Edit m for Print Routine - Edits the m address for printing.

EDX - Edit X routine - Establishes the Tentative Next Best Band Relative Address for clock option.

FIE - Further Input Edit Routine - Provides additional input editing for card types 2 through 6.

GNC - Get Next Card Routine - Obtains next card image from HSR.

GNE - Get Next Entry Routine - Provides next entry from card types 2 through 6.

IA1
IA2
IA3
IA4
IA5 } Interlace Routines - Used by Input/Output interlace routines to determine interlace locations.

IAH - RPU Interlace Routine - Converts a symbolic reference to an HSR interlace address to its real address equivalent.

IAO - RPU Output Interlace Routine - Converts a symbolic reference to an RPU punch interlace address to its real address equivalent.

IAP - Printer Interlace Routine - Converts a symbolic reference to a printer interlace address to its real address equivalent.

IAR - Reader Interlace Routine - Converts a symbolic reference to an HSP interlace address to its real address equivalent.

IAT - Converts a symbolic reference to a tape word address to its interlace position equivalent.

ICA - Instruction Code Analysis Routine - Examines symbolic instruction codes for validity and obtains the corresponding computer code information word for processing.

IFT - Initial Fill Tables Routine - Initially fills the internal X-6 Assembly tables with proper bit configurations.

KWS - K-Constant Working Storage Routine - Assigns initial location to symbolic Working Storage or K-Constants and obtains this address at time of later symbolic reference.

MAR - Memory Availability Routine - Keeps a record of assigned locations through use of a single bit position-one location table scheme. Also differentiates between Fast and Normal access areas and ensures consecutive location assignments for c+1 conditions.

MC - Main Chain Routines - Provides the main line of logic flow for the X-6 Assembly System. Consists of subroutines: MC1, MC2, MC3, MC4, MC5, MC6, MC7, MC8, MC9, MCX, and MCK.

MLC - Modify Latency Counter Routine - Modifies the Latency Counter when a clock option is detected.

PAP - Print and Punch Routine - Provides additional editing prior to printing and/or punching.

PDC - Process Detail Card Routine - Provides the processing of the X-6 symbolic instructions contained on the Detail Card, Card Type 8.

PIE - Process Interlace Entry - Sets up restricted input/output interlaces as defined on the Interlace Card, Card Type 4.

PRE - Prepare Restrict Entry Routine - Edits restrict entry prior to processing as specified on the Restrict Card, Card Type 2.

PRN - Print Routine - Controls the printer listing of the initial specifications and the parallel listing of symbolic input and computer code instruction output.

PSE - Process Specifications Entry Routine - Processes the specification entries on the Specifications Card, Card Type 6.

PTE - Process Tag Equals Routine - Processes the tag equals entries as defined on the Tag Equals Card, Card Type 3.

PTR - Process Table Restrict Routine - Coordinates the restriction of locations defined in restrict and Table specification entries.

PTS - Permanent Tag Search - Assigns an address when initial reference is made to a permanent tag and locates this address at time of later references. Includes subroutine PTT for filing permanent tag entry in table.

PUN - Punch Routine - Controls punching of X-6 machine coded output instructions.

RES - Restrict Routine - Restricts memory table as entries on Card Types 2 through 5 are processed and as locations are assigned during assembly.

STS - Specifications Table Search Routine - Searches specifications table for an identity when symbolic reference is made to an X-entry.

200 - Band Relative Address Routine - Creates a band relative address from a four digit absolute address.

TAB - Prepare Table Entry Routine - Processes table entry as defined on Table Card, Card Type 5.

TAS - Table Address Routine - Calculates a specific table address when a symbolic table reference is encountered.

TTS - Temporary Tag Search - Assigns an address when initial reference is made to a temporary tag and locates this address at time of later reference. Includes subroutine TTT for filing temporary tag entry in table.

UO2 - Undigit Two Routine - Eliminates space bit configuration when necessary.

UDC - Update Clock Routine - Updates latency clock according to information contained in clock option.

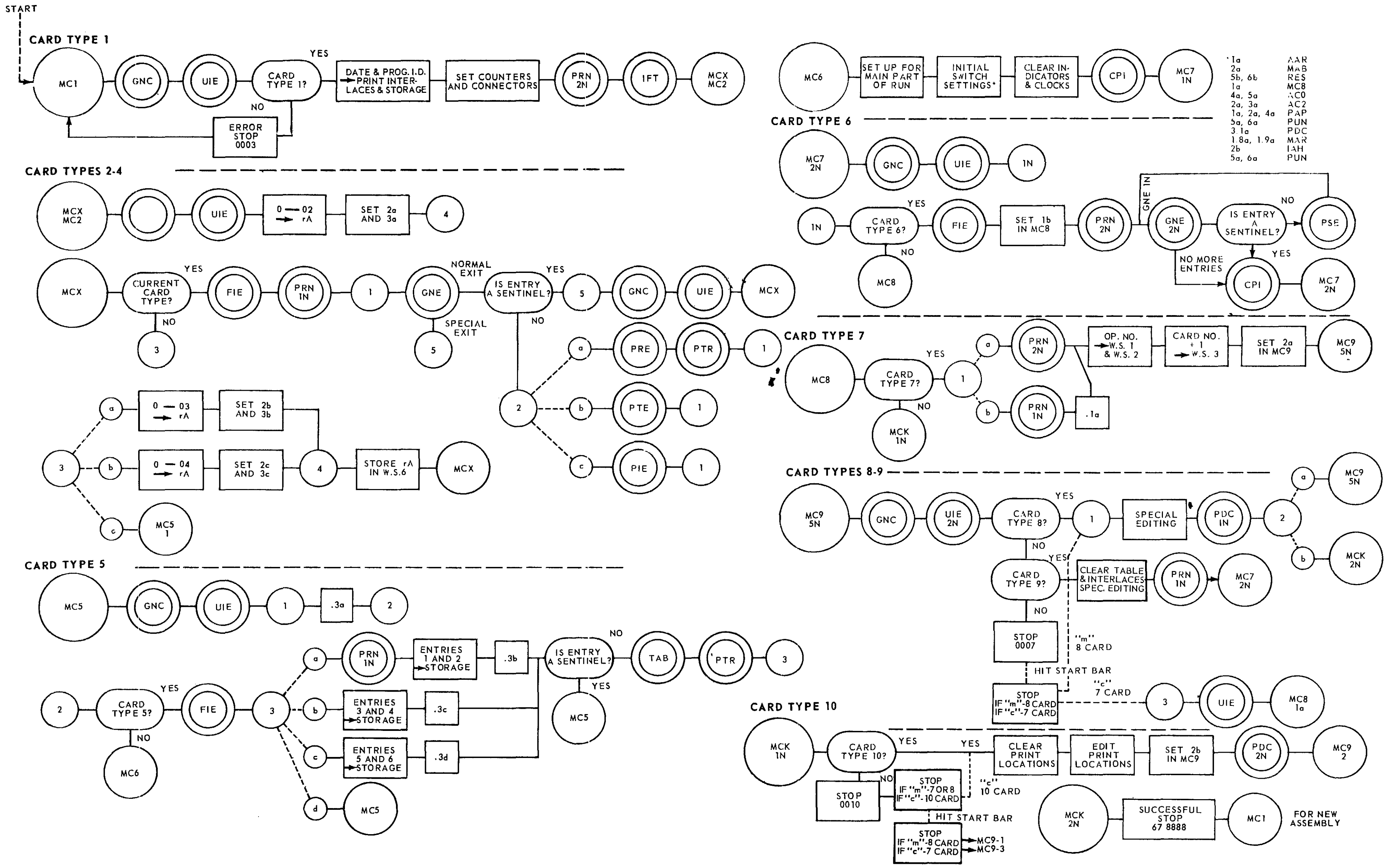
UIE - Universal Input Edit Routine - Edits input card and transfers fields to working storage.

APPENDIX II

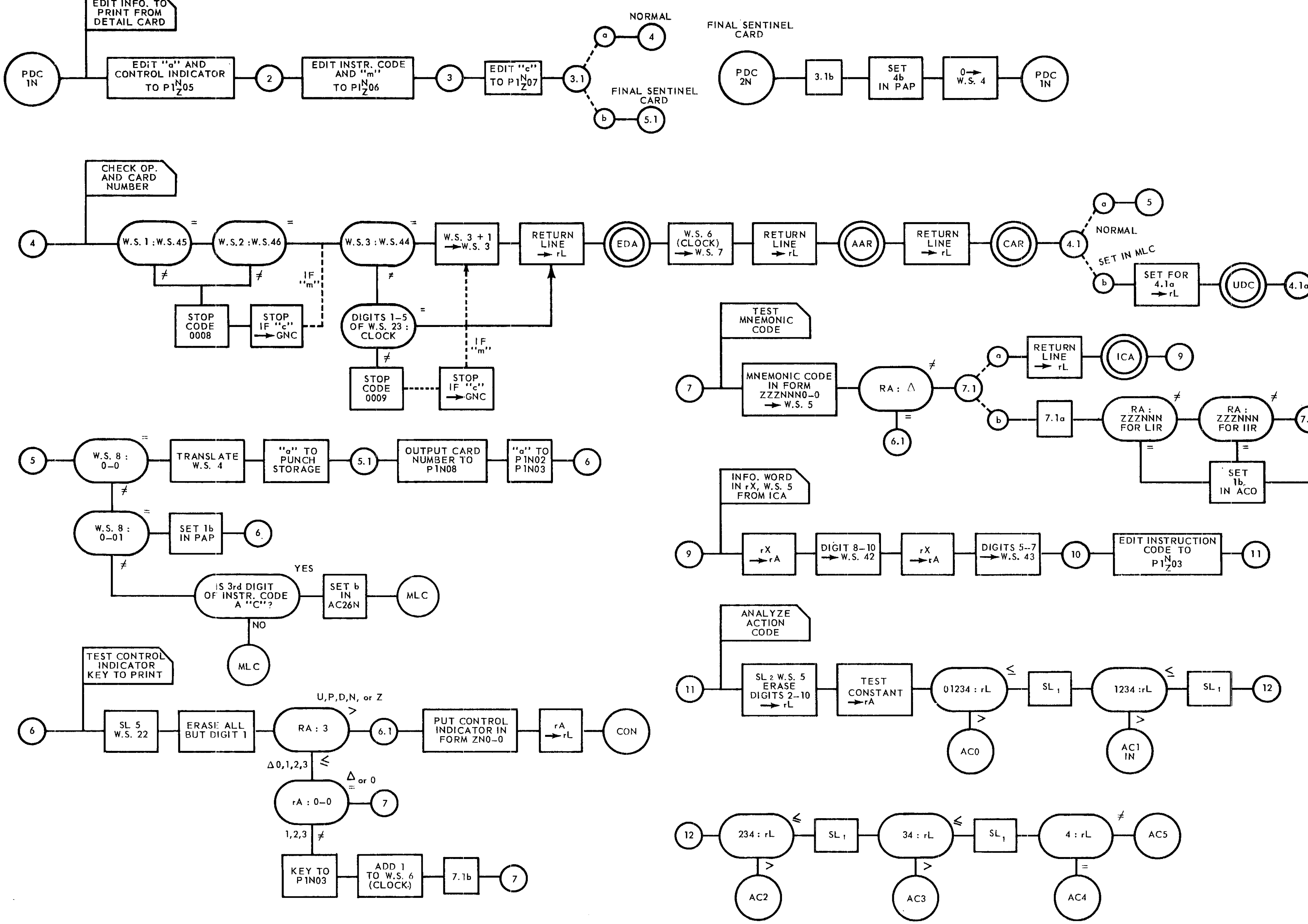
X-6 Assembly System Flow charts

Index To Routines

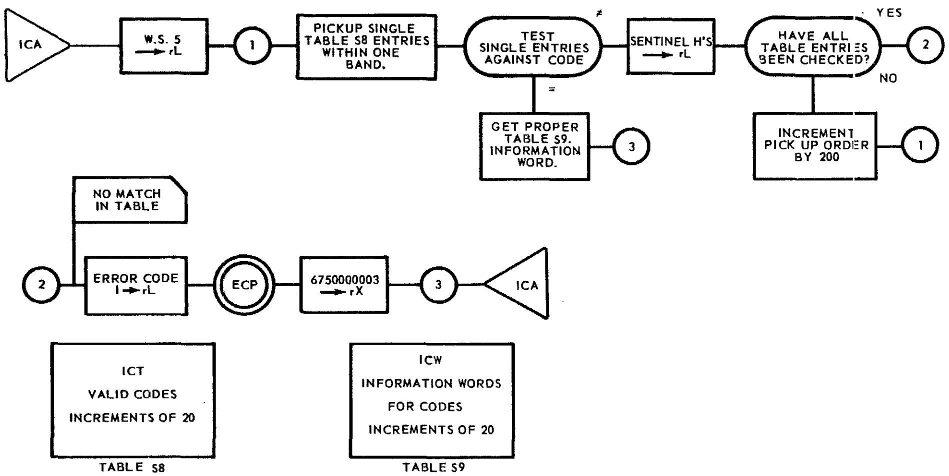
MAIN CHAIN ROUTINES



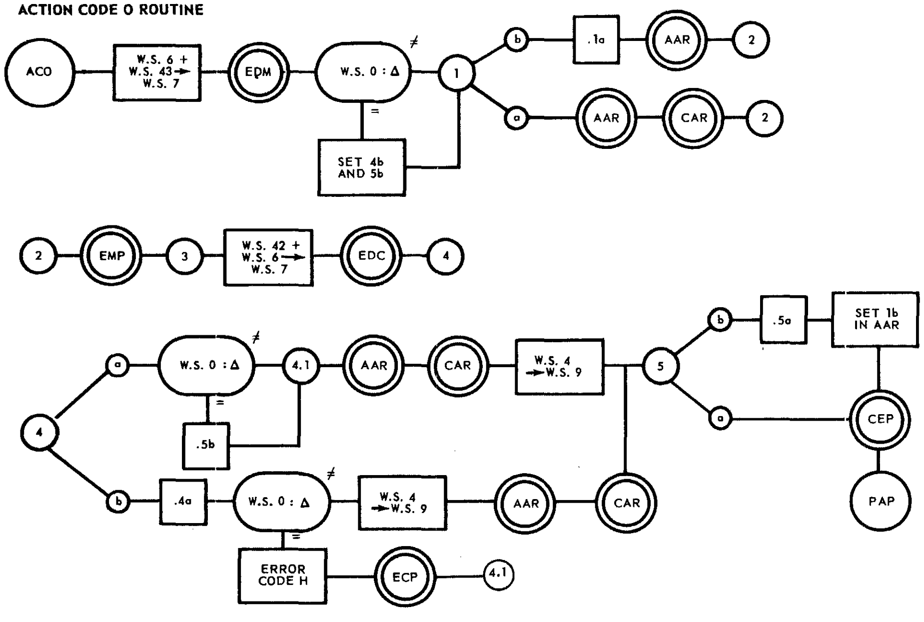
PROCESS DETAIL CARD ROUTINE



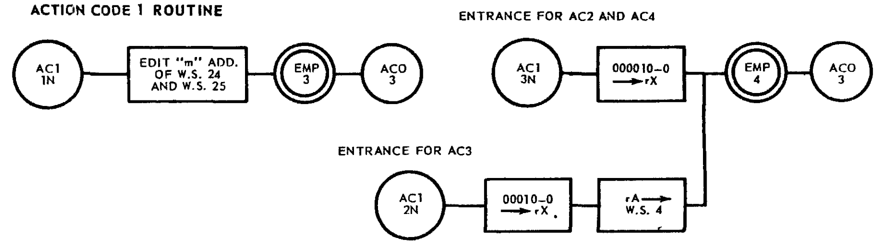
INSTRUCTION CODE ANALYSIS ROUTINE



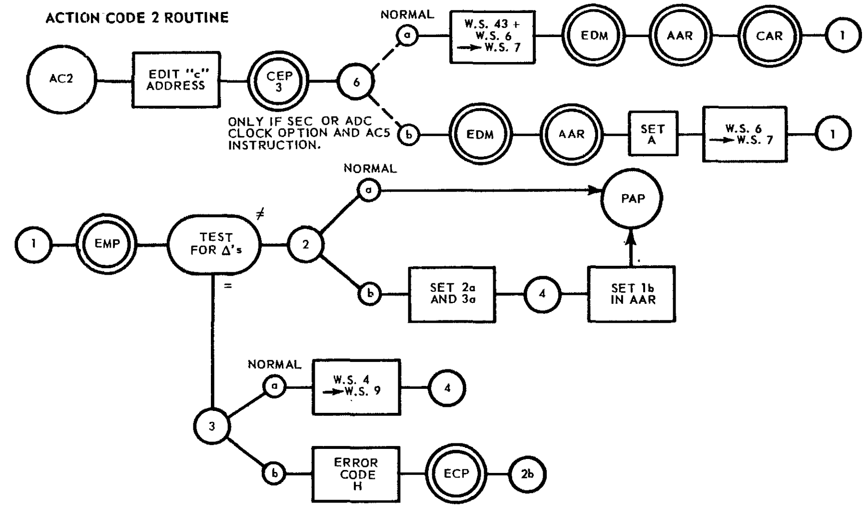
ACTION CODE 0 ROUTINE



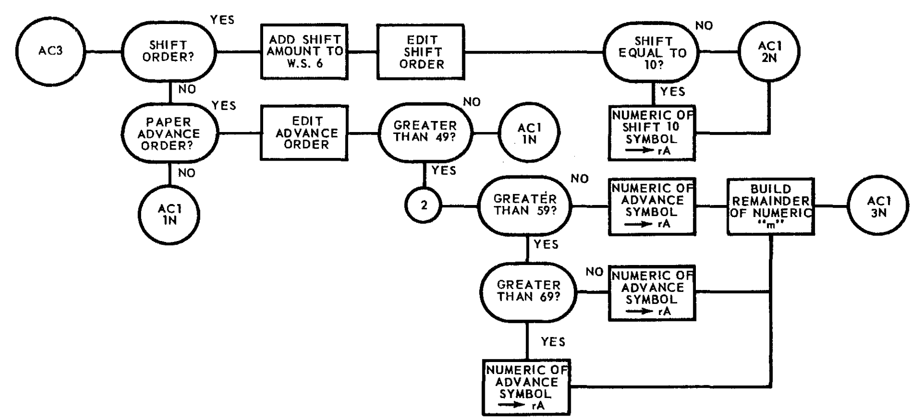
ACTION CODE 1 ROUTINE



ACTION CODE 2 ROUTINE

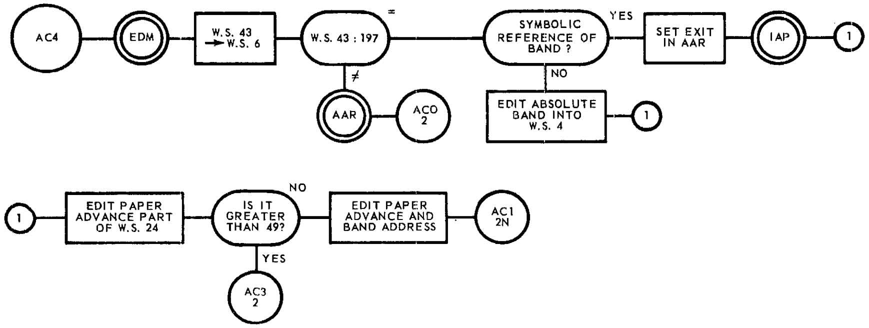


ACTION CODE 3 ROUTINE

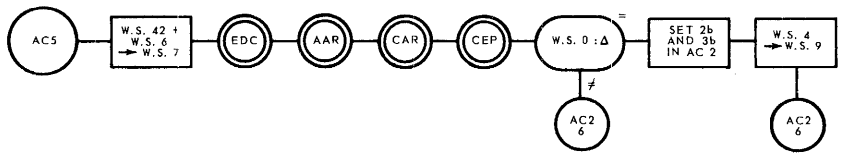


U 1774.1

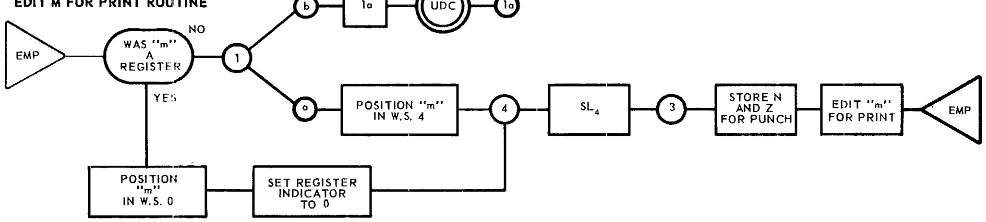
ACTION CODE 4 ROUTINE



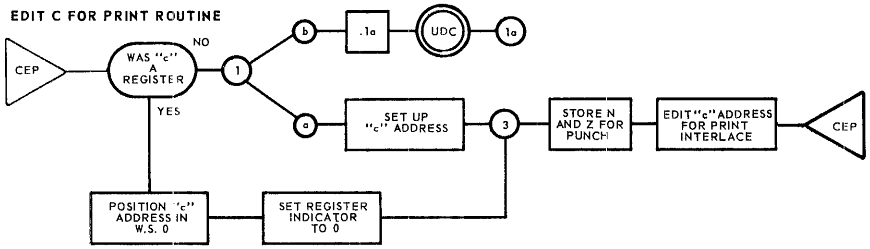
ACTION CODE 5 ROUTINE



EDIT M FOR PRINT ROUTINE



EDIT C FOR PRINT ROUTINE



CLOCK ADJUSTMENT ROUTINE

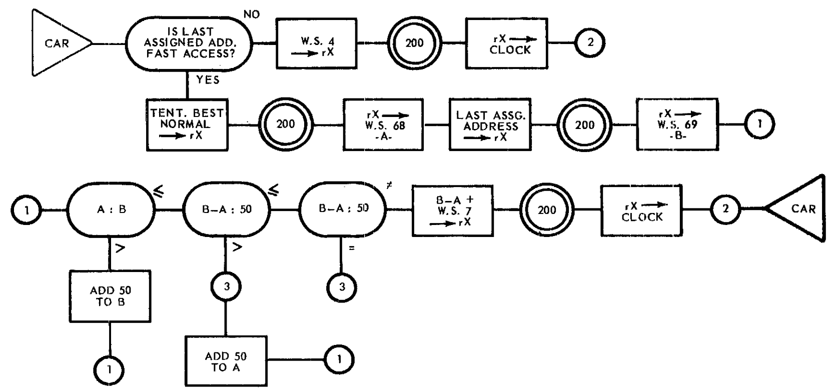
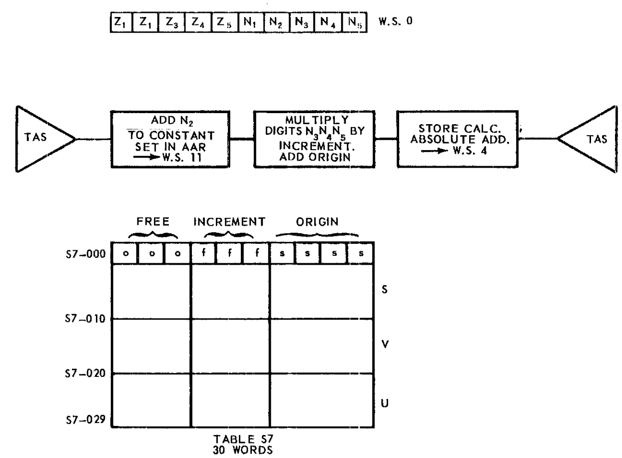
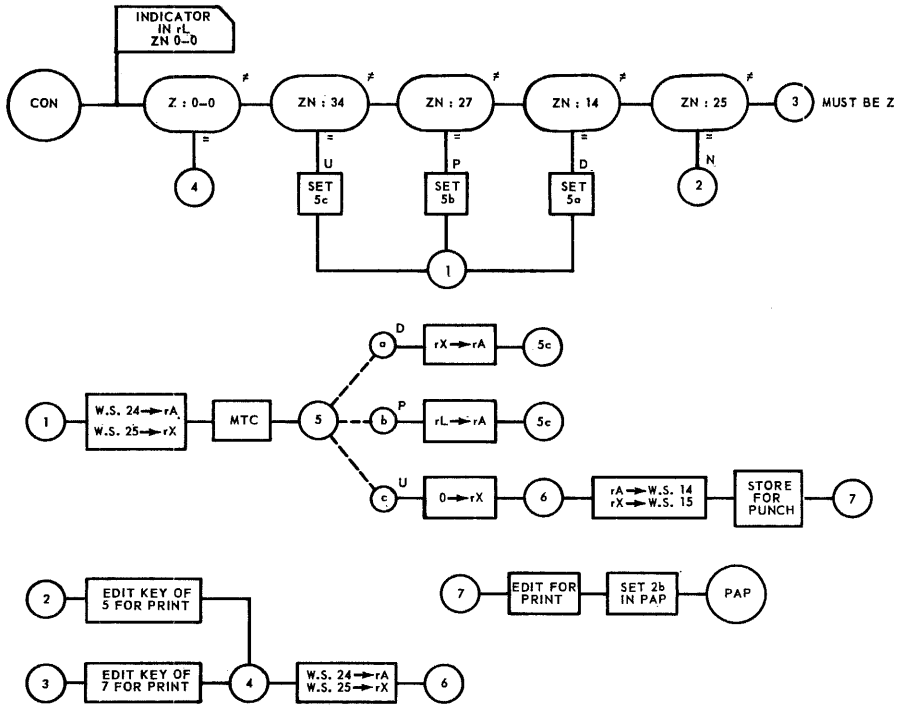


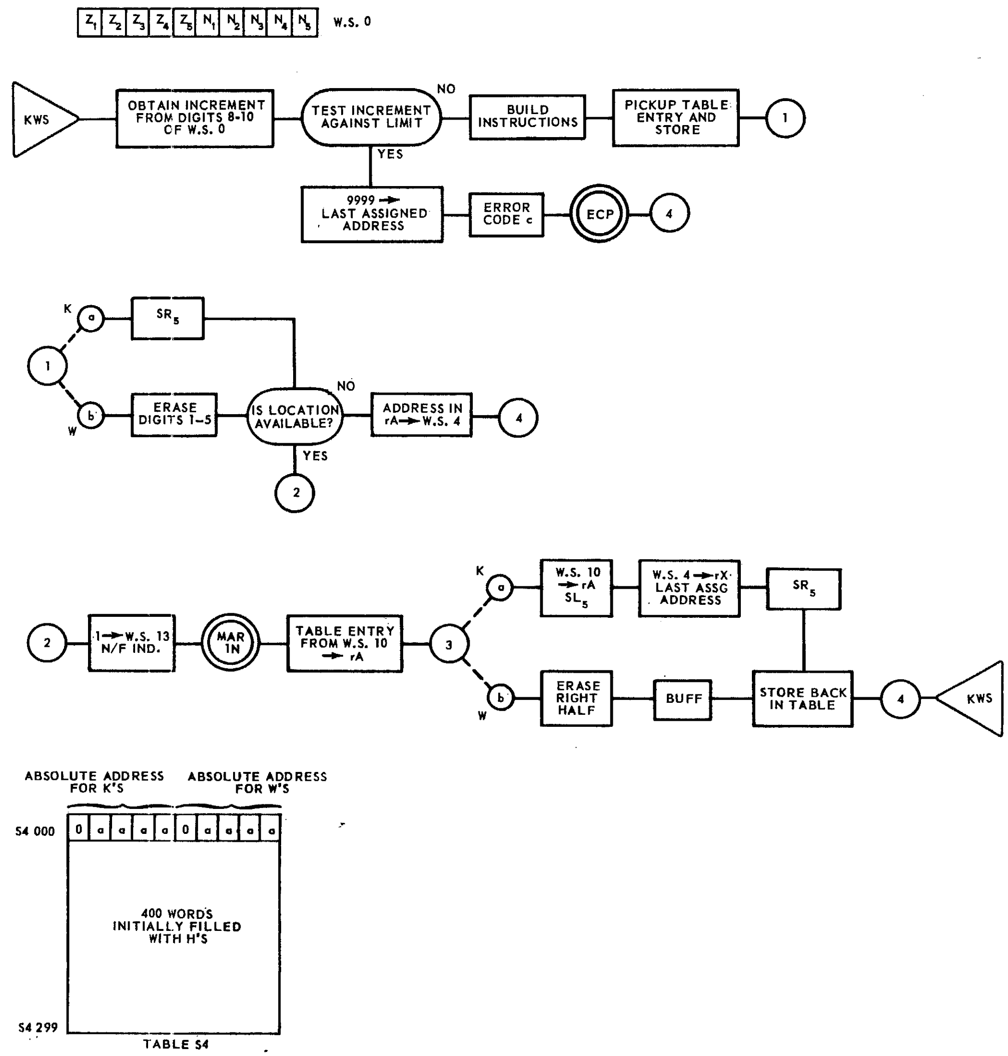
TABLE ADDRESS ROUTINE



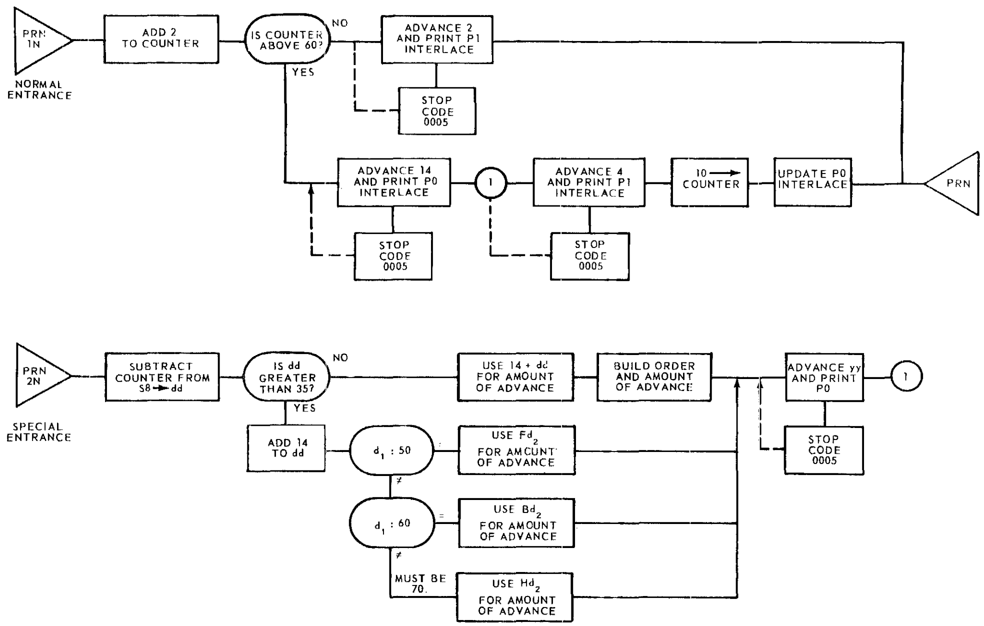
PROCESS CONSTANTS ROUTINE



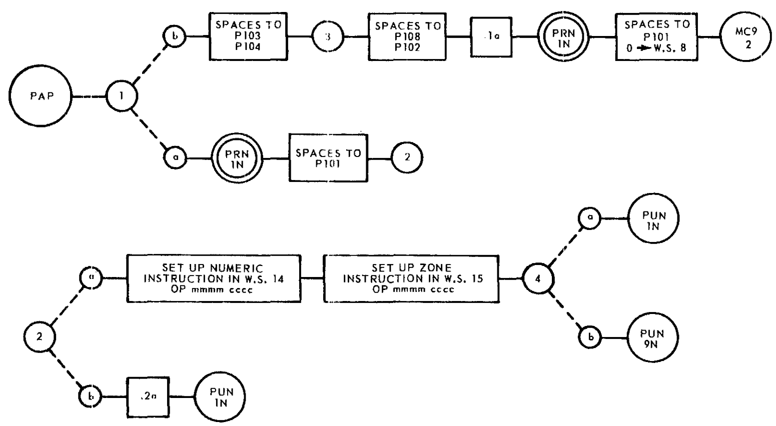
K-CONSTANT WORKING STORAGE ROUTINE



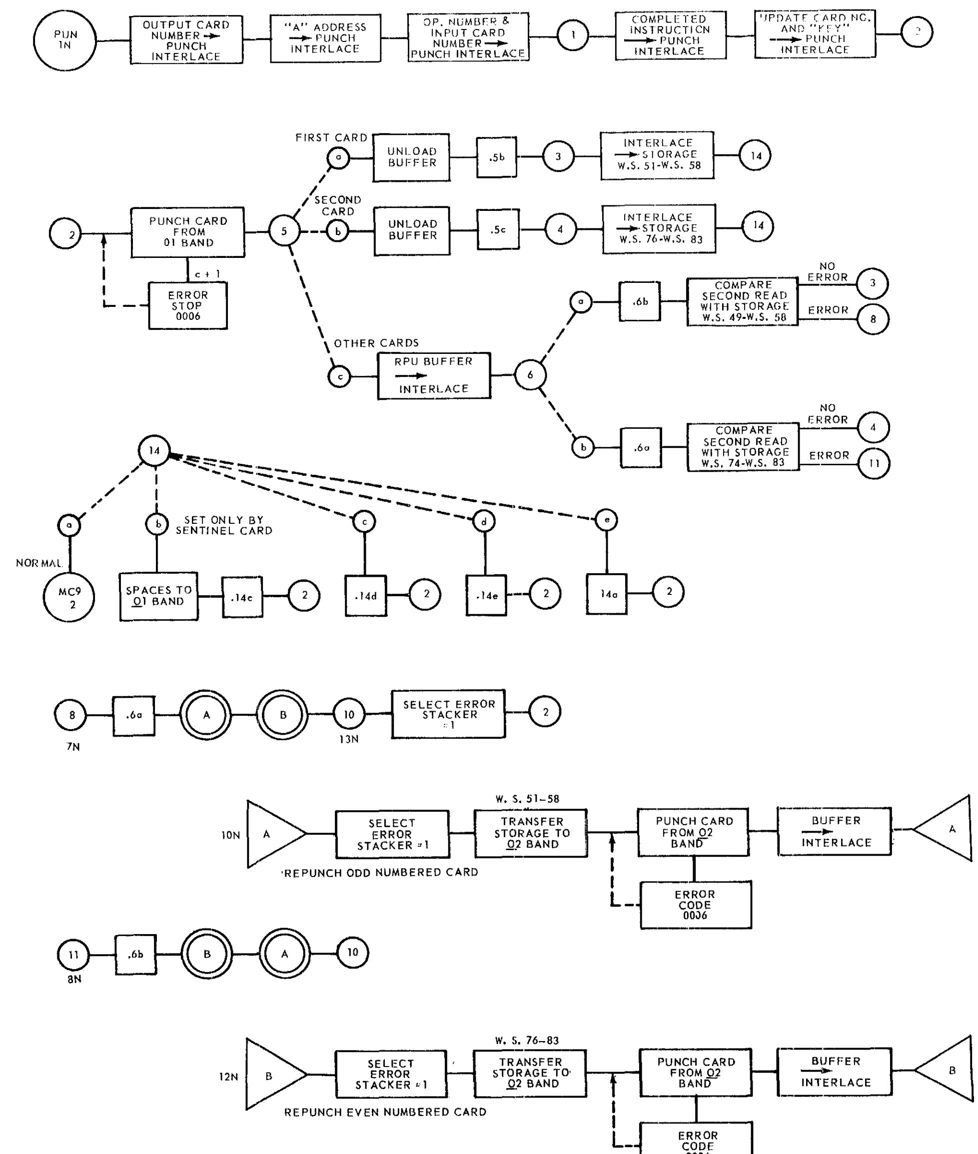
PRINT ROUTINE



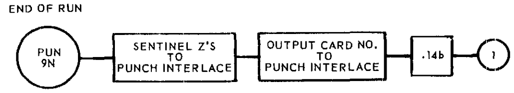
PRINT AND PUNCH ROUTINE



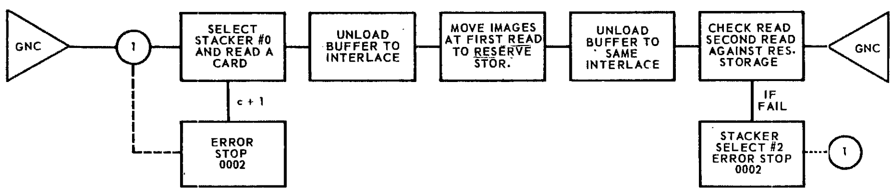
PUNCH ROUTINE



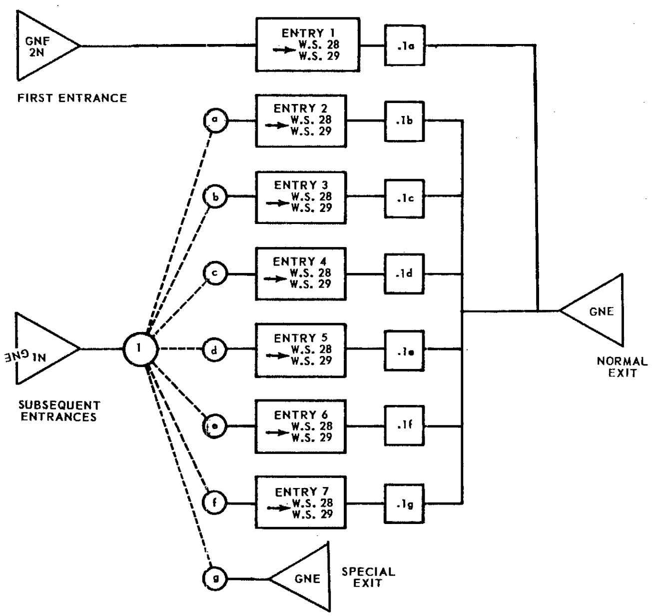
END OF RUN



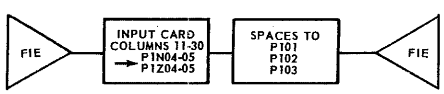
GET NEXT CARD ROUTINE



GET NEXT ENTRY ROUTINE

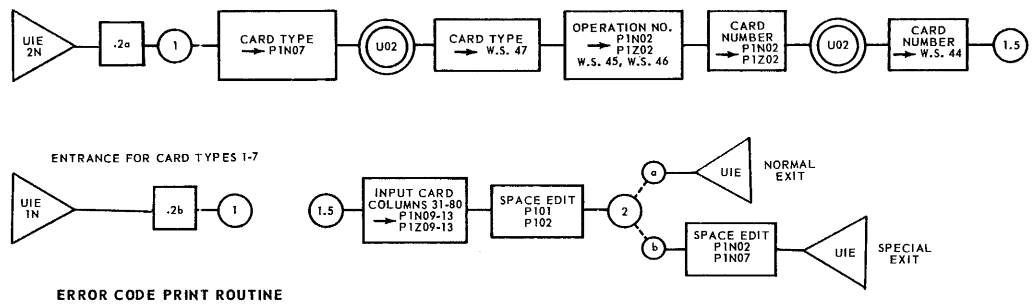


FURTHER INPUT EDIT ROUTINE

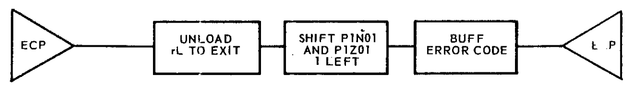


UNIVERSAL INPUT EDIT ROUTINE

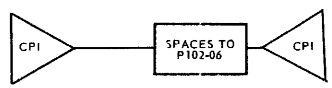
ENTRANCE FOR CARD TYPES 8-9



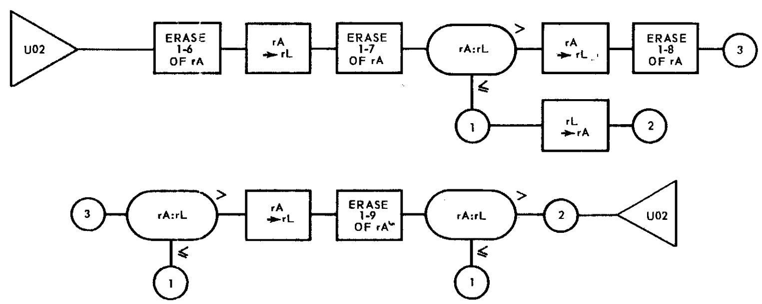
ERROR CODE PRINT ROUTINE



CLEAR PRINT 1 INTERLACE ROUTINE

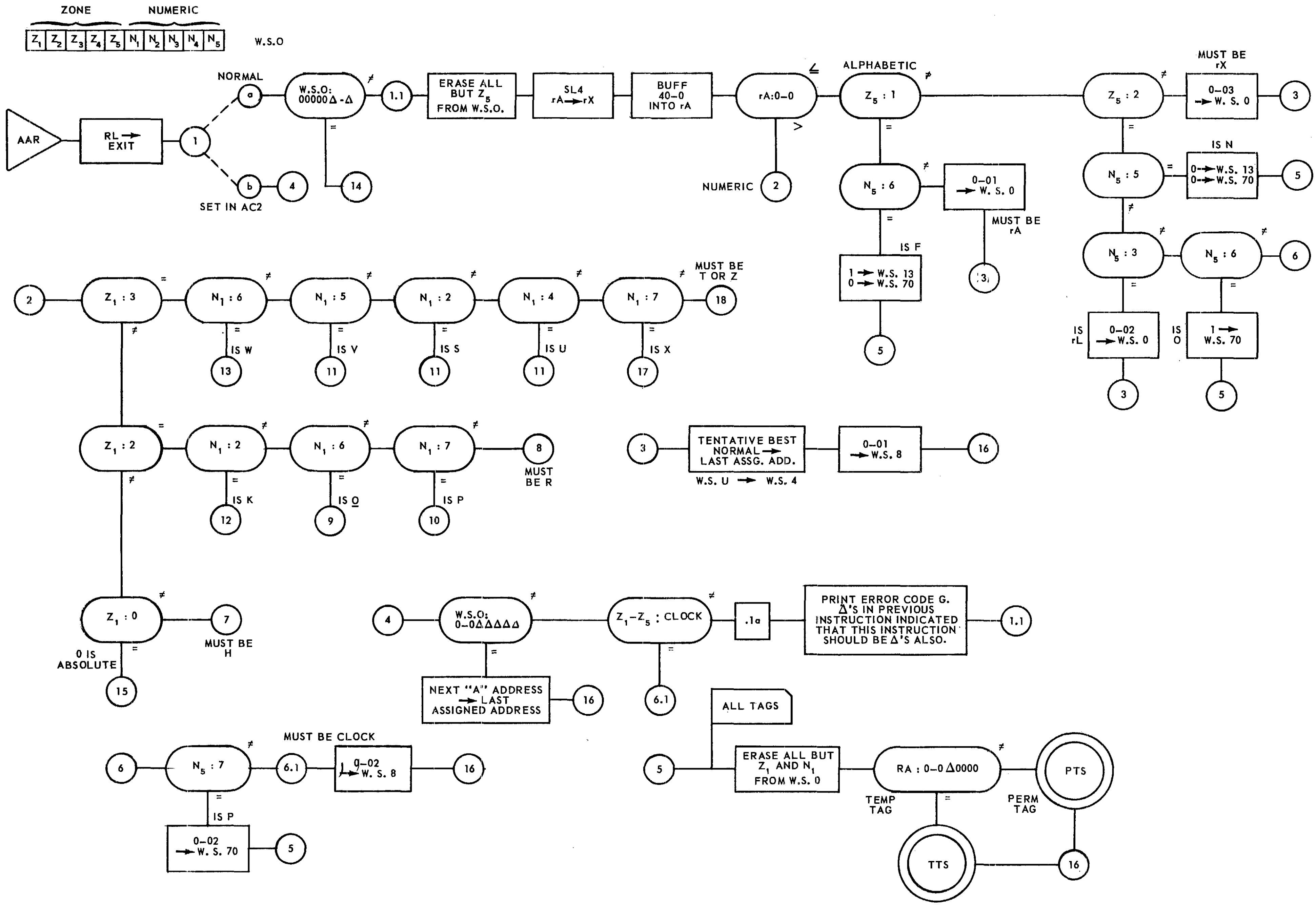


UNDIGIT TWO ROUTINE

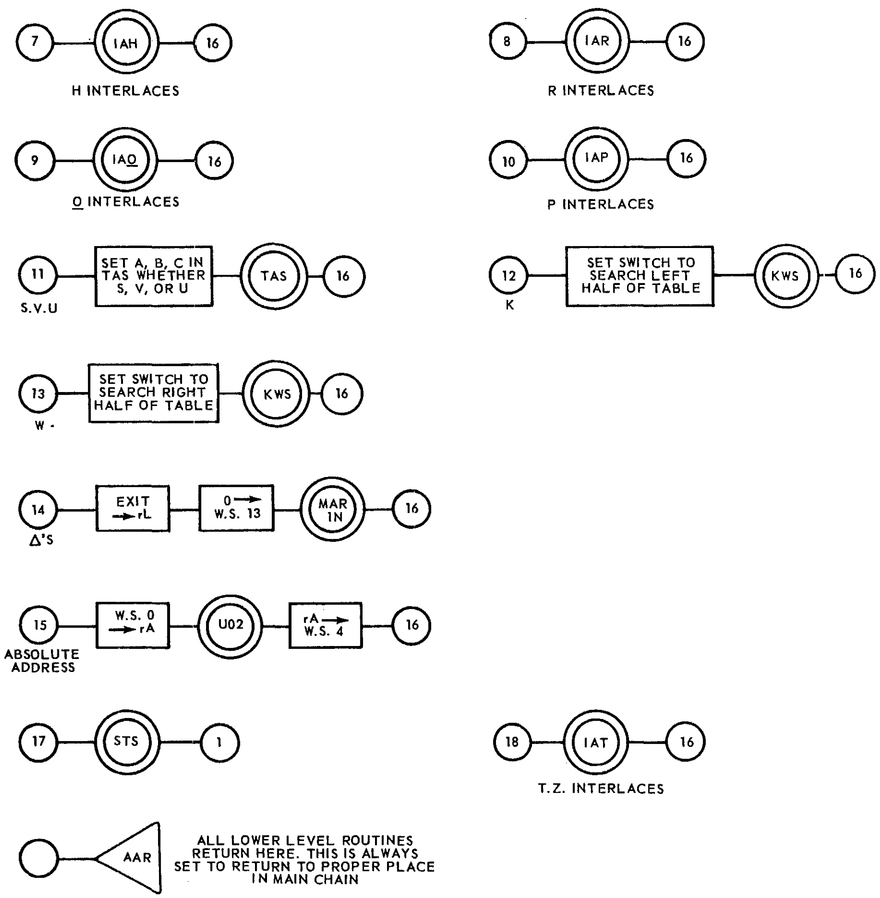


U 1774.1

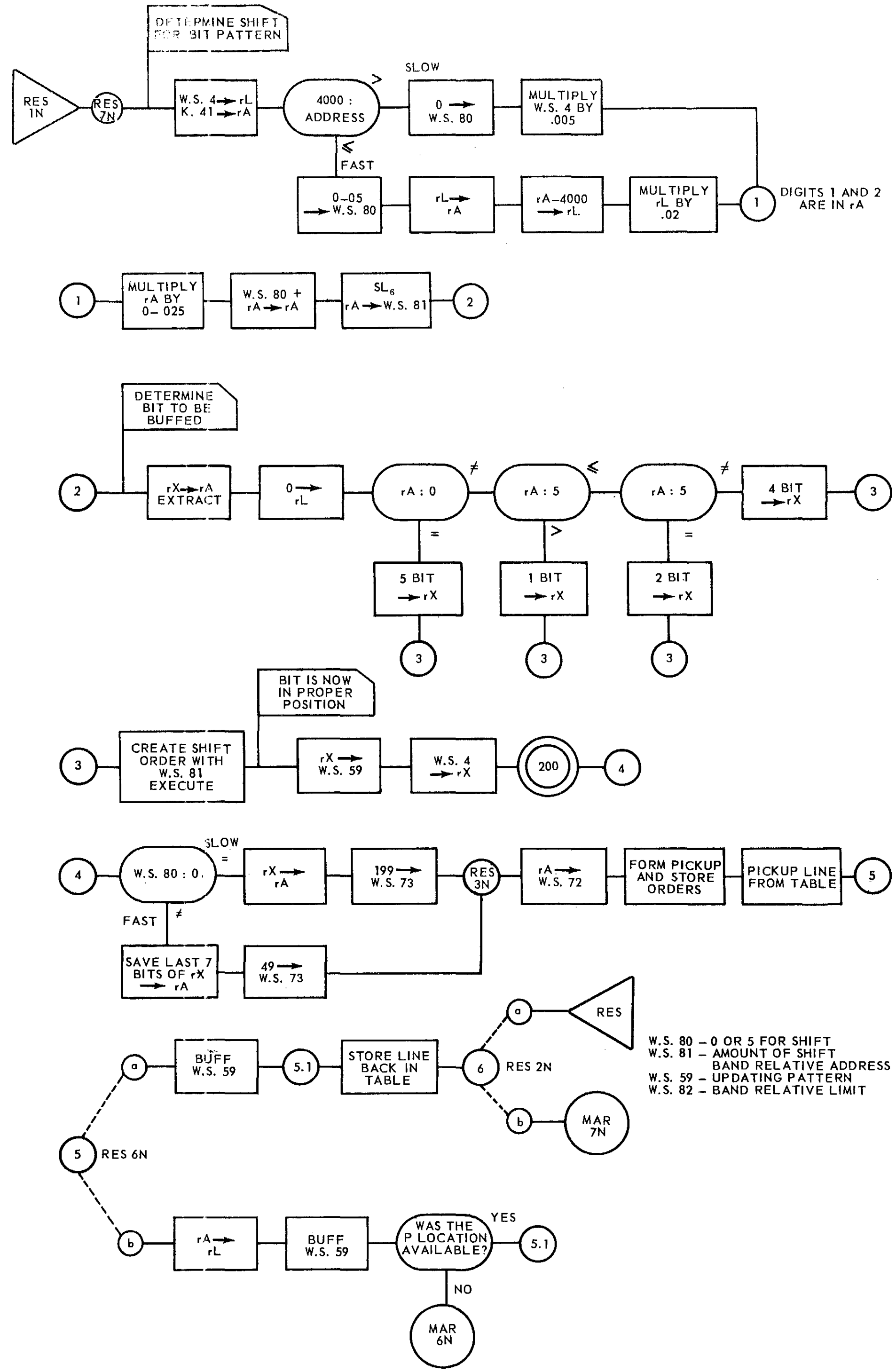
ADDRESS ANALYSIS ROUTINE



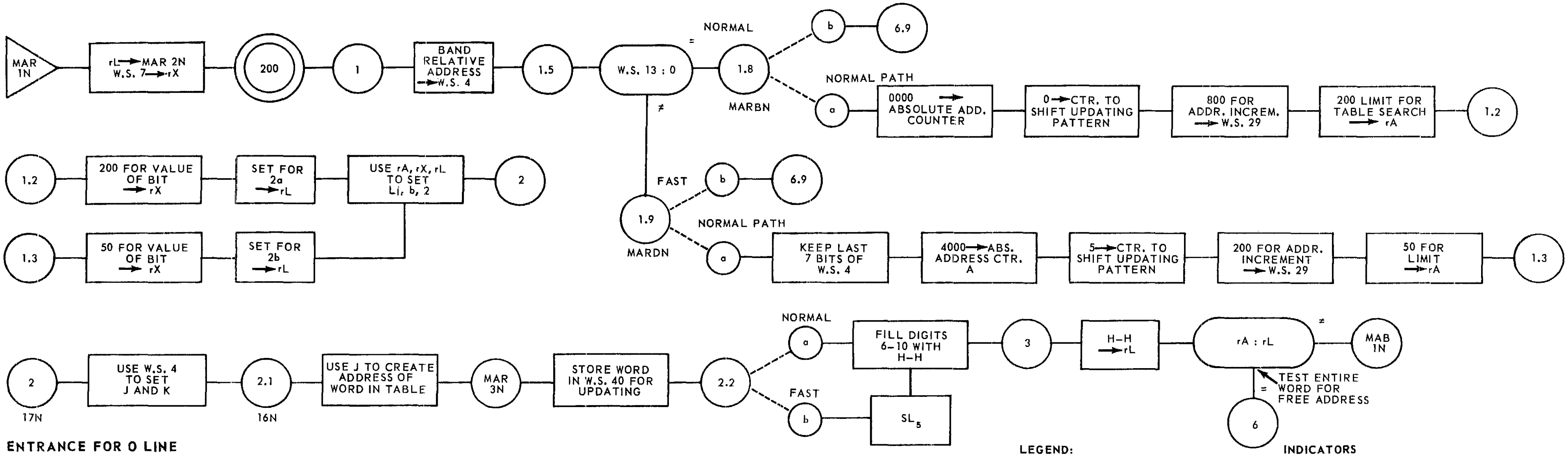
ADDRESS ANALYSIS ROUTINE (CONT'D)

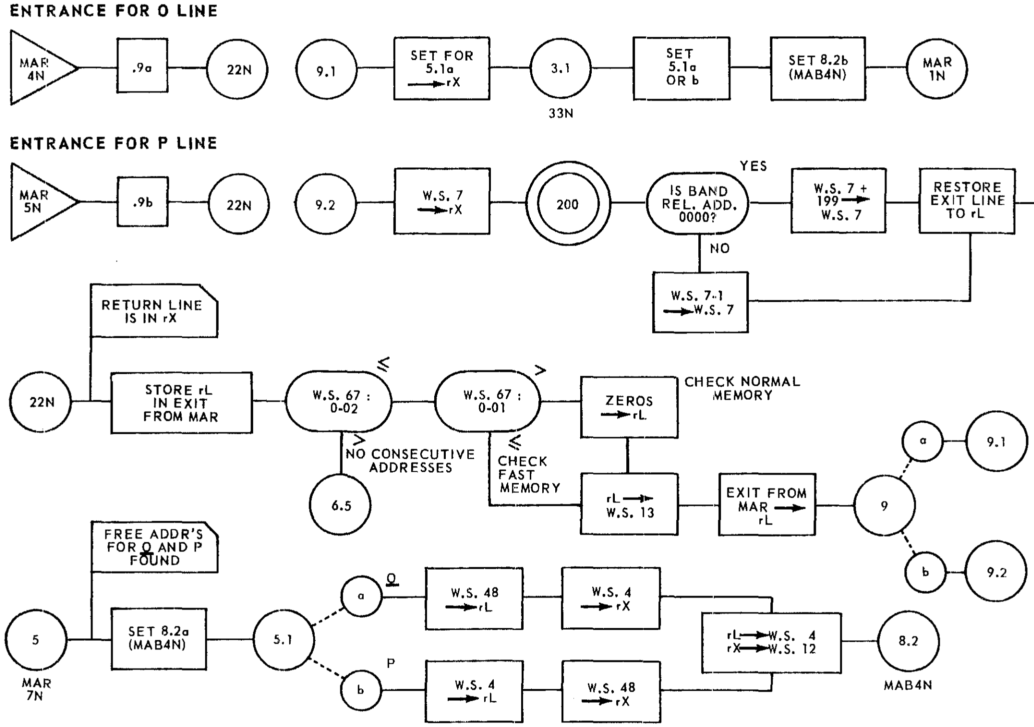


RESTRICT ROUTINE



MEMORY AVAILABILITY ROUTINE - PART ONE



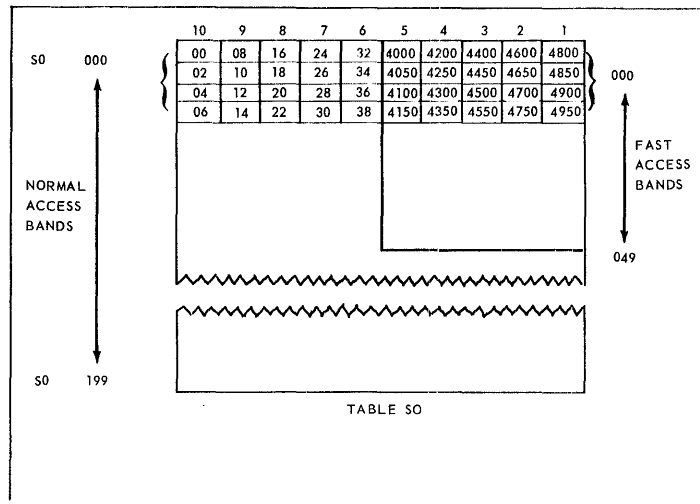


LEGEND:

W.S. 4	Temporary storage at exit: Address assigned to entry being processed
W.S. 7	Tentative best normal band level
W.S. 11	Current table SO word
W.S. 12	For Q and P tags: address of tag which is not being processed
W.S. 28	A – Absolute address counter
W.S. 29	a – Value of a digit in table word
W.S. 38	b – Value of a bit in table word
W.S. 39	j – Band relative address
W.S. 40	Updated table word
W.S. 41	Lj – Limit of table
W.S. 48	Storage for tentative Q address
W.S. 68	K – Counter used to determine when part of storage has been checked
W.S. 69	Bit in position for restricting and updating

INDICATORS

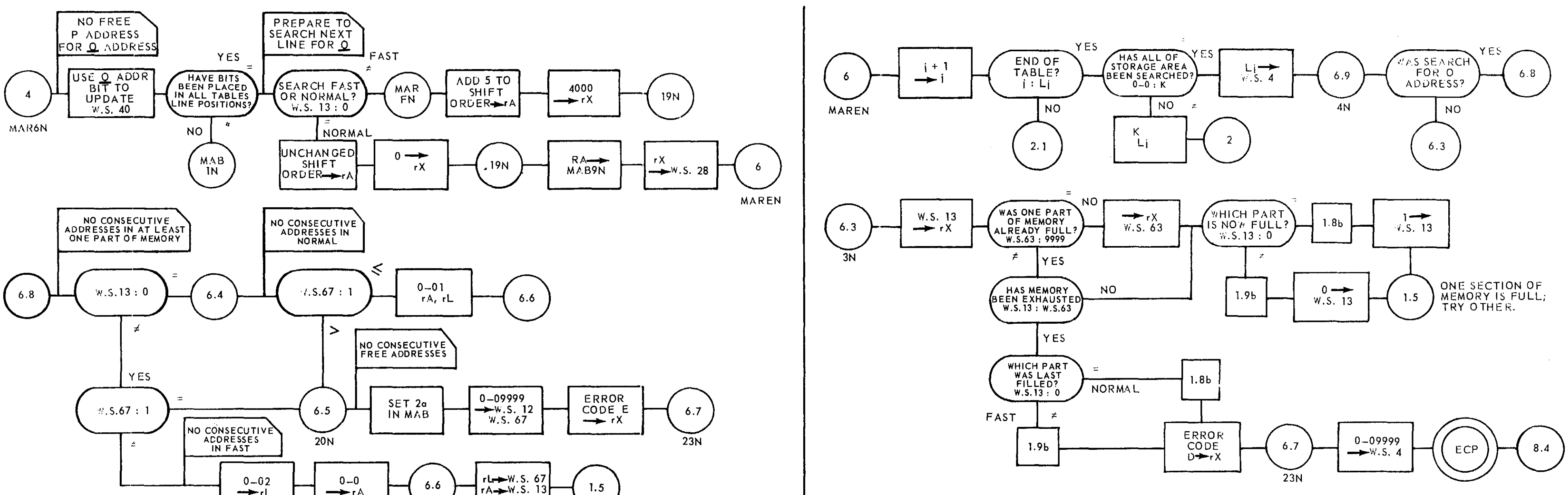
W.S. 13	1 – Fast storage 0 – Normal storage
W.S. 63	9999 – Initial setting 1 – Fast storage gone 0 – Normal storage gone
W.S. 67	0 – Initial setting 1 – No consecutive normal addresses 2 – No consecutive fast addresses 9999 – No consecutive addresses in memory



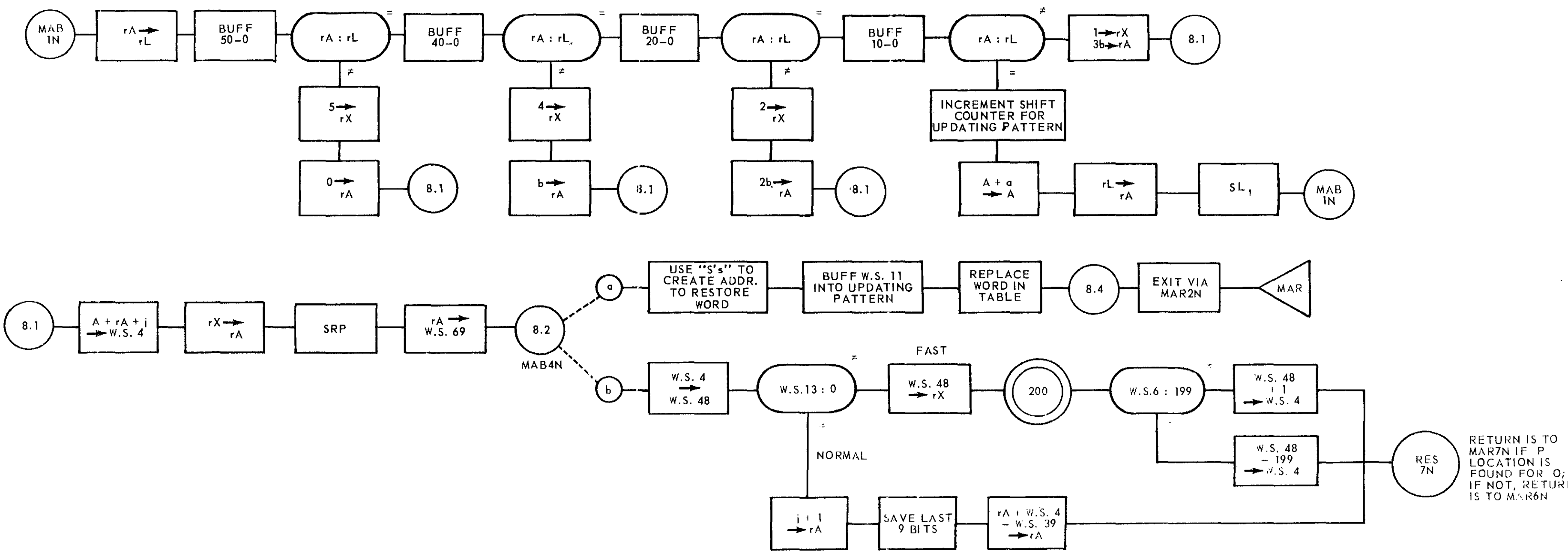
10	9	8	7	6	5	4	3	2	1
00	08	16	24	32	4000	4200	4400	4600	4800
02	10	18	26	34	4050	4250	4450	4650	4850
04	12	20	28	36	4100	4300	4500	4700	4900
06	14	22	30	38	4150	4350	4550	4750	4950

TABLE SO

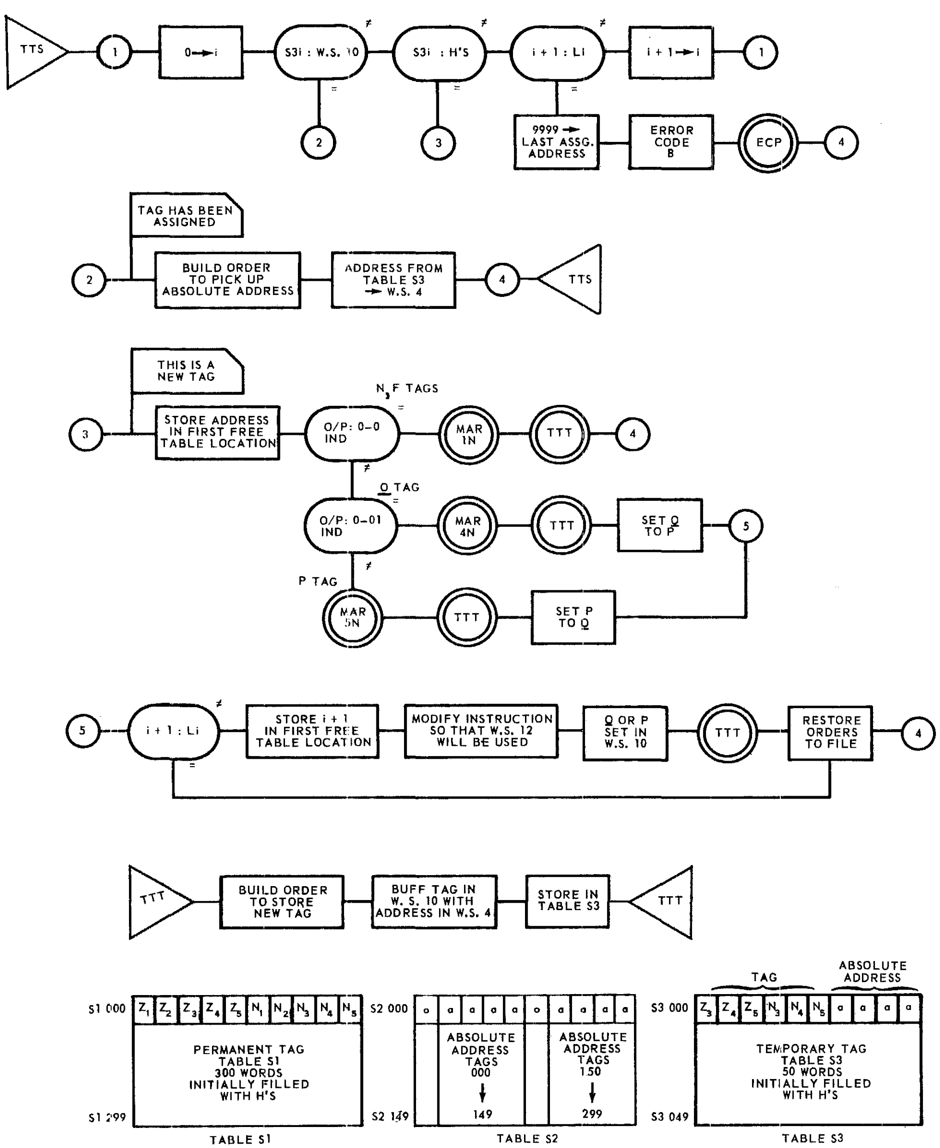
MEMORY AVAILABILITY ROUTINE–PART ONE (CONT'D)



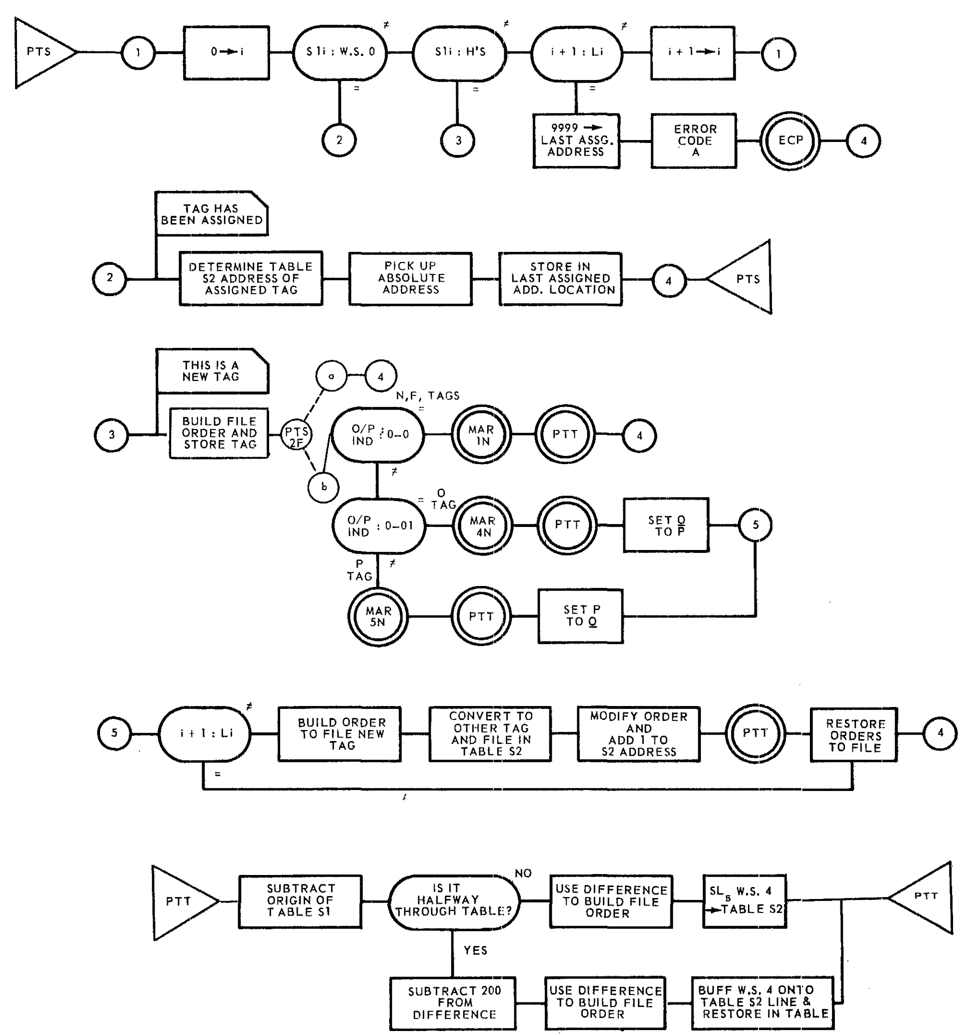
MEMORY AVAILABILITY ROUTINE–PART TWO



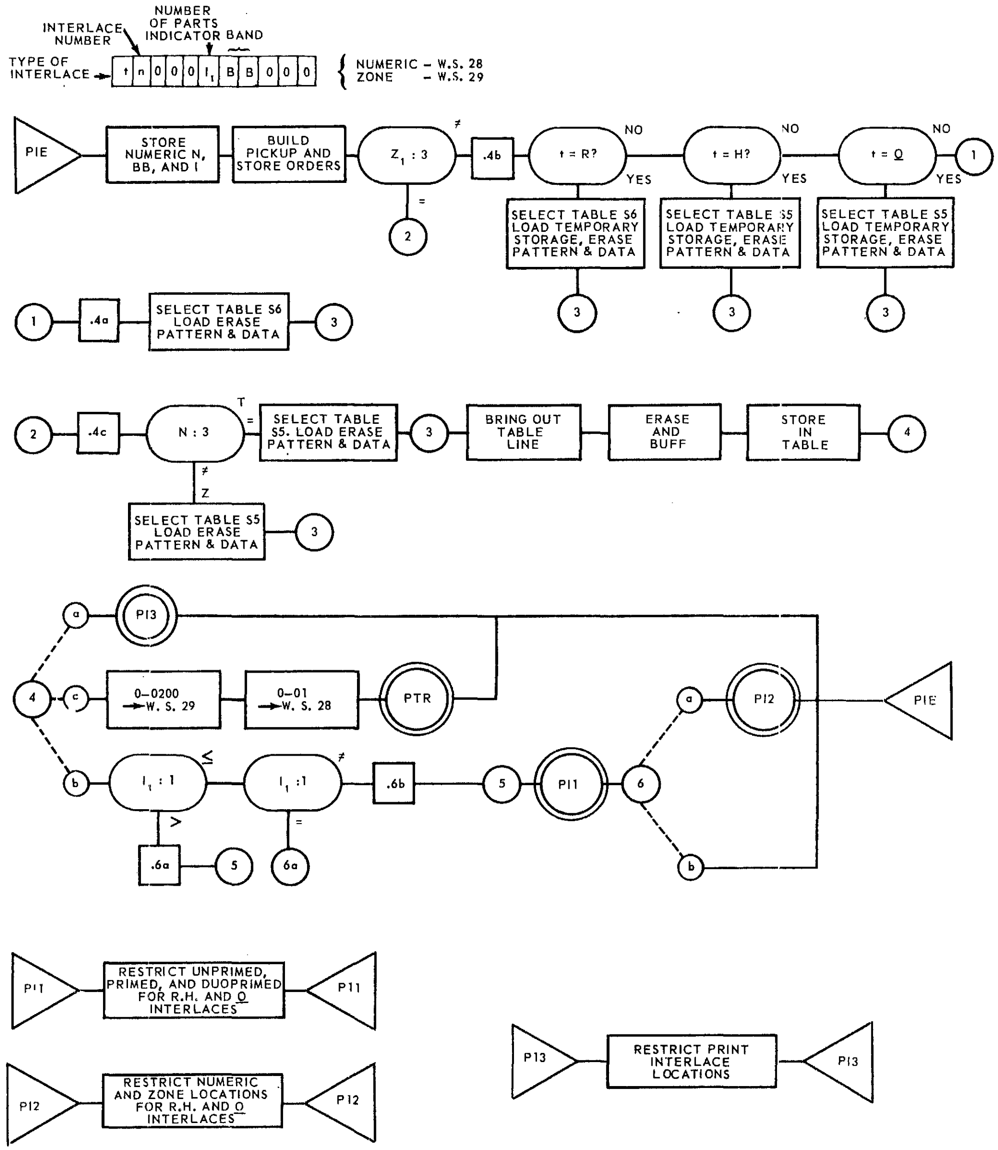
TEMPORARY TAG SEARCH



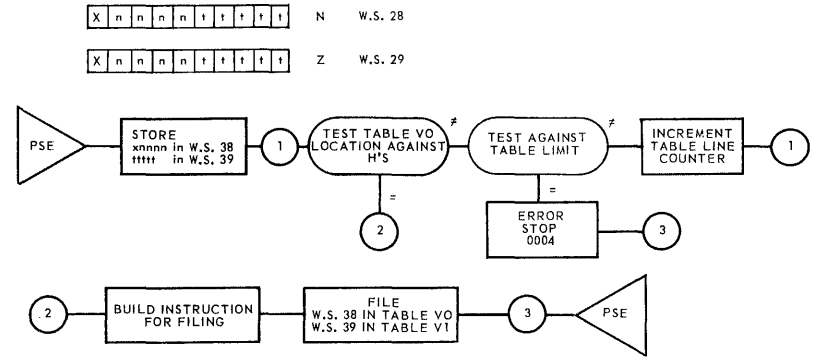
PERMANENT TAG SEARCH



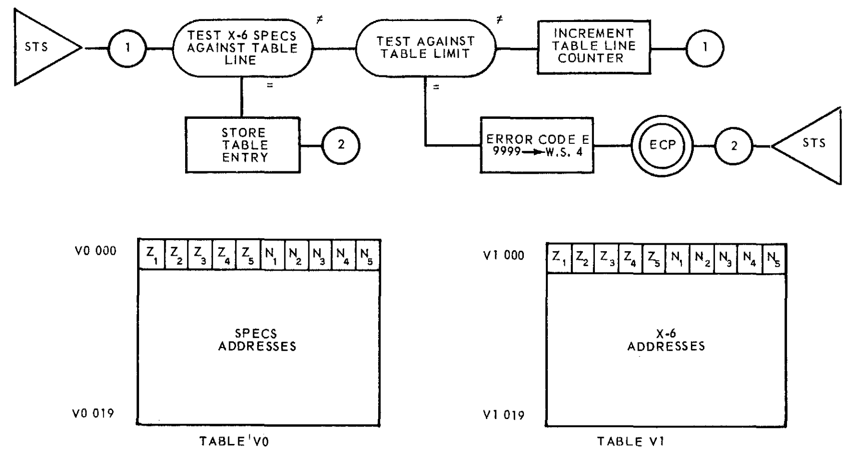
PROCESS INTERLACE ENTRY



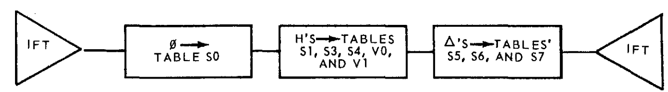
PROCESS SPECS ENTRY ROUTINE



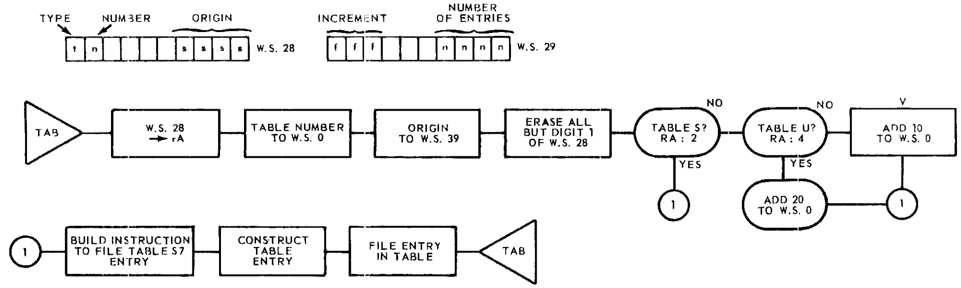
SPECS TABLE SEARCH ROUTINE



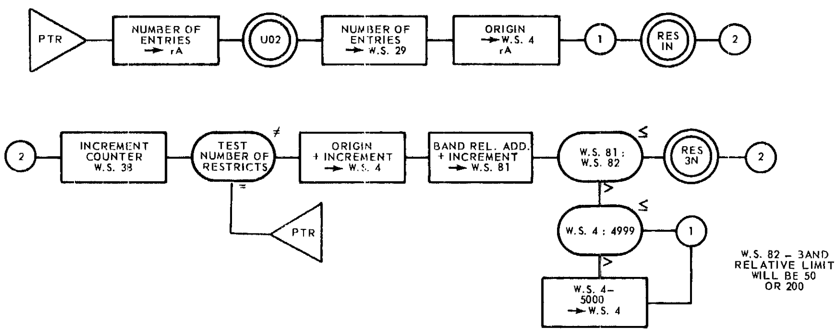
INITIAL FILL TABLES ROUTINE



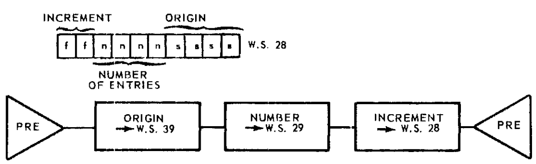
PREPARE TABLE ENTRY ROUTINE



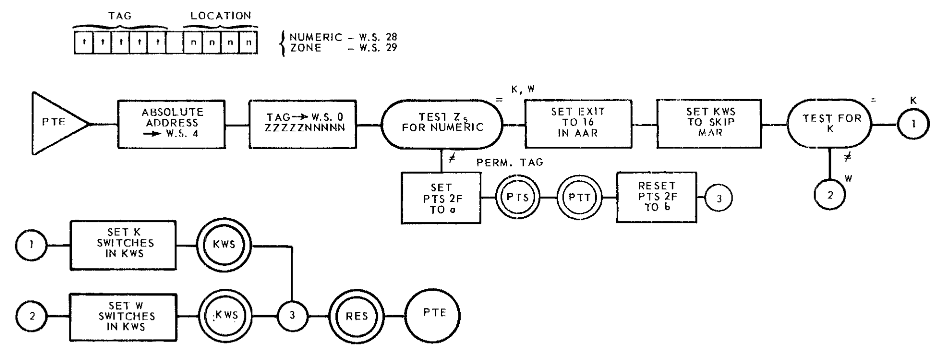
PROCESS TABLE RESTRICT ROUTINE



PREPARE RESTRICT ENTRY ROUTINE

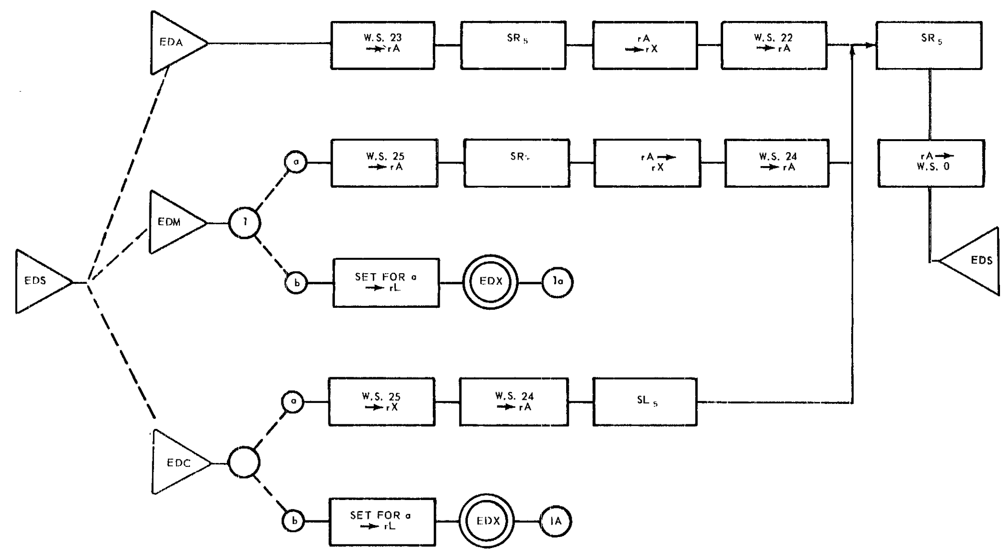


PROCESS TAG EQUALS ROUTINE

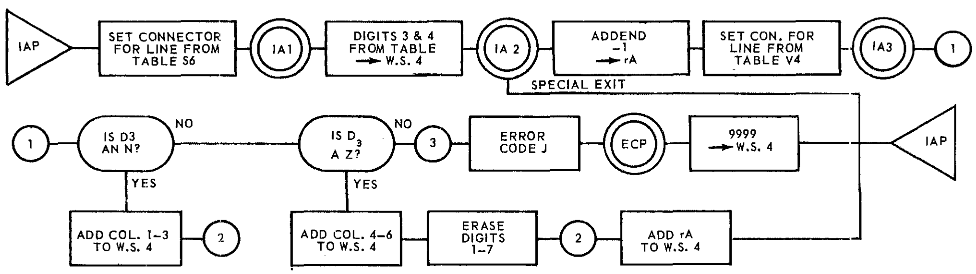


BAND RELATIVE ADDRESS ROUTINE

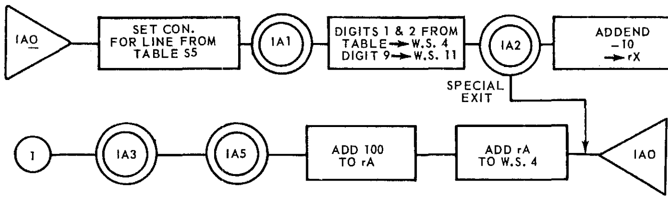
EDIT A, M OR C ROUTINE



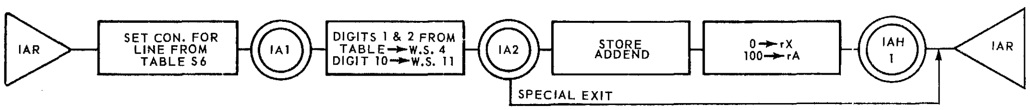
PRINTER INTERLACE ROUTINE



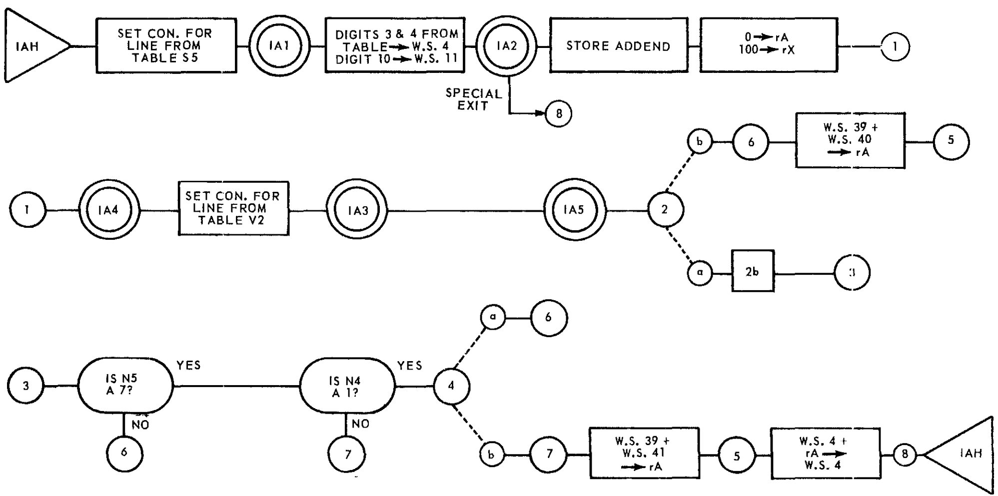
RPU OUTPUT INTERLACE ROUTINE



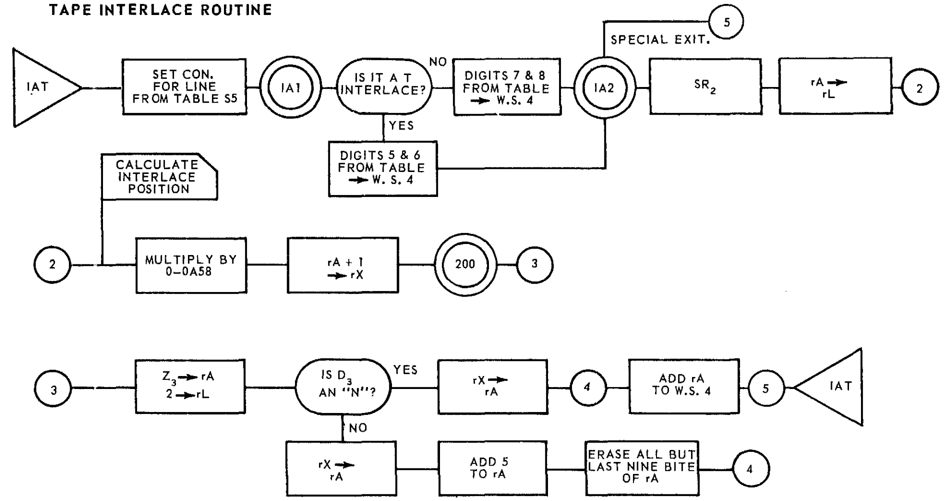
READER INTERLACE ROUTINE



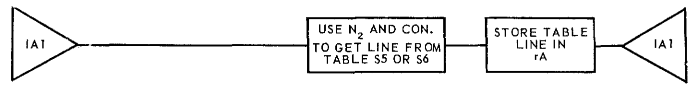
RPU INPUT INTERLACE ROUTINE



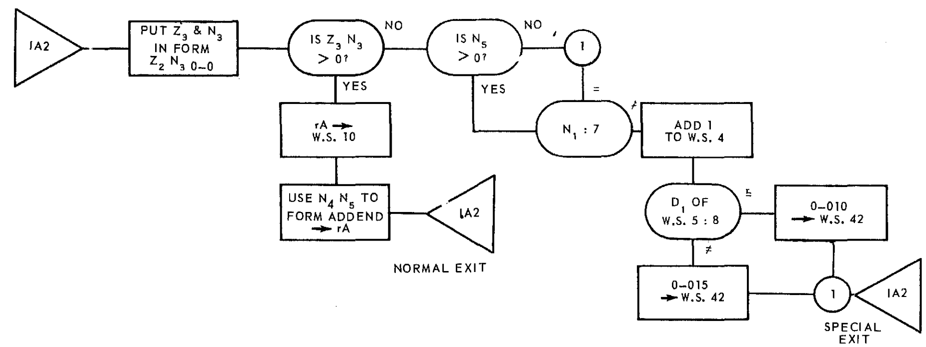
TAPE INTERLACE ROUTINE



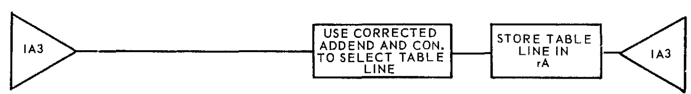
INTERLACE ROUTINE 1



INTERLACE ROUTINE 2

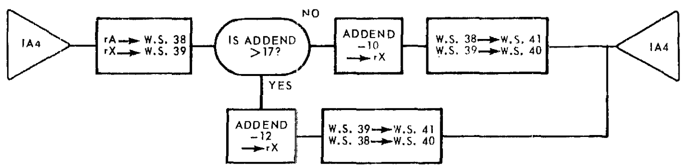


INTERLACE ROUTINE 3

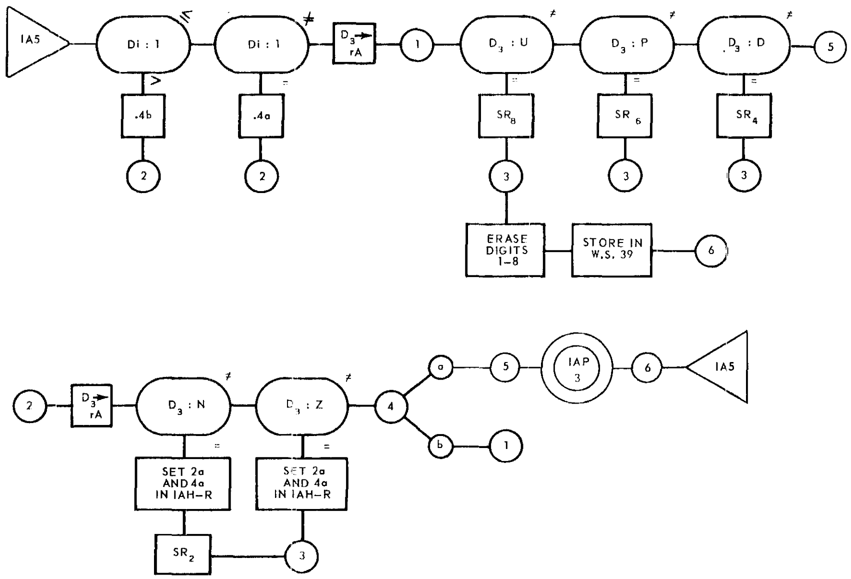


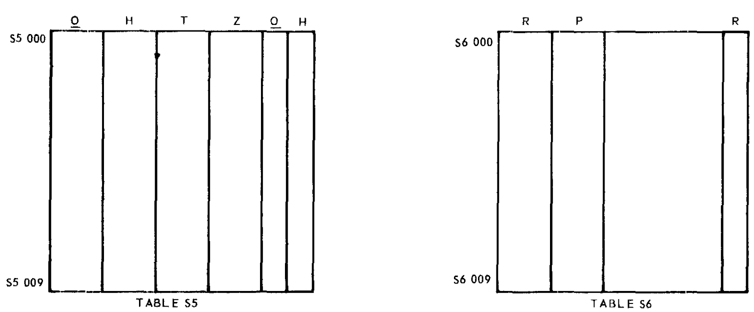
E 1774.1

INTERLACE ROUTINE 4

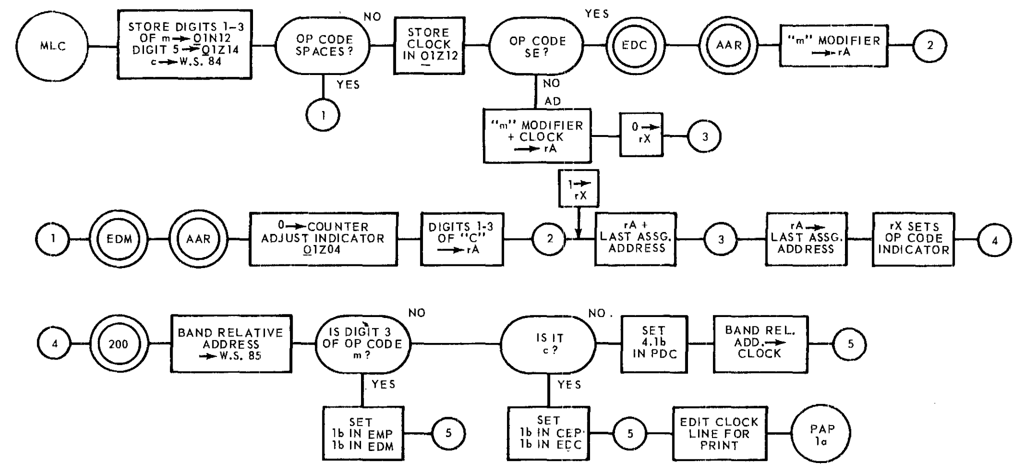


INTERLACE ROUTINE 5

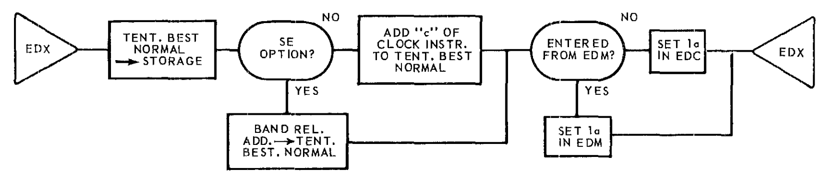




MODIFY LATENCY COUNTER ROUTINE



EDIT X ROUTINE



UPDATE CLOCK ROUTINE