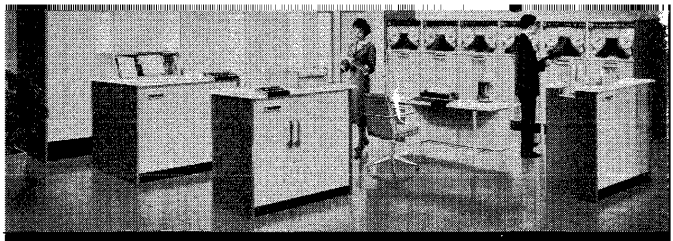


UNIVAC® III

TECHNICAL BULLETIN

SYSTEM CONVENTIONS

Programmers Reference

First Edition

October, 1961

TABLE OF CONTENTS

UNIVAC III CONVENTIONS

A set of Conventions and Documentation Standards has been established for the UNIVAC® III System. The specifications for these are presented in the following pages. They have been designed to provide maximum efficiency and utilization of the system. The assumption underlying these conventions is that CHIEF, the Executive routine for the UNIVAC III System, will be used by each installation.

UNISERVO* III DPL TAPE CONVENTIONS

In the UNIVAC I, II, and Solid-State computing systems, a maximum block size was adopted as part of the computer circuitry. With the greater flexibility of the UNIVAC III computing system, no such hardware restriction is necessary and block size is a function of the program.

Under the UNISERVO III conventions certain standards have been established:

The maximum data block size acceptable to all Input/Output routines is 4096 words of data.

DATA TAPE FORMAT CONVENTIONS

I. The first block on a tape reel and the first block of a file must be a label block. A label block is twelve words long. The label block for each reel of a multi-reel file must contain the number of the reel within the file.

II. The last data block of a file will be followed by two End-of-File sentinel blocks of one word each.

III. When a file is multi-reel, each reel before the last will have the last data block followed by two End-of-Reel sentinel blocks of one word each.

IV. When a file includes information that is not part of the data as such (for example, memory dumps required for re-run purposes), the information block or blocks must be preceded by two bypass sentinel blocks and followed by two bypass sentinel blocks. These are of one word each.

Bypass information with the appropriate Bypass sentinels may only appear:

A. After the label block of a file or tape

and

B. Before the End-of-File or End-of-Reel sentinel of a tape.

V. Thus, the convention blocks associated with a data tape file are:

A. Label block at the beginning of the tape (or each tape, if a multi-reel file).

*Trademark of the Sperry Rand Corporation.

B. Two End-of-Tape Sentinel blocks per tape in a multi-reel file.

C. Two End-of-File sentinel blocks.

D. Two Bypass Sentinel blocks preceding and Two Bypass Sentinel blocks following information to be bypassed in a data tape file.

VI. DATA TAPE BLOCK FORMATS

A. Label Block

WORD				
0	-	0	0	BLOCK IDENTIFIER
1	FILE IDENTIFICATION (4 ALPHABETIC CHARACTERS)			
2	DATE OF CYCLE (FORMAT ARBITRARY)			
3	REEL NUMBER (4 DECIMAL DIGITS)			
4	BLOCK SIZE (IN BINARY)			
5	ITEM SIZE (IN BINARY)			
6	FREE			
7	FREE			
8	FREE			
9	FREE			
10	FREE			
11	-	0	0	BLOCK IDENTIFIER

The Label Block is of twelve word length and must appear at the beginning of each tape reel and of each tape file.

Word 0 must contain the identifier "-00" in the three high order bit positions. The minus identifies this as a non-data block. The 00 specifies a Label Block.

Word 1 must be the file identifier expressed as four (4) alphabetic characters.

Word 2 Date of Cycle. Each reel of a multi-reel file must contain the same date in the label blocks.

Word 3 Reel Number expressed in four (4) decimal digits. Each reel of a multi-reel file must contain the number of the reel within the file.

Word 4 Size of data blocks expressed in the 12 low order binary bits.

Word 5 Item size of data expressed in the 12 low order binary bits.

Note: If block size and item size within a file are variable, words 4 and 5 will contain binary zeros.

Word 6
Word 7
Word 8
Word 9
Word 10
} Unused. Available for installation use as desired.

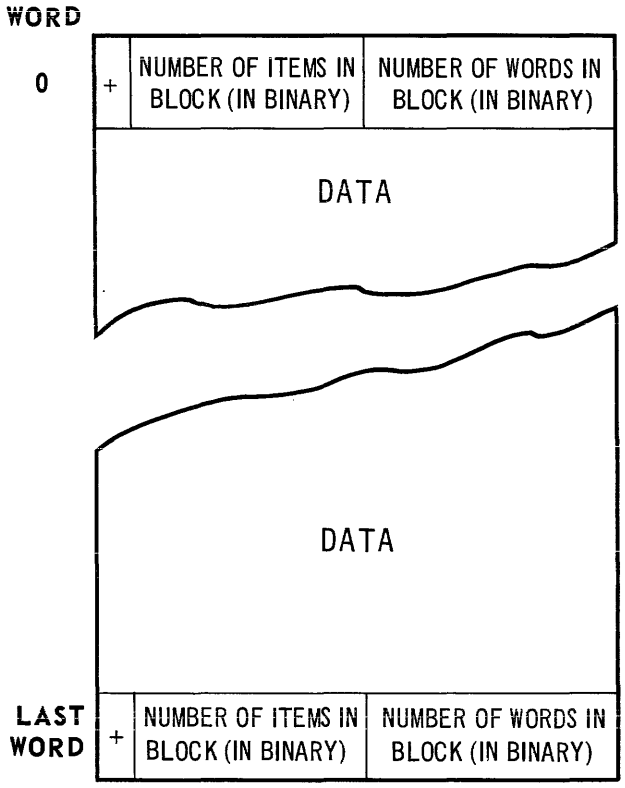
Word 11 same as Word 0.

Any Input/Output close-out subroutine will type out the following label information:

File Identification Number
Data of Cycle
Reel Number
} (Converted to alphanumeric for type-out purposes.)
Number of blocks per reel (including bypass and sentinel blocks)

This type-out will occur for each reel of a multi-reel file.

B. Data Block

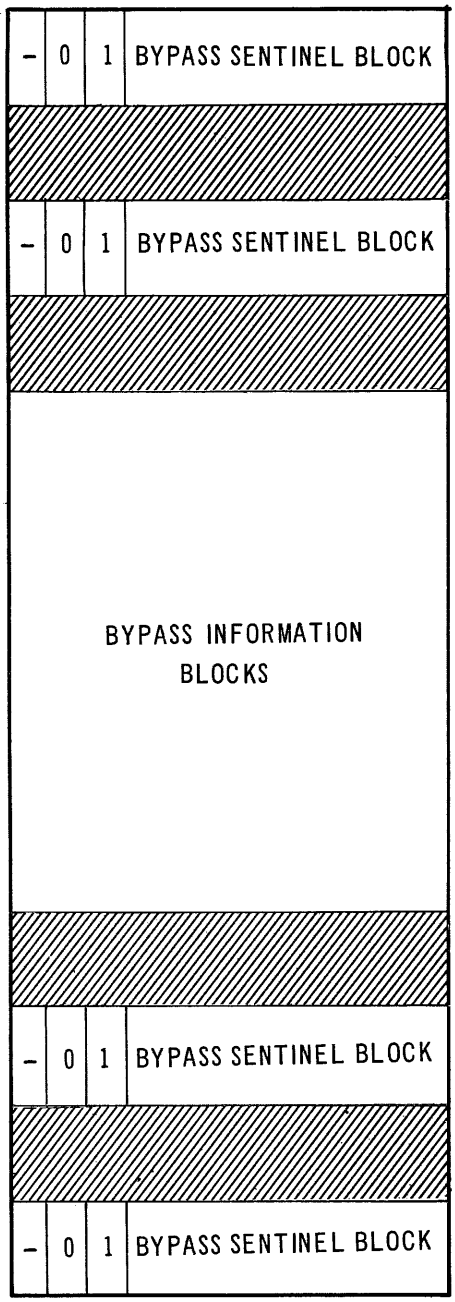


Each data block of a file must contain in the first and in the last word:

1. + in the sign position to indicate a data block.
2. The binary expression of the number of data items in the block in positions 13 through 24.
3. The binary expression of the number of words in the block in positions 1 through 12.

The numbers in the first and last words of the block includes the first and last word.

C. Bypass Sentinel Block

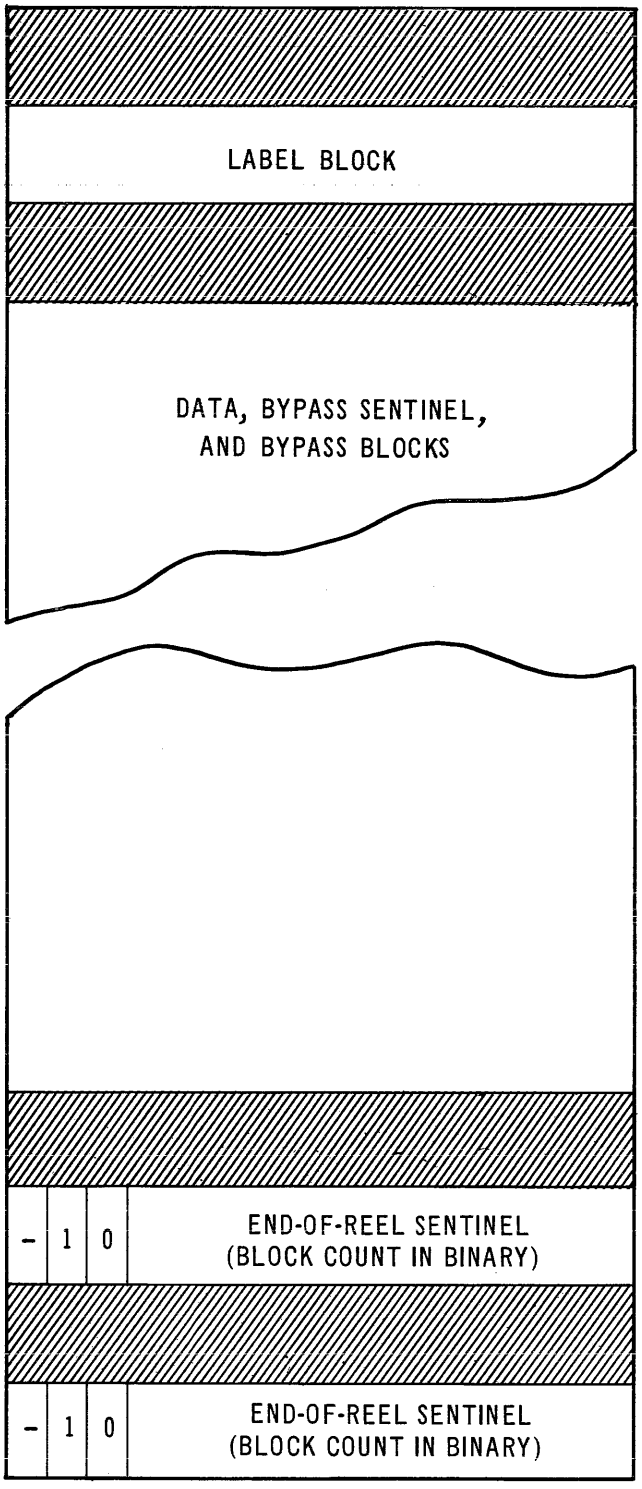


Two blocks of one word each must precede and succeed any information to be bypassed on a data tape. Each of these blocks must contain "-01" in the three high order bit positions of the word. The minus identifies a non-data block. The 01 specifies a Bypass Sentinel.

The information contained in the Bypass blocks, when a memory dump of a program, is in the basic Master Instruction Tape format:

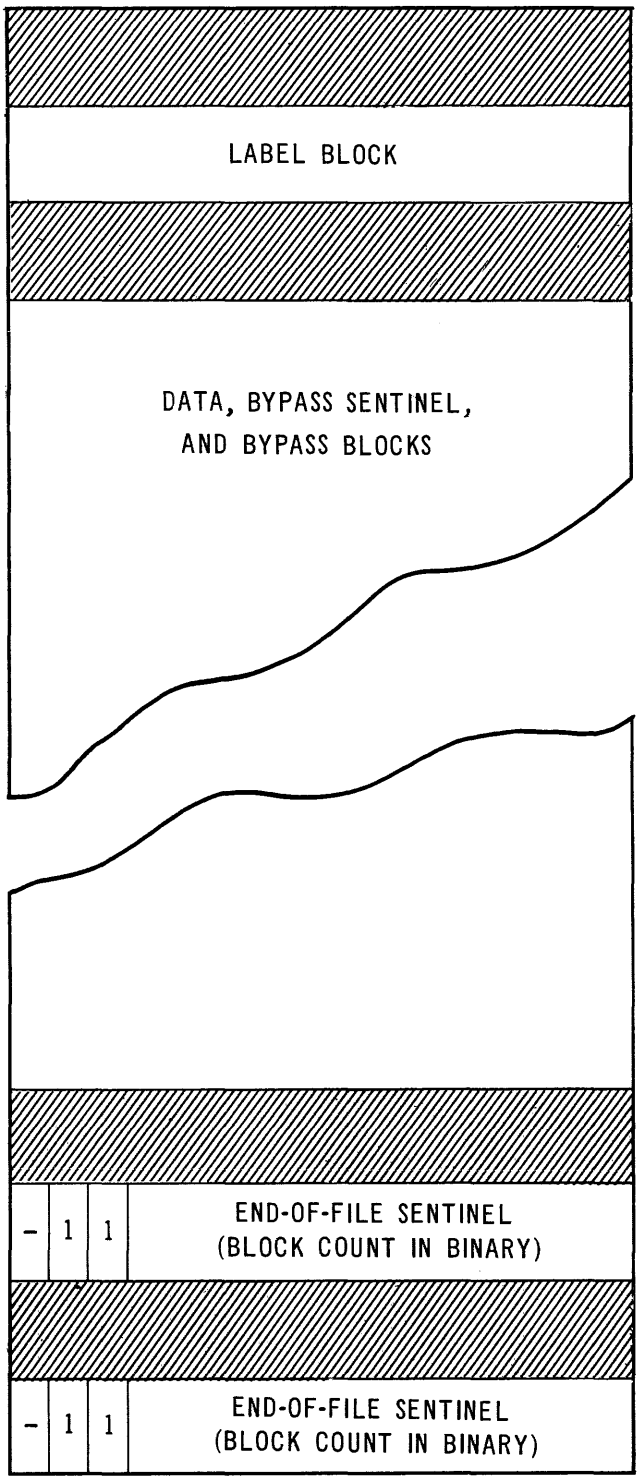
1. The first block or blocks will be a Program Identification block followed by Program Identification Trailer blocks as nes-necessary.
2. The program instructions in blocks of 400 words each followed by dummy Load Modification blocks.

D. End-of-Reel Sentinel Block



Two End-of-Reel Sentinel blocks of one word each must follow the last data block on intermediate tapes of a multi-reel file. Each of these End-of-Reel Sentinel blocks must contain "-10" in the three high order bit positions. The minus identifies a non-data block. The 10 specifies an End-of-Reel sentinel block. The remaining bit positions of the word must contain the number of blocks on the tape (including label, bypass sentinel, bypass blocks, and sentinel block.) Thus, if 62 blocks preceded the first End-of-Reel Sentinel, this sentinel would contain the number 63, and the second End-of-Reel Sentinel block would contain the number 64. This number is expressed in binary. The least significant bit of the binary number appears in bit position one of the sentinel block word.

E. End-of-File Sentinel Block



Two End-of-File Sentinel blocks of one word each must follow the last data block of a file. Each of these End-of-File Sentinel blocks must contain "-11" in the three high order bit positions. The minus identifies a non-data block. The 11 specifies an End-of-File block. The remaining bit positions of the word must contain the number of blocks in the file (Label, Data, Bypass Sentinels, Bypass Blocks, and End-of-Reel Sentinels if any) including the End-of-File sentinels. Thus, if a file contained 500 blocks before the End-of-File Sentinels, the first sentinel would contain the number 501; the second, 502. These numbers are expressed in binary. The least significant bit of the binary number appears in bit position one of the sentinel block word.

MASTER INSTRUCTION TAPE CONVENTIONS

The programs to be scheduled for a computer shift are extracted from a Master Reference File (MRF) and placed on a Master Instruction Tape (MIT) by an Object Code Service Run. Each MIT will begin with a standard label block immediately followed by a preload block for the Executive routine, CHIEF, which is always the first program on the MIT.

After the Executive routine is loaded, control is transferred to it. The remainder of the programs on the MIT become input to the Executive routine which will allocate facilities and areas of computer memory for the programs in turn. To enable CHIEF to perform its functions, each program must be preceded by a Program Identification block containing information concerning memory area necessary, any other loads that should be in memory with the program, and a listing of the facilities required. Should a program consist of more than one load, Load Identification blocks precede each load.

The necessary identification block is followed by the instruction blocks of the load. These are followed by Load Modification blocks which contain the address of each instruction that will need modification for correct functioning and a key indicating what modification must be made.

The information appearing in the identification and modification blocks is a result of the SALT assembler processing and exists in the MRF at the time the MIT is created. The format of these various blocks follows.

A. Program Identification Block

Each Program Identification block is of 100 words and contains the following information:

Word	
0	Block sentinel.
1	Same as Word 0.

Word

2, 3, 4 — Program Identification in twelve alpha-numeric characters:

Word 2 = aaaa, the routine name expressed in four alphanumeric characters. The first character must be alpha.

Word 3 = iicc, with ii being the installation code or the programmer's initials and cc being a control number, used to record the copy number of routine.

Word 4 = x$\Delta\Delta$z with x being a rerun number in binary and $\bar{z}$ being an alphanumeric code inserted by the service run that created the MIT. $\Delta\Delta$ are unused positions.

5, 6 — Load Identifier eight alphanumeric characters.

7 — Bit positions 1 through 17 contain the binary representation of the number of memory locations that are needed for this load.

Bit position 17 contains a one or a zero. If 0, there is only one load; if 1, there is at least one more load to be read into memory whenever the first load is read in.

8, 9 — Next Load Identifier. This is the 8 alpha-numeric character identification of the next instruction load of this program that will occupy memory with the first load. Binary zeros if there is only one load.

10 — Bit positions 1 through 15 contain the binary representation of the starting location of this load. If there is no next load, this area is blank.

11 — Bit positions 1 through 4 contain binary zeros or, in the case of rerun, an internal routine designator. This designator is created by the Locator Section of the Executive Routine. If a rerun, the designator assigned during original execution of the program is inserted.

Word

Bit positions 10 through 24 will contain either binary zeros or, in the case of a rerun, the absolute address assigned to relative line zero of the program. If binary zeros are present, the Allocator Section of the Executive Routine will assign this address before further processing is begun.

12 . . . 97 — List of facilities required by the program. Word 12 must be the binary representation of the greatest number of memory locations that the program must have at any one time for its execution. Word 13 may contain, in binary, the maximum number of additional memory locations desirable for quicker, or more efficient, program execution (as in sorting); otherwise, word 13 will be binary zeros.

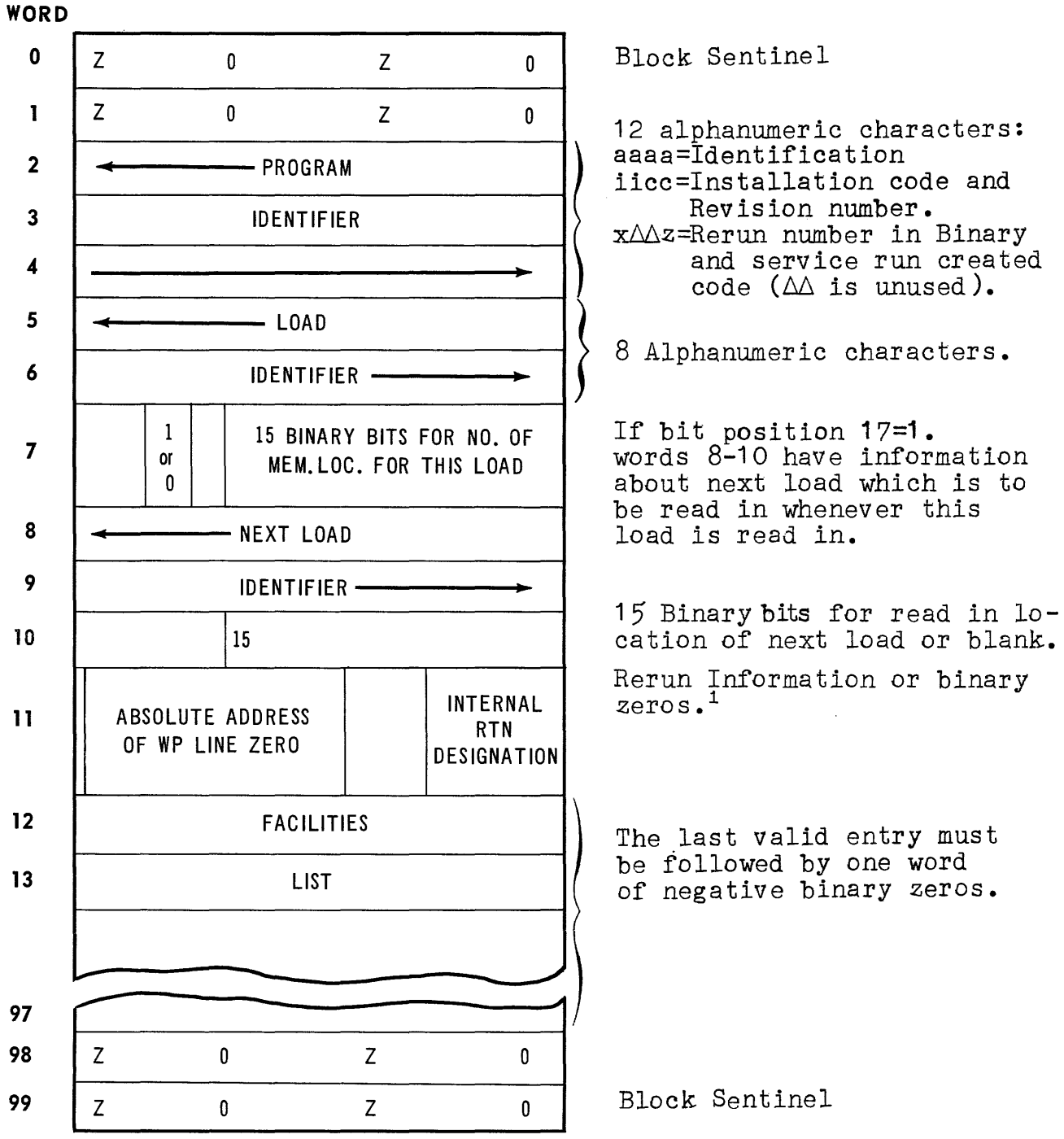
The last valid item of the facilities list must be followed by a word of negative binary zeros.

98 — Same as Word 0.

99 — Same as Word 1.

Note that the information contained in the Program Identification block (as well as the information contained in the blocks that follow on the MIT) is assumed to be the result of a SALT assembly as stored in the MRF. If the program is a rerun, word 11, as assigned by the Executive routine during the previous processing, will exist).

Program Identification Block - 100 Words



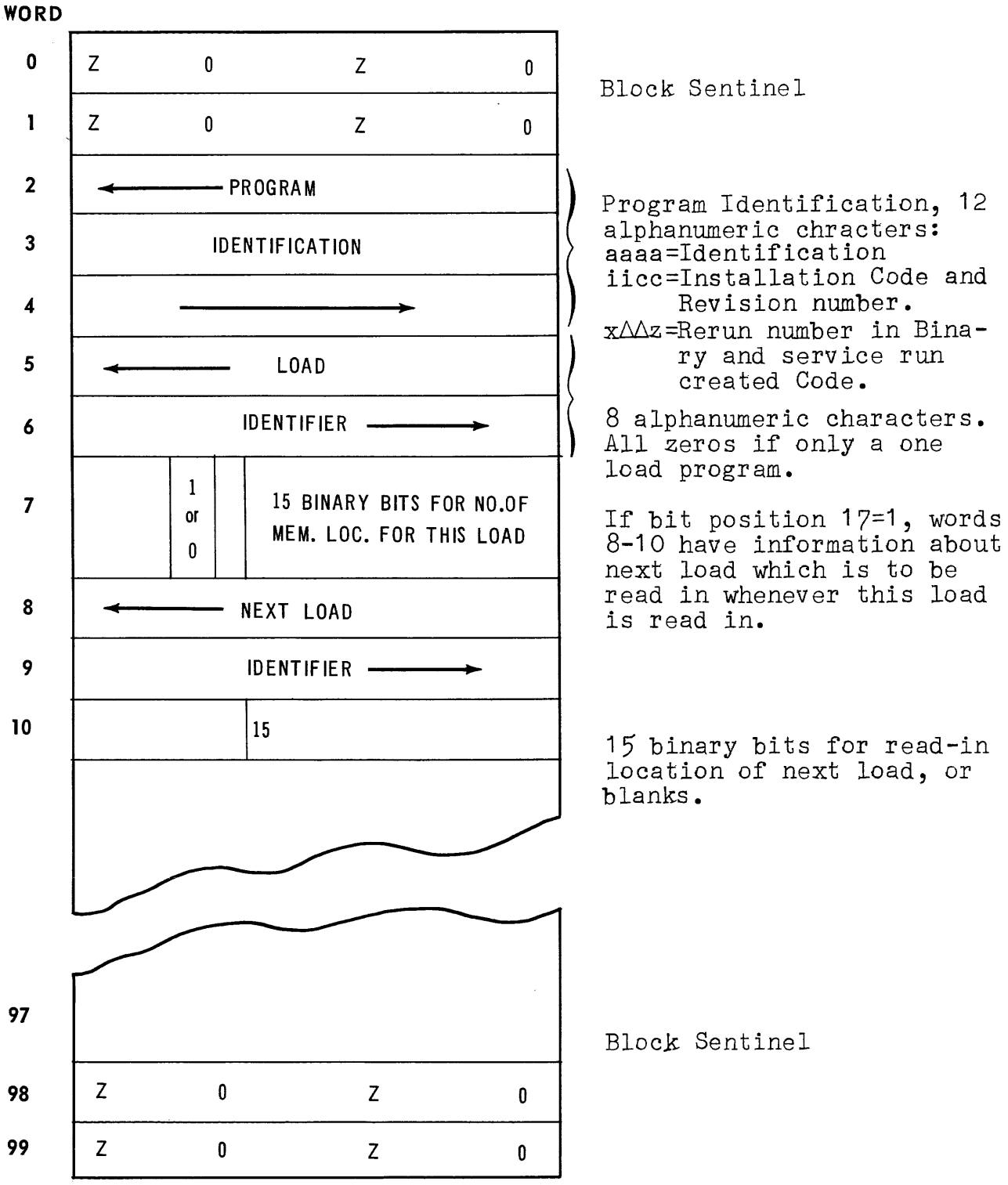
[1]Word 11 is assigned by Executive Routine in memory unless this program is a rerun. If a rerun, the previously assigned Word 11 is used.

B. Load Identification Block

A Load Identification block is used before the second load, and before each of any succeeding loads, of a program when these loads must occupy memory at the same time. Basically, it is of the same format as the Program Identification block with the following exceptions:

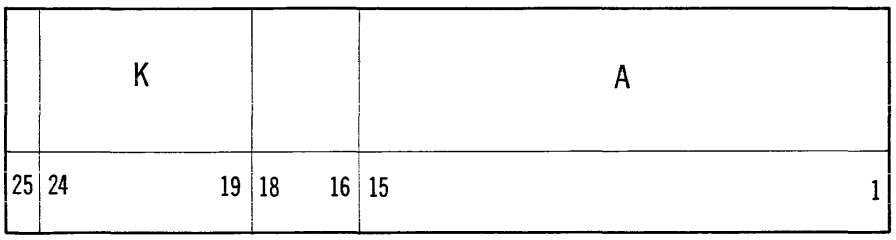
Words 11 through 97 are not used (no facilities can be specified).

C. Load Identification Block - 100 Words



C. Load Modification Blocks

Load Modification blocks follow the instruction blocks of a load and are of 100 words each. Each word is in the following format:



A is the binary address (zero relative to the starting address of the program) of the instruction to be modified.

K is a binary key specifying the modification to be made to the instruction designated by A.

The possible load keys, and their interpretations are:

KEY	MEANING
1	The absolute starting address (or base address) of the program loaded, as assigned by the Executive routine, will be added to the zero relative address contained in bit positions 1 through 15 of the instruction specified by A.
2	The base address of the program loaded will be added to bit positions 10 through 24 of the instruction specified by A.
4	The internal routine designator (assigned by the Executive routine) will be inserted in bit positions 22 through 25 and the base address will be added to bit positions 1 through 15 of the instruction specified by A.
5	The internal file designator (assigned by the Executive routine) will be placed in bit positions 16 through 21 of the instruction specified by A. This is a replacement of the external file designator assigned during the SALT assembler processing.
6	The internal file designator (assigned by the Executive routine) will replace the external file designator in bit positions 16 through 24 of the instruction specified by A.

KEY	MEANING
7	The external file designator (bit positions 11 through 16) will be replaced with an associated channel number (assigned by the Executive routine) by the insertion of this channel number in bit positions 11 through 14 of the instruction specified by A. The correct operation code will be inserted in bit positions 15 through 20 and is determined by an examination of bit positions 17 through 20 of the instruction before insertion is made. If bit positions 17 through 20 contain a binary number: 1 an operation code of 04 is inserted. 2 an operation code of 07 is inserted. 3 an operation code of 64 is inserted. 4 an operation code of 65 is inserted. 5 an operation code of 70 is inserted.
8	In the instruction specified by A; the internal file designator will replace the external file designator in bit positions 16 through 21; the internal routine designator will be inserted in bit positions 22 through 25; and the base address will be added to bit positions 1 through 15.
9	In the instruction specified by A; the external file designator (bit positions 19 through 24) will be replaced by the assigned channel number, which will be inserted in bit positions 21 through 24; the base address will be added to bit positions 1 through 15; and the content of bit positions 18, 17, 16 will be moved to bit positions 19, 18, 17 respectively.

The last valid entry of a load modification block will be followed by a word of binary zeros.

TYPEWRITER PAPER FORMAT CONVENTIONS

There are four categories of information (type-outs, type-ins) that will be printed by the console typewriter.

First position - Actual clock time

Second position - Executive routine

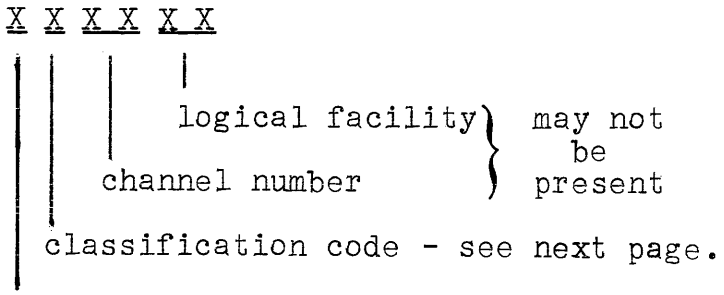
Third position - Input/output subroutines

Fourth position - Worker programs

Appropriate tabulator settings will be made for the starting position of each category. These tabulator settings will be supplied by the requesting program in TPAK. Typewriter messages may extend to the end of the line.

I. TYPE-OUT MESSAGE FORMAT

A. Each type-out message must be preceded by the following alphanumeric encoded message:



1. Flag for operator action or space.

   One alphanumeric character ">", denotes operator action, supplied by the requesting program.

2. Classification Code

   One alphanumeric character
   Supplied by the requesting program.

3. Channel number

   Two alphanumeric characters
   Present for classification codes, A,C,H, if applicable
   Supplied by the requesting program.

4. Logical Facility

   Two alphanumeric characters
   Present for classification codes A,C,H, if ap-

plicable. Supplied by the requesting program.

If channel number and facility code do not apply to a particular type-out message, they need not be present.

An encoded message can be followed by: "Δ English message" or "Δ serial nr", whichever the requesting program prefers.

The Executive Routine will automatically supply an asterisk "*" to indicate completion of ENTIRE type-out message:

encoded message Δ english message*
or
encoded message Δ serial number*

B. Classification Codes

A = Allocation information

C = Computer errors - including read/write errors, etc.

D = Data errors - encountered during validity check

E = End of program

H = Historical data file processing information

J = Jettison of run

O = Option - must be selected

P = Program control errors

S = Start of program

T = Type-in data request (small volume)

II. TYPE-IN MESSAGE FORMAT

Each type-in must be preceded by an alphanumeric encoded message.

There are two kinds of type-in messages:

Solicited

An answer to a program request for type-in.

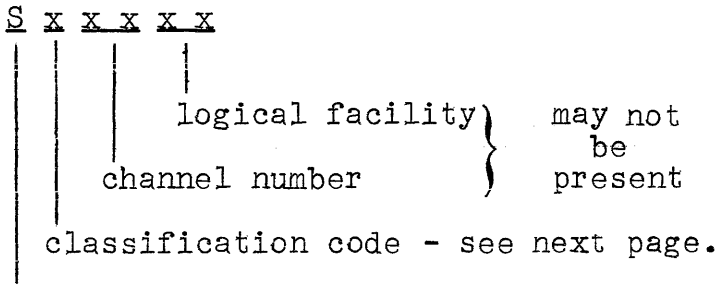
or

Unsolicited

This is an external request initiated by the operator.

A. Solicited Messages

1. Immediate answer to a type-out



If channel number and facility code do not apply to a particular type-in message, they need not be present.

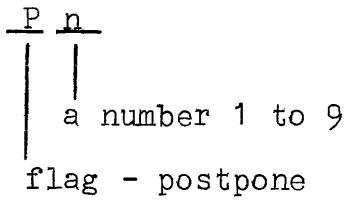
The encoded message can be followed by "Δ English message" or "Δ serial number", whichever is preferred.

The Executive Routine will automatically supply a plus sign "+" to indicate completion of ENTIRE type-in message.

2. Postpone the answer to a type-out

In order to postpone the answer to a type-out, the operator need only use the typewriter release key. The Executive Routine will then automatically assign a postpone number to the unanswered request, and type this number out.

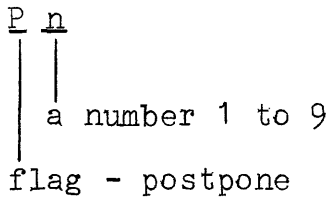
The postpone number will be in the following format:



B. Unsolicited Messages

1. Postponed answer (to a type-out)

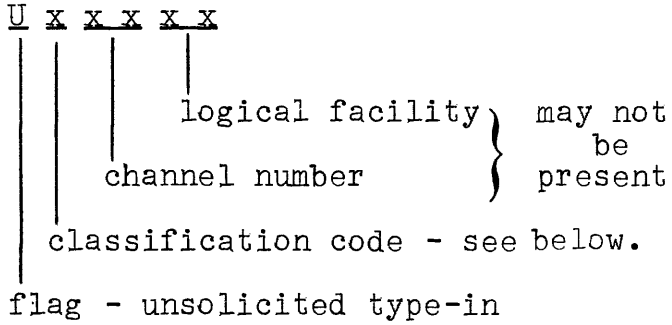
In order to relate the reply to the original type-out request, the operator, when ready, must type in:



The Executive Routine will again type-out the original request, which the operator is now prepared to answer.

2. External request message

An external request (type-in) can be made by the operator whenever he wishes to interrupt the original MIT schedule or whenever he wishes to enter additional logging information.



If channel number and facility code do not apply to a particular type-in message, they need not be present.

The encoded message can be followed by: "Δ English message" or "Δ serial number", whichever is preferred.

The Executive Routine will automatically supply a plus sign "+" to indicate completion of ENTIRE type-in message.

C. Classification Codes

A = Allocation information

C = Computer error corrected

D = Data error corrected

H = Historical data file processing information

I = Ignore condition

J = Jettison run

L = log information (i.e. idle time, down time, lost time, test time, scheduled or unscheduled maintenance, training, etc.).

M = Memory dump for printing

O = Option selected

P = Program error corrected

R = Repeat condition

S = Start program

T = Type-in of data

PROGRAM DOCUMENTATION REQUIREMENTS

The following conventions for program documentation can be used. Their implementation at installations will, of course, depend on the requirements and needs of the user. It is recommended that they be followed, so far as possible, to allow interchange of programming information and maintain program standards.

For each program, the documents listed below should be prepared, using the established conventions.

Contents Check List
Program Design Specifications
Environmental Flow Chart/Systems Flow Chart
Block Diagram
Detailed Flow Chart
Memory Allocation Chart
Listing (source-object code)
Data Designs
Sample type-outs
Sample print-outs
Operator's Instructions
Listing of library routines used

No program is complete until all twelve of the component parts of programming documentation are finished, and the set assembled in a standard binder.

I. Contents Check List

The suggested form to be used as the "CONTENTS CHECK LIST" is shown in Figure I, Contents Check List. There are six columns on the form as follows:

Column 1 - DOCUMENTS - This is a preprinted list of the documents that the manual, for each program should include.

Column 2 - PREPARED BY - This column is reserved for the name of the Programmer originally preparing the document.

Column 3 - COMPLETION DATE - The month, day and year that the document being recorded was fully completed.

Column 4 - REPRINT NUMBER - Should it become necessary to revise any document in the manual, the revision number should be entered in this column. For example, the first time it is necessary to revise the specification a "1" would be entered in column four.

Column 5 - DATE REVISED - The month, day, and year that a document is revised.

Column 6 - REVISED BY - The name of the programmer making the revision.

CONTENTS CHECK LIST

TITLE	IDENTIFICATION NO.					
DOCUMENTS	PREPARED BY	COMPLETION	DATE	REPRINT NO.	REVISION DATE	REVISED BY
Contents Check List						
Operator's Instructions						
Environmental Flow-Chart						
Final Design Specifications						
Listing (Source-Object Code)						
Listing of Library Routines Used						
Block Diagram						
Data Designs						
Detailed Flow-Chart						
Memory Layout						
Sample Typeouts						
Sample Printouts						

FIGURE 1

II. Program Design Specifications

Basic items of information which are a component part of all Program Design Specifications include:

Computer Configuration
Purpose of the Program
General Information
Mathematical Formulae
Procedures
Input (general description).
Output (general description)
Phased Run
Codes

To facilitate the locating of information in the specifications without reading them in their entirety, the order shown above should be the order of presentation.

1. Computer Configuration - List the computer configuration on which this program is designed to run.

2. Purpose - This is to describe in general terms what the program will accomplish in terms of inputs created for related machine runs and reports produced.

3. General Information - Since each program is a unique problem, all situations and pertinent information not covered elsewhere will be stated in this section.

4. Mathematical Formulae - All formulae used will be set forth. If the mathematical technique is a programming device only, it will be so noted.

5. Procedure - Give a general narrative description of the program and point out the basic approaches used (Matrices, Summarization, etc.).

6. Input - Describe input to the run. If card input, give the card identification numbers and the source of the cards. If tape input, give the tape identification number and the program number which created the tape. If the tape input is a result of a card-to-tape operation, give the card identification number and the source of the cards. For all inputs, give the maximum volume of cards and/or tape records to be used, if possible.

7. Output - Describe output. If card output, give the identification number, the use to which the card will be put. If tape output, give the tape identification number. If the output tape is edited for printing, give the form number of the report to be printed. Give the maximum record volume when known.

8. Phases - If the program is accomplished in one pass of data, state that the run is not phased. If the machine run is phased, state which of the inputs are introduced in each phase and which of the outputs are produced in each phase. Further, give a brief description of intermediate tapes, which are in effect, scratch tapes merely used to facilitate computer operation.

9. Codes - In some instances, it may be advantageous to assign codes internally in the memory to facilitate the programming of the machine run. If this is necessary, give a detailed explanation of the codes and keys used. If, in a machine run, a coded output is developed which has not previously been documented, a set of the newly developed codes must be documented and distributed to appropriate areas.

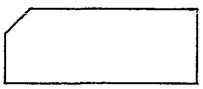
III. Environmental Flow Chart/Systems Flow Chart

An Environmental Flow Chart graphically describes the facilities employed by a particular program. It shows simply the type and flow of data between specific units of the machine. It emphasizes the types of input and output facilities. An Environmental Flow Chart is used to depict input and output facilities for one run (see example A). A Systems Flow Chart depicts the flow of information involving an entire system of runs (see example B). Special symbols have been designed to visually represent the various types of units.

The flow of data through the computer is shown by drawing arrow heads on the connecting lines of the peripheral equipment to show direction to and from the frame.

The file name and identification number should be entered in the appropriate symbols on the flow chart.

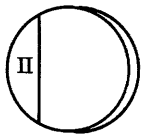
A. Standard Symbols used in Environmental Flow Charts/ Systems Flow-Charts.

Card Reader 

Card Punch 

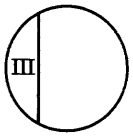
Tapes

UNIVAC II Tape

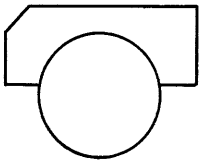


Second circle would indicate a multi-reel file.

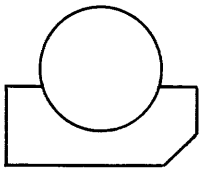
UNIVAC III Tape



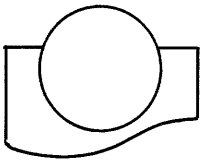
Tape Prepared from Cards



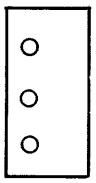
Cards Prepared from Tape



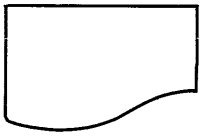
Tape edited for printing



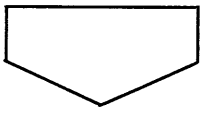
Paper Tape



Printer



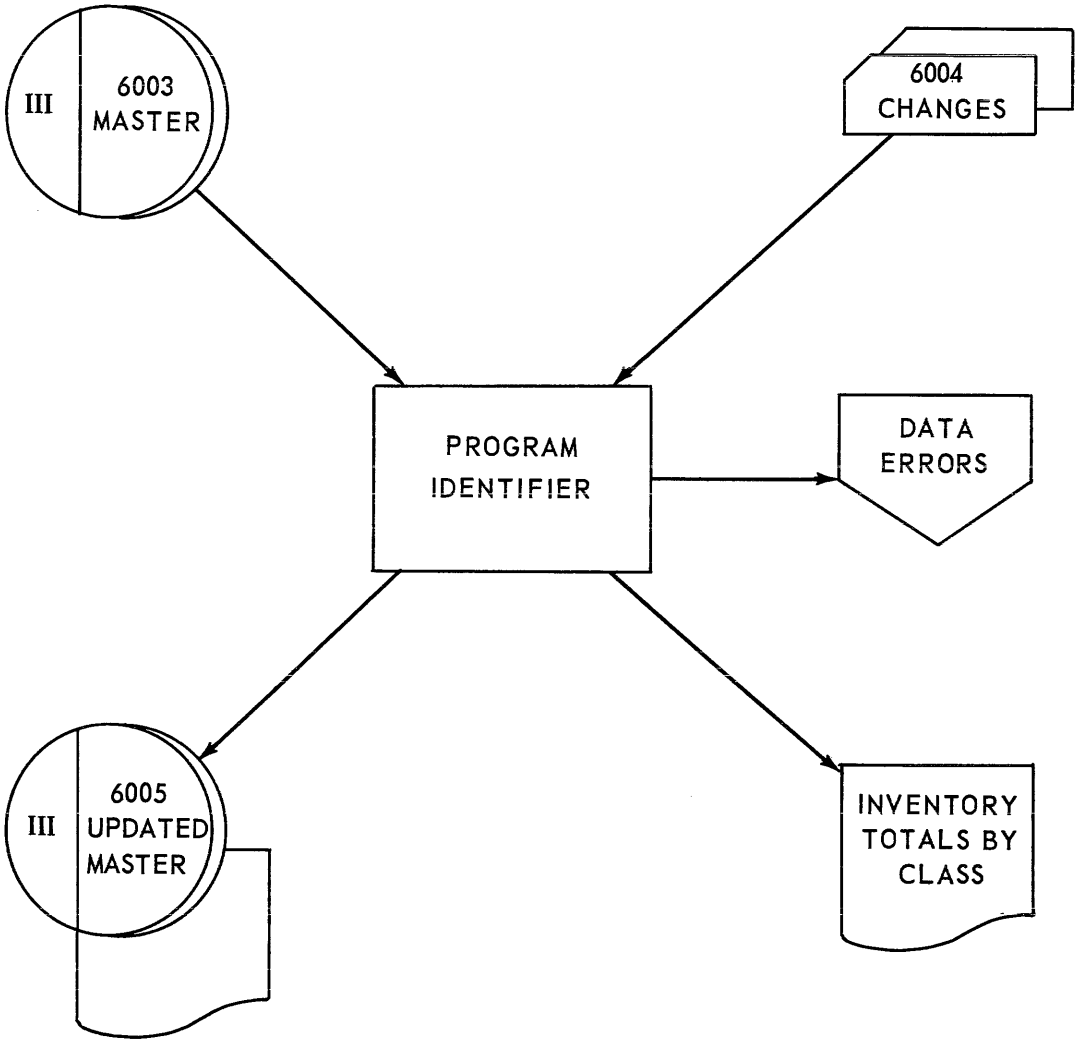
Typewriter



Computer

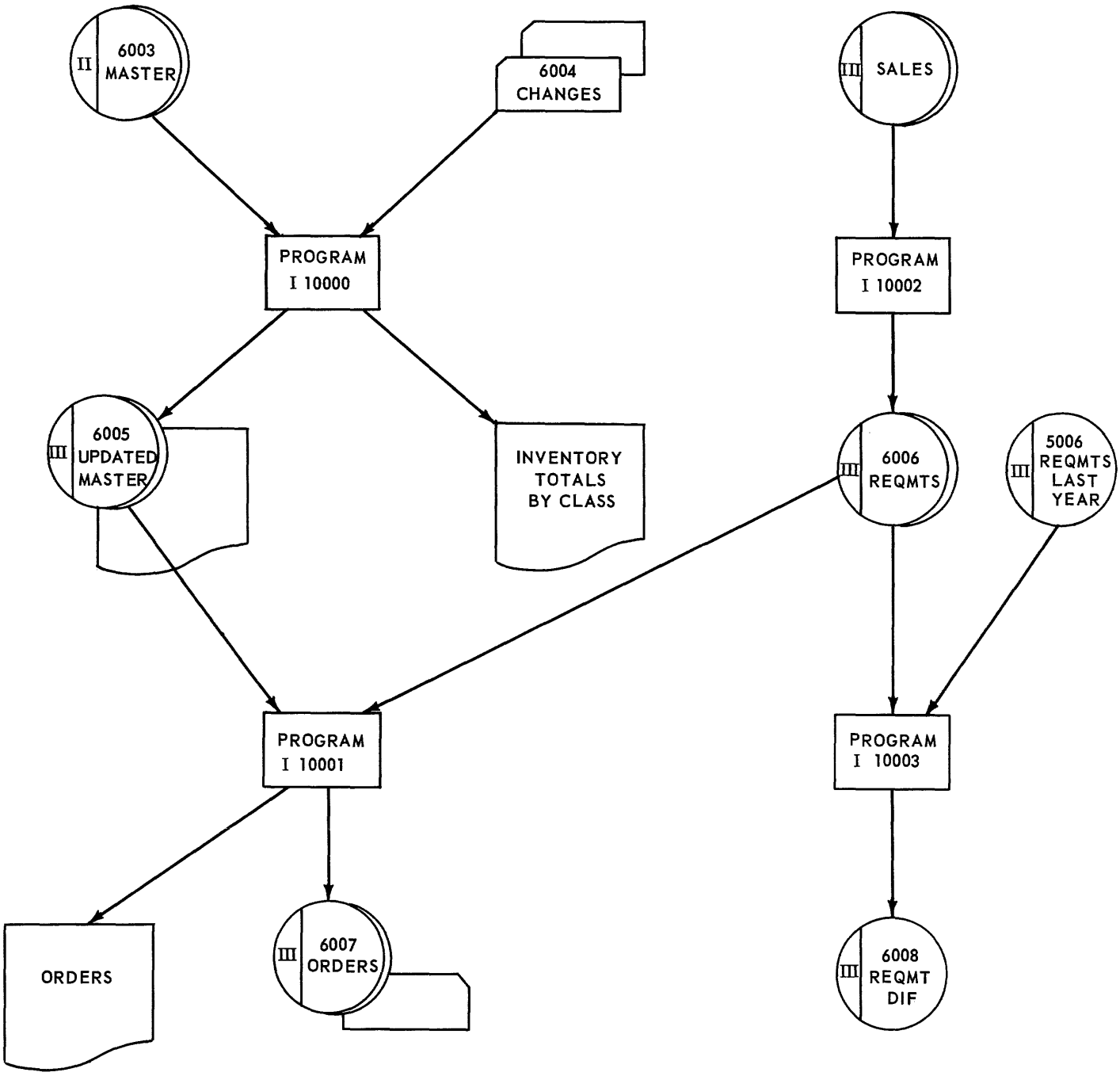
Path of flow

ENVIRONMENTAL FLOW CHART



EXAMPLE A

SYSTEMS FLOW CHART



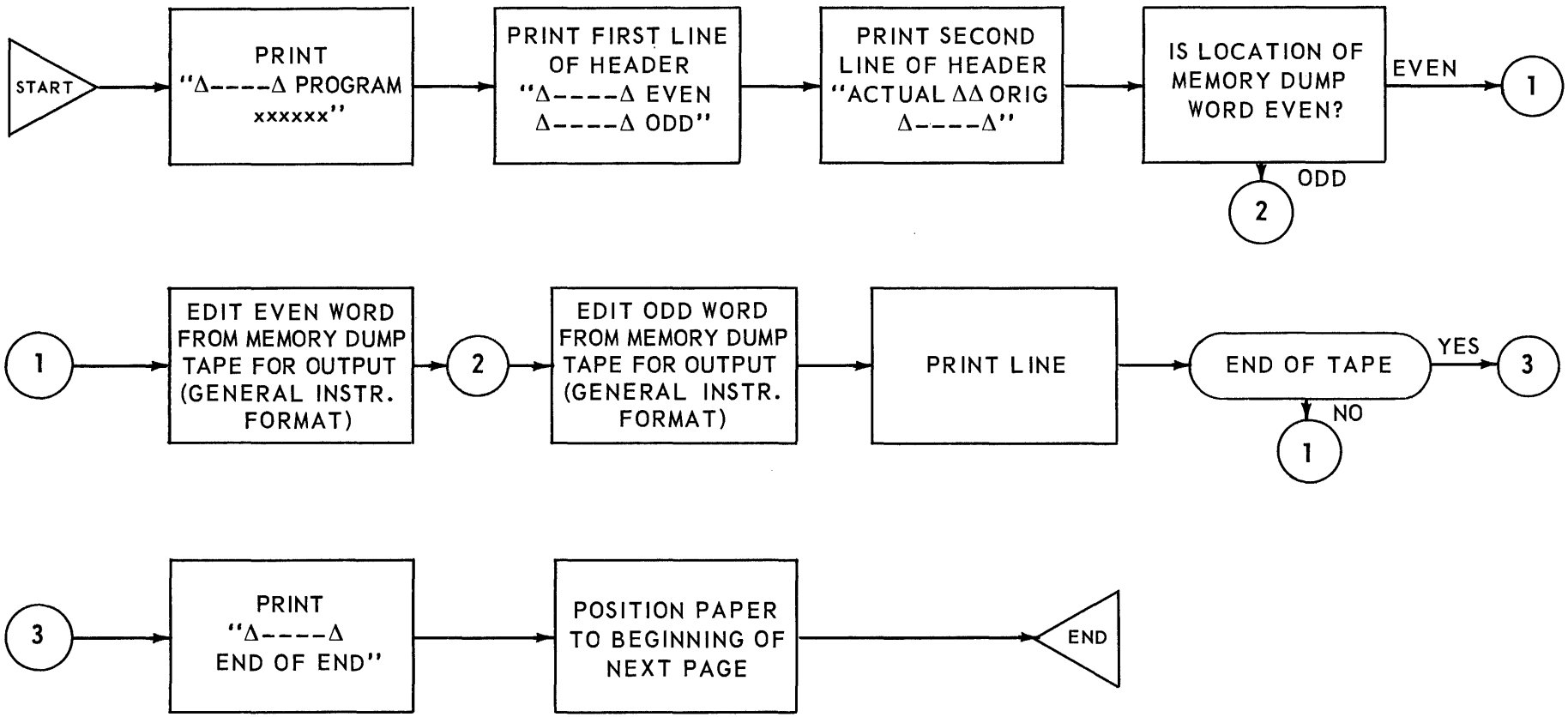
EXAMPLE B

B. Block Diagram

A block diagram (see Example C) describes the specifications in concise common language block diagram form. It presents the logical solution of the problem based strictly on the inherent elements of information and the criteria for their interrelationship and processing. The purpose of the block diagram is to represent step-by-step processing required to accomplish the objectives of the machine run. It depicts a completely general, logical treatment of the information and ideally not restricted by specific computer limitations or characteristics.

A block diagram emphasizes the processing steps required to produce a specific output from a specific input.

EXAMPLE BLOCK DIAGRAM



EXAMPLE C

C. Detailed Flow Chart

After the Block Diagram has been created, the logic of the machine must be applied. The Detailed Flow Chart is an explosion of each of the blocks in the Block Diagram. It is used to describe the sequence of logical steps which must be performed by a specific computer and is, therefore, concerned with the detailed machine operations. The Detailed Flow Chart is the medium from which the written program is derived.

IV. Block Diagram, Detailed Flow Chart Format and Techniques

A. 1. Flow charting is made easier if standard rules are adopted. The use of standard symbols is not only an aid in preparation, but is visually suggestive of the function being performed. A systematic approach to flow charting through standard rules and symbols makes for charts that are both efficiently organized and well-expressed.

2. In general, the flow of data or documents is charted across the page from left to right, or down the page. Flow lines from right to left or up the page are to be avoided, although permissible if they are relatively short.

3. Some flow charts could require many flow lines from one section to another; putting all of these lines in could clutter a chart making it difficult to follow. These lines are omitted and the proper flow is indicated by circled numbers called "connectors."

4. A standard flow charting paper should be adopted and used.

5. Each flow chart should be identified in the lower right hand corner. The identification should include:

   a. Program Name
   b. Date
   c. Ident.
   d. Programmer(s)
   e. Date Revised
   f. Revised By

   This may be expanded as necessary.

6. Should non-standard abbreviations or symbols be used in the flow chart, a legend giving the complete definition of such symbols or abbreviations must be stated on the flow chart page on which they occur.

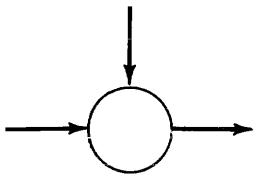
7. Operations on a flow chart should be self-explanatory. In preference to symbolic statements, English statements should be used in decision boxes and operation boxes.

8. If a major routine uses a subroutine that has been completely documented, the detailed flow chart of the subroutine should not be included in the documentation of the major routine.

9. Some open subroutines, because of their simplicity, may not require flow charts, but all higher levels of programming products require flow charts for complete documentation.

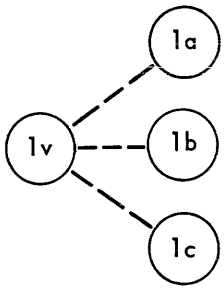
10. Flow charts must be related to the written source code by writing the coding tag, label, sequence, or operation number at the beginning of each non-branching path on the chart.

A. Standard Symbols Used in Preparing a Block Diagram or a Detailed Flow Chart.



Connectors

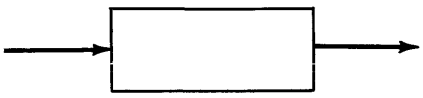
The Connector is a means of connecting one path of flow to another. This system eliminates the necessity of crossing lines to obtain direct connections. If connector transfers from or to another page, page number should be written beside the connector.



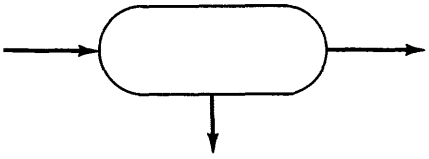
Variable Connector or Switch

A switch indicates a point in an operation at which several courses of action are possible, the correct one being determined by a condition prevailing elsewhere in a routine.

Operation Box



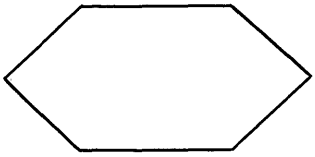
This symbol represents a processing operation, such as a mathematical operation, data transfer, or the setting of a switch. It does <u>not</u> include the operation of comparison



Decision Box

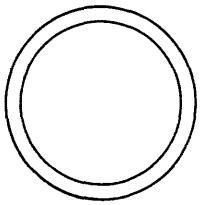
The Decision Box symbol indicates a logical choice.

The colon (:) is used to separate two factors being compared.



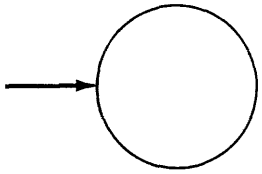
Subroutine

The execution of a separately defined, open or closed subroutine at a point in the flow. The function performed by this subroutine should be described in brief within the symbol. The detailed flow chart for the subroutine must appear elsewhere in this routine's overall chart.



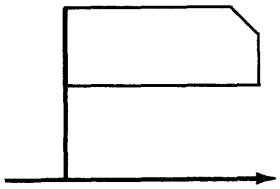
Library Subroutine

The execution of an open or closed library subroutine at a point in the flow. The identification of the library routine should be written within the symbol. The detailed flow chart for the library routine should not appear in the overall chart for the routine calling on the library.



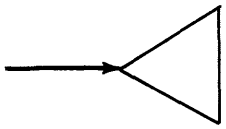
Start, Stop, Alarm Stop

The circle used for this symbol should be larger than the circles used for connectors. The Stop number should be printed within the symbol.



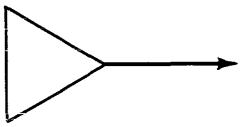
Flag

Assertion or explanation. Conditions existing at this point in the flow are described by statements written within this symbol.



End Subroutine

The ending point of a closed subroutine. The name of the subroutine is written outside the symbol.



Start Subroutine

The starting point of a closed subroutine.

The name of the subroutine is written outside the symbol.

B. Descriptive Characters Used in Preparing Charts.

To minimize writing on flow charts the following descriptive characters should be used where applicable.

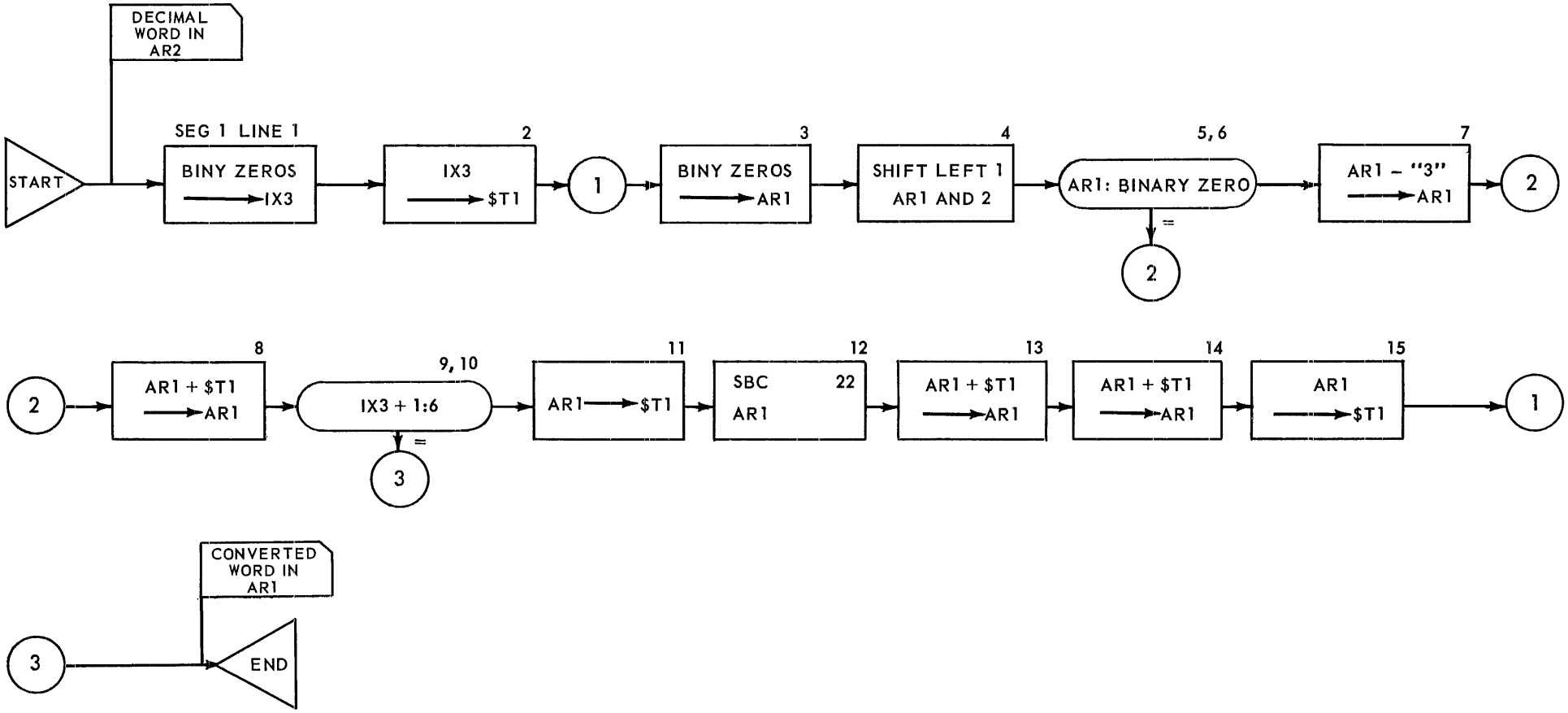
Character	Description
:	Compare
>	Greater than
<	Less than
=	Equal
≠	Not equal
≤	Less than or equal to
≥	Greater than or equal to
→	Used within an operation box to signify transfer of data
-	Minus
+	Plus
0	Zero
•	Set
Σ	Sum
TS	Temporary Storage
FS	Field Select
IA	Indirect Address

C. To avoid confusing certain alphabetic characters with numeric digits in written form, in communications within the department and externally when necessary (primarily for ease and clarity for keypunch operators or unitypists), the following conventions should be practiced by all personnel:

O Alphabetic	written with a O
Numeric	written as 0 (spoken as zero)
1 Alphabetic	written as I or i
Numeric	written as | (straight vertical line)
S Alphabetic	written with tails to distinquish from numeric 5.

Z Alphabetic	written with slash to distinguish from numeric 2: Ƶ
Δ Blank	Used to represent blank position.

EXAMPLE: DETAILED FLOW CHART



EXAMPLE D

V. Program Listing (Source-Object Code)

As an output of the SALT assembly routine, a listing is created on tape. This tape will be used to create two listings (original and duplicate).

The programmer will include the above listings in the program documentation.

VI. Data Designs

A. Cards

1. Drawing of card format indicating field position, field names, structure (alphabetic or numeric) etc.. (See suggested form, Example E.)

2. Sequence

B. Tapes

Define the fields and show the order of the information on the tape (see suggested form, example F). The following information will also be shown:

1. Field size

2. Define fields as alphabetic, numeric, alphanumeric, or blank.

3. The significance of numbers or zero justification.

4. Inter-record gap must be shown to indicate end-of-record format.

5. If the formats shown on the forms are edited for card-to-tape, tape-to-card, or tape-to-printer operations, check C/T, T/C, T/P as applicable.

6. Additional descriptive information will be entered in the upper right-hand corner of the form as shown.

7. Number of words per item.

8. Number of items per block.

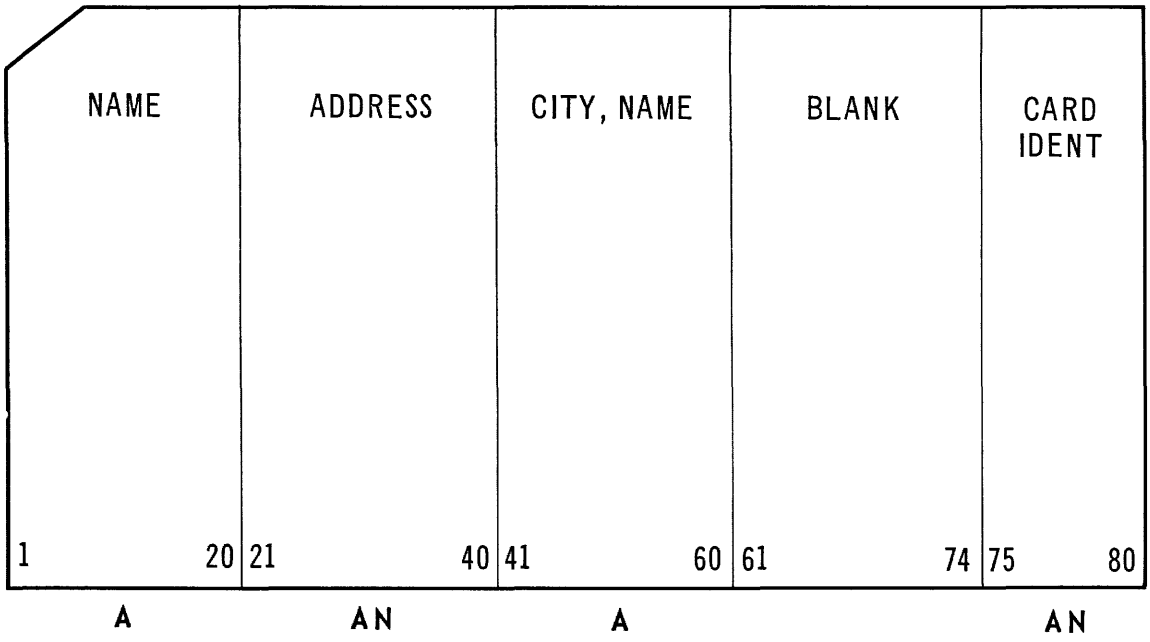
9. Sequence.

10. Format should not reflect standard label block or standard sentinel blocks.

C. Printer

Show actual headings, page number and field positions on printer paper form.

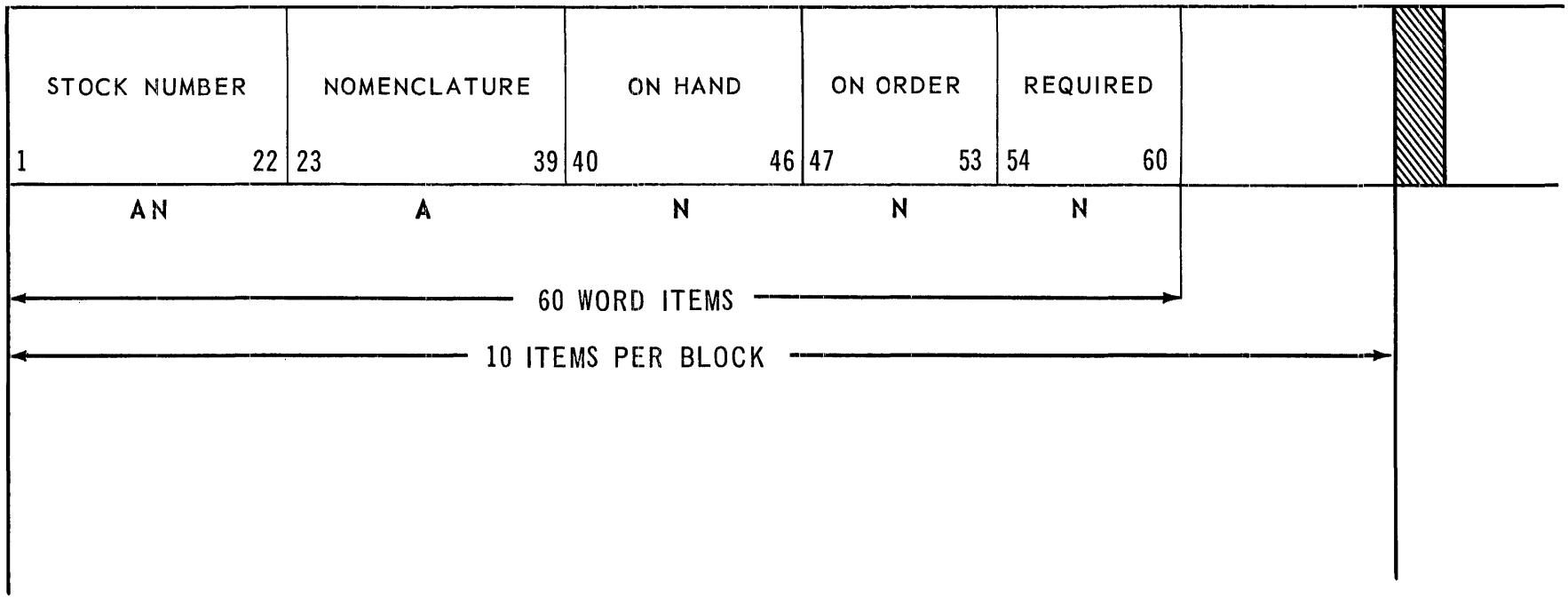
CARD FORMAT



SEQUENCE: (1) NAME

EXAMPLE E

TAPE FORMAT



SEQUENCE: (1) STOCK NUMBER

EXAMPLE F

D. Sample Typeouts

A copy of the complete actual typeouts of each run will be included in the program documentation.

E. Sample Reports (Print Outs)

After the first production run, a specimen of all printed reports will be obtained from tape to printer operations. The sample will be subject to the following restrictions:

1. The complete report will <u>not</u> be included in the program documentation.

2. Only a sample of different type pages will be included (Title page, sub-total page, last page if Grand Total, etc.).

VII. Operators Instructions

A. For each phase of a program, an operator's instruction sheet should be completed. The form allows for a description of the operating status of the following:

1. Tape Units (Read Status)
2. Card Reader
3. Tape Units (Write Status)
4. Card Punches
5. Printers
6. Typewriters
7. Options
8. Dating Parameters
9. Start
10. Rerun

The form is self-explanatory and all of the items listed must be completed, either with operating information, or if the machine component is unused by the program, enter the word "none" on the form. A suggested Operator's Instruction Sheet is shown in Example G.

B. In addition, a list of the stops and/or error type-outs should be included. The list of type-outs will state the following:

1. Stop number of Error type-out.
2. Assembly item-nr. or actual address.
3. Explanation of type-out with detailed steps to be taken to achieve an efficient and accurate restart.
4. Type-in.

OPERATOR'S CHECK LIST

Run Identification
Sub-Project
Project
Date

Prepared By
Approved

Total Memory
Words Required

OPTION PARAMETERS	EFFECT OF SETTING	DATING PARAMETERS	

		IDENT	REELS	SOURCE	REMARKS
INPUT	TAPE UNIT TAPE UNIT TAPE UNIT TAPE UNIT TAPE UNIT TAPE UNIT TAPE UNIT				
	CARD READER	CARD NR			

		IDENT	REELS	REMARKS
OUTPUT	TAPE UNIT TAPE UNIT TAPE UNIT TAPE UNIT TAPE UNIT TAPE UNIT TAPE UNIT			
	PUNCH	CARD NR		
	PRINTER	PAPER FORM		
	TYPEWRITER			

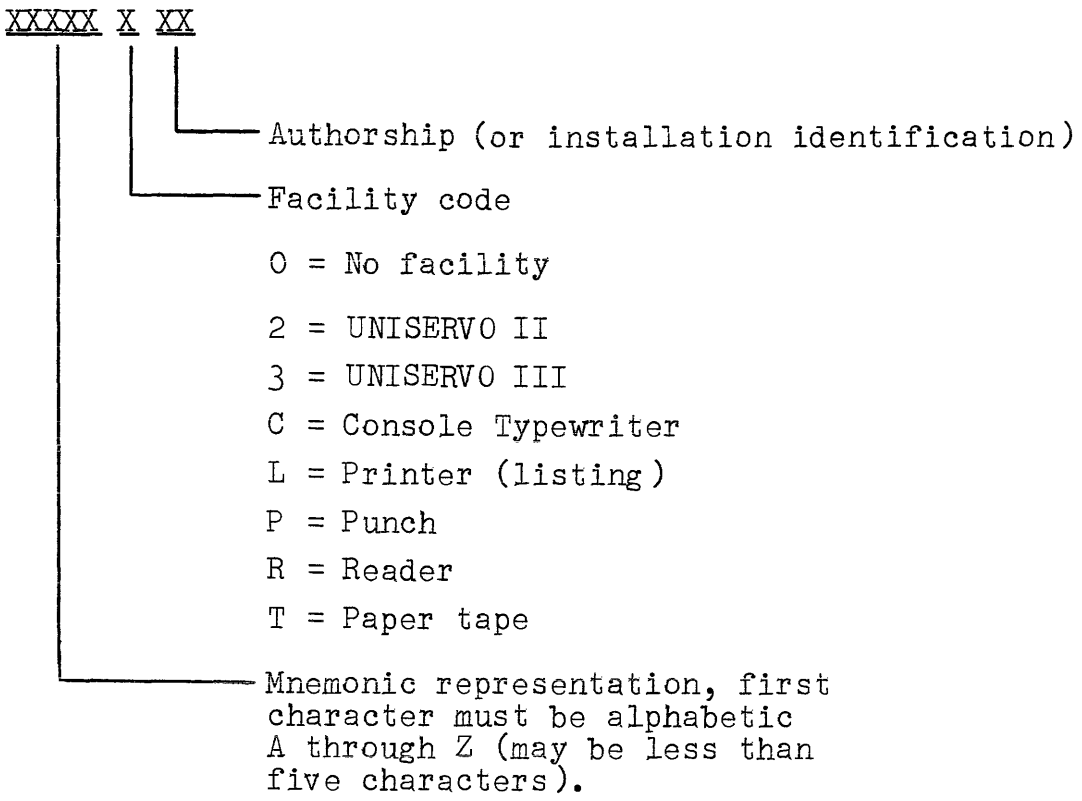
INSTRUCTIONS	START INSTRUCTIONS	RERUN INSTRUCTIONS

EXAMPLE G

VIII. Subroutines

A subroutine is a computer routine which cannot by itself perform a useful function until it is incorporated into a larger routine. A subroutine not stored in the direct path of a main routine is called a CLOSED SUBROUTINE. Such a subroutine is entered from the main routine by a jump instruction, and provision is made to jump back to the main routine after the subroutine has performed its function. A subroutine which is inserted directly into the path of a main routine where needed, is called an OPEN SUBROUTINE.

A. A Subroutine name cannot exceed eight characters. The first character must be alphabetic A through Z. Only letters A through Z and numbers 0 through 9 may be used.



B. Subroutine Documentation

Written documentation for subroutines should include:

1. Subroutine Name
2. Purpose
3. Description
4. Programming Procedures

    a. Subroutine call

b. Entrance Requirements
c. Entrance
d. Exit Conditions
e. Execution Time
f. Length of Coding
g. Constants
h. List of other subroutines referred.

5. Operating Procedures

6. Flow-charts

7. Coding

IX. Generator Naming

A generator is a routine that can be referenced by an assembly language source program as though it were a machine instruction. When used, it implies the generation of two or more lines of machine coded instructions in the object program. The availability of generators, extends the instruction repertoire of an assembly language and, in effect, makes of an assembly system a compiler.

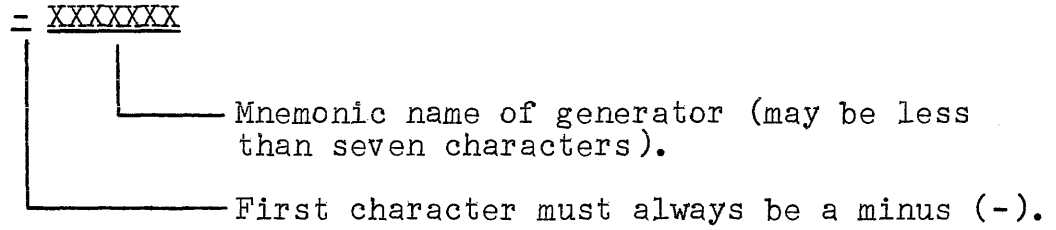
Generator Naming for Generators on the SALT Library Tape

A generator name cannot exceed eight characters.

The first character must be a minus (-).

The remaining characters should be a meaningful mnemonic.

Only letters A through Z and numbers 0 through 9 may be used in the mnemonic.



X. Initializing Procedures

Programmers, when preparing a program should never assume memory to be blank or zero. Blank areas or zero areas must be set up by the program either by reading in blanks or zeros from tape or creating them in memory by programming.

Programmers should never assume that the input or output tapes have been rewound when commencing a machine run. Also, tapes should be rewound when they are completed.

All counters and all switches must be initialized by programming. (Initial switch settings and counters should not be read in from tape.)

Manual operations should be kept to an absolute minimum by programming. Any necessary manual operations must be positively checked by the program, and the typewriter automatically used to call the operator's attention to double check the manual operations performed.

XI. Memory Dump for Rerun

Purpose - To provide a standard method of determining:

1. Which programs are to include a restart procedure.

2. How often a rerun point is to be written in these runs.

Procedure - Any program requiring tape changing previous to the final end of job, must contain a provision for rerun point writing. When multiple reels of input are to be read, or multiple reels of output are to be produced, a rerun-point must be written shortly after the tape label is read (or written) on the second and succeeding reels. A check may be made on the block counter for the tape unit involved, and when it is equal to 00002, write a rerun-point. This procedure will eliminate the necessity of "backing up" to some previous reel, should a memory failure occur.

In those runs where tape changing, as outlined above, does not occur, a rerun-point must be written when the processing time exceeds fifteen minutes.[2]

If tape changing does take place, and the processing time of an input reel (or input cards) exceeds fifteen minutes, a rerun-point must be taken every fifteen minutes, as well as every tape change. The rerun-point at fifteen-minute intervals is taken by approximating the number of major input or output records that will be processed in fifteen minutes, and setting up a counter containing this number, reducing it to zero, while processing, then writing a rerun-point.

XII. Validity Checking of Raw Input Data

Each program (other than card to tape, tape to printer, etc., service routines) reading raw data from cards, input card-image tapes or unityped reels, must perform validity checks on each field that will ultimately be processed.

Raw data is defined as data which has been created manually, then keypunched on cards or typed on tape.

Input validity checking is not required on data generated by a computer program.

---

[2]This fifteen minute interval may be varied according to installation and/or program needs.

Remington Rand Univac

DIVISION OF SPERRY RAND CORPORATION

PRINTED IN U.S.A. UT2500