N-23-4-3

TM-891/007/00

TECHNICAL MEMORANDUM

(TM Series)

This document was produced by SDC in performance of contract AF 19(628)-1648, Space Systems Division Program, for Space Systems Division, AFSC.

Utility System Program Specifications

Design Specifications for Symbolic Assembly Routine (SABER) Milestone 4

By
F. J. LaChapelle

25 May 1963

Approved
J. A. Kneemeyer

SYSTEM DEVELOPMENT CORPORATION
2500 COLORADO AVE.
SANTA MONICA
CALIFORNIA

A-1159

TABLE OF CONTENTS

INTRODUCTION

This document is one of a series of TM-891 volumes established for Utility System program specifications.

Comments on this document must be received by 1 July 1963 to be reflected in the final design criteria. It is anticipated that this feature will be available in the AF/CPL on 1 August 1963. The publication of a volume in the TM-705 (Systems Manual) series will officially announce the completion of this project.

1.0 PURPOSE

SABER will be used to assemble a special format symbolic card deck on the CDC 1604 computer. The binary output of a SABER assembly will be loaded by the MTCII (Master Tape Control) program.

Currently LARII (Lockheed Assembly Routine II) is being used for assemblies, but the expansion of the COPII system has necessitated a more comprehensive assembly technique. The result has been extensive modification and additions to LARII to produce SABER.

2.0 USAGE

2.1 FUNCTION CARD

SABER is called and operated by the function card

*SABER L T_1 T_2 T_3 T_4 T_5 T_6

where

L is to force a LARII type assembly. If this parameter is an S, a SABER type assembly will be forced. (See the Section on compatibility for an explanation of LARII and SABER type assemblies).

T_1 is the prestore tape to be assembled (nominal selection is 2).

T_2 is the scratch tape (nominal selection is 3).

T_3 is the list tape (nominal selection is 5).

T_4 is the binary output tape (nominal selection is 6).

T_5 is the second list tape (nominal selection is 4).

T_6 is the second binary output tape (nominal selection is 7).

If a nominal selection is to be changed, the parameters preceding it must be present.

Input can be from the card reader ($T_1 = 0$).

Output can be on the printer ($T_3 = 13$) and the punch ($T_4 = 0$).

2.2 LARII AND SABER COMPATIBILITY

The new features which will be added to LARII to transform it into SABER are of two types. The first type is a brand new capability such as literals or the VFD pseudo operation. The second type is an extension or re-configuration of an old capability, e.g., M-term arithmetic. In the case of the new features, there is no compatibility problem since the LARII language is merely a subset of the SABER language.

On the other hand, where existing LARII specifications have been altered, incompatibilities do exist. The two areas of incompatibility are:

1. LARII allows symbols with special characters in them (, ( ) * / = $). SABER recognizes symbols constructed from the 26 letters and ten digits only.

2. LARII has a fixed M-term (cols 20-40) and it may contain blanks in any position. Remarks start in column 41. SABER terminates the M-term on the first blank and the remarks field may start any place following this blank.

Since it is the intent of SABER to be as compatible with LARII as possible, a compatibility feature must be added to SABER. First of all, two distinct modes of assembly, SABER and LARII, will be recognized. Programs which must be assembled with LARII will be assembled in the LARII mode and all other programs will be assembled in the SABER mode. The selection of modes will be automatically determined by SABER except in one instance, i.e., problem area #2 above. For this case and for the programmer who has a particular preference for one mode or the other, a mode parameter will be provided on the SABER function card. The mode of the assembly, LARII or SABER, will be printed at the end of the assembly listing (see Section 7.1.9).

The mode switch is set only once per assembly; by the function card or the decision process described above. If special characters are encountered in location terms while in the SABER mode, those symbols will be flagged as an error. All special characters in M-terms will assume their special SABER meanings as described under the separate pseudo operation definitions.

If SABER specific type pseudos are encountered while in the LARII mode, they will be flagged as an error and no space (machine words) will be reserved for them.

3.0 SYMBOLIC CARD FORMAT

The general card format of the source program is:

Cols 1-8	Location field - This field is either blank, or a symbol of 8 characters or less (no imbedded blanks), or + or - in column one and blank elsewhere.

Col 9	Always blank
Cols 10-15	Operation field - This field contains a 1604 machine instruction or pseudo instruction left justified with no imbedded blanks. The octal equivalents can be substituted for 1604 machine instructions.
Col 16	Always blank
Cols 17-18	B-term field - This field contains a numeric index designator or a symbol or blanks.
Col 19	Always blank
Cols 20-69	M-term field and/or comments field. If the M-term field is not zero, it must begin before column 25. It extends to the first blank. The comments field begins immediately following the first blank and extends for the remainder of the field.
Cols 70-80	Gangpunch identification and sequence numbering.

4.0 M-TERMS OF 1604 MACHINE INSTRUCTIONS

An M-term field is constructed from symbols and constants separated by the binary operators "+, -, /. *".

4.1 CONSTRUCTION OF SYMBOLS AND ELEMENTS

A symbol consists of a string of one to eight characters. Each character must be an alphabetic character or a digit and these may be combined in any fashion with the following exceptions:

1. At least one character must be a letter.

2. A string of all digits followed by the letter "B" is an octal constant, not a symbol.

No special characters (+ - , ( ) * / = $) will be recognized as a part of a symbol.

There are four types of elements. They are: (1) Symbols, (2) Decimal integers, (3) a string of octal digits followed by a "B" which is treated as an octal integer, and (4) an asterisk. An asterisk is recognized as an element meaning "this location" when it is the first element in the M-term field or follows a binary operator.

4.2 CONSTRUCTION OF EXPRESSIONS

Terms are formed by performing multiplication and division on elements which are separated by the characters "*" and "/", respectively. Each term is evaluated from left to right and only the integer part of a division result is retained at each step. Terms are combined by the characters "+" and "-" to form the entire M-term expression value. The expression ** will receive the value absolute zero.

In the case of relocatable programs, the relocation characteristics of all elements are combined to arrive at an expression which is either absolute or relocatable, or in error. M-term arithmetic, then, is general integer arithmetic with the exclusion of parentheses. This type of M-term is legal in the M-term of any 1604 machine instruction or in any pseudo operation which does not require a special format M-term. The M-term format described above will be referred to hereafter as an arithmetic M-term.

5.0 LITERALS

Any field which requires an arithmetic M-term may instead contain a literal.

A literal M-term contains the data to be operated on rather than the address of the data as with other types of M-terms. The appearance of a literal causes SABER to form a full word constant, equivalent in value to the content of the literal subfield; store this constant in the next available location at the end of the program; and replace the address field of the instruction containing the literal with the address of the constant which was just generated.

There are seven types of literals:

1. Decimal
2. Octal
3. M-term
4. Hollerith
5. Flexowriter
6. Teletype
7. VFD

The presence of a literal is characterized by the first character of the M-term being an "=". The information following the "=" (the literal subfield) determines the value of the literal and its type.

5.1 DECIMAL LITERALS

A literal is evaluated as a decimal constant if the characters following the "=" form a valid decimal data item and there is no other information given. See the explanation of the DEC Pseudo-op for a definition of decimal data items. A decimal literal must not end with a B (although a B can be imbedded).

5.2 OCTAL LITERALS

A literal is evaluated as an octal constant if the characters following the "=" form a valid octal data item and this data item is followed by the character "B". See the OCT pseudo for a definition of octal data items. Binary scaling is allowed in octal literal subfields as it is in the OCT pseudo. In this case there are two "B's" in the subfield (one before the scaling factor and one following it). However, an "S" can be substituted for the first "B".

5.3 M-TERM LITERALS

An M-term literal provides a method for specifying the upper and/or lower address of a cell. There are three possible formats:

1. =ME1
2. =ME1,
3. =ME1,E2

where E1 and E2 are any valid arithmetic M-term expressions.

Example 1 would form a full word with the value of E1 in the lower address and a zero in the upper address.

Example 2 would form a full word with the value of E1 in the upper address and a zero in the lower address.

Example 3 would form a full word with the value of E1 in the upper address and the value of E2 in the lower address.

These words have relocation bits as all other words do, and the addresses formed will retain their true relative value if they are relocatable. The operation and B-Term of both upper and lower halves will always be zero.

5.4 HOLLERITH, FLEXOWRITER, AND TELETYPE LITERALS

A Hollerith Literal has the following format: =HCCCCCCCC, where CCCCCCCC represents eight characters which may be anything, blanks included. It is evaluated in the same manner as the M-term of the BCD pseudo op (with an understood word count of one) and its purpose is to provide a cell containing

Hollerith information. Flexowriter and teletype literals have the same format and they are evaluated in the same manner as the M-term of a FLX or TEL pseudo op, respectively. Special FLX and TEL codes are recognized as explained in Appendix B.

5.5 VFD LITERALS

A VFD literal consists of any number of subfields legal for the VFD pseudo (except literal subfields) for which the total combined bit count is 48 or less. This literal is specified by the character "=" followed by a V.

6.0 PSEUDO-OPERATIONS

6.1 IDENT PSEUDO

The IDENT pseudo operation is used to identify each program and must be the first card in the deck. Its M-term consists of the name of the routine followed by one or more blanks and identification information not to extend beyond column 40. The identification information is a maximum of seven characters with no imbedded blanks. It is optional.

6.2 ORG, ORGR PSEUDOS

To start a sequence of instructions at a particular location, an operation code, ORG(R), is followed by an arithmetic M-term. If the operation code, ORGR, is used, the binary output will be relocatable. If ORG is used, the output is absolute. If the M-term is a symbolic value, the symbol must have been defined as a previous location term. If this pseudo does not precede the first storage assignment, the program will be assembled relocatable at location $10,000_8$. Any number of ORG(R) cards may exist in one deck.

6.3 BSS PSEUDO

A BSS pseudo operation is used to reserve a block of consecutive cells. If a location symbol is present, it is assigned to the first location in the block. The arithmetic M-term specifies the number of locations to be reserved. The block is reserved at the place where the BSS card occurs with no change in the instruction sequence. Any symbols occuring in the M-term must have been previously defined.

6.4 BES PSEUDO

The BES pseudo operation is used to reserve a block of consecutive cells and assign a location symbol to the last location of the block. The arithmetic M-term specifies the number of locations to be reserved. The block is reserved at the place where the BES card occurs with no change in the instruction sequence. Any symbols occuring in the M-term must have been previously defined.

6.5 BCD PSEUDO

To insert packed, binary-coded, decimal characters into consecutive words, the BCD operation code is used. A location term is optional. The M-term consists of a single digit N ($1 \leq N \leq 6$) and the succeeding 8 x N characters, including blanks. This results in N computer words, each containing eight BCD characters. The location symbol (if any) is assigned to the first word. Any information following the 8 x N characters is treated as remarks.

6.6 FLX PSEUDO

To insert Flexowriter codes into conseuctive words, the operation code, FLX, is used (location term is optional). The results and format are similar to the BCD pseudo, except that resulting codes are Flexowriter codes. Flexowriter functions such as shift-up, shift-down, etc., are represented by the use of special codes which are listed in Appendix B.

6.7 TEL PSEUDO

To insert teletype codes into consecutive words, the operation code, TEL, is used (location term is optional). The results and format are similar to the BCD function, except that resulting codes are Teletype codes, with each five-bit Teletype code re-expressed as a six-bit code (i.e., an extra 2^5 = "0" bit inserted). Teletype functions, such as shift for numbers, are represented by use of special codes which are listed in Appendix B.

6.8 OCT PSEUDO

To insert octal constants, the operation code, OCT, is used. A location term is optional. The M-term consists of one or more consecutive sub-terms, separated by commas, with each sub-term consisting of an octal data item. Octal data items consist of a sign (+, -, or none) followed by up to 16 octal digits. The octal digits may be followed by a "B" or "S" and a positive decimal integer which will result in a binary scaling (shift to the left) of the data word specified.

6.9 DEC PSEUDO

To generate a decimal constant at a location, the operation code, DEC, is used. A location term is optional. The M-term consists of one or more consecutive sub-terms (decimal data items), separated by commas. A decimal data item consists of up to four parts: 1) a sign (+, -, or none, interpreted as +), 2) a value of up to 14 decimal digits, 3) a decimal scaling consisting of a "D" or an "E" followed by a sign (+, or -, or none and one or more decimal digits), and 4) a binary scaling consisting of a "B" or "S" followed by a sign (+, -, or none) and one or more decimal digits. If the constant contains a decimal point in any position, it is packed into floating point format. If no decimal point is present, an integral constant will result. Both the decimal and binary scaling are optional, and if both are present they can appear in either order.

6.10 REM PSEUDO

If it is desired to enter a line of remarks on the assembly listing, the operation code, REM, is followed by a maximum of 61 characters, starting in column 20 or any subsequent column. On the assembly listing, the operation code REM is suppressed. If one or two REM cards immediately follow the IDENT card, the information on these cards will be used to head each page of the assembly listing. The identification field (cols 70-80) is suppressed on these heading cards.

6.11 SPACE PSEUDO

The operation code, SPACE, is used to insert spaces between lines of print in the assembly listing. The arithmetic M-term contains the number of spaces desired. This pseudo is not printed.

6.12 EJECT PSEUDO

A card containing the pseudo operation code, EJECT, will cause the printer to eject paper to the top of the next page. This pseudo is not printed.

6.13 FWZ PSEUDO

To define the upper and/or lower address of a cell, the FWZ pseudo operation can be used. FWZ can have a location term. If the M-term contains one arithmetic M-term, its value is placed in the lower address. If this M-term is followed by a comma, its value is placed in the upper address. In the case of two M-terms which are separated by a comma, the value of the expression before the comma goes in the upper address of the word and the value following the comma goes in the lower address. If the upper address is a VFD literal, the total bit count must be exactly 48. If a B-term is present, it is placed in the lower B-term. If there is no B-term or M-term(s), a zero word is generated. The upper operation code and B-term and the lower operation code are always zero.

6.14 VFD PSEUDO

A Variable Field Definition pseudo will generate one or more words of data made up of one or more VFD subfields. If a symbol appears in the location term, it is assigned to the location of the first word generated. The B-term field must be blank.

Seven types of subfields exist:

1. M-term
2. Octal
3. Decimal
4. Hollerith

5. Flexowriter
6. Teletypewriter
7. Literal

The format of a subfield is:

1. The number of bits of data this subfield is to generate. This is specified by an unsigned decimal integer between one and 96.

2. A single letter to specify the type of information contained in the subfield. This letter follows the decimal integer.

   The letter "M" specifies an arithmetic M-term format subfield.
   The letter "B" specifies an Octal data item format subfield.
   The letter "D" specifies a Decimal data item format subfield.
   The letter "H" specifies a Hollerith format subfield.
   The letter "F" specifies a Flexowriter format subfield.
   The letter "T" specifies a Teletypewriter format subfield.
   The letter "L" specifies a Literal format subfield.

   The type letter is followed by a slash, "/", and the actual data subfield is contained between this slash and the next blank or comma (whichever is encountered first). In the case of a comma, another subfield begins immediately following the comma. A blank terminates the VFD pseudo op M-term.

The type of information following the slash depends on the type letter.

1. In the case of an M, the data item consists of any legal arithmetic M-term expression (except a literal).

2. In the case of a B, the data item consists of any legal octal data item.

3. In the case of a D, the data item consists of any legal decimal data item.

4. In the case of an H, F, or T, the data item consists of any legal M-term for the BCD, FLX, or TEL pseudo operation, respectively, except that no character can be a <u>comma</u> or a <u>blank</u>. Also, a word count cannot be specified.

5. In the case of an L, the data item consists of a legal literal subfield. If the literal is of the M-term type, it must contain a comma to specify upper and lower addresses of the literal word (even if one is zero) unless it is the last subfield in the VFD. (VFD literals are not allowed in VFD subfields.

   Any number of subfields may be used.

The subfields are taken in order and are left justified. A subfield may take up more than one word. If the bit count for the first subfield is 50, the first 48 bits would go in word one, and the last two bits in Bit 47 and Bit 46 of word two. This procedure applies independently of the starting place of the subfield. One subfield may require up to three words to contain it. Each subfield is exactly the length of its bit count, regardless of the number of bits required to generate the data item. If the total number of bits in all subfields is not a multiple of 48, the low order bits of the last word are filled out with zeros.

6.14.1 M-term Data Items. The data item in a M-term subfield is evaluated as if it were a standard arithmetic M-term, but without cognizance of literals (a "C" error will be flagged if the character "=" exists in this data item). For example, let the specified bit count be n. If the data item just generated is less than n bits in length, it will be right adjusted in an n bit field, with enough leading zeros to make an n bit byte. The n bits are packed in the manner described above. If the data item is greater than n bits in length, only the rightmost n bits are used. Neither of these conditions is regarded as an error. If the data item generated is a relocatable expression, then the subfield must be so situated with respect to other subfields that its rightmost bit falls in Bit 0 or Bits 24 of a computer word. Furthermore, the bit count must be 15 or more. If this requirement is not met, a relocation error will be flagged.

6.14.2 Octal Data Items. An Octal subfield data item may consists of any valid M-term for the OCT pseudo operation. For example, let n be the bit count for the subfield. If the data item as generated occupies less than n bits, sufficient zeros are placed to the left of the value to form an n bit subfield. If the data item occupies more than n bits, only the rightmost n bits are used. These conditions will not be flagged as errors.

6.14.3 Decimal Data Items. A Decimal subfield data item may consists of any valid M-term for the DEC pseudo operation. After the value is generated, it is treated exactly the same as an octal subfield.

6.14.4 Hollerith Data Items. The data item in a Hollerith subfield consists of a string (up to 16) legal BCD characters, none of which is a comma or a blank. Let n be the bit count for the subfield. If the data item generated is longer than n bits, only the rightmost n bits are used. If it is less than n bits, sufficient blanks are placed to the left of the value to form an n bit subfield. If the bit count is not a multiple of six, then the leftmost blank has its left bits truncated. The bit count will not be flagged as an error in either case.

6.14.5 Flexowriter Data Items. The data item of a FLX subfield is converted in the same manner as the M-term of a FLX pseudo op. There can be up to 16 characters not counting the special codes (minus zero). Minus zero retains its meaning (shift mode), but none of the characters can be a comma or a blank. See Appendix B for the use of minus zero. For example, let n be the bit count. If the data item generated occupies less than n bits, sufficient blanks (flexowriter code for blanks) are placed to the left of the value to create an n bit subfield. Truncation of the leftmost character follows the same rules as in the case of Hollerith data items. None of these conditions will be flagged as an error.

6.14.6 Teletypewriter Data Items. The data item of a TEL subfield is converted as the M-term of a TEL pseudo operation. The same rules apply as in the case of the Flexowriter data item.

6.14.7 Literal Data Items. The data item in a Literal subfield consists of a type letter followed by the literal data item. There are two special cases with literals. An octal literal has its type letter (B) at the end and a decimal literal has no type letter at all. The rules which govern the formation of normal literals apply in the case of VFD literal subfields also. The appearance of this type of subfield directs the assembler to generate a data item according to the rules governing literal subfields, and to then let the subfield in the VFD instruction receive the location of the literal which was just generated. This subfield will be relocatable if the program is relocatable. The bit count for a literal subfield must be 15 or more. Furthermore, the rightmost bit of the subfield must fall in Bit 0 or Bit 24 of a word. A violation of either of these rules will be flagged as a relocation error, even if the program is absolute. If the bit count is greater than 15, the leftmost bits are filled with zeros. This is not flagged as an error.

The element asterisk may be used in an M-term subfield or an M-term literal subfield of a VFD M-term. The value assigned to the asterisk in both cases is the location of the word which contains the leftmost bit of the VFD subfield when it is assigned. The remainder of this subfield may be in the next word (or the next two words), but the first word is always chosen.

6.15 CTU PSEUDO

The CTU pseudo is used to extend a VFD field. This pseudo can not have a location term or a B-term. It must appear immediately following a VFD or another CTU. Any number of CTU's can follow a VFD, although the variable field definition must be started with a VFD. The M-term has exactly the same format as that of a VFD pseudo operation code. When a CTU is encountered, the last word generated by the previous VFD or CTU is not terminated and the effect is as if the M-term of the original VFD was extended. The last subfield on a VFD or CTU card does not have to be followed by a comma, whether a CTU follows or not.

6.16 DUP PSEUDO

The DUP pseudo operation will duplicate the information N times on the following M cards. The DUP M-term has the form "M, N", where M and N are arithmetic M-terms. If M and the comma are omitted, the card count is assumed to be one. A DUP pseudo operation can be followed immediately by any machine instructions or pseudo operations with the exception of the following pseudos: DUP, IDENT, FINIS, ORG, ORGR, REF, REFC, REFD, RST, WST, END, and EQU. The normal rules for forcing upper or lower apply to each instruction being duplicated. Either the DUP pseudo operation, or the following cards, can have a location symbol. The location symbol on the DUP pseudo will be assigned to the first word generated. Location symbols on instructions within the DUP field will be assigned the first time they are encountered; however, the subsequent repetitions of these instructions will be forced upper also.

6.17 EQU PSEUDO

The EQU pseudo operation is used to assign a location symbol to one of a group of working storages, to assign a value to a location symbol, or to equate two location symbols. An EQU must have a location term to be followed by an arithmetic M-term. If the M-term is evaluated as an absolute value, a "fixed" value is entered into the symbol table. Symbols used in the M-term do not have to be previously defined. The order of equate cards in a deck is not important. A literal cannot be substituted for the arithmetic M-term.

6.18 BLOCK PSEUDO

The BLOCK pseudo is used to logically separate parts of a program for assembly purposes. It consists of the mnemonic BLOCK in the operation field and a single symbol in the M-term. The symbol is the name of this program block. A program which is to consist of separate portions has each of its parts identifiable as either a BLOCK or SHARE. A SHARE is defined as a block with no name, e.g., a BLOCK pseudo with a blank M-term. Each BLOCK pseudo with a distinct name will originate another block. Blocks with duplicate names will add to the existing block of the same name. Ordinary references to symbols are treated in the following manner. If the reference is within a block, the symbols in that block are searched first. If the symbol is not found, the SHARE area is searched. If the symbol is not found there, it is treated as undefined. If the reference is within the SHARE area, the symbol is searched for in only the SHARE area and is undefined if not found.

Communication between blocks is accomplished by the following notation: A(B), where A is the symbol to be found, and B is the block which contains the correct A. If A is not found in the B block, the share area is searched automatically with a resulting undefined if it is not found there.

If the first portion of the program is not blocked, it will automatically be a share portion. Similarly, a program which contains no BLOCK pseudo is equivalent to one share block. If a blocked program contains no share block, then a reference within a block is searched for in that block only. Also, any reference to the share block will be undefined.

6.19 RST PSEUDO

This pseudo operation is used to modify the reference symbol table in SABER and/or to provide intercommunication between two sections of a program. The RST card must precede the first location term in the program and is followed directly by a binary symbol table deck punched by the WST pseudo-operation.

The reference symbol count is set to zero prior to the read of the binary symbol table deck. See Section 5 under BINARY OUTPUT CARDS for a description of the binary symbol table.

6.20 WST PSEUDO

The WST pseudo operation is used to punch a binary symbol table deck for use with the RST pseudo operation. Both reference symbols and internal program symbols are punched. The WST operation is executed when encountered during the first pass of the assembly. Only reference symbols are punched if the WST operation precedes the first location term in the program. The format of the binary symbols table deck is described in Section 8.0, BINARY OUTPUT CARDS.

6.21 REF PSEUDO

The REF pseudo operation is used to define a new entry in the reference symbol table. It must have a location term distinct from any symbol in the reference symbol table. The arithmetic M-term defines the symbol. Symbols used in the M-term must be previously defined.

6.22 REFC PSEUDO

The REFC pseudo operation is used to change the value of a reference symbol. It must have a location term identical to a symbol in the reference symbol table. The arithmetic M-term redefines the symbol. Symbols used in the M-term must be previously defined.

6.23 REFD PSEUDO

The REFD pseudo operation is used to delete a reference symbol. It must have a location term identical to a symbol in the reference symbol table. The M-term is not used.

6.24 END PSEUDO

To end the assembly of one program, the last card must contain the operation code, END. If a binary transfer card with the starting location (transfer address) of the program is desired, a reference to that location should be contained in the END card M-term. If no effective transfer card is desired, the M-term is left blank.

6.25 TCD PSEUDO

The TCD pseudo operation is used to break the binary deck into two or more parts. The M-term contains a name and identification in the same format as the M-term of the IDENT pseudo. The effect of a TCD is to punch a binary transfer card and a binary ident card with the identification specified in the M-term. The identification will continue through the remainder of the deck (or until the next TCD card is encountered). Any number of TCD pseudos can be used in one assembly.

6.26 TITLE PSEUDO

The TITLE pseudo operation is used to reduce the quantity of listing in a checked-out program. Some pseudo operations produce more than one word of binary information. Normally the assembly listing will contain every word. If TITLE is in effect, only the first half word of binary information will appear on the listing this has no effect on the binary output). In the case of the DUP pseudo only the first iteration will be present. If TITLE is in effect at the end of the program, the literal printout will be suppressed. In case of an error, the binary word will be printed, regardless. The TITLE card will not appear in the assembly listing.

6.27 DETAIL PSEUDO

The DETAIL pseudo is used to nullify the effect of the previous TITLE pseudo operation. If a TITLE is not in effect, the DETAIL instruction is ignored. The DETAIL card will not appear in the assembly listing.

6.28 FINIS PSEUDO

A card entry having a FINIS operation code is used to end a series of assemblies and to cause an exit from the assembly routine. It should be placed after the last END card.

7.0 PROGRAM LISTING

7.1 DESCRIPTION OF LISTING

1. The output listing consists of 120-character records containing, from left to right, certain error indicators, the octal machine language Program, the symbolic card (input program), and an error flag.

2. After the line containing the END entry, a one-line record gives the octal range of the program.

3. Following the range, DUPLICATE SYMBOLS will appear. All symbols used more than once in the LOC field will be listed under this heading.

4. Following the Duplicate Symbols, UNDEFINED SYMBOLS will appear. All symbols referenced in the M-term but not appearing in the LOCN field, reference symbol table, or CODES table, will be listed following this heading.

5. The Undefined Symbols listing is followed by a decimal count of the assembly errors in the program and a listing showing their type and location. If the program is error free, NO ERRORS will be printed.

6. The error count is followed by a decimal count of the sequence errors and a listing showing their locations. If the program has no sequence errors or no sequence numbers, no SEQUENCE ERRORS will be printed.

7. The sequence count is followed by a decimal count of the number of symbolic cards in the program.

8. The symbolic card count is followed by a decimal count of the number of binary cards in the program.

9. The binary card count is followed by a mode message which tells in what mode the program was assembled (LARII or SABER). The location in the program and the reason for the decision is included.

10. The mode message is followed by the symbol listing described in the following section.

8.0 SYMBOL LISTING

The symbol listing is divided into three sections: internal symbols, reference symbols, and CODES symbols. The format of each section is the same. That format is described below. (Comments on the EQU pseudo apply only to the internal symbol listing.)

The format of the symbol index is the location of every symbol, followed by the symbol itself (symbols are in alphabetical order), followed by the location of each cell which makes a reference to this symbol.

Each one of these references is followed by a comma and one letter, which gives some sort of indication of the operation which was performed on the symbolic cell at the reference location.

The letter "L" indicates that this cell was loaded or entered into one of the central registers.

The letter "D" indicates that this cell was destroyed, that is, changed in some fashion.

The letter "T" indicates that a program transfer was made to this location.

The letter "U" indicates that the cell was combined with a central register, e.g., added.

The letter "P" indicates that this cell was used as a machine count, e.g., shift instruction.

The letter "B" indicates that this cell contained a symbolic B-term reference.

The letter "Z" indicates that this cell was referenced by a ZRO, NOP, SVN, or appears in a pseudo operation.

All EQU's and references made in the M-term of an EQU are put in this index. A reference made by an EQU is followed by an asterisk and the letter "E". The reference location is the location counter at the time the EQU was encountered.

Each symbol defined by an EQU card is put in the index. The symbol's value and its location go to the left of the symbol. An asterisk appears between the value and the symbol. The value in the first reference location is the location counter followed by two asterisks, at the time the EQU was encountered. All cells which referenced the symbol follow in the usual manner.

9.0 ERROR CODES

These error codes appear on the listing to the left of the location digits. Six asterisks (******) are printed at the far right of each line that contains an error code.

"R" error -	This M-term expression is neither relocatable nor absolute and would not be handled correctly at load time. An example would be A+B, where A and B are relocatable elements. The M-term is cleared to absolute zero.
"C" error -	Incorrect use of a special character in an M-term. The M-term is cleared to absolute zero and examination of that card ceases.
"V" error -	Division by zero has occurred in an arithmetic expression. It is equivalent to division by one.
"N" error -	The symbolic B-term is undefined. It is defined as zero.
"G" error -	The B-term specified is greater than seven. It is calculated modulo eight.
"E" error -	Too many temporarily undefined EQU cards. All those following must remain undefined. The limit would be about 400.

"P" error	-	A pseudo instruction such as DUP or BSS has a relocatable M-term. It is defined as one. This error printout will also occur if a symbol is undefined when it is used in an ORG, DUP, or BSS, but it is defined later. The ORG will get a value of 10000B. The DUP and BSS will get the value one. The symbol will be defined correctly and used as it is defined, if it is referenced thereafter.
"T" error	-	The literal table is full. All references to literals which do not already exist in the table must go undefined. The limit on distinct literals is 1,000 octals.
"K" error	-	A reference has been made to a block which does not exist. The share area is searched (if one exists) and the symbol is undefined if it is not found there.
"B" error	-	A B-term exists in a pseudo operation which does not require one or does not appear in an index operation which does require one.
"D" error	-	The same symbol has appeared in two different location terms. All references will be to the location assigned to the first occurrence of the symbol.
"F" error	-	The symbol table is full. Any future symbols occuring in location terms will go undefined.
"I" error	-	The IDENT entry is not the first in the program or more than one IDENT entry occurs in the program.
"L" error	-	The absence of a location term in an EQU entry, the presence of a location term where none is required, or the use of a + or - forcing mark on data generating instructions will cause this error code. No location symbol assignment is made.
"M" error	-	The M-term format is illegal for this instruction.
"O" error	-	Illegal operation code.
"S" error	-	Sum-check error in reading symbol table during the processing of an RST entry.
"U" error	-	A symbol is referenced which does not appear in a location term. Undefined symbols are assigned to successive locations in COMMON.

10.0 BINARY OUTPUT CARDS

10.1 BINARY IDENT CARD

Each binary program deck will be preceded by a binary IDENT card with the following format:

Column 1	- row 12, 11, 0, 1, 2, 3 - 77 to indicate IDENT card. - row 7, 8, 9 - column binary card control information.
Column 3, 4	- 24 binary bit folded checksum of card.
Columns 9-12	- program name in BCD taken from symbolic IDENT card.
Columns 74-80	- program identification in 12-bit Hollerith taken from symbolic IDENT card (columns 74-80 are not included in the checksum).

10.2 ABSOLUTE BINARY CARD

Column 1	- row 11, 0, 1, 2, 3 - word count of binary card - row 4, 5, 6 - first 3 bits of address of this card (first of 5 digits of address) - row 7, 8, 9 - column binary control card information.
Column 2	- 12 bits of address (last 4 digits)
Column 3, 4	- 24 bit folded checksum of card
Columns 5-8	- first binary word to be loaded into specified address
Columns 9-12	- second binary word to be loaded
.	
.	
.	
Columns 69-72	- 17th binary word to be loaded
Columns 74-80	- program identification in 12-bit Hollerith (columns 74-80 are not included in the checksum).

10.3 RELOCATABLE BINARY CARD

Column 1	- row 12 - relocatable card designator. - row 11, 0, 1, 2, 3 - word count of binary card. - row 4, 5, 6 - first 3 bits of address for this card. (First of 5 digit of address). - row 7, 8, 9 - column binary card control information.

Column 2	- 12 bits of address (last four digits).
Column 3, 4	- 24 bit folded checksum of card.
Columns 5-8	- relocatable control information.
Columns 9-12	- first binary word to be loaded into specified address.
.	
.	
.	
Columns 69-72	- 16th binary word to be loaded
Columns 74-80	- program identification in 12-bit Hollerith (columns 74-80 are not included in the checksum)

10.4 BINARY TRANSFER CARD

The binary transfer card will be the last card in a binary program deck.

Column 1	- row 12, 0 if an absolute program, 1 if a relocatable program. - row 11, 0, 1, 2, 3 - 0 to indicate transfer card. - row 4, 5, 6 - first 3 bits of transfer address. (First of 5 digits.) - row 7, 8, 9 - column binary card control information.
Column 2	- 12 bits of transfer address (last four digits).
Column 3,4	- 24 bit folded checksum of card.
Column 10	- count of binary cards in this deck.
Column 11	- row 7, 8, 9 and column 12 - start address of an absolute program, length if a relocatable program.
Columns 74-80	- program identification in 12-bit Hollerith (columns 74-80 are not included in the checksum)

10.5 BINARY SYMBOL TABLE

The binary symbol table is punched when a WST pseudo operation is encountered by SABER. The format of this deck follows:

1. All cards are in absolute binary format.

2. The first card has one word containing the number of unused cells in the reference symbol table.

3. The remainder of the cards contains symbols in BCD or symbol equivalents in binary. All program symbols preceding the WST pseudo operation and all reference symbols are included in the deck.

4. The binary transfer card will not contain the number of binary cards or the beginning address/length information.

APPENDIX A

SUMMARY OF OPERATION CODES

1.0 INSTRUCTION MNEMONICS

DIGITS	MNEMONICS	DESCRIPTION
00	ZRO	Zero in Operation Code Position
01	ARS	A Right Shift
02	QRS	Q Right Shift
03	LRS	Long (AQ) Right Shift
04	ENQ	Enter Q
05	ALS	A Left Shift
06	QLS	Q Left Shift
07	LLS	Long (AQ) Left Shift
10	ENA	Enter A
11	INA	Increase A
12	LDA	Load A
13	LAC	Load A, Complement
14	ADD	Add
15	SUB	Subtract
16	LDQ	Load Q
17	LQC	Load Q, Complement
20	STA	Store A
21	STQ	Store Q
22	AJP	A Jump
23	QJP	Q Jump
24	MUI	Multiple Integer
25	DVI	Divide Integer
26	MUF	Multiply Fractional
27	DVF	Divide Fractional
30	FAD	Floating ADD

DIGITS	MNEMONICS	DESCRIPTION
31	FSB	Floating Subtract
32	FMU	Floating Multiply
33	FDV	Floating Divide
34	SCA	Scale A
35	SCQ	Scale AQ
36	SSK	Storage Skip
37	SSH	Storage Shift
40	SST	Selective Set
41	SCL	Selective Clear
42	SCM	Selective Complement
43	SSU	Selective Substitute
44	LDL	Load Logical
45	ADL	Add Logical
46	SBL	Subtract Logical
47	STL	Store Logical
50	ENI	Enter Index
51	INI	Increase Index
52	LIU	Load Index, Upper
53	LIL	Load Index, Lower
54	ISK	Index Skip
55	IJP	Index Jump
56	SIU	Store Index, Upper
57	SIL	Store Index, Lower
60	SAU	Substitute Address, Upper
61	SAL	Substitute Address, Lower
62	INT	Input Transfer
63	OUT	Output Transfer
64	EQS	Equality Search
65	THS	Threshold Search
66	MEQ	Masked Equality

DIGITS	MNEMONICS	DESCRIPTION
67	MTH	Masked Threshold
70	RAD	Replace Add
71	RSB	Replace Subtract
72	RAO	Replace Add One
73	RSO	Replace Subtract One
74	EXF	External Function
75	SLJ	Selective Jump
76	SLS	Selective Stop
77	SEV	Sevens in Operation Code Position

2.0 ALTERNATE MNEMONICS

2.1 A-JUMP AND A-RETURN JUMP*

AJP	Z	same as	AJP	0
AJP	N	same as	AJP	1
AJP	P	same as	AJP	2
AJP	M	same as	AJP	3
ARP	Z	same as	AJP	4
ARP	N	same as	AJP	5
ARP	P	same as	AJP	6
ARP	M	same as	AJP	7

2.2 Q-JUMP AND Q-RETURN JUMP*

QJP	Z	same as	QJP	0
QJP	N	same as	QJP	1
QJP	P	same as	QJP	2
QJP	M	same as	QJP	3
QRP	Z	same as	QJP	4
QRP	N	same as	QJP	5

* Z = Zero, N = Non-zero, P = Plus, M = Minus

QRP	P	same as	QJP	6
QRP	M	same as	QJP	7

2.3 RETURN JUMP

RTJ	0	same as	SLJ	4
RJT	1	same as	SLJ	5
RTJ	2	same as	SLJ	6
RTJ	3	same as	SLJ	7

2.4 STOP AND RETURN JUMP

SRJ	0	same as	SLS	4
SRJ	1	same as	SLS	5
SRJ	2	same as	SLS	6
SRJ	3	same as	SLS	7

2.5 NO OPERATION

NOP		same as	ENI	0

3.0 PSEUDO OPERATIONS

Code	Meaning
IDENT	Insert Program Identifier
RST	Read Symbol Table
WST	Write Symbol Table
EJECT	Eject Line Printer Page
REM	Enter Remarks Only
EQU	Equate or Define Location Symbol
ORG	Set Origin Address
DEC	Insert Decimal Constant
OCT	Insert Octal Constant
BCD	Insert BCD Characters

Code	Meaning
FLX	Insert Flexowriter Characters
TEL	Insert Teletype Characters
BSS	Block Starting Symbol, Reserve
BES	Block Ending Symbol, Reserve
END	End Assembly (Each Program)
FINIS	End Assembly (Series of Programs)
SPACE	Insert Spaces Between Lines of Print
SEL	Inserts Operation Code 74, Selecting External Equipment
SEN	Inserts Operation Code 74, Sensing External Equipment
ACT	Insert Operation Code 74, Activating External Equipment
CALL	Call Subroutine
ORGR	Set Origin Address, Relocatable
REF	Add Reference Symbol
REFC	Change Reference Symbol
REFD	Delete Reference Symbol
DUP	Duplicate the Following Cards
NOP	No Operation
VFD	Define Variable Field
CTU	Continue Variable Field
BLOCK	Begin Putting Symbols in this Block
FWZ	Full Word Zero, Define Two M-terms
TITLE	Suppress Extra Binary Words
DETAIL	Resume Listing of all Information
TCD	Punch Transfer and Ident Card

APPENDIX B

BCD, FLX, AND TEL CODES

CHARACTER	CARD CODE	BCD CODE	FLX CODE	TEL CODE	FLEXOWRITER		SOROBAN TYPEWRITER		TELETYPE-WRITER	
					UC	LC	UC	LC	FC	KC
A	12-1	61	30	30	A	a	A	a	-	A
B	12-2	62	23	23	B	b	B	b	?	B
C	12-3	63	16	16	C	c	C	c	:	C
D	12-4	64	22	22	D	d	D	d	$	D
E	12-5	65	20	20	E	e	E	e	3	E
F	12-6	66	26	26	F	f	F	f	!	F
G	12-7	67	13	13	G	g	G	g	&	G
H	12-8	70	05	05	H	h	H	h	#	H
I	12-9	71	14	14	I	i	I	i	8	I
J	11-1	41	32	32	J	j	J	j	'	J
K	11-2	42	36	36	K	k	K	k	(	K
L	11-3	43	11	11	L	l	L	l	)	L
M	11-4	44	07	07	M	m	M	m	.	M
N	11-5	45	06	06	N	n	N	n	,	N
O	11-6	46	03	03	O	o	O	o	9	O
P	11-7	47	15	15	P	p	P	p	0	P
Q	11-8	50	35	35	Q	q	Q	q	1	Q
R	11-9	51	12	12	R	r	R	r	4	R
S	0-2	22	24	24	S	s	S	s	bell	S
T	0-3	23	01	01	T	t	T	t	5	T
U	0-4	24	34	34	U	u	U	u	7	U
V	0-5	25	17	17	V	v	V	v	;	V
W	0-6	26	31	31	W	w	W	w	2	W
X	0-7	27	27	27	X	x	X	x	/	X
Y	0-8	30	25	25	Y	y	Y	y	6	Y
Z	0-9	31	21	21	Z	z	Z	z	"	Z

	CARD	BCD	FLX	TEL					TELETYPE-	
CHARACTER	**CODE**	**CODE**	**CODE**	**CODE**	**FLEXOWRITER**		**SOROBAN TYPEWRITER**		**WRITER**	
					UC	LC	UC	LC	FC	KC
0	0	12	56	15	0	0	)	0	0	P
1	1	01	74	35	1	1	*	1	1	Q
2	2	02	70	31	2	2	@	2	2	W
3	3	03	64	20	3	3	#	3	3	E
4	4	04	62	12	4	4	$	4	4	R
5	5	05	66	01	5	5	%	5	5	T
6	6	06	72	25	6	6	¢	6	6	Y
7	7	07	60	34	7	7	&	7	7	U
8	8	10	33	14	8	8	½	8	8	I
9	9	11	37	03	9	9	(	9	9	O
/	0-1	21	44	27	'	/	?	/	/	X
blank	blank	20	04	04	space	space	space	space	space	space
=	3-8	13	42	Note 1	=	.	.	.		
-	11	40	52	30	-	-	-	-	-	A
-	4-8	14	52	30	-	-	-	-	-	Á
+	12	60	46	Note 1	+	,	,	+		
.	12-3-8	73	42	07	=	.	.	.	.	M
)	12-4-8	74	54	11	(	)	"	'	)	L
$	11-3-8	53	62	22	4	4	$	4	$	D
*	11-4-8	54	74	Note 1	1	1	*	1		
,	0-3-8	33	46	06	+	,	,	+	,	N
(	0-4-8	34	54	36	(	)	"	'	(	K
$\bar{0}$	11-0	52	Note 3							
$\overset{+}{0}$	12-0	72	Note 2							

Notes:

1. The EQUALS (=), PLUS (+), and ASTERISK (*) are not available on the Teletypewriter; the assembler indicates an M error and inserts a space (04). See note 3 for special codes.

2. The $\overset{+}{o}$ is not translatable; the assembler indicates an M error and inserts a space (04).

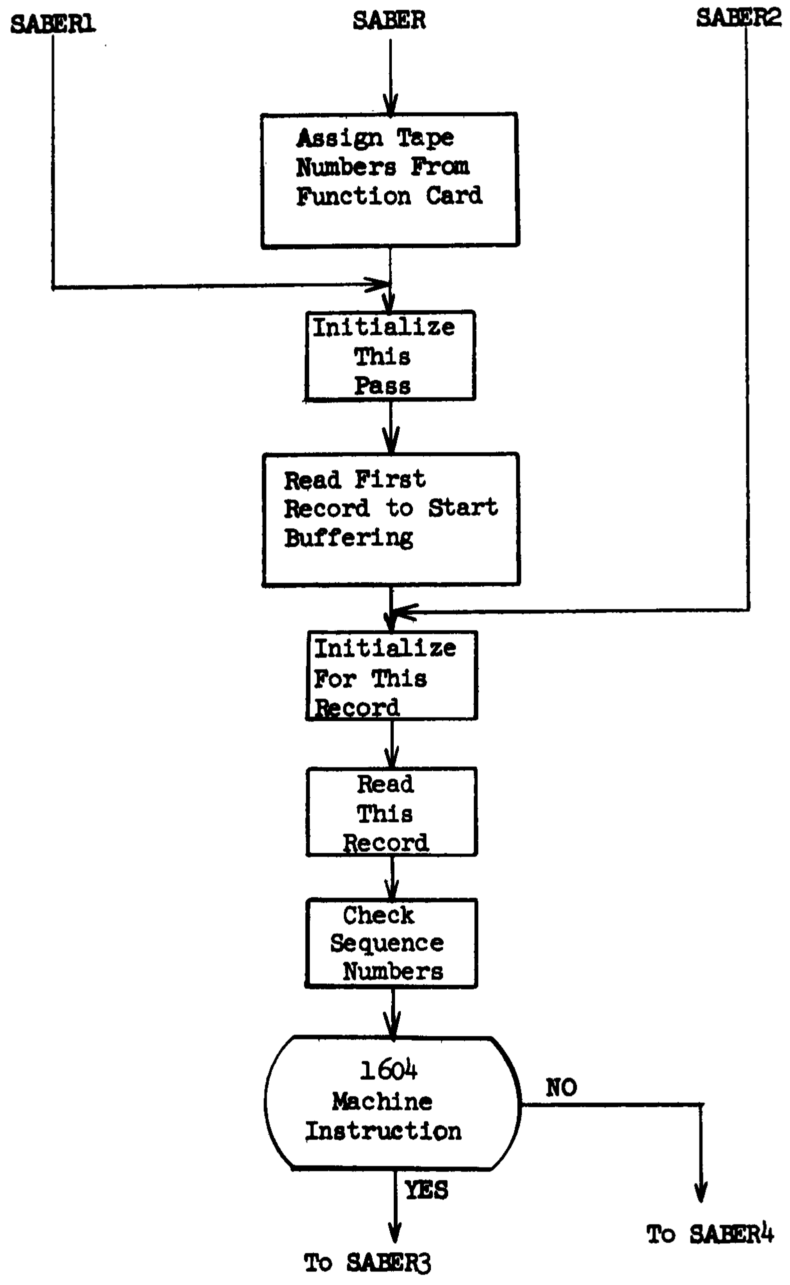
3. In a FLX or TEL entry or subfield, a $\bar{o}$ (zero with minus overpunched) changes the mode, and causes the succeeding codes to be translated in a special manner, until the next $\bar{o}$ code is encountered. These $\bar{o}$ brackets are not considered part of the word or character count. The special manner of translation is given below:

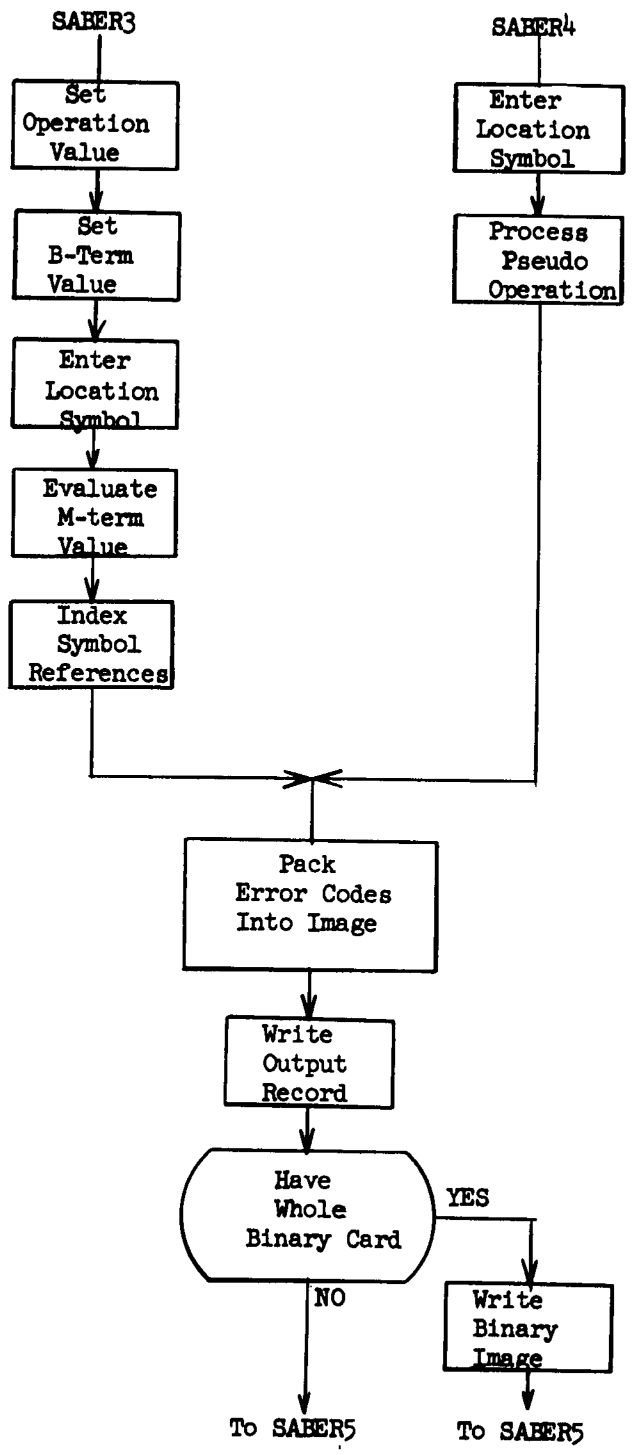
   a. For FLX:

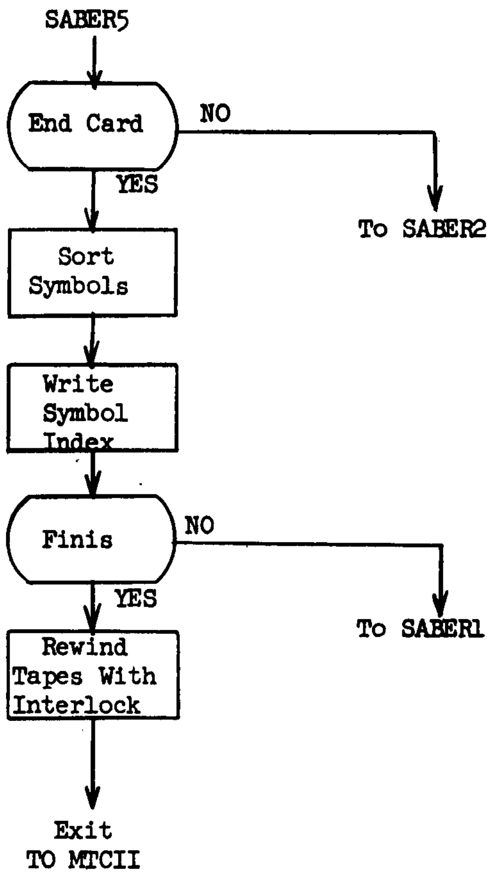
BCD	FLX Meaning	FLX Code
R (51)	Carriage Return	(45)
U (24)	Shift to Upper Case	(47)
L (43)	Shift to Lower Case	(57)
B (62)	Backspace	(61)
C (63)	Color Shift	(02)
T (23)	Tab	(51)
S (22)	Stop	(43)
D (64)	Delete	(77)
F (66)	Tape Feed	(00)

   b. For TEL:

BCD	TEL Meaning	TEL Code
R (51)	Carriage Return	(02)
U (24)	Shift to Figure	(33)
L (43)	Shift to Letters	(37)
F (66)	Line Feed	(10)
I (71)	Ignore	(00)


SABER1
SABER
SABER2
Assign Tape Numbers From Function Card
Initialize This Pass
Read First Record to Start Buffering
Initialize For This Record
Read This Record
Check Sequence Numbers
1604 Machine Instruction
NO
YES
To SABER3
To SABER4


SABER3
Set Operation Value
Set B-Term Value
Enter Location Symbol
Evaluate M-term Value
Index Symbol References
SABER4
Enter Location Symbol
Process Pseudo Operation
Pack Error Codes Into Image
Write Output Record
Have Whole Binary Card
YES
NO
Write Binary Image
To SABER5
To SABER5


SABER5
End Card
NO
YES
To SABER2
Sort Symbols
Write Symbol Index
Finis
NO
YES
To SABER1
Rewind Tapes With Interlock
Exit TO MTCII

DISTRIBUTION LIST

External

Space Systems Division
(Contracting Agency)
Major C. R. Bond (SSOCD)

6594th Aerospace Test Wing
(Contracting Agency)
Col. A. W. Dill (TWRD)
Lt. Col. M. S. McDowell (TWRU) (4)
TWACS (28)

PIR-E1 (Lockheed)
B. J. Jones
C. H. Finnie
H. F. Grover
H. R. Miller
G. D. Lawrence (5)
461 Program Office
698BK Program Office

PIR-E2 (Philco)
J. A. Bean
J. A. Isaacs
R. Morrison
S. M. Stanley

PIR-E3 (LFE)
D. F. Criley
K. B. Williams (5)

PIR-E8 (Mellonics)
F. Druding

PIR-E5 (Aerospace)
F. M. Adair
R. D. Brandsberg
L. H. Garcia
G. J. Hansen
C. S. Hoff
L. J. Kreisberg
T. R. Parkin
E. E. Retzlaff
H. M. Reynolds
D. Saadeh
V. White

PIR-E7 (STL)
A. J. Carlson (2)
R. L. Mills

PIR-E4 (Sunnyvale)
D. Alexander (2)

PIR-E4 (GE-Box 8555)
J. S. Brainard
H. G. Klose
J. D. Selby

PIR-E4 (GE-3198 Chestnut)
J. F. Butler
H. D. Gilman

PIR-E4 (GE- Bethesda)
W. L. Massey

PIR-E4 (GE-Box 8661)
J. D. Rogers

TOTSCHEK, R. A.	24120	WINSOR, M. E.	22084
VORHAUS, A. H.	24076A	WINTER, J. E.	24117
WAGNER, I. T.	24093	WISE, R. C.	22158
WARSHAWSKY, S. B.	24097	WONG, J. P.	SUNNYVALE
WEST, G. D.	SUNNYVALE	ZACHTE, S. A.	24086
WEST, G. P.	22116A	ZUBRIS, C. J.	24075
WILSON, G. D.	24124		

DISTRIBUTION LIST

Internal

AFCPL	(5)	14059	KEY, C. D.	22083
ALLFREE, D.		24083	KEYES, R. A.	24073
ALPERIN, N. I.		22088B	KINKEAD, R. L.	22155
ARMSTRONG, E.		24123	KNEEMEYER, J. A.	22153A
BERNARDS, R. M.		SUNNYVALE	KNIGHT, R. D.	22119
BIGGAR, D.		24118A	KOLBO, L. A.	22079
BILEK, R. W.		22101	KOSTINER, M.	14056B
BLACK, H.		14039	KRALIAN, R. P.	14039
BRENTON, L. R.		24103B	KRISTENSEN, K.	SUNNYVALE
BURKE, B. E.		24086	LACHAPELLE, F. J.	22156
BURKE, R. F.		22082	LAUGHLIN, J. L.	24073
BUSCH, R. E.		22153B	LAVINE, J.	24091
CHAMPAIGN, M. E.		22091A	LEWIS, H. L.	22095
CHIODINI, C. M.		24082B	LITTLE, J. L.	24090A
CIACCIA, B. G.		24090B	LONG, F.	22078
CLINE, B. J.		24127	MADRID, G. A.	23014
COGLEY, J. L.		22122	MAHON, G. A.	24089
CONGER, L.		24082	MARIONI, J. D.	24076B
COOLEY, P. R.		24081	MARSHALL, R. D.	22099
COURT, T. D.		24086B	MARTIN, W. P.	24127B
DANT, G. B.		24086B	MCKEOWN, J. C.	22083
DECUIR, L. E.		24053A	MICHAELSON, S. A.	14039
DERANGO, W. C.		24094A	MILANESE, J. J.	22078
DEXTER, G. W.		25026	MUNSON, J. B.	22096A
DISSE, R. J.		22082	MYERS, G. L.	14056A
DOBRUSKY, W. B.		22150	NELSON, P. A.	24075
ELLIS, R. C.		22131A	NG, J.	22148
EMIGH, G. A.		14039	NGOU, L.	24127
ERICKSEN, S. R.		22113	PADGETT, L. A.	24110A
FELKINS, J.		24097	PATIN, O. E.	SUNNYVALE
FOSTER, G. A.		14039	PERRY, G. H.	24034
FRANKS, M. A.		24122	POLK, T. W.	24113
FREY, C. R.		23110	PRUETT, B. R.	22157
FRIEDEN, H. J.		23013	REILLY, D. F.	24121
GARDNER, S. A.		22160	REMSTAD, C. L.	25030
GREENWALD, I. D.		22116B	RUSSELL, R. S.	14054
GRIFFITH, E. L.		22157	SCHOLZ, J. W.	14039
HAAKE, J. W.		22088A	SCOTT, R. J.	24110
HARRIS, E. D.		24081	SEACAT, C. M.	SUNNYVALE
HENLEY, D. E.		22094B	SHAPIRO, R. S.	24110B
HILL, C. L.		22161	SKELTON, R. H.	22087
HILLHOUSE, J.		23110	SOLOMON, J. D.	23007
HOLMES, M. A.		24103	SPEER, N. J.	24085
HOLZMAN, H. J.		24065B	STONE, E. S.	24058B
HOUGHTON, W. H.		24103B	SWEENEY, M. J.	25030
HOYT, R. L.		14039	TABER, W. E.	22155
IMEL, L. E.		14050	TENNANT, T. C.	27029
KASTAMA, P. T.		23007	TESTERMAN, W. D.	14039
KAYSER, F. M.		24109	THOMPSON, J. W.	24082A
KEDDY, J. R.		24105	THORNTON, R. L.	14050

UNCLASSIFIED

System Development Corporation,
Santa Monica, California
UTILITY SYSTEM PROGRAM SPECIFICATIONS
DESIGN SPECIFICATIONS FOR SYMBOLIC
ASSEMBLY ROUTINE (SABER) MILESTONE 4.
Scientific rept., TM-891/007/00, by
F. J. LaChapelle. 25 May 1963, 31p.
(Contract AF 19(628)-1648, Space
Systems Division Programs, for Space
Systems Division, AFSC)
Unclassified report

DESCRIPTORS: Programming (Computers).
Satellite Networks.

UNCLASSIFIED

UNCLASSIFIED

Reports that SABER (Symbolic Assembly
Routine) will be used to assemble a
special format symbolic card deck on
the CDC 1604 computer. Also reports
that the binary output of a SABER
assembly will be loaded by the MTCII
(Master Tape Control) program.

UNCLASSIFIED