SECTION C
SYSTEM 399 PROGRAMMING

OBJECTIVES:

Explain the steps involved in creating a user program.

Describe the programming worksheets and explain the use of each.

Explain each type of worksheet entry and show how each entry is made.

Describe the object level data format.

Describe the object level user instruction format.

Illustrate the bit string of each type of user instruction and relate each bit string to its worksheet entries.

Explain the interpretive mode of operation.

Explain the console keys used in conjunction with loading and executing a user program.

INTRODUCTION

There are two levels of System 399 programs, source and object. At the source level, the program consists of handwritten statements on three types of worksheets. At the object level, the program is in its final format, stored on a magnetic tape cassette and ready to be used with the customer's system.

The source level statements are written in mnemonics, a space-saving system of abbreviating English words; data statements and instruction statements are both written in this manner. The information on the worksheets is encoded on punched tape or punched cards, which are input to an assembler program for conversion into an object level program, as shown in figure C-1. The data on the worksheets is punched in a definite order: assembler specification worksheet first, data layout worksheets second, and coding worksheets last.

The assembler makes a one-for-one conversion from the source level to the object level. For each source statement written on the worksheets, one instruction or data definition is produced in object form. The bit string of an object instruction or data definition contains all the pertinent information written in the corresponding source statement.

Memory locations for the data fields and instructions are assigned by the assembler, and these addresses are inserted in the operand address portions of the instructions by the assembler.

SOURCE LEVEL

ASSEMBLER SPECIFICATION WORKSHEET

The entries on the Assembler Specification Worksheet constitute the assembler control statement, which must be the first statement in every System 399 source program.

This statement controls the assembly process for a particular program by providing the assembler with such information as the name of the program, the type of program listing, the type of output, and the object memory size of the system on which the program is to be run.

System 399 programs may be assembled by either an NCR Century assembler or a System 399 assembler, and the proper assembler specification worksheet must be used. A large logo is printed in the center of each worksheet to identify which assembler the worksheet is to be used with.

NCR CENTURY ASSEMBLER SPECIFICATION WORKSHEET (FIG. C-2)

If punched tape is used as input to the assembler, the format code must be punched before the page-line number.

1. Page-Line

The assembler control statement must be the first statement in a source program. Page-line number 000000, which is the lowest possible page-line number, is preprinted on the worksheet and must be entered to ensure proper placement of the assembler control statement when sorting is requested.

2. Program Name

The letter P is preprinted in position 7, and must be entered.

The program name, entered in positions 8 through 15, identifies the program on each page of the assembler listing.

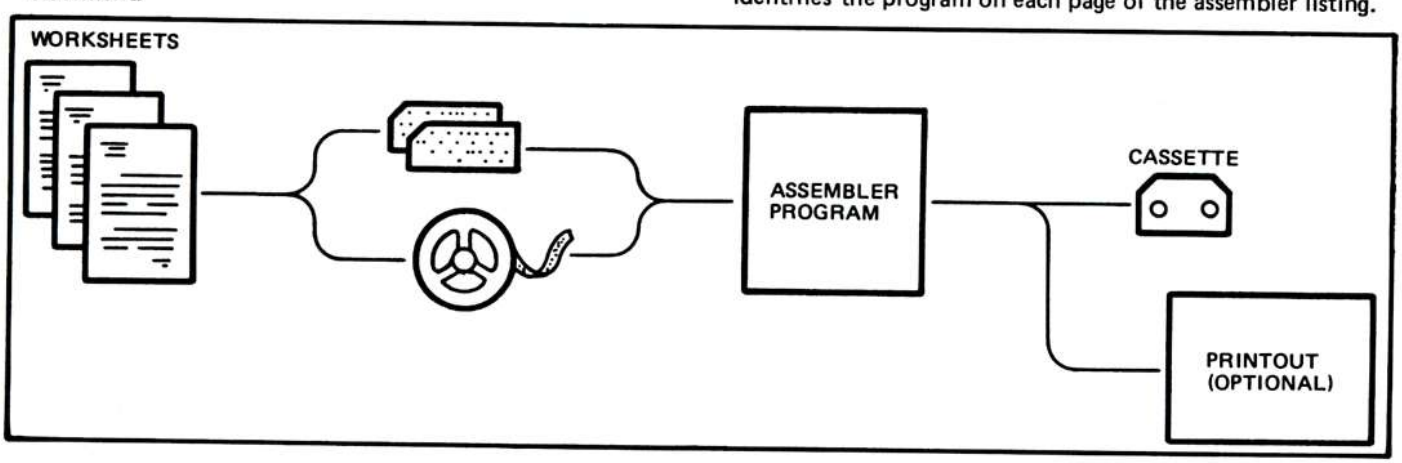

Fig. C-1 Program flow

399

399 ASSEMBLER SPECIFICATION WORKSHEET
FOR NCR CENTURY ASSEMBLER

NCR*

Program _____ *Prepared by* _____

_____ *Date* _____ *Page* _____ *of* _____

ALL SYMBOLIC REFERENCES MUST BE LEFT-JUSTIFIED AND MUST CONTAIN AT LEAST ONE ALPHABETIC CHARACTER.
ALL NUMERIC ENTRIES MUST BE RIGHT-JUSTIFIED AND MUST BE ZERO-FILLED TO THE LEFT.

(Shaded Boxes Are Optional) Punched Tape Format Code /99

1. Page-Line — 1 to 6: 000000
2. Program Name — 7 to 15: P
3. Language Name — 18 to 23: C399
4. Reassembly Name (Enter name of program to be reassembled; enter N in position 24 for initial assembly) — 24 to 33
5. Should source input be sorted? (Y-source lines will be sorted if out of sequence; N-source lines will be renumbered but not sorted; Y must be entered for reassembly) — 35
6. Should source statements be renumbered? (Enter 1 thru 0 for renumbering increments 10 thru 100; enter N for no renumbering; if position 35 contains N, statements will be renumbered) — 36
7. Should program listing be double spaced? (Y-double spacing; N-single spacing; N if blank) — 44
8. Should program listing include cross reference? (Y or N; N if blank) — 45
9. Indicate type of object output desired. — 50
   1 for Punched Cards
   2 for Punched Tape
   3 for Magnetic Tape
   N or blank for none
10. Indicate type of source output desired. — 51
    1 for Punched Cards
    2 for Punched Tape
    3 for Magnetic Tape
    N or blank for none
11. Object memory size (Enter 04, 06, 08, 10, 12, 14 or 16 to represent increments of 1024 bytes) — 54
12. (Reserved) — 74
13. Identification — 75 to 80

NOTE: THIS WORKSHEET IS TO BE USED WHEN ASSEMBLING NCR 399 PROGRAMS ON AN NCR CENTURY SYSTEM.

ST-61650 5-1-72 NCR

*TRADEMARK REG. U.S. PAT. OFF.

Fig. C-2 NCR Century assembler specification worksheet

The program name may consist of any standard characters, but must include at least one alphabetic character. A single letter N in position 8 has special significance during reassembly (see item 4, Reassembly Name) and cannot be used as a program name.

NOTE: During an initial assembly, version number 00 is automatically assigned to the program, except as noted under item 4.

During reassembly, if the program name is the same as the reassembly name, the assembler automatically increments the version number of the program being reassembled by one and assigns the new version number to the reassembled program.

If the version number of the program being reassembled is 99, enter a new (different) program name in positions 8 through 15. The reassembled program thus gets a new program name with a version number of 00. Renaming the program in this way is necessary because a current version number of 99 would be incremented to 00 in the reassembled program, and the highest version number would no longer indicate the newest version of the program.

Refer to the examples of program name entries and reassembly name entries in table C-1.

Positions 16 and 17 must be left blank.

3. Language Name

The language name, C399, is preprinted in positions 18 through 21 and must be entered.

4. Reassembly Name
(Enter name of program to be reassembled; enter N in position 24 for initial assembly)

If an initial assembly is being performed, enter the letter N in position 24 and leave positions 25 through 33 blank.

NOTE: During an initial assembly, if the programmer wishes to assign a version number other than 00 to the program, he must enter the letter N in position 24 and the desired version number, minus one, in positions 32 and 33. During assembly, the number in positions 32 and 33 is incremented by one and this version number is assigned to the new program.

If a reassembly is being performed, enter the name and version number of the program to be reassembled. The program name must begin in position 24 and the version number must be entered in positions 32 and 33. The acceptable range for the version number is 00 through 99.

Consider the following examples (table C-1) of possible program name and reassembly name entries and the resulting assembled program name and version number.

PROGRAM NAME (Entry 2)	REASSEMBLY NAME (Entry 4)	NEW PROGRAM NAME
ACCTRECD	N	ACCTRECD00
ACCTRECD	N⌀⌀⌀⌀⌀⌀⌀⌀49*	ACCTRECD50*
ACCTRECD	ACCTRECD02	ACCTRECD03**
RECDACCT	ACCTRECD99	RECDACCT00

Table C-1 Program names and reassembly names

* Initial assembly only (see preceding NOTE).
** If, prior to the reassembly, the program disc already contained versions 00, 01, 02, and 03, the reassembly results in a new generation of version 03. The previous generation of version 03 is then no longer accessible.

5. Should source input be sorted? (Y-source lines will be sorted if out of sequence; N-source lines will be renumbered but not sorted; Y must be entered for reassembly).

Enter Y if out-of-sequence source statements are to be sorted by page-line number. A reassembly always requires a Y in position 35.

Enter N if (1) this is an initial assembly, (2) a sort is not desired, and (3) source statements are to be renumbered in their order of presentation (see item 6).

If an invalid character is entered in item 5, the assembler assumes N.

6. Should source statements be renumbered? (Enter 1 thru 0 for renumbering increments 10 thru 100; enter N for no renumbering; if position 35 contains N, statements will be renumbered).

Source lines will be renumbered to reflect their order of presentation if N is entered in position 35. The entry in position 36 specifies the increment to be used for renumbering. Entering 1, 2, 3, . . . 9, or 0 specifies renumbering increments of 10, 20, 30, . . . 90, or 100 respectively.

Enter N if renumbering is not desired and item 5 contains a Y.

If an invalid character is entered in item 6, the assembler assumes 1.

7. Should program listing be double-spaced? (Y-double spacing; N-single spacing; N if blank).

Enter Y to indicate that the assembler output listing should be double-spaced.

Enter N to indicate that the assembler output listing should be single-spaced.

If left blank, or an invalid character is entered, N is assumed.

8. Should program listing include cross reference? (Y or N; N if blank)

The cross-reference listing is an alphabetical listing of all references named in the program. Listed with each reference are the page-line numbers of all source statements containing that reference as an operand.

Enter Y in position 45 to obtain the cross-reference listing.

Enter N in position 45 if the cross-reference listing is not desired.

If left blank, or an invalid character is entered, N is assumed.

9. Indicate type of object output desired.

1 for Punched Cards
2 for Punched Tape
3 for Magnetic Tape
N or blank for none

This entry is self-explanatory.

If an invalid character is entered, the assembler assumes N.

10. Indicate type of source output desired.

1 for Punched Cards
2 for Punched Tape
3 for Magnetic Tape
N or blank for none

This entry is self-explanatory.

If an invalid character is entered, the assembler assumes N.

11. Object memory size (Enter 04, 06, 08, 10, 12, 14, or 16 to represent increments of 1024 bytes).

Enter 04, 06, 08, 10, 12, 14,or 16 in positions 56 and 57 to specify the memory size (in increments of 1024 bytes) of the NCR 399 System on which the assembled program is to be run.

If a value other than those specified for item 11 is entered, the assembler assumes 08 and indicates an error.

12. (Reserved)

Position 74 is not used and must be left blank.

13. Identification

Columns 75 through 80 are optional. They can be used when the source statements are punched on cards to identify each card as belonging to a given program. From one to six characters may be used in making up the identification label.

SYSTEM 399 ASSEMBLER SPECIFICATION WORKSHEET (FIG. C-3)

If punched tape is used as input to the assembler, the format code must be punched before the page-line number.

1. Page-Line

The assembler control statement must be the first statement in a source program. Page-line number 000000, which is the lowest possible page-line number, is preprinted on the worksheet and must be entered to ensure proper placement of the assembler control statement.

2. Program Name

The letter P is preprinted in position 7, and must be entered.

The program name, entered in positions 8 through 15, identifies the program on each page of the assembler listing. The program name may consist of any standard characters. Spaces are allowed only to the right of the name; a space must not be embedded in the name. A single letter N in position 8 has special significance during reassembly (see item 4, Reassembly Name) and cannot be used as a program name.

NOTE: During an initial assembly, version number 00 is automatically assigned to the program, except as noted under item 4.

During reassembly, if the program name is the same as the reassembly name, the assembler automatically increments the version number of the program being reassembled by one and assigns the new version number to the reassembled program.

If the version number of the program being reassembled is 99, enter a new (different) program name in positions 8 through 15. The reassembled program thus gets a new program name with a version number of 00. Renaming the program in this way is necessary because a current version number of 99 would be incremented to 00 in the reassembled program, and the highest version number would no longer indicate the newest version of the program.

Refer to the examples of program name entries and reassembly name entries in table C-2.

3. Language Name

The language name, C399, is preprinted in positions 18 through 21 and must be entered.

4. Reassembly Name (Enter name of program to be reassembled; enter N in position 24 for initial assembly).

If an initial assembly is being performed, enter the letter N in position 24 and leave positions 25 through 33 blank.

NOTE: During an initial assembly, if the programmer wishes to assign a version number other than 00 to the program, he must enter the letter N in position 24 and the desired version number, minus one, in positions 32 and 33. During assembly, the number in positions 32 and 33 is incremented by one and this version number is assigned to the new program.

If a reassembly is being performed, enter the name and

399

399 ASSEMBLER SPECIFICATION WORKSHEET
FOR NCR 399 ASSEMBLER

NCR*

*Program*_____ *Prepared by*_____

_____ *Date*_____ *Page*_____ *of*_____

ALL SYMBOLIC REFERENCES MUST BE LEFT-JUSTIFIED AND MUST CONTAIN AT LEAST ONE ALPHABETIC CHARACTER.
ALL NUMERIC ENTRIES MUST BE RIGHT-JUSTIFIED AND MUST BE ZERO-FILLED TO THE LEFT.

(Shaded Boxes Are Optional) Punched Tape Format Code /,9,9

1. Page-Line — 1 … 6: 0,0,0,0,0,0
2. Program Name — 7 … 15: P
3. Language Name — 18 … 23: C,3,9,9,_,_
4. Reassembly Name (Enter name of program to be reassembled; enter N in position 24 for initial assembly) — 24 … 33
5. Should source input be sequence checked? (Y or N) — 35
6. Should source statements be renumbered? (Enter 0 for increments of 100; enter N for no renumbering) — 36
7. Error statement, re-entry option for keyboard input (Y or N) — 38
8. Should program listing be double spaced? (Y-double spacing; N-single spacing; leave blank if no line printer on 399 System) — 44
9. Should program listing include cross reference? (Y or N; N if blank) — 45
10. Assembly memory size (Enter 04, 06, 08, 10, 12, 14 or 16 to represent increments of 1024 bytes) — 53
11. Object memory size (Enter 04, 06, 08, 10, 12, 14 or 16 to represent increments of 1024 bytes) — 56
12. (Reserved) — 74
13. Identification — 75 … 80

NOTE: THIS WORKSHEET IS TO BE USED WHEN ASSEMBLING NCR 399 PROGRAMS ON AN NCR 399 SYSTEM.

ST-61650 5-1-72 NCR

*TRADEMARK REG. U.S. PAT. OFF.

Fig. C-3 System 399 assembler specification worksheet

version number of the program to be reassembled. The program name must begin in position 24 and the version number must be entered in positions 32 and 33. The acceptable range for the version number is 00 through 99.

Consider the following examples (table C-2) of possible name and reassembly name entries and the resulting assembled program name and version number.

PROGRAM NAME (Entry 2)	REASSEMBLY NAME (Entry 4)	NEW PROGRAM NAME
ACCTRECD	N	ACCTRECD00
ACCTRECD	N⌀⌀⌀⌀⌀⌀⌀⌀49*	ACCTRECD50*
ACCTRECD	ACCTRECD02	ACCTRECD03**
RECDACCT	ACCTRECD99	RECDACCT00

Table C-2 Program names and reassembly names

* Initial assembly only (see preceding NOTE).
** If, prior to the reassembly, the program disc already contained versions 00, 01, 02, and 03, the reassembly results in a new generation of version 03. The previous generation of version 03 is then no longer accessible.

5. Should source input be sequence checked? (Y or N)

Enter the letter Y in position 35 to specify sequence checking. The letter P will be printed in the assembler listing above every statement whose page-line number is equal to or less than the page-line number of the preceding statement.

Enter the letter N in position 35 if no sequence checking is desired.

6. Should source statements be renumbered? (Enter 0 for increments of 100; enter N for no renumbering)

Source statements will be renumbered in increments of 100 to reflect their order of presentation if 0 is entered in position 36.

Enter the letter N in position 36 if no renumbering is desired.

7. Error statement, re-entry option for keyboard input (Y or N)

Specify Y in position 38 when program source lines are to be entered through the NCR 399 keyboard and the re-entry option for keyboard input is desired. This option permits the operator to re-enter the same source lines that were incorrectly entered through the keyboard and found to contain errors by the assembler.

Enter the letter N in position 38 if the keyboard re-entry option is not desired or if source input is through media other than the keyboard.

8. Should program listing be double spaced? (Y-double spacing; N-single spacing; leave blank if no line printer on 399 System)

Complete item 8 only if the NCR 399 system has a line printer; otherwise leave blank.

Enter the letter Y in position 44 if a double-spaced listing is desired on the line printer.

Enter the letter N in position 44 if a single-spaced listing is desired on the line printer.

9. Should program listing include cross reference?

Option deleted – leave blank.

10. Assembly memory size (Enter 04, 06, 08, 10, 12, 14, or 16 to represent increments of 1024 bytes).

Enter 04, 06, 08, 10, 12, 14,or 16 in positions 53 and 54 to specify the memory size (in increments of 1024 bytes) of the NCR 399 System on which the source program is to be assembled.

11. Object memory size (Enter 04, 06, 08, 10, 12, 14, or 16 to represent increments of 1024 bytes).

Enter 04, 06, 08, 10, 12, 14,or 16 in positions 56 and 57 to specify the memory size (in increments of 1024 bytes) of the NCR 399 System on which the assembled program is to be run.

12. (Reserved)

Position 74 is not used and must be left blank.

13. Identification

Columns 75 through 80 are optional. They can be used when the source statements are punched on cards to identify each card as belonging to a given program. From one to six characters may be used in making up the identification label.

DATA LAYOUT WORKSHEET

Each line of the data layout worksheet (fig. C-4) contains all the information necessary to form a data definition in object code. In addition, each line contains a page and line number for sequencing and reference information for the programmer's convenience.

Columns 1-6 of each source line contain a unique page and line number. This number indicates the sequence of the source statements to the assembler and to anyone working with the source program.

These numbers must be in ascending order, and they must be higher than the assembler control statement number (000000) and lower than the first instruction statement number (coding worksheet). It is advisable to use an increment of 10 or more (30 by convention) between source line numbers to allow the addition of source lines within the program without renumbering all subsequent lines.

399

DATA LAYOUT WORKSHEET

NCR

Program _____ Prepared by _____

/,4,8 = PAPER TAPE FORMAT CODE _____ Date _____ Page ____ of ____

X=TABS	PAGE	LINE		DESCRIPTION	REFERENCE	TYPE	LENGTH	DP	PRINT COL.	SLEW	SIGN	VALUE ALPHANUMERIC NUMERIC		IDENT.
	1 2 3	4 5 6	7	8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31	32 33 34 35 36 37	38	39 40 41	42 43	44 45 46	47	48	49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64	65 66 67 68 69 70 71 72 73 74	75 76 77 78 79 80
			D											
			D											
			D											
			D											
			D											
			D											
			D											
			D											
			D											
			D											
			D											
			D											
			D											
			D											
			D											
			D											
			D											
			D											
			D											
			D											
			D											
			D											
			D											
	1 2 3	4 5 6	7	8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31	32 33 34 35 36 37	38	39 40 41	42 43	44 45 46	47	48	49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64	65 66 67 68 69 70 71 72 73 74	75 76 77 78 79 80

ST-61618 5-1-71 NCR *TRADEMARK PRINTED IN U. S. A.

Fig. C-4 Data layout worksheet

Column 7 contains a preprinted D on each line to indicate to the assembler that the source statement on that line is a data definition.

Columns 8-31 are used for descriptive comments. These comments generate no object code, but they are printed on the program listing to provide an aid in debugging and documenting the program. They may contain any character in the 399 code chart, including space (☑).

Comments should describe the data field accurately enough to permit another programmer to understand the program data without assistance from the original programmer.

An asterisk (*) in column 8 indicates that the entire line is a comment. This may be done on several consecutive lines to allow a lengthy comment. These comment lines are printed on the program listing, but they have no effect on the object program.

Asterisks in both columns 8 and 9 indicate that the entire line is a comment, and also cause the comment to be printed at the top of the next page of the listing. In this way, the comment can be used as a heading.

Columns 32-37, headed REFERENCE, contain the mnemonic data reference name of the data field. This reference name is used by the assembler to identify the field when it is used as an operand in an instruction.

With a few exceptions, each field defined must have a data reference which describes the data associated with it.

Data reference names must be unique, left-justified, and must range from one to six characters in length. The first character must be alpha (A-Z), with the remaining characters alphanumeric (A-Z, 0-9). Other characters and embedded spaces are not permitted.

DATA DESCRIPTIONS

Column 38, headed TYPE (table C-3), identifies the type of data the source line generates. Six types of data are possible, and each source line must contain the symbol of one of them.

TYPE	MEANING
B	Buffer allocation statement
N	Numeric data
X or blank	Alphanumeric data
T	Table definition
F	Format for redefinition
–	Continuation (associated with X type)

Table C-3 Types of data definitions

B Type

The letter B in column 38 specifies a buffer for online communications. When the program includes this type of source statement, it must be the first statement on the data layout worksheet.

One of two types of communications buffers can be used, depending on the communications system used. Polled Device (fig. C-5) or Point-to-Point (fig. C-6) communications are available.

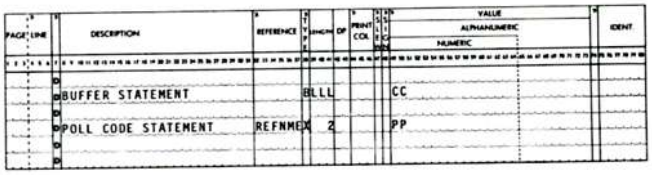

Fig. C-5 Polled device statement

B – Buffer allocation statement, must be first data statement, when used.

LLL
- Number of 8-bit characters in buffer (01-999)
- Right justified
- Spaces or invalid assumes 100

CC
- Two-character driver code
- 01 = 270 ASCII
- 02 = TC-500 Compatible Central
- 03 through 32 = not used
- Left justified

REFNME
- Valid symbolic reference name
- Left justified
- First character alpha, remaining characters alphanumeric
- No special characters
- No embedded spaces
- Non-duplicate

X
- Alphanumeric poll code statement
- Length of 2 (column 41)

PP
- Poll code
- Value may be any two alpha characters greater than hexadecimal 20.

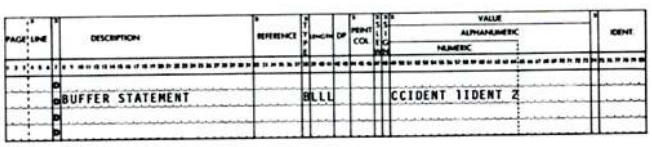

Fig. C-6 Point-to-point statement

B – Buffer allocation statement, must be first data statement, when used

LLL
- Number of 8-bit characters in buffer (01-999)
- Right justified
- Spaces or invalid assumes 100

CC
- Two-character driver code
- 33 = 399 to 399
- 34 = 2780 (399-615/360 Compatible O.C.D.-EBCD)
- 35 - 64 = not used

IDENT 1 — Central address
— Seven alpha characters, greater than hex 1F

IDENT 2 — Terminal address
— Seven alpha characters, greater than hex 1F

N Type

The letter N in column 38 specifies a data description of a field of numeric data (fig. C-7). One numeric character is represented by one digit in 605 memory. The maximum length of a numeric field is 16 characters.

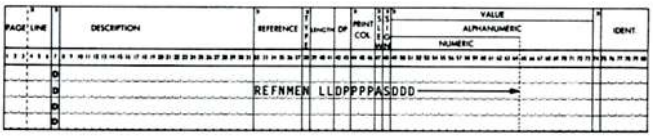

Fig. C-7 Numeric data statement

REFNME
— Valid symbolic reference name
— Left justified
— First character alpha, remaining characters alphanumeric
— No special characters
— No embedded spaces
— Non-duplicate

N
— Numeric data field, 4-bit characters
— Space or invalid indicates alphanumeric

LL
— Data length (number of characters in field) 1-16
— Right justified

DP
— Number of characters (0-15) to right of decimal point
— Right justified
— Must be equal to or less than data length
— Space or invalid indicates 0

PPP
— Print column for serial printer
— 1-255 (12 characters per inch)
— 1-221 (10 characters per inch)
— Right justified
— Space indicates no print head movement
— Invalid indicates column 1

A
— Paper advance after printing field
— Line printer, 1-3 lines
— Serial printer
L — Advance left platen one line
R — Advance right platen one line
B — Advance both platens one line
— Space or invalid indicates no paper advance

S
— Sign of data field
— Plus (+), space, or invalid indicates positive value
— Minus (-) indicates negative value

DDD
— Numeric data in field
— Columns 49-64
— Left justified
— Leading zeros must be entered
— Blanks generate zeros

X Type

The letter X or a blank in column 38 specifies a data description of a field of alphanumeric data (fig. C-8). One alphanumeric character is represented by two digits (ASCII) in 605 memory. The maximum length of an alphanumeric field is 256 8-bit characters.

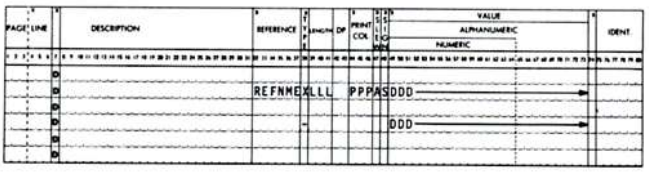

Fig. C-8 Alphanumeric data statement

REFNME
— Valid symbolic reference name
— Left justified
— First character alpha, remaining characters alphanumeric
— No special characters
— No embedded spaces
— Non-duplicate

X
— Alphanumeric data field, 8-bit characters
— Space or invalid indicates alphanumeric

LLL
— Data length (number of characters in field) 1-256
— Right justified

DP
— Not used for alphanumeric field

PPP
— Print column for serial printer
— 1-255 (12 characters per inch)
— 1-221 (10 characters per inch)
— Right justified
— Space indicates no print head movement
— Invalid indicates column 1

A
— Paper advance after printing field
— Line printer, 1-3 lines
— Serial printer
L — Advance left platen one line
R — Advance right platen one line
B — Advance both platens one line
— Space or invalid indicates no paper advance

S
— Sign of data field
— Plus (+), space, or invalid indicates positive value
— Minus (-), indicates negative value

DDD
— Alphanumeric data in field
— Columns 49-73
— Left justified
— Leading spaces must be entered
— Only 25 characters can be entered on this line
— Additional characters, up to 231, are entered in the value columns of data lines immediately following (continuation lines).
— A hyphen (-) in the TYPE column identifies each continuation line.
— All 25 characters in a continuation line are assembled into the field, unless the field length

is satisfied short of the end of the value field.
– If the number of continuation lines is not sufficient to fill the field, the remainder of the field is space filled.
– If no value is entered for the data field, no continuation lines are necessary to generate the specified number of spaces, even though the field may be longer than 25 characters.

T Type

The letter T in column 38 indicates that the source line contains the definition of a table of data (fig. C-9). The line defines the name, width, and length of the table.

A table is a structure consisting of a series of fields which can be repeated a given number of times. Data can be conveniently organized in a table for uses such as accumulation of data and the storage of constants. The tables used in the 399 take two forms, single-line and multiple-line.

The single-line table is made up of a number of contiguous data fields identified by a table reference name. The table is accessed by the table instructions by using the beginning address of the table, plus the number of the field within the table. The single-line table is limited in number of fields only by memory available. However, an entry of 1-999 must be made in the length column to establish the table. Each field definition can include a reference name, decimal point position, print column, slew code, sign, and value. The reference name makes it possible to access the field with instructions other than the table instructions. The only difference between the single-line table fields and normal data fields is the table definition preceding them, making them accessible to the table instructions.

A multiple-line table is essentially a single-line table with the line repeated a specific number of times. No values may be used in the data fields defined, and the decimal point position, print column, etc., defined in the data field definitions for the first line,are assigned to the subsequent lines. The number of contiguous fields on the first line is limited to 63, and the multiple-line table can contain a maximum of 256 lines. As in the single-line table, the data fields in the first line of the multiple-line table may be accessed by instructions other than the table instructions, if a reference name is entered.

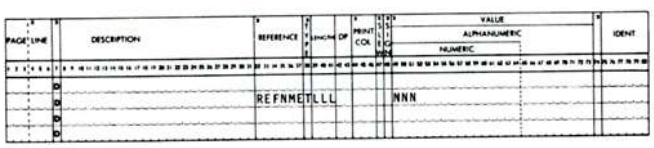

Fig. C-9 Data table statement

REFNME – Valid symbolic reference name
– Left justified
– First character alpha, remaining characters alphanumeric
– No special characters
– No embedded spaces
– Non-duplicate

T – Data table definition
– Space or invalid indicates alphanumeric data field

LLL – The number of data definitions to be used to make up the width of the table, 1-999 (any entry; size limited only by available memory) for single and 1-63 for multiple. These data definitions immediately follow the table definition line.
– Right justified
– Space or invalid indicates no data definitions

NNN – The length (number of lines) of the table (1 for single and 2-256 for multiple). The width specified by LLL is used for each of the lines.
– Left justified
– Space or invalid entry indicates 1

The entries in figure C-10, for example, generate a multiple-line data table the size of the one shown in figure C-11.

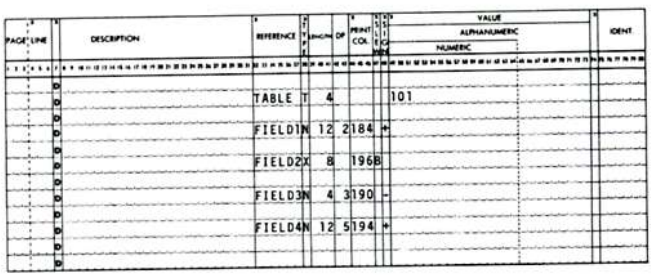

Fig. C-10 Typical table entries

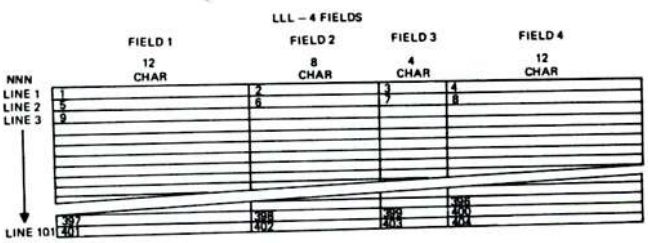

Fig. C-11 Conceptual layout of data table

The table shown in figure C-11 contains 404 fields. Any one of these fields may be accessed by a table instruction.

F Type

The letter F in column 38 indicates that the source line contains a format for redefinition (fig. C-12). The redefinition format source line is immediately followed by a number of X and/or N type data definition lines, which contain no value; only descriptive data is permitted, including reference names, if desired.

This data is used to redefine an area of memory when used with a Redefine instruction, described in the Coding Worksheet portion of this section.

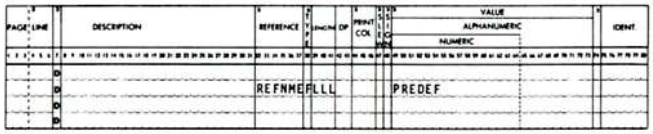

Fig. C-12 Format for redefinition statement

REFNME – Valid symbolic reference name
- Left justified
- First character alpha, remaining characters alphanumeric
- No special characters
- No embedded spaces
- Non-duplicate

F – Format for redefinition
- Space or invalid assumes alphanumeric

LLL – Number of fields to be defined on subsequent source lines (1-256)

PREDEF – Valid symbolic name of predefined data area to be redefined
- Left justified
- If spaces or undefined reference, zero address is assigned by assembler.

CODING WORKSHEET

The 399 instructions are entered on the coding worksheet (fig. C-13) in much the same way the data statements are entered on the data layout worksheet. Each source line contains the information necessary for the assembler to produce an object instruction.

Columns 1-6 (page and line) are used for sequencing, just as they are on the data layout worksheet. The first page and line number must be higher than the last number on the last data worksheet. The numbers must be in ascending order.

Column 7 contains a preprinted C on each line to indicate to the assembler that the source statement on that line is an instruction.

Columns 8-31 are used for descriptive comments. These comments generate no object code, but they are printed on the program listing to provide an aid in debugging and documenting the program.

Columns 32-37, headed REFERENCE, contain the unique symbolic reference name of the instruction entered on that line. A reference name is entered when the instruction is to be used as an operand in another instruction, such as a branch. Otherwise, the reference is not necessary.

Reference names must be unique, left-justified, and must range from one to six characters in length. The first character must be alpha, with the remaining characters alphanumeric (A-Z, 0-9). Special characters and embedded spaces are not permitted.

The mnemonic name of the instruction is entered in columns 38-43, headed OPERATION.

The A OPERAND, columns 44-49, may contain the symbolic reference name of a data definition predefined on the data layout worksheet, or it may contain indicator codes or device codes. The type of entry is determined by the instruction associated with it.

The B OPERAND, when used, always contains the symbolic reference name of a data definition predefined on the data layout worksheet.

The C OPERAND may contain the symbolic reference name of either a data definition, predefined on the data layout worksheet, or an instruction defined elsewhere on the coding worksheet.

Columns 62-67, headed ACTION, contain codes that direct various functions of the processor or peripheral devices. These codes modify the instruction.

N-FIELDS entries, columns 68-73, usually indicate a specific number of consecutive fields to be handled in the execution of the instruction. Certain instructions use this entry differently, as shown in the instruction descriptions.

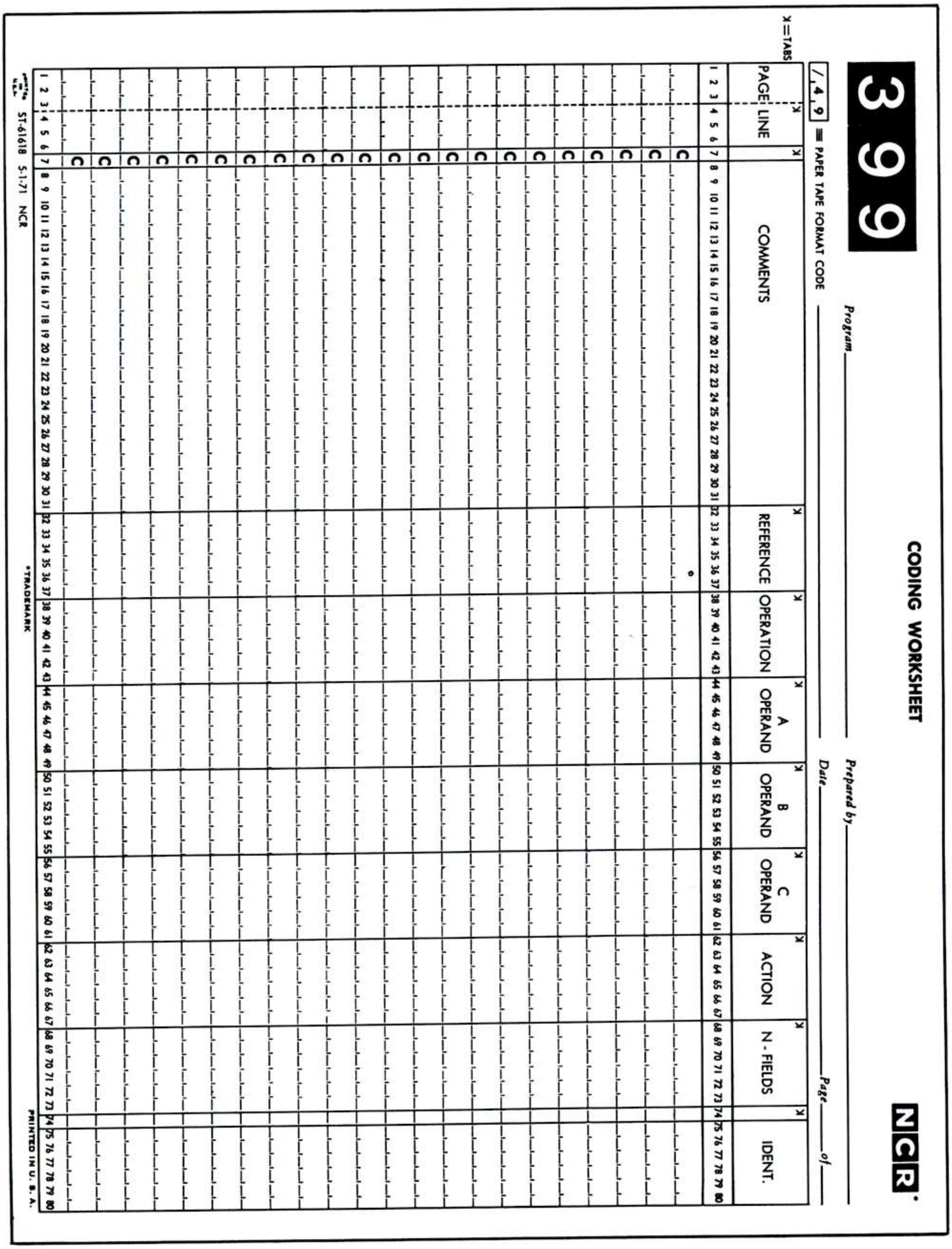

Fig. C-13 Coding worksheet

INSTRUCTION DESCRIPTIONS

ADD INSTRUCTIONS (FIG. C-14)

- ADD B + A → C

The data field specified by the A operand is added to the field specified by the B operand. The sum is put away in the field designated by the C operand. The contents of the A and B operand fields are not altered.

- ADDN $B_N + A_N \rightarrow B_N$

A specific number of fields (N-FIELDS), starting with the data field designated by the A operand, are added to the same number of fields, starting with the data field specified by the B operand. The sums are put away in the B operand fields. The contents of the A operand fields are not altered.

- ADDNA $B + A_N \rightarrow B$

A specific number of fields (N-FIELDS), starting with the data field designated by the A operand, are added to the field specified by the B operand. The sum is put away in the field designated by the B operand. The contents of the A operand fields are not altered.

- ADDNB $B_N + A \rightarrow B_N$

The field designated by the A operand is added to each field of a specific number of fields (N-FIELDS), starting with the field designated by the B operand. Each field of the B operand has put away in it the sum of its original contents plus the A operand contents. The contents of the A operand field are not altered.

Restrictions

No rounding can be specified.

If the result is shorter than the destination field, unused positions to the left and right of the decimal point are filled with 0's.

If the result is longer than the destination field, the result is truncated at either or both ends to fit the field.

No overflow indication is used.

PAGE	LINE	COMMENTS	REFERENCE	OPERATION	A OPERAND	B OPERAND	C OPERAND	ACTION	N FIELDS	IDENT
			REFNME	ADD	NDATA	NDATA	NDATA			
			REFNME	ADDN	NDATA	NDATA			NN	
			REFNME	ADDNA	NDATA	NDATA			NN	
			REFNME	ADDNB	NDATA	NDATA			NN	

Fig. C-14 Add statements

REFNME
- Valid symbolic reference name
- Left justified
- First character alpha, remaining characters alphanumeric
- No special characters
- No embedded spaces
- Non-duplicate

NDATA
- Symbolic reference name of previously defined N type data definition
- Left justified
- Space or undefined assigns address 00

NN
- Number of fields to be affected (1-31)
- Right justified
- Blank or invalid assumes 1

BR (UNCONDITIONAL) (FIG. C-15)

The program branches to the instruction named in operand C.

PAGE	LINE	COMMENTS	REFERENCE	OPERATION	A OPERAND	B OPERAND	C OPERAND	ACTION	N FIELDS	IDENT
			REFNME	BR			NXTINS			

Fig. C-15 Branch unconditional statement

REFNME
- Valid symbolic reference name
- Left justified
- First character alpha, remaining characters alphanumeric
- No special characters
- Non duplicate

NXTINS
- Reference name of instruction to be branched to
- Undefined or blank assigns zero address

BRE/BRU/BRL/BRG (CONDITIONAL) (FIG. C-16)

The conditional branch instructions compare the data field specified by the A operand with the data field specified by the B operand. The A operand field is the field being tested, and the B operand is the test value. BRE tests A for being equal to B, BRU tests A for being unequal to B, BRL tests A for being less than B, and BRG tests A for being greater than B. Operand C designates the instruction to be branched to when the condition tested for is satisfied.

PAGE	LINE	COMMENTS	REFERENCE	OPERATION	A OPERAND	B OPERAND	C OPERAND	ACTION	N FIELDS	IDENT
			REFNME	BRE	XNDATA	XNDATAN	XTINS			
			REFNME	BRU	XNDATA	XNDATAN	XTINS			
			REFNME	BRL	XNDATA	XNDATAN	XTINS			
			REFNME	BRG	XNDATAN	XDATAN	XTINS			

Fig. C-16 Branch conditional statements

REFNME
- Valid symbolic reference name
- Left justified
- First character alpha, remaining characters alphanumeric
- No special characters
- No embedded spaces
- Non-duplicate

XNDATA – Symbolic reference name of previously defined X or N type data definition
– Left justified
– Operand A names field to be tested
– Operand B names test value field
– Blank or invalid assigns address 00

NXTINS – Symbolic reference name of instruction to be branched to
– Left justified
– Blank or invalid assigns address 00

BRION/BRIOFF/BRS

The BRION or BRIOFF instruction (fig. C-17) tests the indicator specified in operand A for being ON or OFF, respectively. If the indicator is in the condition tested for, the program branches to the instruction specified in the C operand. If the indicator is not in that condition, the program continues its normal flow.

The BRS instruction (fig. C-17) tests the status of the line printer page sentinel. If the page sentinel condition exists, the program branches to the instruction specified in the C operand.

PAGE	LINE	COMMENTS	REFERENCE	OPERATION	A OPERAND	B OPERAND	C OPERAND	ACTION	N - FIELDS	IDENT
			REFNME	BRION	1111		NXTINS			
			REFNME	BRIOFF	1111		NXTINS			
			REFNME	BRS	DDD		NXTINSSSS			

Fig. C-17 Branch on indicator and branch on status statements.

REFNME – Valid symbolic reference name
– Left justified
– First character alpha, remaining characters alphanumeric
– No special characters
– No embedded spaces
– Non-duplicate

DDD – Device code (LPR – Line printer)
– Left justified
– Spaces or invalid indicates zeroes and the status field is ignored

NXTINS – Symbolic reference name of instruction to be branched to
– Left justified
– Spaces or invalid assigns address 00

SSS – Status code (PGS – Page sentinel)
– Left justified
– Spaces or invalid indicates zeroes

iiii – Code of the indicator to be tested
– Left justified
– Space or invalid assigns branch key 00
– The following codes may be entered
ALL – All branch keys (BRIOFF only)
ANY – Any branch keys (BRION only)
LEB – Communications buffer empty (load enabled buffer)
GEB – Communications buffer full (get enabled buffer)
MCE – Mag cassette error (read/write)
BK00 - BK19 – Branch key 0-19
KBF – Keyboard buffer full
LL1 – Last line continuous form 1
LL2 – Last line continuous form 2
FLL – Full ledger left
FLR – Full ledger right
P1 – Program modify key 1
P2 – Program modify key 2
MLE – Magnetic ledger error (read or write)
CSE – Communication send error
CRE – Communication receive error
CET – Cassette EOT
CTM – Cassette tape mark
PTE – Paper tape reader error
LES – Communication line established
RVI – Communication reverse interrupt
RBF – Card reader buffer full
PBA – Card punch buffer available
CFO – Cassette field overflow
EOT – End of transmission (Communications)
ETX – End of text (Communications)
HDR – Header field (Communications)
TMS – Transmit message status (Communciations)
CBT – Cassette blank tape (22 inch blank read)
SGN – Sign (Cassette)
SB5 – Cassette separator bit 5
SB6 – Cassette separator bit 6

CNTL

The CNTL instruction (fig. C-18) causes the peripheral device to perform the function specified in the instruction. CNTL is strictly a function instruction; user data is never transferred.

PAGE	LINE	COMMENTS	REFERENCE	OPERATION	A OPERAND	B OPERAND	C OPERAND	ACTION	N - FIELDS	IDENT
			REFNME	CNTL	DDDD			AAAA	NNN	

Fig. C-18 Control statement

REFNME – Valid symbolic reference name
– Left justified
– First character alpha, remaining characters alphanumeric
– No special characters
– No embedded spaces
– Non-duplicate

DDDD – CAS1 Cassette unit 1
CAS2 Cassette unit 2

AAAA – BACK Back Cassette 1 block
– CTM Cassette tape mark
– RWND Rewind Cassettes
– SFE Separator flag enable (Cass. 1)
– SFD Separator flag disable (Cass. 1)

DDDD – COM Communications

AAAA – CONN Establish Connections
– ECOM Enable Communications
– GEB Get Enable Buffer
– HDR Enable Header
– ICOM Inhibit Communications

	– ID1	Change Identification Code 1
	– ID2	Change Identification Code 2
	– LAST	Input Data Terminator
	– LEB	Load Enable Buffer
	– MBM	Enter Multi-Block Mode
	– RING	Monitor for Ring In
	– RTB	Retransmit Communications Buffer
	– SBM	Enter Single Block Mode
	– INEM	Insert end of media
	– INHT	Insert horizontal tab
	– RESC	Receive escape character
	– STX	Start of text
	– TESC	Transmit escape character
	– BEL	Insert BEL
	– DSGN	Disable sign
	– DSS	Disable space suppress
	– DUS	Disable unit separator
	– DZS	Disable zero suppress
	– EOT	End of transmission
	– ESS	Enable space suppress
	– EZS	Enable zero suppress
DDDD	– CP	Card Punch
AAAA	– FSEL	Field Select Card Column
	– PLZ	Print leading zeros
	– ETX	End of text
	– EM	End of message
NN	– 1-80, right-justified	
DDDD	– CR	Card Reader
AAAA	– FSEL	Field Select Card Column
	– EMTX	End of text/message
NN	– 1-80, right justified	
DDDD	– FH	Forms Handler
AAAA	– FF1	Feed Forms Field 1
	– FF2	Feed Forms Field 2
	– FH1	Feed Forms Home 1
	– FH2	Feed Forms Home 2
	– LE	Left Eject
	– RE	Right Eject
	– RRE	Right Rear Eject
	– LFB	Line Feed Both
	– LFL	Line Feed Left
	– LFR	Line Feed Right
	– PCB	Platen Close Both
	– PCL	Platen Close Left
	– PCR	Platen Close Right
	– POB	Platen Open Both
	– POL	Platen Open Left
	– POR	Platen Open Right
NN	– 1-15 with LFB, LFL, or LFR – Right justified	
DDDD	– LITE	Console Lights
NN	– 0-31, right justified	
DDDD	– LPR	Line Printer
AAAA	– PFP	Position Pointer in Buffer
	– PRNT	Print
	– SLEW	Slew Paper
NNN	– 1-132 for PFP action code. – 1-127 for SLEW action code. – Right justified.	
DDDD	– SPR	Serial Printer
AAAA	– PFP	Position Serial Printer Ball
NNN	– 1-265 – Right justified	
DDDD	– TP	Paper Tape Punch
AAAA	– 046	IBM 046 Translation Table
	– 315	NCR 315 Translation Table
	– 500	NCR 500 Translation Table
	– ASC	ASCII Standard Translation Table
	– KAT	Katakana Translation Table
	– DEL	Punch Delete Characters
	– NUL	Punch NULL Characters
NN	– 1-63 for DEL and NUL action codes only – Right justified	
DDDD	– TR	Paper Tape Reader
AAAA	– 046	IBM 046 Translation Table
	– 315	NCR 315 Translation Table
	– 500	NCR 500 Translation Table
	– ASC	ASCII Standard Translation Table
	– KAT	Katakana Translation Table
	– RWND	Rewind Paper Tape to Leader

If DDDD or AAAA is spaces or invalid, binary ZEROS are inserted into the object program for this instruction.

If NN or NNN is spaces or invalid, where applicable, a binary ONE is inserted into the object program for this instruction; if DDDD is LITE then the insertion value will be a binary ZERO.

DIVIDE INSTRUCTIONS (FIG. C-19)

- DIV B ÷ A → C

The data in the field specified by operand B is divided by the data in the field specified by the A operand. The result is stored in the field designated by operand C. The contents of the A and B operand fields are not altered.

- DIVN $B_N \div A_N \rightarrow B_N$

A specific number of fields, starting at the field specified by the B operand, are divided by the same number of fields, starting at the address specified by the A operand. The quotients are put away in the fields of the B operand. The contents of the A operand fields are not altered.

Restrictions

If the result is shorter than the destination field, unused positions to the left and right of the decimal point are filled with 0's.

If the result is longer than the destination field, the result is truncated at either or both ends to fit the field.

If a division by 0 is attempted, 0's are moved into the

quotient field, with no error indication.

No overflow indication is used.

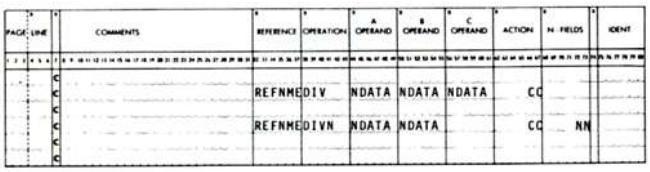

Fig. C-19 Divide statements

REFNME
- Valid symbolic reference name
- Left justified
- First character alpha, remaining characters alphanumeric
- No special characters
- No embedded spaces
- Non duplicate

NDATA
- Symbolic reference name of previously defined N type data definition
- Left justified
- Space or undefined assigns address 00
- A operand is divisor
- B operand is dividend; also putaway field on DIVN
- C operand is putaway field (DIV)

CC
- Enter RD when quotient is to be rounded
- Right justified
- Space or invalid entry indicates no rounding

NN
- Number of fields to be affected (1-31)
- Right justified
- Blank or invalid assumes 1

FILL

The FILL instruction (fig. C-20) clears an area of memory, starting at the data field specified in operand A and continuing for the number of fields specified in N-FIELDS. Numeric fields are filled with zeroes, and alpha fields are filled with spaces.

RESTRICTIONS

The execution of the FILL instruction halts, displaying Alert Code 15 if the instruction encounters an address which should be a header, but is not. This can happen if more fields are specified than exist in memory; if the clearing operation enters a communications buffer; or if the clearing operation enters a REDEF header area.

In computing the number of consecutive fields to be cleared by the FILL instruction, T type data definitions must not be included. T type definitions are used only as address constants during assembly, and they are not stored in memory.

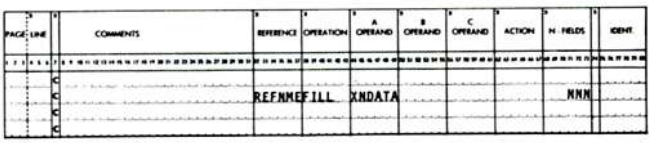

Fig. C-20 Fill statement

REFNME
- Valid symbolic reference name
- Left justified
- First character alpha, remaining characters alphanumeric
- No special characters
- No embedded spaces
- Non-duplicate

XNDATA
- Symbolic reference name of previously defined X or N type data definition
- Left justified
- Space or invalid assigns address 00

NNN
- Number of contiguous fields to be space or zero filled (1-999)
- Right justified
- Blank or invalid indicates 1

GET

The GET instruction (fig. C-21) transfers data from the peripheral device to memory. Data is accepted from the peripheral specified in operand A and stored in memory, starting with the data field specified by the B operand. The number of fields involved is specified by N-FIELDS.

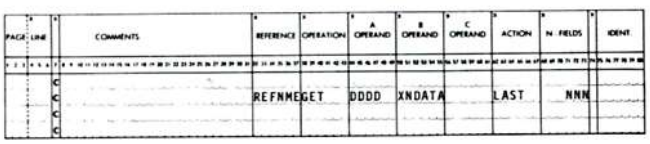

Fig. C-21 Get statement

REFNME
- Valid symbolic reference name
- Left justified
- First character alpha, remaining characters alphanumeric
- No special characters
- No embedded spaces
- Non duplicate

DDDD
- Device codes, left justified
- CAS1 Data Cassette 1
- CAS2 Data Cassette 2
- CR Card Reader
- COM Communications
- DISC Disc
- KB Numeric Keyboard
- ML Magnetic Ledger
- MLFR Magnetic Ledger Feeder-Reader
- LNCR Line Number Count Right (N-FIELD column not used)
- LNCL Line Number Count Left (N-FIELD column not used)
- TR Paper Tape Reader
- Spaces or invalid code indicates keyboard

XNDATA – Symbolic reference name of previously defined X or N type data definition (KB, LNCR, and LNCL always use N type)
– Left justified
– Space or invalid assigns address 00

NNN – Number of contiguous putaway fields (1-255)
– Card reader can use maximum of 80
– Right justified
– Blank or invalid assumes 1

LAST – Indicates that the last unit of data is being processed from the buffer (card reader and communications)

PUT

The PUT instruction (fig. C-22) transfers data from memory to a peripheral device. Data is transferred from memory, starting with the data field specified in operand B, to the peripheral specified in operand A. The number of fields to be transferred is specified in N-FIELDS.

PAGE	LINE	COMMENTS	REFERENCE	OPERATION	A OPERAND	B OPERAND	C OPERAND	ACTION	N FIELDS	IDENT
	C C C C C C C		REFNME	PUT	DDDD	XNDATA		EEEe or CCCC	NNN	

Fig. C-22 Put statement

REFNME – Valid symbolic reference name
– Left justified
– First character alpha, remaining characters alphanumeric
– No special characters
– No embedded spaces
– Non duplicate

DDDD – Device codes, left justified
– CAS1 Data Cassette 1
– CAS2 Data Cassette 2
– CP Card Punch
– COM Communications
– DISC Disc
– ML Magnetic Ledger
– LNCR Line Number Counter Right (N-FIELD column not used)
– LNCL Line Number Counter Left (N-FIELD column not used)
– TP Paper Tape Punch
– LPR Line Printer
– SPR Serial Printer
– Spaces or undefined assigns address 00

XNDATA – Symbolic reference name of previously defined X or N type data definition (LPR, LNCR, and LNCL always use N type)
– Left justified
– Space or invalid assigns address 00

EEE – Primary printer edit codes, left justified
– Serial printer
– ABN, spaces, or invalid – All black, no symbol
– ARN All red, no symbol
– PBC Positive black, negative red with CR
– PBD Positive black, negative red with ◊
– PRC Positive red, negative black with CR
– PRD Positive red, negative black with ◊

– Line Printer
– LPN Do not print sign
– LPS Print sign

e – Secondary serial printer edit codes
– $ Dollar protect
– E Dollar edit
– S Zero suppress
– ☑ or invalid – Absolute print with decimal

CCCC – Action codes other than print edit
– SLEW Causes line printer to slew paper the number of lines (1-127) dictated by the contents of NDATA. After the line printer prints the contents of its buffer, the slew is executed.
– LAST Indicates that the last unit of data is being processed from the buffer (card punch and communications)

NNN – Number of contiguous source fields
– Right justified
– 1-31 for serial and line printers
– Not used with line printer SLEW instruction or LNCR and LNCL
– 1-255 for other instructions
– Blank or invalid indicates 1

GBP

The GBP instruction (fig. C-23) combines either the GET:KB and BRION instructions or the GET:KB and PUT:SPR instructions.

The numeric keyboard is activated, and the data entered through it is stored in the field designated in the instruction. The keyboard entry is terminated by the operator, who presses either the ENTER bar or a branch key.

When the keyboard entry is terminated through the branch key specified in the instruction, the program branches to the instruction specified in the instruction, and no print occurs.

When the keyboard entry is terminated by the ENTER bar or a branch key other than the one called for in the instruction, the keyboard entry stored in memory is printed by the serial printer. The format of the print is controlled by the data definition of the field printed, and by the printer edit codes in the instruction. If the field contains nothing but 0's, no print occurs.

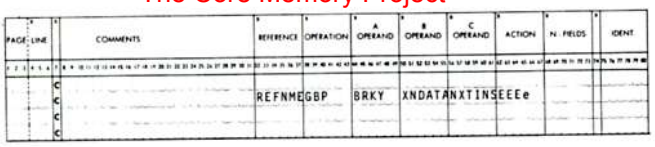

Fig. C-23 Get-branch-put statement

REFNME
- Valid symbolic reference name
- Left justified
- First character alpha, remaining characters alphanumeric
- No special characters
- Non duplicate

BRKY
- Branch key indicator code to be tested
- BK00 - BK19
- Blank or invalid indicates branch key 00

XNDATA
- Symbolic reference name of previously defined X or N type data definition
- Field in which keyboard entry data is to be stored
- Left justified
- Blank or undefined assigns address 00

NXTINS
- Symbolic reference name of instruction to be branched to
- Destination when branch is taken
- Left justified
- Blank or invalid assigns address 00

EEE
- Primary printer edit codes — Left justified
- ABN, blank or invalid — All black, no symbol
- ARN All red, no symbol
- PBC Positive black, negative red with CR
- PBD Positive black, negative red with ◊
- PRC Positive red, negative black with CR
- PRD Positive red, negative black with ◊

e
- Secondary printer edit codes
- $ Dollar protect
- E Dollar edit
- S Zero suppress
- ☑ or invalid — Absolute print with decimal

MOVE INSTRUCTIONS (FIG. C-24)

- MOVE $A_N \rightarrow \bar{B}_N$

A specific number of fields, 1-31, starting with the A operand field, are moved into the same number of fields, starting with the B operand field. The contents of the A operand fields are not altered.

When moving numeric data fields, the decimal point definition in the header of each destination field determines the decimal position for that field.

The method of data movement is determined by the type, X or N, of each destination field. If it is X (alphanumeric), data is moved from the origin field into the destination field eight bits at a time. An N (numeric) destination field receives data four bits at a time.

Restrictions (MOVE, alpha field to numeric field)

An alpha source field length greater than eight characters must not be used.

The source field length must be equal to the destination field length. If they are unequal, an internal error occurs, and the ALERT 1, 2, 3, and 4 indicators are set ON.

If the destination field is smaller than the source field, the data is truncated at either or both ends to fit the field.

If the destination field is larger than the source field, unused positions to the right and left of the decimal point are filled with 0's.

Restrictions (MOVE, numeric field to alpha field)

An alpha destination field length greater than eight characters must not be used.

The source field length must be equal to the destination field length. If they are unequal, an internal error occurs, and the ALERT 1, 2, 3, and 4 indicators are set ON.

If the destination field is smaller than the source field, the data is truncated on the left end.

If the destination field is larger than the source field, unused high-order digits are space-filled.

Any data to the right of the decimal point in the source field is lost.

Restrictions (MOVE, numeric field to numeric field)

If the destination field is smaller than the source field, the data is truncated at either or both ends to fit the field.

If the destination field is larger than the source field, unused positions to the right and left of the decimal point are zero-filled.

The rounding option is effective only when the destination field contains fewer places to the right of the decimal than the source field. For example, 123.456 moved into NNN.N becomes 123.5.

Restrictions (MOVE, alpha field to alpha field)

The source field length must be equal to the destination field length. If they are unequal, an internal error occurs and the ALERT 1, 2, 3, and 4 indicators are set ON.

- MOVENA $A_N \rightarrow B$

The MOVENA instruction moves a specific number of small data fields into one larger field. Data from the origin fields is left-justified in the destination field; that is, the data from the first small field is stored in the leftmost positions of the larger field, data from the next small field is stored to the right of the first, and so on, until the last small field has been stored. The contents of the small fields are not altered.

The headers of the small data fields are not transferred to the larger field, which has its own header. The header of the larger field defines all the data moved.

Restrictions (MOVENA)

If the move without sign option is not specified, the sign of the most significant small field is moved into the large field.

If the source fields length is greater than the destination length, fields following the destination fields are written over.

If the source fields length is less than the destination length, the unused low order digits are zero or space filled.

● MOVENB A → B_N

The MOVENB instruction distributes portions of a large data field into a specific number of smaller fields. Data is transferred, starting with the leftmost character of the large data field, into the first small data field, left justified. When the first small field is filled, data is transferred into the next small field, and so on until all the small fields have been filled. The contents of the large field are not altered.

The length of the data transferred is determined by the total length of the small fields. Therefore, this total length must not exceed the length of the large field.

Restrictions (MOVENB)

If the move without sign option is not specified, the sign of the source field is moved into each of the destination fields.

If the source field length is greater than the destination fields length, the excess low order source digits are not moved.

If the source field length is less than the destination fields length, digits from the data field following the source field are moved.

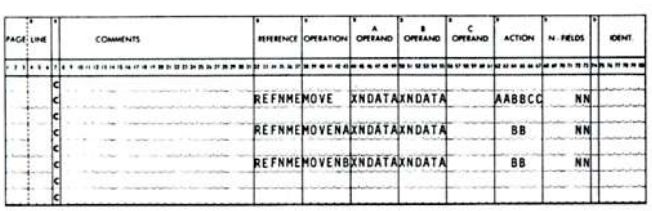

Fig. C-24 Move statements

REFNME
- Valid symbolic reference name
- Left justified
- First character alpha, remaining characters alphanumeric
- No special characters
- Non duplicate

XNDATA
- Symbolic reference name of previously defined X or N type data definition
- Left justified
- A operand names source field(s)
- B operand names destination field(s)
- Blank or undefined assigns address 00

AA
- Fill option
- ⊠⊠ Space-fill destination field(s) if X type, zero-fill if N type
- NF Do not fill destination field(s)

BB
- Sign option
- ⊠⊠ Move the sign source field(s) to the destination field(s)
- NS Do not move sign to destination field(s)

CC
- Round option (N type data definitions only)
- ⊠⊠ Do not round the destination field(s)
- RD Round the destination field(s)

NN
- Number of fields to be affected (1-31)
- Blank or invalid assumes 1

MULTIPLY INSTRUCTIONS (FIG. C-25)

● MULT B x A → C

The data field named by the B operand is multiplied by the data field named by the A operand. The product is put away in the data field named by the C operand. The contents of the A and B operand fields are not altered.

● MULTN B_N x A_N → B_N

A specific number of fields, starting with the data field named in the B operand, are multiplied by the same number of fields, starting with the data field named in the A operand. The products are put away in the B operand fields. The contents of the A operand fields are not altered.

Restrictions

If the result is longer than the destination field, the result is truncated at either or both ends to fit the field.

If the result is shorter than the destination field, unused positions to the left and right of the decimal point are zero filled.

No overflow indication is used.

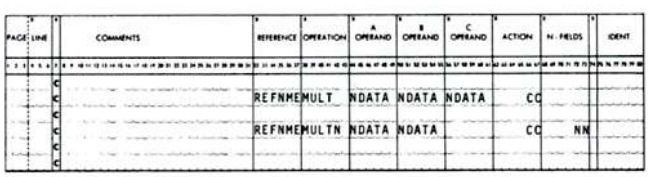

Fig. C-25 Multiply statements

REFNME
- Valid symbolic reference name
- Left justified
- First character alpha, remaining characters alphanumeric
- No special characters
- Non duplicate

NDATA
- Symbolic reference name of previously defined N type data definition
- Left justified
- A operand names the multiplier
- B operand names the multiplicand; in MULTN, also names putaway fields
- C operand names the putaway field (MULT)
- Blank or undefined assigns address 00

CC
- Round option
- Right justified
- ⊠⊠ Do not round the putaway field(s)
- RD Round the putaway field(s)

NN
- Number of fields to be affected (1-31)
- Blank or invalid assumes 1

REDEF

The REDEF instruction (fig. C-26) makes it possible to change the data configuration of a part of the user data area of memory. For instance, a given portion of memory may contain three large data fields. The REDEF instruction can change this to one or two larger fields or many smaller fields.

A previously defined format for redefinition (F type data definition) contains a group of headers. The REDEF instruction moves the first of these headers into the position occupied by the header of the first field of the area to be redefined. The new header specifies the length of the new field; the new field, which may be shorter, longer, or equal to the original field, is then space or zero filled. The new field is written over any previously recorded data within its length, including headers or print position bits for other fields.

The second header from the redefine area is then stored in memory on the first word boundary following the field just created. As with the first new field, this field is space or zero filled.

This procedure is repeated until the number of fields specified in the instruction are created. The redefine area is not altered.

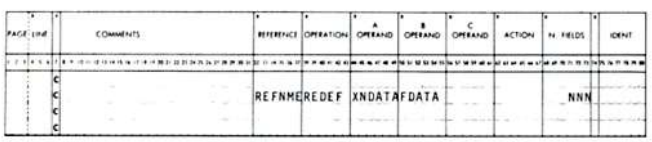

Fig. C-26 Redefine statement

REFNME
- Valid symbolic reference name
- Left justified
- First character alpha, remaining characters alphanumeric
- No special characters
- Non duplicate

XNDATA
- Symbolic reference name of previously defined X or N type data definition
- Left justified
- Name of first data field of area to be redefined
- Blank or invalid assigns address 00

FDATA
- Symbolic reference name of previously defined F type data definition
- Left justified
- Name of packed header area
- Blank or invalid assigns address 00

NNN
- Number of fields to be created in area to be redefined (1-999)
- Right justified
- Blank or invalid assumes 1

SAVERA/RETURN (FIG. C-27)

When a branch is executed in a program, the address of the user instruction following the branch instruction is stored in a special memory address called YBRLNK.

The SAVERA instruction transfers the contents of YBRLNK to a memory address called SAVECR.

The RETURN instruction causes program execution to continue at the address specified in SAVECR, which puts the program back into its normal flow.

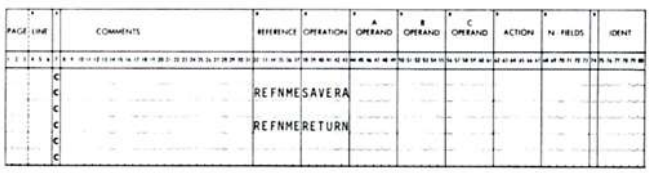

Fig. C-27 Savera and Return statements

REFNME
- Valid symbolic reference name
- Left justified
- First character alpha, remaining characters alphanumeric
- No special characters
- Non duplicate

SUBTRACT INSTRUCTIONS (FIG. C-28)

● SUB B - A → C

The data field specified by the A operand is subtracted from the field specified by the B operand. The result is put away in the field specified by the C operand. The contents of the A and B operand fields are not altered.

● SUBN B_N - A_N → B_N

A specific number of fields, starting with the field designated by the A operand, are subtracted from the same number of fields starting with the field designated by the B operand. The results are put away in the fields of the B operand. The contents of the A operand fields are not altered.

● SUBNA B - A_N → B

A specific number of fields, starting with the field designated by the A operand, are subtracted from the field designated by the B operand. The result is put away in the B operand field. The contents of the A operand fields are not altered.

● SUBNB B_N - A → B_N

The field designated by the A operand is subtracted from each of a specific number of fields, starting with the field designated by the B operand. The results of the subtractions are put away in the B operand fields. The contents of the A operand field are not altered.

Restrictions

If the result is shorter than the destination field, unused positions to the left and right of the decimal point are filled with 0's.

If the result is longer than the destination field, the result is truncated at either or both ends to fit the field.

No overflow indication is used.

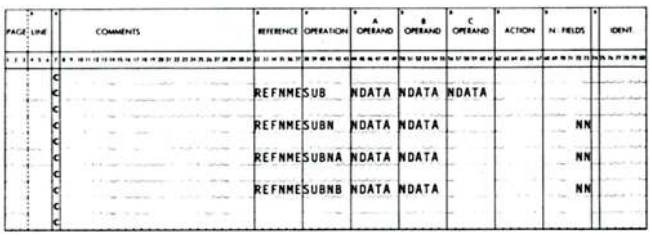

Fig. C-28 Subtract statement

REFNME
- Valid symbolic reference name
- Left justified
- First character alpha, remaining characters alphanumeric
- No special characters
- No embedded spaces
- Non-duplicate

NDATA
- Symbolic reference name of previously defined N type data definition
- Left justified
- Space or undefined assigns address 00

NN
- Number of fields to be affected (1-31)
- Right justified
- Blank or invalid assumes 1

TABLE INSTRUCTIONS

These instructions (fig. C-29) deal with tables set up on the data layout worksheet (T type data definition). Data can be moved into a table from a work area, be moved from a table into a work area, or be taken from a work area and added to data within a table.

●TBLIN $B_N \rightarrow (A + C)_N$

The TBLIN instruction moves a number of contiguous work area fields into a corresponding number of table fields. The first field to be moved is determined by the B operand. The A operand specifies the starting point of the table, and the C operand indicates the position, within the table, of the first field to receive data. Up to 63 contiguous fields, specified in N-FIELDS, can be moved.

Data is transferred from one field to another in the form of numeric digits (four bits) or alphanumeric characters (eight bits). The type of transfer is governed by the data type of each table field, which must be the same as the corresponding work area field.

The sign is moved with the data, and no fill is performed.

Restrictions (TBLIN)

Data moved from a work area field to a smaller table field may be truncated at both ends to fit the table field; no error is indicated.

Data moved from a work area field into a larger table field may include part of the data field following the work area field.

If the decimal definitions of the work area field and the table field differ, the result may be truncated at either end, with no error indication.

●TBLOUT $(A + C)_N \rightarrow B_N$

The TBLOUT instruction moves a number of contiguous table fields into a corresponding number of work area fields. The first field to be moved is determined by the A and C operands. The A operand specifies the starting point of the table, and the C operand indicates the position within the table of the first field to be moved. The first field to receive data is specified by the B operand. Up to 63 contiguous fields, specified in N-FIELDS, can be moved.

Data is transferred from one field to another in the form of numeric digits (four bits) or alphanumeric characters (eight bits); the type of transfer is governed by the data type of each work area field, which must be the same as that of the corresponding table field.

The sign is moved with the data, and no fill is performed.

Restrictions (TBLOUT)

The work area field length must be equal to or greater than the corresponding table field length.

The decimal definition of the work area field must be equal to or greater than the corresponding table field decimal definition.

If either of the previously mentioned restrictions are violated, an error is detected and the data is not moved. The system halts in a M05 WAIT, with the ALERT 1, 2, and 8 lights ON.

●TBLADD $B_N + (A + C)_N \rightarrow (A + C)_N$

The TBLADD instruction adds a number of contiguous work area fields to a corresponding number of table fields. The first fields to be added are determined by the B operand for the work area field and by the A operand, indexed by the C operand, for the table field. Up to 63 contiguous fields, specified in N-FIELDS, can be added to the table fields.

Restrictions (TBLADD)

Data moved from a work area field to a smaller table field may be truncated at both ends to fit the table field; no error is indicated.

Data moved from a work area field into a larger table field may include part of the data field following the work area field.

If the decimal definitions of the work area field and the table field differ, the result may be truncated at either end, with no error indication.

No overflow indication is used.

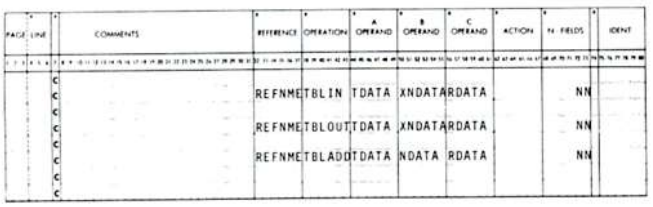

Fig. C-29 Table statements

REFNME
- Valid symbolic reference name
- Left justified
- First character alpha, remaining characters alphanumeric
- No special characters
- No embedded spaces
- Non-duplicate

TDATA
- Symbolic reference name of previously defined T type data definition
- Names table to be affected
- Left justified
- Blank or invalid assigns address 00

XNDATA
- Symbolic reference name of previously defined X or N type data definition
- Names first work area field (TBLIN and TBLOUT)
- Left justified
- Blank or invalid assigns address 00

NDATA
- Symbolic reference name of previously defined N type data definition
- Names first work area field (TBLADD)
- Left justified
- Blank or invalid assigns address 00

RDATA
- Symbolic reference name of previously defined N type data definition, containing no more than four digits (1-7999)
- Location of first table field to be affected, relative to the beginning of the table
- Left justified
- Zero, negative, or greater than 7999 indicates value of 1
- Blank or invalid assigns address 00

NN
- Number of fields to be affected (1-63)
- Right justified
- Blank or invalid indicates 1

TYPE/TYPEK/TYPEL/TYPEM (FIG. C-30)

The TYPE instruction allows data entered on the alphanumeric keyboard to be printed by the serial printer. The numeric keyboard is locked out.

The TYPE instruction is terminated by the RESUME bar or a branch key; the amount of typing possible prior to termination is entirely up to the operator.

The TYPEK instruction differs from TYPE in that the instruction ables the alphanumeric keyboard for typing only if the TYPE key has been pressed. If the key has not been pressed, the program skips the TYPEK instruction and advances to the next instruction.

With the TYPEL instruction, a specific number of characters, up to 999, may be typed. The N-FIELDS column contains the maximum number of characters allowed for a particular TYPEL instruction. When the number of characters specified has been typed, the instruction terminates. The operator can also terminate the instruction at any time by pressing the RESUME bar or a branch key.

The characters typed are not stored in memory.

With the TYPEM instruction, data entered through the alphanumeric keyboard is printed by the serial printer and stored in memory. The storage area starts at the data field named by operand A and continues, up to a maximum of 63 fields, as specified by N-FIELDS.

A count is maintained to determine when the number of fields specified in the instruction is satisfied. The instruction terminates at the completion of the final field.

Pressing the RESUME bar terminates entry of a field; repeated depression of the RESUME bar until the field count is satisfied terminates the instruction. The instruction can also be terminated by pressing a branch key one time.

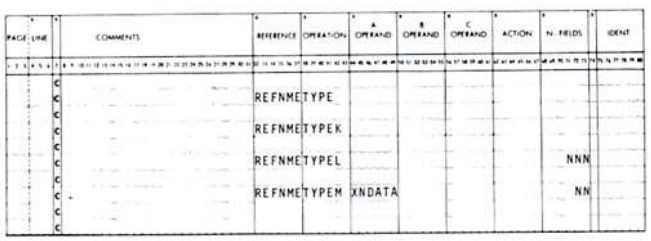

Fig. C-30 Type statements

REFNME
- Valid symbolic reference name
- Left justified
- First character alpha, remaining characters alphanumeric
- No special characters
- No embedded spaces
- Non-duplicate

XNDATA
- Previously defined symbolic reference name of X or N type data definition
- First field of memory storage area
- Left justified
- Blank or invalid assigns address 00

NNN
- Number of characters (1-999) that can be typed
- Right justified
- Blank or invalid assumes 1

NN
- Number of fields (1-63) that can be typed and stored in memory
- Right justified
- Blank or invalid assumes 1

WAIT

The WAIT instruction (fig. C-31) halts all processing until the ENTER bar or a branch key is pressed.

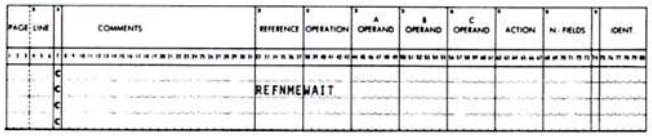

Fig. C-31 Wait statement

REFNME – Valid symbolic reference name
– Left justified
– First character alpha, remaining characters alphanumeric
– No special characters
– No embedded spaces
– Non-duplicate

END$

The END$ statement (fig. C-32) indicates to the assembler the end of the source program; it is the last statement in the program.

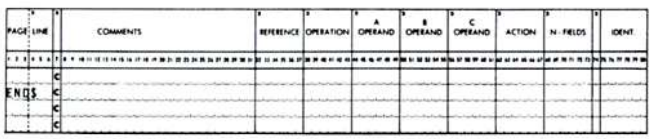

Fig. C-32 END$ statement

OBJECT LEVEL

DATA FORMATS

The user data is encoded on the cassette tape and stored in memory in serial bit fashion, on a word boundary basis. Word boundary means that a field of data must start at the beginning of a 16-bit word, not within the word. If the last word of a field is not filled, the next field does not begin until the next word boundary.

A data field is a number of 16-bit words making up a unit of data. The field can contain either numeric data or alphanumeric data, but not a mixture of the two. A 16-bit word can contain four numeric characters or two alphanumeric characters, stored in ASCII code.

ASCII coding is simply a standard binary coding of 128 characters, numbers, and symbols used in data processing. Figure C-33 is a chart showing the code configuration of each of the 128. Seven bits are needed to provide the 128 configurations; for simplification, two digits (eight bits) are allotted to make up one ASCII character.

B_4-B_1 → / B_7-B_5 ↓	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0000	0	NUL	SOH	STX	ETX	EOT	ENQ	ACK	BEL	BS	HT	LF	VT	FF	CR	SO	SI
0001	1	DLE	DC1	DC2	DC3	DC4	NAK	SYN	ETB	CAN	EM	SUB	ESC	FS	GS	RS	US
0010	2	SP	!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
0011	3	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
0100	4	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
0101	5	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
0110	6	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
0111	7	p	q	r	s	t	u	v	w	x	y	z	{	¦	}	~	DEL

Fig. C-33 ASCII code chart

Using the two digits per character ASCII system, a 16-bit word can contain two characters, as is the case with alphanumeric data. All the ASCII numeric characters, however, use the same binary configuration in the high order digit, and the low order digit is the binary form of the number represented. When a data field is numeric, then, the high order digit is unnecessary, since it is always the same.

By eliminating the high order digit, four numeric characters can be represented in a word; this is called packed ASCII or binary coded decimal (BCD).

A method of identifying numeric and alphanumeric fields is needed to implement the storage of four numeric characters in a word. The data field header contains that information.

Immediately preceding every data field is a 16-bit data field header, which contains information defining the field which follows it. The numeric header differs in some ways from the alpha header; a breakdown of the two headers is shown in figure C-34.

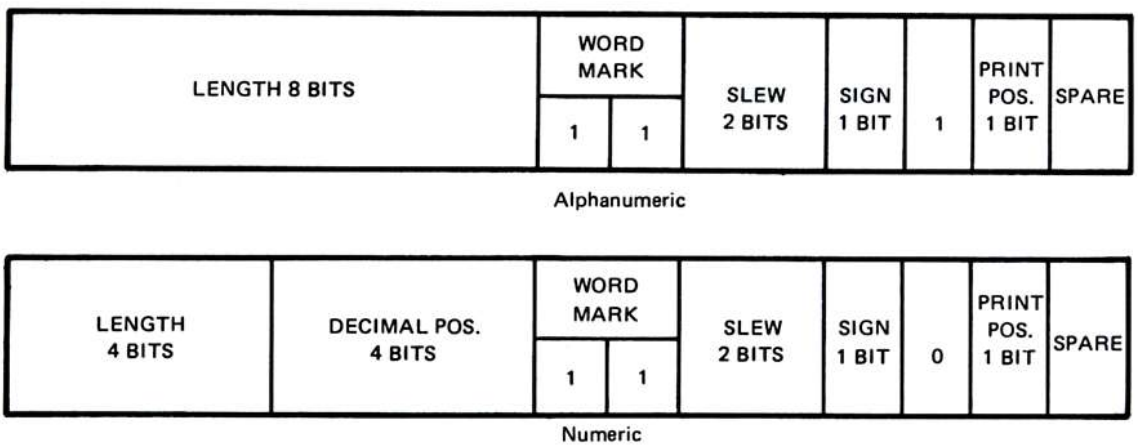

Fig. C-34 Alphanumeric and numeric headers

The high order digit of the numeric header indicates the number of numeric characters the field can contain, up to a maximum of 16. The number of characters in the field is always one more than the header length digit specifies, to take advantage of all 16 possible configurations of the length digit; 0 in the header defines a field length of one character, 1 in the header defines two characters, and so on, to 15 (1111) in the header for the maximum of 16 characters. It follows that a one character field is the smallest data field possible.

The decimal digit in the numeric header defines the position of the decimal point within the field. The binary value in the decimal digit (0-15) represents the number of characters to the right of the decimal point. The decimal position must never be greater than the field length.

In the alpha header, no decimal position is needed; the two high order digits define a field length of up to 256 8-bit characters. As in the numeric header, the actual number of characters in an alpha field is one more than the length portion of the header indicates.

The two word-mark bits always contain 1's in both types of headers. These bits identify the header as not being a word of data; no data configuration used in the 399 has both these bit positions on at the same time. Therefore, any word with 1's in both bit positions is recognized as a data field header by the system logic.

The two slew bits are used when the data field is put to the serial or line printer. Table C-4 illustrates the printer functions controlled by these bits.

SERIAL PRINTER	SLEW BITS	LINE PRINTER
NO PAPER FEED	00	NO PRINT, NO SLEW
PAPER FEED RIGHT	01	PRINT, SLEW ONE LINE
PAPER FEED LEFT	10	PRINT, SLEW TWO LINES
PAPER FEED BOTH	11	PRINT, SLEW THREE LINES

Table C-4 Printer slew bits configurations

The sign bit designates the sign of the data field. A positive sign is indicated by 0, and a negative sign is indicated by 1.

The bit position to the right of the sign bit indicates whether the field is alpha or numeric. A 0 in this bit position specifies numeric and a 1 specifies alpha.

The print position (P/P) bit indicates whether or not a particular starting column for the serial printer is called for. If a 0 is present in the P/P bit, the printing begins in the column in which the print head came to rest following its last movement. A 1 in the P/P bit of the header results in an eight-bit print position immediately following the header. The eight print position bits specify a print column from 1 to 255, causing the print head to assume that column before starting to print the data field. If a column higher than 255 is required, a Control instruction must be used.

The last bit of the header is not used.

Figures C-35 through C-38 depict various data field configurations in memory.

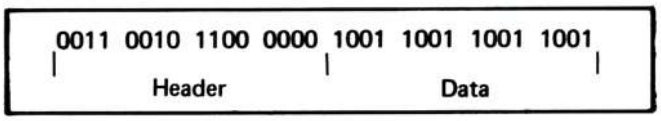

Fig. C-35 Numeric data field

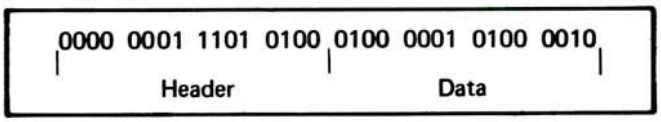

Fig. C-36 Alphanumeric data field

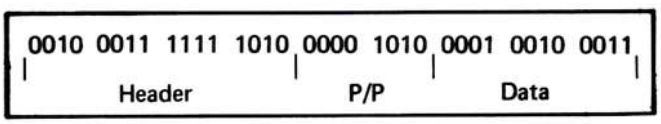

Fig. C-37 Numeric data field with print position

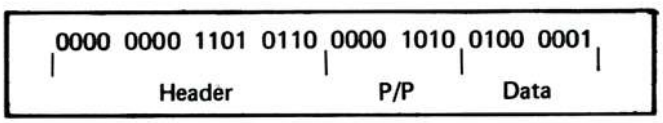

Fig. C-38 Alphanumeric data field with print position

INSTRUCTION FORMATS

A 399 instruction at the object level (fig. C-39) consists of a minimum of two 4-bit digits to a maximum of 14 digits. The leftmost six bits identify the instruction and are called the Q code. The remaining two bits of the first two digits are referred to as mode bits (M bits). The M bits indicate certain miscellaneous information about the instruction; they are further explained later.

All 399 instructions contain these two digits, and they are used in the same way in each instruction.

Three M digits are available in addition to the two standard M bits. One, two, or all three M digits can be used in an instruction. These digits serve the same purpose as the M bits; they contain information about the instruction.

Q CODE	M BITS	M DIGITS	A OPERAND ADDR.	B OPERAND ADDR.	C OPERAND ADDR.
6 BITS	2	0 TO 3 DIGITS	3 DIGITS WHEN USED	3 DIGITS WHEN USED	3 DIGITS WHEN USED

Fig. C-39 Instruction format

Up to three operand addresses can be incorporated into an instruction, depending on how many operands that instruction makes use of. Some instructions do not use any operands, and others use one, two, or three.

Each operand address contains 12 bits (three digits) to specify the address, in binary, of the location in memory containing the first word of the first data field in the operand. This first word is the header for that field, as explained in Data Formats.

It is possible for 14 digits to be used in a 399 instruction, but no existing instruction exceeds 12 digits. Each instruction is made up of a specific number of digits, and it occupies that number of digits in memory each time it appears in the program. For instance, the DIV instruction is 10 digits in length and is always that length; it cannot vary.

The following descriptions break down the bit strings of each instruction. The entries under OBJECT pertain to the bit string of the instruction in object form. SOURCE COLUMN entries indicate the corresponding entry on the Coding Worksheet.

Keep in mind that the operands in the bit strings are binary addresses of data fields or instructions, and are not the fields or instructions themselves. The Q code, mode bits, and mode digits are binary representations in their final form.

ADD B + A → C

0010 0000 AAAA AAAA AAAA BBBB BBBB BBBB
CCCC CCCC CCCC

OBJECT	SOURCE COLUMN
Q - 8	OPERATION
A - first operand	OPERAND A
B - second operand	OPERAND B
C - putaway operand	OPERAND C

ADDN $B_N + A_N \rightarrow B_N$

0000 110N NNNN AAAA AAAA AAAA
BBBB BBBB BBBB

OBJECT	SOURCE COLUMN
Q - 3	OPERATION
N - number of contiguous fields (1-31)	N-FIELDS
A - first operand	OPERAND A
B - second operand	OPERAND B

ADDNA $B + A_N \rightarrow B$

0000 010N NNNN AAAA AAAA AAAA
BBBB BBBB BBBB

OBJECT	SOURCE COLUMN
Q - 1	OPERATION
N - number of contiguous fields (1-31)	N-FIELDS
A - first operand	OPERAND A
B - second operand	OPERAND B

ADDNB $B_N + A \rightarrow B_N$

0000 100N NNNN AAAA AAAA AAAA
BBBB BBBB BBBB

OBJECT	SOURCE COLUMN
Q - 2	OPERATION
N - number of contiguous fields (1-31)	N-FIELDS
A - first operand	OPERAND A
B - second operand	OPERAND B

BR (UNCONDITIONAL)

0010 10aa aaaa aaaa aaaa

OBJECT	SOURCE COLUMN
Q - 10	OPERATION
a - destination address	OPERAND C

BRE

0011 00aa aaaa aaaa aaaa AAAA AAAA AAAA
BBBB BBBB BBBB

OBJECT	SOURCE COLUMN
Q - 12	OPERATION
a - destination address	OPERAND C
A - field to be tested	OPERAND A
B - comparator field	OPERAND B

BRU

0011 01aa aaaa aaaa aaaa AAAA AAAA AAAA
BBBB BBBB BBBB

OBJECT	SOURCE COLUMN
Q - 13	OPERATION
a - destination address	OPERAND C
A - field to be tested	OPERAND A
B - comparator field	OPERAND B

BRL

0011 10aa aaaa aaaa aaaa AAAA AAAA AAAA
BBBB BBBB BBBB

OBJECT	SOURCE COLUMN
Q - 14	OPERATION
a - destination address	OPERAND C
A - field to be tested	OPERAND A
B - comparator field	OPERAND B

BRG

0011 11aa aaaa aaaa aaaa AAAA AAAA AAAA
BBBB BBBB BBBB

OBJECT	SOURCE COLUMN
Q - 15	OPERATION
a - destination address	OPERAND C
A - field to be tested	OPERAND A
B - comparator field	OPERAND B

BRS

1001 01DD DDDT TTTT 0ccc cccc cccc cccc

OBJECT	SOURCE COLUMN
Q - 37	OPERATION
D - device code (see table C-5, page C-30)	OPERAND A
T - status code (see table C-10, page C-31)	ACTION
c - destination address	OPERAND C

BRION

0010 11ii iiii iaaa aaaa aaaa aaaa

OBJECT	SOURCE COLUMN
Q - 11	OPERATION
i - indicator code (see table C-7, page C-30)	OPERAND A
a - destination address	OPERAND C

BRIOFF

0100 00ii iiii iaaa aaaa aaaa aaaa

OBJECT	SOURCE COLUMN
Q - 16	OPERATION
i - indicator code (see table C-7, page C-30)	OPERAND A
a - destination address	OPERAND C

DIV B ÷ A → C

0110 10R0 AAAA AAAA AAAA BBBB BBBB BBBB CCCC CCCC CCCC

OBJECT	SOURCE COLUMN
Q - 26	OPERATION
R - 1 = round, 0 = do not round	ACTION
A - divisor field	OPERAND A
B - dividend field	OPERAND B
C - putaway field	OPERAND C

DIVN $B_N \div A_N \rightarrow B_N$

0110 11RN NNNN AAAA AAAA AAAA BBBB BBBB BBBB

OBJECT	SOURCE COLUMN
Q - 27	OPERATION
R - 1 = round, 0 = do not round	ACTION
N - number of fields (1-31)	N-FIELDS
A - first divisor field	OPERAND A
B - first dividend field, first putaway field	OPERAND B

FILL

1001 00NN NNNN NNNN AAAA AAAA AAAA

OBJECT	SOURCE COLUMN
Q - 36	OPERATION
N - number of contiguous fields to be filled (1-999)	N-FIELDS
A - address of first field to be filled	OPERAND A

GET

AUTO LINE FIND – Left or Right Line Number Counter

0101 00DD DDD0 0000 000r AAAA AAAA AAAA

All Other Devices

0101 00DD DDDm NNNN NNNN AAAA AAAA AAAA

OBJECT	SOURCE COLUMN
Q - 20	OPERATION
D - device code (see table C-5, page C-30)	OPERAND A
*m - indicates last get	ACTION
N - number of fields (1-255)	N-FIELDS
A - first field of putaway area	OPERAND B
r - left or right line number counter (1 = left, 0 = right)	OPERAND A

PUT

SERIAL PRINTER
0101 01DD DDEE EeeN NNNN AAAA AAAA AAAA

LINE PRINTER – Data
0101 01DD DDD0 EEEN NNNN AAAA AAAA AAAA

LINE PRINTER – Slew
0101 01DD DDD1 0000 0001 AAAA AAAA AAAA

AUTO LINE FIND – Left or Right Line Number Counter
0101 01DD DDD0 0000 000r AAAA AAAA AAAA

ALL OTHER DEVICES
0101 01DD DDDm NNNN NNNN AAAA AAAA AAAA

OBJECT	SOURCE COLUMN
Q - 21	OPERATION
D - device code (see table C-5, page C-30)	OPERAND A
E - primary edit code (see table C-6, page C-30)	ACTION
e - secondary printer edit code (see table C-8, page C-31)	ACTION
N - number of fields (1-31 for printers, 1-255 for others)	N-FIELDS
A - 1. first field from memory to be put	OPERAND B
2. number of lines to slew (Line Printer - slew)	OPERAND B
3. number to be put to the line number counter	OPERAND B
r - left or right line number counter (1 = left, 0 = right)	OPERAND A
*m - indicates last put	ACTION

*Used only with communications or card punch

GBP

0101 10EE Eeei iiii iaaa aaaa aaaa aaaa
AAAA AAAA AAAA

OBJECT	SOURCE COLUMN
Q - 22	OPERATION
E - primary printer edit code (see table C-6, page C-30)	ACTION
e - secondary printer edit code (see table C-8, page C-31)	ACTION
i - indicator code (branch key number)	OPERAND A
a - destination address	OPERAND C
A - field in which keyboard data is stored	OPERAND B

MOVE $A_N \rightarrow B_N$

0111 1100 FRSN NNNN AAAA AAAA AAAA
BBBB BBBB BBBB

OBJECT	SOURCE COLUMN
Q - 31	OPERATION
F - 1 = fill, 0 = no fill	ACTION
R - 1 = round, 0 = no round	ACTION
S - 1= move sign, 0 = do not move sign	ACTION
N - number of contiguous fields (1-31)	N-FIELDS
A - first field to be moved	OPERAND A
B - first destination field	OPERAND B

MOVENA $A_N \rightarrow B$

0111 0100 10SN NNNN AAAA AAAA AAAA
BBBB BBBB BBBB

OBJECT	SOURCE COLUMN
Q - 29	OPERATION
S - 1 = move sign, 0 = do not move sign	ACTION
N - number of contiguous fields (1-31)	N-FIELDS
A - first field to be moved	OPERAND A
B - destination field	OPERAND B

MOVENB $A \rightarrow B_N$

0111 1000 10SN NNNN AAAA AAAA AAAA
BBBB BBBB BBBB

OBJECT	SOURCE COLUMN
Q - 30	OPERATION
S - 1 = move sign, 0 = do not move sign	ACTION
N - number of contiguous fields (1-31)	N-FIELDS
A - field to be moved	OPERAND A
B - first destination field	OPERAND B

MULT B x A → C

0111 00R0 AAAA AAAA AAAA BBBB BBBB BBBB
CCCC CCCC CCCC

OBJECT	SOURCE COLUMN
Q - 28	OPERATION
R - 1 = round, 0 = no round	ACTION
A - multiplier field	OPERAND A
B - multiplicand field	OPERAND B
C - putaway field	OPERAND C

MULTN B_N x $A_N \rightarrow B_N$

0110 01RN NNNN AAAA AAAA AAAA
BBBB BBBB BBBB

OBJECT	SOURCE COLUMN
Q - 25	OPERATION
R - 1 = round, 0 = no round	ACTION
N - number of contiguous fields (1-31)	N-FIELDS
A - first multiplier field	OPERAND A
B - first multiplicand field, first putaway field	OPERAND B

REDEF

1000 00NN NNNN NNNN AAAA AAAA AAAA
BBBB BBBB BBBB

OBJECT	SOURCE COLUMN
Q - 32	OPERATION
N - number of fields to be created (1-999)	N-FIELDS
A - first field in area to be redefined	OPERAND A
B - start of redefine area (packed headers)	OPERAND B

SAVERA

0101 1100

OBJECT	SOURCE COLUMN
Q - 23	OPERATION

RETURN

0110 0000

OBJECT	SOURCE COLUMN
Q - 24	OPERATION

SUB B - A → C

0001 0000 AAAA AAAA AAAA BBBB BBBB BBBB CCCC CCCC CCCC

OBJECT	SOURCE COLUMN
Q - 4	OPERATION
A - subtrahend field	OPERAND A
B - minuend field	OPERAND B
C - putaway field	OPERAND C

SUBN B_N - A_N → B_N

0001 110N NNNN AAAA AAAA AAAA BBBB BBBB BBBB

OBJECT	SOURCE COLUMN
Q - 7	OPERATION
N - number of contiguous fields (1-31)	N-FIELDS
A - first subtrahend field	OPERAND A
B - first minuend field, first putaway field	OPERAND B

SUBNA B - A_N → B

0001 010N NNNN AAAA AAAA AAAA BBBB BBBB BBBB

OBJECT	SOURCE COLUMN
Q - 5	OPERATION
N - number of contiguous fields (1-31)	N-FIELDS
A - first subtrahend field	OPERAND A
B - minuend field, putaway field	OPERAND B

SUBNB B_N - A → B_N

0001 100N NNNN AAAA AAAA AAAA BBBB BBBB BBBB

OBJECT	SOURCE COLUMN
Q - 6	OPERATION
N - number of contiguous fields (1-31)	N-FIELDS
A - subtrahend field	OPERAND A
B - first minuend field, first putaway field	OPERAND B

TBLIN B → $(A + C)_N$

1000 01NN NNNN AAAA AAAA AAAA BBBB BBBB BBBB CCCC CCCC CCCC

OBJECT	SOURCE COLUMN
Q - 33	OPERATION
N - number of contiguous fields (1-63)	N-FIELDS
A - base address of table	OPERAND A
B - first work area field	OPERAND B
C - location within the table of first table field (1-9999)	OPERAND C

TBLOUT $(A + C)_N$ → B_N

1000 10NN NNNN AAAA AAAA AAAA BBBB BBBB BBBB CCCC CCCC CCCC

OBJECT	SOURCE COLUMN
Q - 34	OPERATION
N - number of contiguous fields (1-63)	N-FIELDS
A - base address of table	OPERAND A
B - first work area field	OPERAND B
C - location within the table of first table field (1-9999)	OPERAND C

TBLADD B_N + $(A + C)_N$ → $(A + C)_N$

1000 11NN NNNN AAAA AAAA AAAA BBBB BBBB BBBB CCCC CCCC CCCC

OBJECT	SOURCE COLUMN
Q - 35	OPERATION
N - number of contiguous fields (1-63)	N-FIELDS
A - base address of table	OPERAND A
B - first work area field	OPERAND B
C - location within the table of first table field (1-9999)	OPERAND C

TYPE

0100 0100

OBJECT	SOURCE COLUMN
Q - 17	OPERATION

TYPEK

0100 0101

OBJECT	SOURCE COLUMN
Q - 17	OPERATION

TYPEL

0100 11NN NNNN NNNN

OBJECT	SOURCE COLUMN
Q - 19	OPERATION
N - maximum number of characters to be typed (1-9999)	N-FIELDS

TYPEM

0100 10NN NNNN AAAA AAAA AAAA

OBJECT	SOURCE COLUMN
Q - 18	OPERATION
N - number of contiguous fields to be typed into (1-63)	N-FIELDS
A - first putaway field	OPERAND A

WAIT

0000 0000

OBJECT	SOURCE COLUMN
Q - 0	OPERATION

CNTL (Q = 9)

The CNTL instruction calls on any one of a number of peripheral devices to perform a variety of functions. Some of the functions require a count, such as number of lines to space paper.

To avoid confusion, all configurations of the CNTL instruction are represented. The high-order six bits are always the Q code and the next five bits are the device code. The remaining nine bits follow no set pattern.

FORMS HANDLER

0010 0100 0110 0000 0000	EJECT LEFT
0010 0100 0110 0001 0000	EJECT RIGHT
0010 0100 0110 0010 0000	EJECT RIGHT REAR
0010 0100 0110 0011 0000	PLATEN OPEN BOTH
0010 0100 0110 0100 0000	FEED FORMS FIELD 1
0010 0100 0110 0101 0000	FEED FORMS FIELD 2
0010 0100 0110 0110 0000	FEED FORM HOME 1
0010 0100 0110 0111 0000	FEED FORM HOME 2
0010 0100 0110 1000 0000	PLATEN OPEN LEFT
0010 0100 0110 1001 0000	PLATEN OPEN RIGHT
0010 0100 0110 1010 0000	PLATEN CLOSE LEFT
0010 0100 0110 1011 0000	PLATEN CLOSE RIGHT
0010 0100 0110 1100 PPPP	PAPER FEED LEFT (1-15)
0010 0100 0110 1101 PPPP	PAPER FEED RIGHT (1-15)
0010 0100 0110 1111 PPPP	PAPER FEED BOTH (1-15)
0010 0100 0110 1110 0000	PLATEN CLOSE BOTH

SERIAL PRINTER

0010 0100 000c cccc cccc	PRINT POSITION (1-265)

LINE PRINTER

0010 0101 0001 1PPP PPPP	PRINT, SLEW (0-127)
0010 0101 0001 0PPP PPPP	SLEW (1-127)
0010 0101 0000 XXXX XXXX	PRINT POSITION (1-132)

PAPER TAPE READER

0010 0101 0010 0001 0000	REWIND TO LEADER
0010 0101 0010 0000 ffff	TRANSLATION TABLE

PAPER TAPE PUNCH

0010 0110 0000 10nn nnnn	PUNCH NULS (1-64)
0010 0110 0001 00nn nnnn	PUNCH DELETES (1-64)
0010 0110 0000 0000 ffff	TRANSLATION TABLE

COMMUNICATIONS

0010 0100 1110 0000 0000	LOAD ENABLE BUFFER
0010 0100 1110 0001 0000	RETRANSMIT BUFFER
0010 0100 1110 0010 0000	GET ENABLE BUFFER
0010 0100 1110 0100 0000	INHIBIT COMMUNICATIONS
0010 0100 1110 1000 0000	ENABLE INPUT
0010 0100 1110 0011 0000	ESTABLISH CONNECTION
0010 0100 1110 0101 0000	CHANGE ID1 ON NEXT PUT
0010 0100 1110 0110 0000	CHANGE ID2 ON NEXT PUT
0010 0100 1110 0111 0000	ENTER MULTI-BLOCK MODE
0010 0100 1110 1001 0000	ENTER SINGLE BLOCK MODE
0010 0100 1110 1100 0000	ENABLE HEADER LOAD
0010 0100 1110 1011 0000	RING INDICATOR MONITOR
0010 0100 1110 1010 0000	INPUT DATA TERMINATOR
0010 0100 1111 0001 0000	INSERT HT
0010 0100 1111 0010 0000	INSERT EM
0010 0100 1111 0011 0000	STX FIELD OPTION
0010 0100 1111 0100 0000	INSERT ESC TRANSMIT
0010 0100 1111 0101 0000	INSERT EOT/DLE-EOT
0010 0100 1111 0110 0000	SPACE SUPPRESS ENABLE
0010 0100 1111 0111 0000	SPACE SUPPRESS DISABLE
0010 0100 1110 1101 0000	ZERO SUPPRESS DISABLE
0010 0100 1110 1110 0000	ZERO SUPPRESS ENABLE
0010 0100 1110 1111 0000	SIGN DISABLE
0010 0100 1111 0000 0000	UNIT SEPARATOR DISABLE
0010 0100 1111 1000 0000	RECEIVE ESCAPE CHARACTER
0010 0100 1111 1001 0000	COMMUNICATIONS BELL

PROGRAM STATUS LIGHTS

0010 0100 1000 000L LLLL	SEQUENCE CODE (1-31)

CARD READER

0010 0101 0100 0BBB BBBB	FIELD SELECT (1-80)
0010 0101 0101 0000 0000	END OF TEXT/MESSAGE

CARD PUNCH

0010 0101 1110 0BBB BBBB	FIELD SELECT (1-80)
0010 0101 1111 0000 0000	PRINT LEADING ZEROS
0010 0101 1110 1000 0000	END OF TEXT
0010 0101 1111 1000 0000	END OF MESSAGE

CASSETTES

0010 0101 0110 0001 0000	REWIND CASSETTE 1
0010 0101 1000 0001 0000	REWIND CASSETTE 2
0010 0101 0110 0000 0000	BACKSPACE CASSETTE 1
0010 0101 1000 0000 0000	BACKSPACE CASSETTE 2
0010 0101 0110 0010 0000	WRITE TAPE MARK 1
0010 0101 1000 0010 0000	WRITE TAPE MARK 2
0010 0101 0110 0011 0000	SEPARATOR FLAG ENABLE CASSETTE 1
0010 0101 0110 0100 0000	SEPARATOR FLAG DISABLE CASSETTE 1

OBJECT	SOURCE COLUMN
Q - 9	OPERATION
Device codes (see table C-5, page C-30)	OPERAND A
P, c, X, and B	N-FIELDS
L	N-FIELDS
f (see table C-9, page C-31)	ACTION
All other bits	ACTION

CODE	DEVICE	WORKSHEET MNEMONIC
00000	Serial Printer	SPR
00010	Keyboards	KB
00011	Forms Handler	FH
00100	Program Status Lights	LITE
00101	Auto Line Find	LNCR or LNCL
00110	Disc	DISC
00111	Communications	COM
01000	Line Printer	LPR
01001	Paper Tape Reader	TR
01010	Card Reader	CR
01011	Cassette 1	CAS1
01100	Cassette 2	CAS2
01101	Magnetic Ledger	ML
01110	Ledger Card Feeder-Reader	MLFR
01111	Card Punch	CP
10000	Paper Tape Punch	TP

Table C-5 Device codes

CODE	WORKSHEET MNEMONIC	MEANING
Serial Printer		
000	ABN	All black, no symbol
001	ARN	All red, no symbol
010	PBC	Positive black, negative red with CR
011	PRC	Positive red, negative black with CR
100	PBD	Positive black, negative red with ◊
101	PRD	Positive red, negative black with ◊
Line Printer		
000	LPN	No sign print
001	LPS	Print sign

Table C-6 Primary printer edit codes

BINARY CODE	DECIMAL	MEANING	WORKSHEET MNEMONIC
0000000-0001001	0-9	Branch Key 0-9	BK00-BK09
0010000-0011001	16-25	Branch Key 10-19	BK10-BK19
0011110	30	Any Branch Key On	ANY
0011111	31	All Branch Keys Off	ALL
0100001	33	Communications GET-Enabled Buffer	GEB
0100010	34	Keyboard Buffer Full	KBF
0100011	35	Ledger Full Left	FLL
0100100	36	Ledger Full Right	FLR
0100101	37	Last Line CFF1	LL1
0100110	38	Last Line CFF2	LL2
0100111	39	Program Modify Key 1	P1
0101000	40	Program Modify Key 2	P2
0101001	41	Mag Ledger Read/Write Error	MLE
0101010	42	Communications-Enable Buffer Load	LEB
0101011	43	Cassette Read/Write Error	MCE
0101100	44	Communications Send Error	CSE
0101101	45	Communications Receive Error	CRE
0101110	46	Card Reader Buffer Full	RBF
0101111	47	Card Punch Buffer Available to PUT	PBA
0110000	48	Cassette EOT	CET
0110001	49	Cassette Tape Mark	CTM
0110010	50	Communications-Header Field	HDR
0110011	51	Communications-End of Text	ETX
0110100	52	Paper Tape Error	PTE
0110101	53	Communications-Line Establish	LES
0110110	54	Communications-Reverse Interrupt	RVI
0110111	55	Communications-Transmit Message Status	TMS
0111000	56	Communications-End of Transmission	EOT
0111001	57	Cassette Field Overflow	CFO
0111010	58	Cassette Blank Tape Read	CTB
0111110	62	Sign (Cassette)	SGN
0111111	63	Cassette Separator Bit 5	SB5
1000000	64	Cassette Separator Bit 6	SB6

Table C-7 Indicator codes

CODE	WORKSHEET MNEMONIC	MEANING
00	S	Zero suppress
01	$	Dollar protect
10	E	Dollar edit
11	☑	Absolute print with decimal point

Serial Printer Only

Table C-8 Secondary printer edit codes

f BITS	CODE
0000	ASCII
0001	Katakana
0010	315
0011	500
0100	046

Table C-9 Paper tape punch and reader translation table

CODE	STATUS
00100	Page Sentinel

Table C-10 Status codes

LINKING LOADER

The object user program cassette produced by the assembler is not an executable program; before it can be used with a 399 system, the procedure described in this text must be performed.

This procedure makes use of software called Linking Loader. After the unexecutable program is loaded into memory, various special-purpose software must be loaded; this software is resident on the Master Software Cassette (MSC).

Linking Loader determines what additional software is needed and accesses the MSC and loads the required software into memory, creating an executable object user program in memory. This program can be written to a scratch cassette, producing a ready-to-use program cassette.

START PROCEDURES

1. Processor switch ON.
2. Console switch ON.
3. Mount the unexecutable object program cassette on transport 1 and close the lid.

OPERATING PROCEDURES

1. Press Load. The ENA light turns ON following the load; press CLEAR.
2. Following a carriage return and line feed, PROG STAT lights 1, 4, and 16 and the ALPHA light turn ON, indicating that the program name and version number may be entered for verification.
3. If verification is not necessary, press RESUME and skip steps 4 and 5.
4. If verification is to be performed, enter the program name through the alpha keyboard. The program name can include up to eight characters if no version number is used, but a program name and its version number must total 10 characters when both are entered. Press RESUME following entry.
5. Following a carriage return and line feed, PROG STAT lights 1, 2, 8, and 16 turn ON; press RESUME.
6. When the user program is located on the cassette, the LOAD light turns ON; when the loading of the user program is complete, the cassette is rewound and the LOAD light turns OFF.
7. Following a carriage return and line feed, PROG STAT lights 2, 4, 8, and 16 and the ALPHA light turn ON, indicating that the unexecutable object program cassette is to be removed from transport 1 and replaced with the Master Software Cassette (MSC).
8. If DEBUG is to be included, enter D and press RESUME; if FIXER is to be included, enter F and press RESUME; if neither is to be included, press RESUME with no entry.

At this point, the CAS 1 and ALERT 2 lights turn ON, indicating that the executable user program in memory can be run or it can be copied to a scratch cassette to produce a ready-to-use program cassette.

To run the program, press RESET and then COMP.

To copy the program, replace the MSC on transport 1 with a scratch cassette and press COMP.

CAS 1 and ALERT 4 lights ON indicate load error; restart at step 3, START PROCEDURES.

INTERPRETIVE MODE OF OPERATION

Absolute programming of the M05 is difficult, making that format impractical for use in creating customer programs. To minimize programming complexity, user programs are written in the much simpler System 399 language. However, a program in that form cannot be performed by the M05; it must be converted into a format the M05 can recognize. A software program called the interpreter, written in M05 language,interprets the 399 object level user instructions, transforming them into a form the M05 can use. This method of operation is called the interpretive mode of operation.

The interpreter is made up of four types of routines. The executive routine controls the program flow and sets up the instructions; internal routines accomplish operations within the processor; peripheral driver (external) routines control the peripheral units; utility routines perform various commonly-used data-handling sequences for the internal and driver routines. Figure C-40 shows the flow of the interpreter.

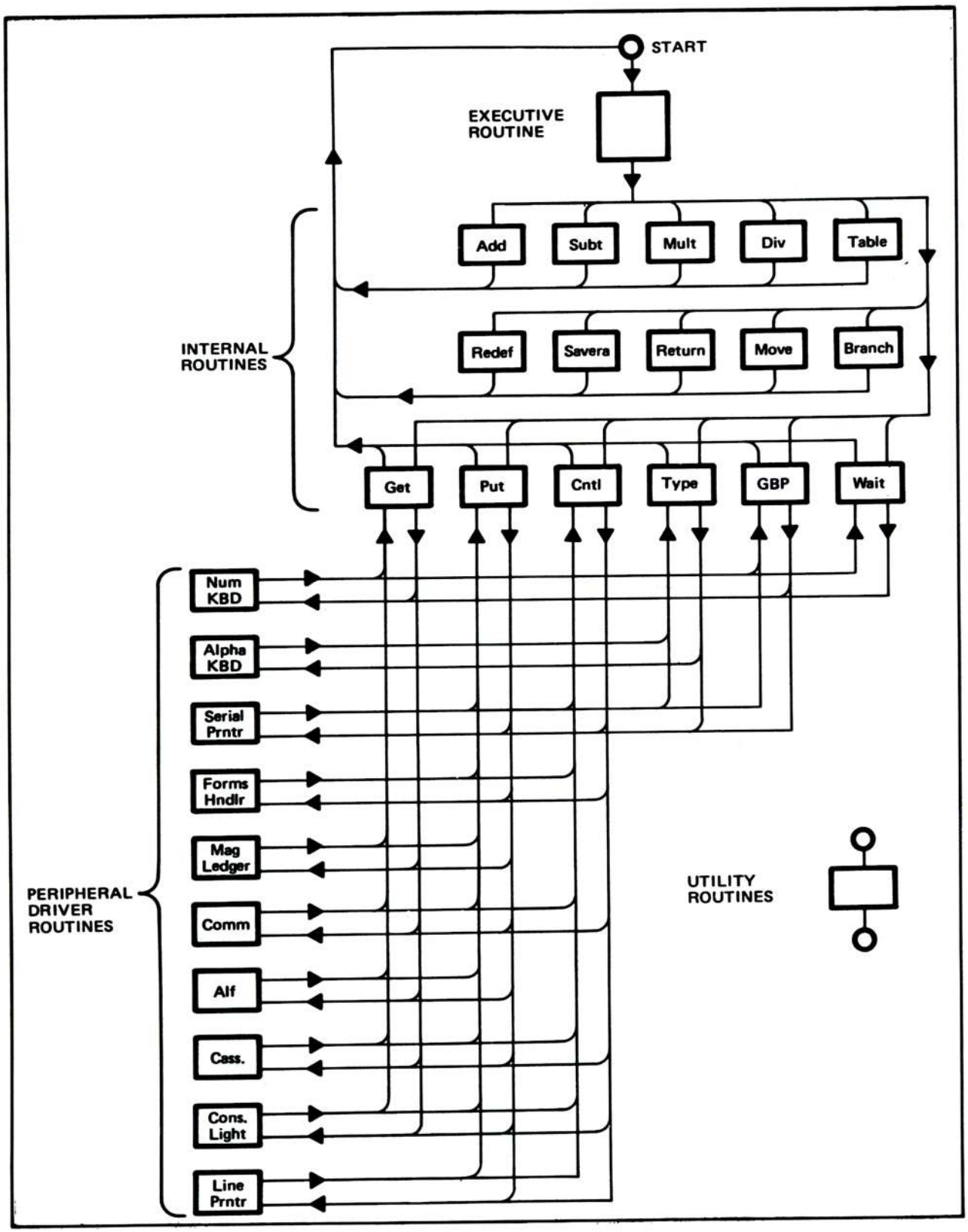

Fig. C-40 Interpreter flow

EXECUTIVE ROUTINE

The executive is performed once each user instruction. During the executive cycle, the user instruction is accessed in memory, taken apart, and loaded into M05 registers for use in the routine that executes the instruction.

The executive begins by extracting the first two digits from the user instruction in memory. The high-order six bits represent the Q code of the instruction. The Q code is used with the Instruction Decode and Transfer Table (table C-11) to determine the number of mode digits and operands in the instruction, as well as the address of the interpreter internal routine to be used.

INSTRUCTION	Q CODE	NUMBER OF MODE DIGITS / NUMBER OF OPERAND ADDRESSES / ADDRESS OF ROUTINE
WAIT	0	0000 XXXX XXXX XXXX
ADDNA	1	0110 XXXX XXXX XXXX
ADDNB	2	0110 XXXX XXXX XXXX
ADDN	3	0110 XXXX XXXX XXXX
SUB	4	0011 XXXX XXXX XXXX
SUBNA	5	0110 XXXX XXXX XXXX
SUBNB	6	0110 XXXX XXXX XXXX
SUBN	7	0110 XXXX XXXX XXXX
ADD	8	0011 XXXX XXXX XXXX
CNTL	9	1100 XXXX XXXX XXXX
BR	10	1100 XXXX XXXX XXXX
BRION	11	1001 XXXX XXXX XXXX
BRE	12	1110 XXXX XXXX XXXX
BRU	13	1110 XXXX XXXX XXXX
BRL	14	1110 XXXX XXXX XXXX
BRG	15	1110 XXXX XXXX XXXX
BRIOFF	16	1001 XXXX XXXX XXXX
TYPE/TYPEK	17	0000 XXXX XXXX XXXX
TYPEM	18	0101 XXXX XXXX XXXX
TYPEL	19	1000 XXXX XXXX XXXX
GET	20	1101 XXXX XXXX XXXX
PUT	21	1101 XXXX XXXX XXXX
GBP	22	1110 XXXX XXXX XXXX
SAVERA	23	0000 XXXX XXXX XXXX
RETURN	24	0000 XXXX XXXX XXXX
MULTN	25	0110 XXXX XXXX XXXX
DIV	26	0011 XXXX XXXX XXXX
DIVN	27	0110 XXXX XXXX XXXX
MULT	28	0011 XXXX XXXX XXXX
MOVENA	29	1010 XXXX XXXX XXXX
MOVENB	30	1010 XXXX XXXX XXXX
MOVE	31	1010 XXXX XXXX XXXX
REDEF	32	1010 XXXX XXXX XXXX
TBLIN	33	0111 XXXX XXXX XXXX
TBLOUT	34	0111 XXXX XXXX XXXX
TBLADD	35	0111 XXXX XXXX XXXX
FILL	36	1001 XXXX XXXX XXXX
BRS	37	1101 XXXX XXXX XXXX

Table C-11 Instruction decode and transfer table

The Instruction Decode and Transfer Table is part of the interpreter; it contains a 16-bit word for each System 399 instruction. Bits 1-12 specify the address in memory of the internal routine to be used; bits 13 and 14 specify the number of 3-digit operand addresses the instruction contains; and bits 15 and 16 specify the number of mode digits.

The proper word in the table is located by adding the Q code of the instruction to the table base. Thus, if the table base were address 1000, the word containing information concerning the ADD instruction, Q code of 8, would be in address 1008.

After the proper word in the table is located, the address of the routine is stored in a memory location to be used at the end of the executive cycle. The specified number of mode digits and operand addresses are taken from the user instruction and loaded into M05 registers. At this point, control is transferred to the internal routine whose address was taken from the table. Figure C-41 shows the general flow of the executive, and figure C-42 is a command by command detailed flow.

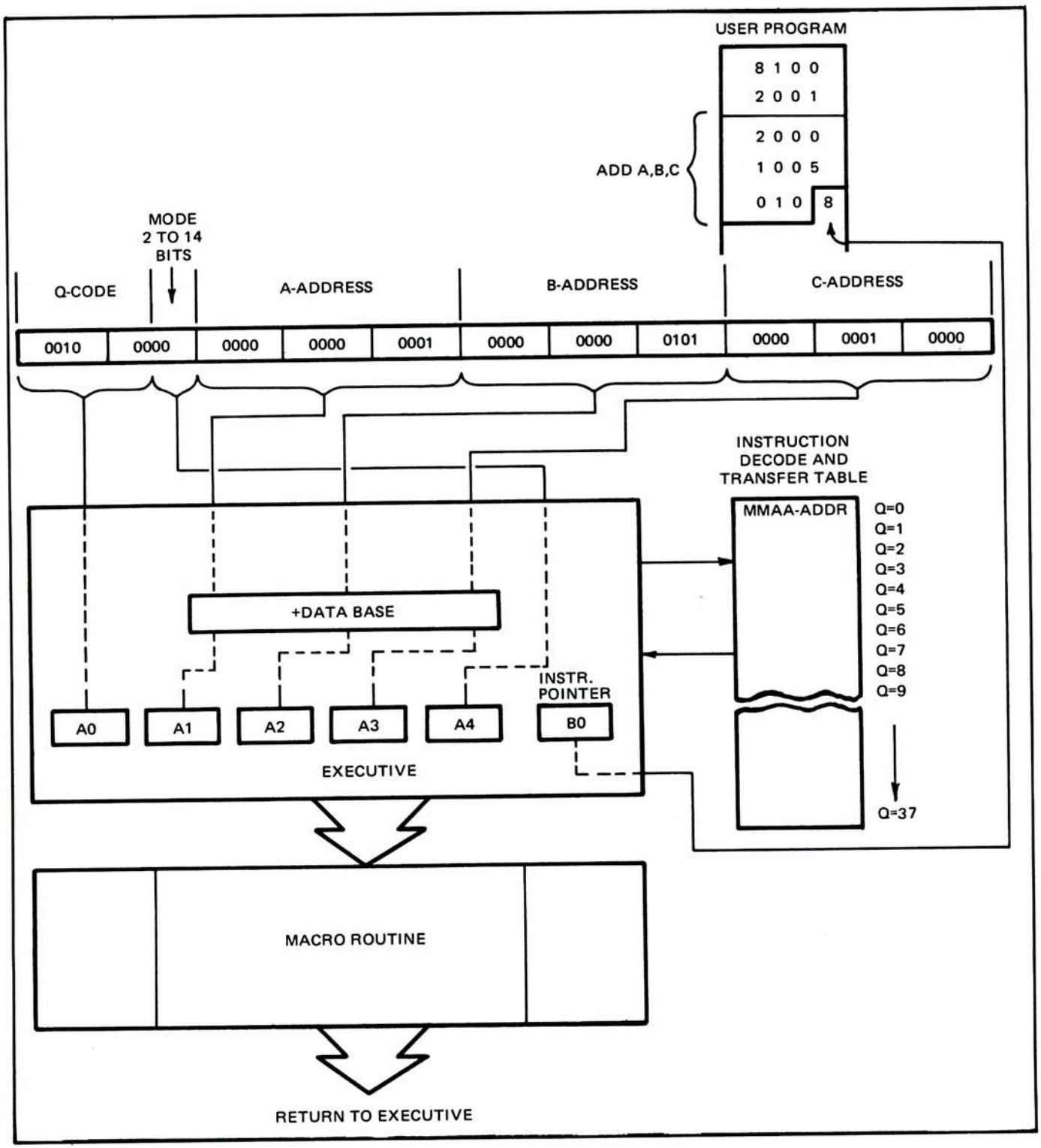

Fig. C-41 General executive flow

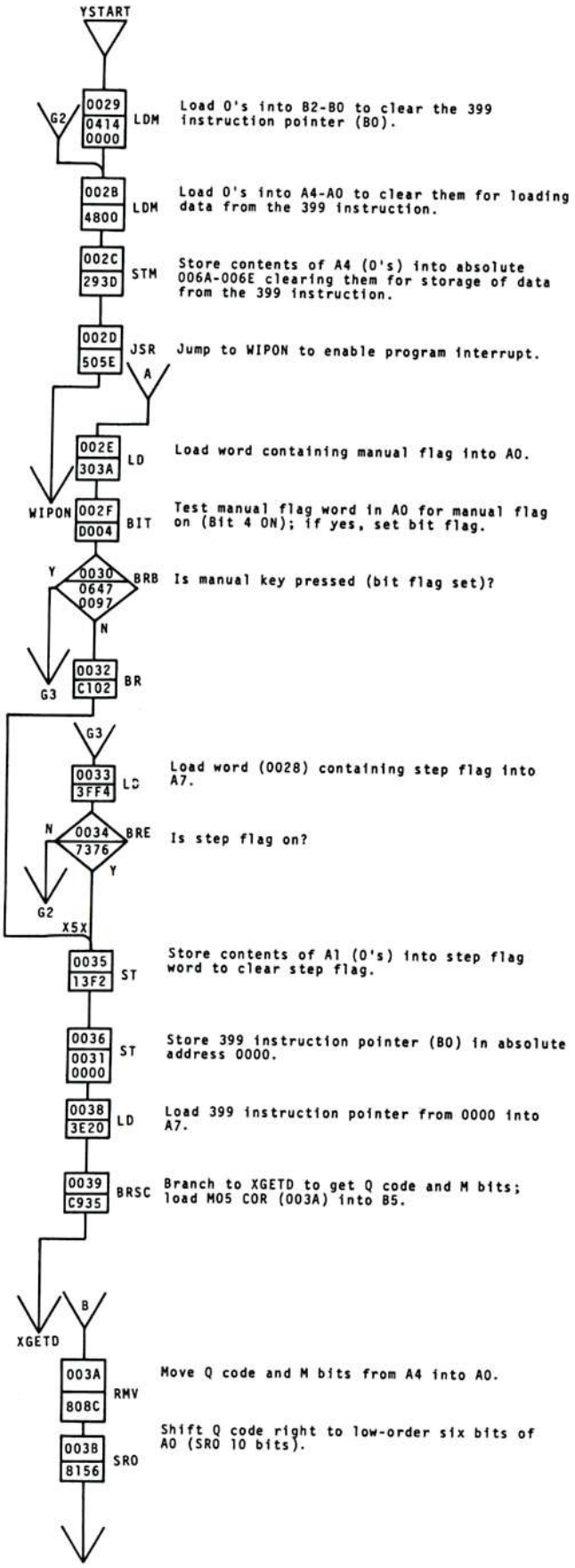

Fig. C-42 Detailed executive flow

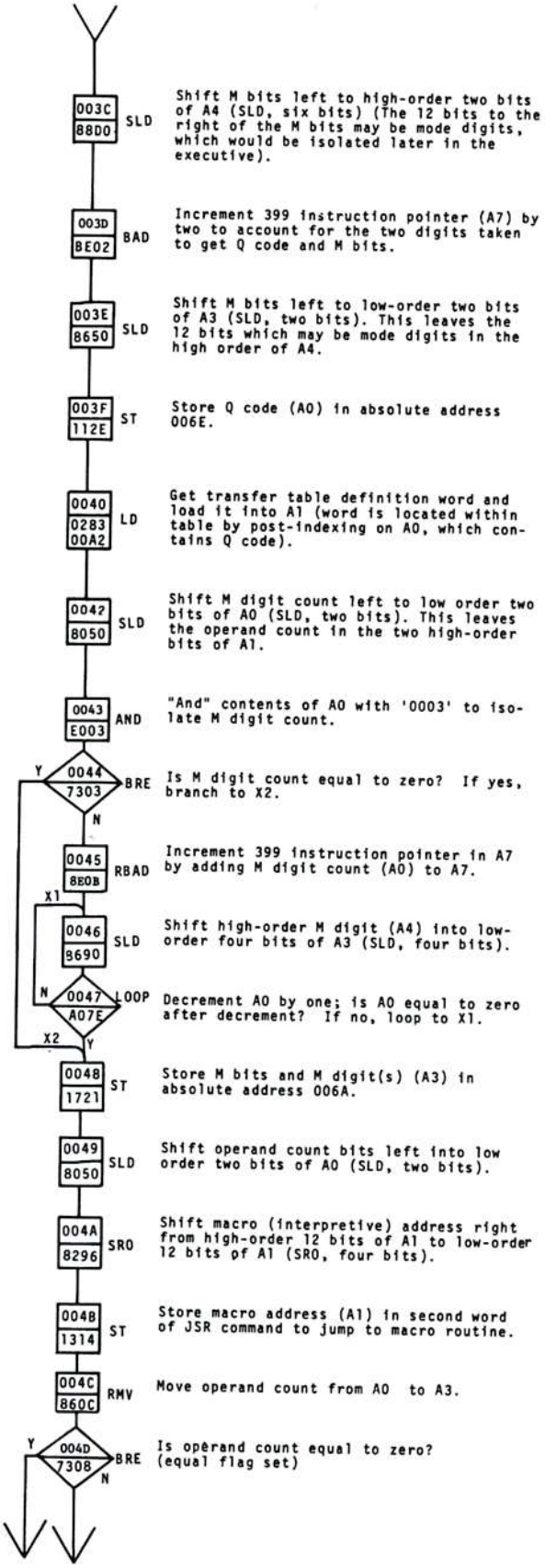

Fig. C-42 Continued

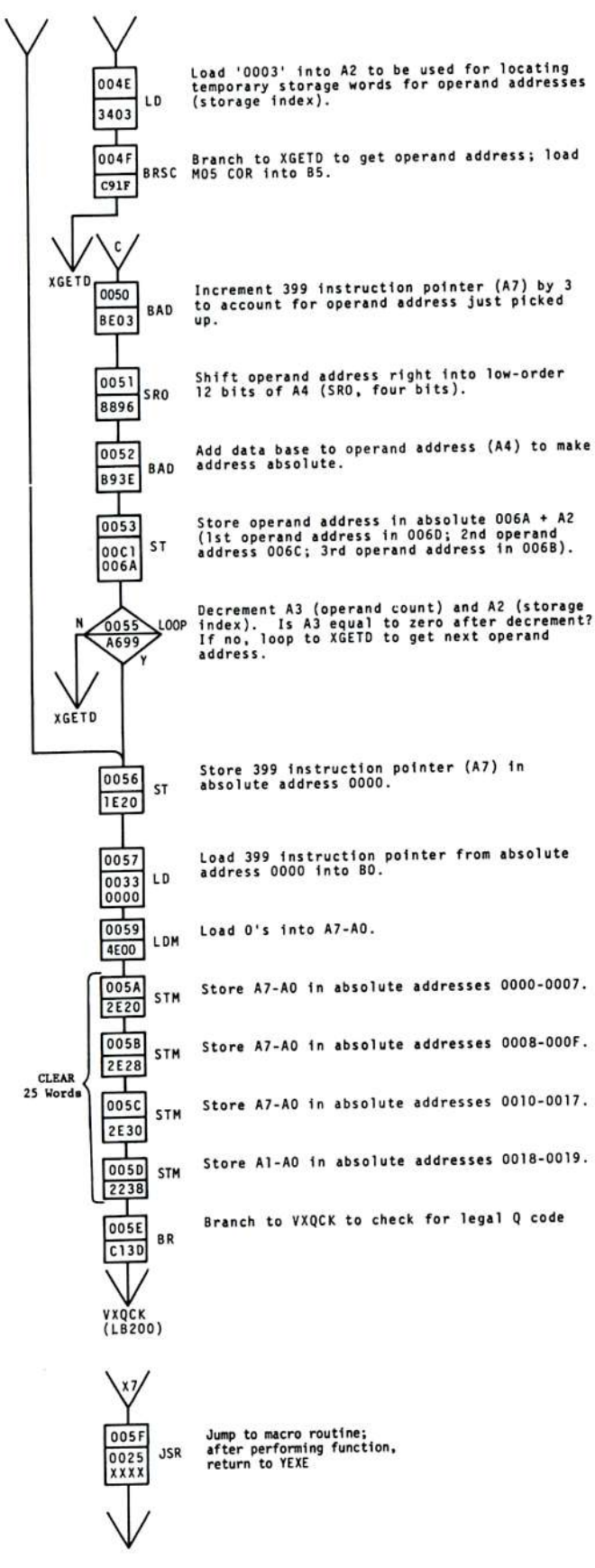

Fig. C-42 Continued

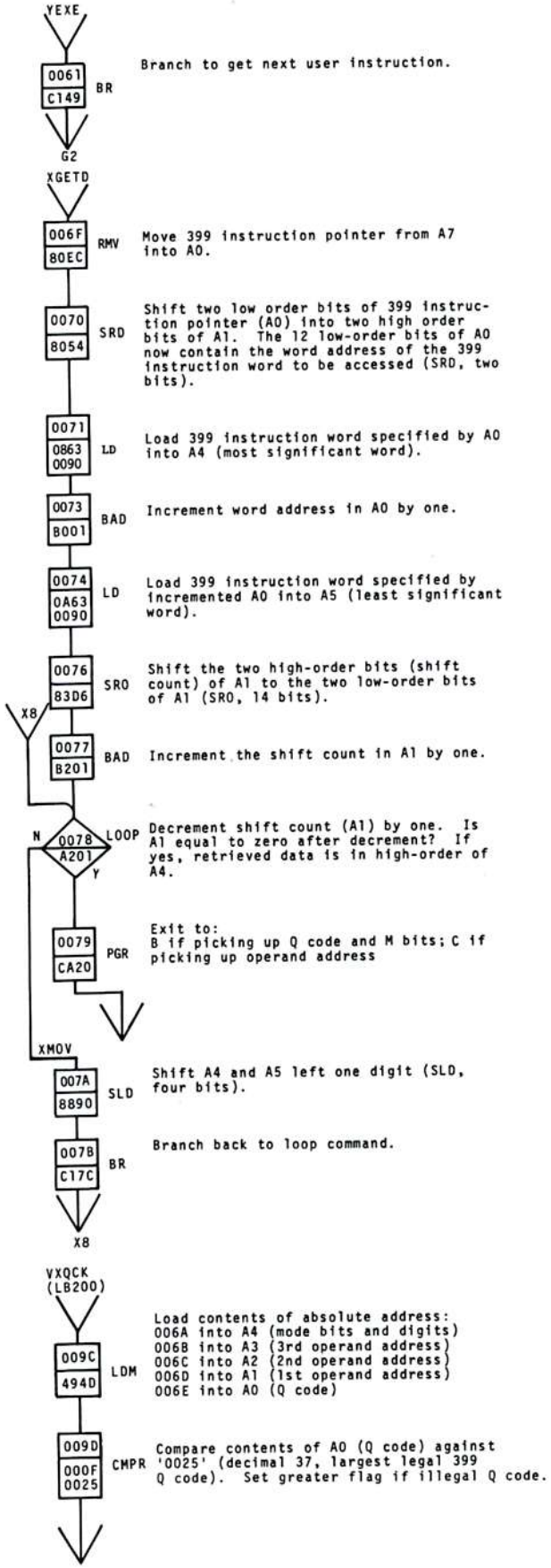

Fig. C-42 Continued

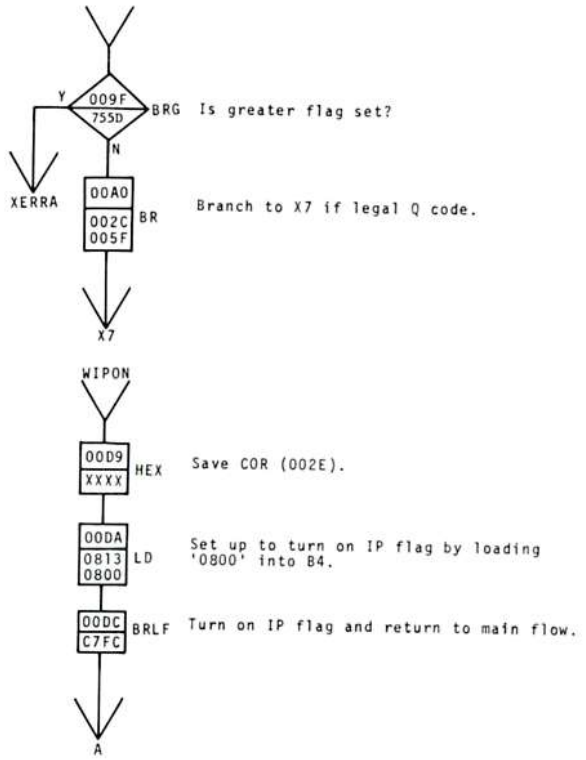

Fig. C-42 Continued

INTERNAL ROUTINES

An internal routine is provided for each 399 instruction, with similar instructions sharing a single routine. As an example, all the various add and subtract instructions (ADD, ADDN, ADDNA, ADDNB, SUB, SUBN, SUBNA, and SUBNB) use the same internal routine. The Q codes differentiate among the instructions, causing variations in the flow of the routine to accomplish the instruction.

The user instructions dealing with a peripheral device are GET, PUT, GBP, TYPE, WAIT, and CNTL. Each of these has an internal routine of the same name, which performs preliminary data-handling and control functions before passing control to the appropriate peripheral driver routine. The driver routine is selected through the use of a table of addresses and a device code, in the same way the internal driver is selected by the Q code and the Instruction Decode and Transfer Table.

PERIPHERAL DRIVER ROUTINES

A peripheral driver routine is provided for each integrated or free-standing peripheral device. The driver routine handles the data or control being transmitted between the processor and the peripheral involved.

UTILITY ROUTINES

Many minor routines are used repeatedly in the course of performing a user program. For obvious reasons, these routines are put together once and assigned locations within the interpreter portion of memory. When one of these routines is needed, the internal or peripheral driver routine requiring it branches to the utility routine; after the utility routine completes its function, control is returned to the calling routine, and the program continues.

CONSOLE KEYS

Certain keys and indicators on the console (fig. C-43) affect the loading and executing of the user program. These keys and their functions are the subject of the following text.

RESUME

This key affects the typing instructions as follows.

Type — The resume bar terminates this instruction.

Type Limited — The resume bar terminates this instruction even if the specified number of characters has not been typed.

Type To Memory — The resume bar terminates each field of this instruction even if all positions have not been filled. Upon termination, all unfilled character positions are space filled. When used with the last field, the instruction is terminated.

The resume bar also terminates a user WAIT instruction.

TYPE

This key sets a flag bit in the M05 memory. Its status is tested in a subsequent TYPEK instruction. If the flag is ON, the alphanumeric keyboard is enabled and a TYPE instruction is in effect. If the flag is OFF, the next user instruction is executed. The type key can be pressed at any time and its flag bit remains ON until a TYPEK instruction is executed.

Fig. C-43 Console keys and indicators

COMP (Compute Key)

This key causes the M05 to execute the current M05 instruction and step to the next M05 instruction. The M05 program execution may be interrupted by an M05 WAIT instruction or by the depression of the Halt key. While in the halt mode, each depression of the Compute key causes the next M05 instruction to be executed. On a serial printer print error, depression of this key conditions the system to reposition the printer carrier to reprint the entire field and reset the indicator. On an M05 hard halt condition, memory parity, or address error, the Compute key must be pressed after the Reset key to reset the user program to the starting instruction.

HALT

This key halts the execution of the M05 instructions after the present one is completed. The Halt Indicator Lamp is ON while in this mode and can be reset by a second depression of the Halt key or by depressing the Reset key. On an M05 hard halt condition, the Halt key must be pressed before the Reset key.

LOAD

This key generates a master reset pulse that initializes the M05 and all peripherals to a program load mode, so that a program from the Program Tape Cassette may be entered into M05 memory. The Load Indicator Lamp is ON while the program is being loaded. This key also turns the ENA Lamp ON.

MAN (Manual Key)

This key interrupts the user program and allows the operator to step through the user program one instruction at a time with each depression of the Step key. The manual mode is maintained until the Manual key is pressed a second time. The Manual Indicator Lamp is ON while in the manual mode.

STEP

This key allows the operator to step through the user program one instruction at a time while the 399 is in a manual mode.

RESET

This key is used in conjunction with the Halt key to reset the 399 from an M05 memory error or a power interruption. It leaves the 399 in a rest state, with the ENA lamp ON, the trunk deselected, and all M05 flags reset. The Compute key resets the user program to the starting position. The sequence of key depressions to recover from an M05 hard halt condition is: (1) Halt; (2) Reset; (3) compute.

START

This key interrupts the user program and resets it to the starting instruction with the ENA lamp ON.

. (Decimal Key)

This key emits a special character for decimal point location during a numeric entry. When the data is transferred to the user data field, the decimal location specified during the keyboard entry is aligned on the decimal location specified in the field header. If more digits are entered to the right of the decimal than are available in the user data field, an ENA condition results. If fewer spaces to the right of the decimal are used, the remaining spaces are zero filled.

R (Reverse Key)

This key sets a flag bit in the M05 memory. When data is tranferred from the keyboard buffer to the user data field, this flag controls the setting of the sign bit in the user data field header to indicate a negative field. If the flag bit is set ON, the sign bit is set negative. This key is active when the numeric keyboard is enabled and may be used any time before the Enter Bar or a branch key is depressed.

CLEAR

This key has two functions.

1. On a numeric entry, when used before the Enter Bar or a branch key, it clears the keyboard buffer and turns OFF the reverse flag. This allows the operator to recover if a wrong digit was entered. This key is disabled from the time the Enter Bar or a branch key is pressed, until the data is transferred to the user data field.

2. This key clears the ENA condition from either the numeric or the alphanumeric keyboard and causes the system, via an interpreter error correcting routine, to revert to a condition whereby a re-entry can be made.

ENTER BAR

When this key is used on a contiguous field instruction, on other than the last field, it signifies the end of that field. When used on a single field instruction or at the last field of a contiguous instruction, it signifies termination of the instruction. This key is active only when the numeric keyboard is enabled.

P1

This key sets a flag bit in the M05 memory and turns ON the P1 indicator lamp. This condition remains until a second depression of the P1 key. The flag bit is tested by a Branch on Indicator instruction in the user program to branch to a program option.

P2

This key functions the same as the P1 Key, except a different flag and indicator is set.

0-9 (Branch Keys)

These keys give 10 branch options to terminate a keyboard instruction, either numeric or alphanumeric, or a user Wait instruction. If the GET:Keyboard instruction calls for contiguous fields, a branch key terminates the instruction and zero or space fills any unused fields.

Each branch key also sets a flag bit in the M05 memory. The flag bits are tested later in the user program with a BRION or BRIOFF instruction. Depending on the status of the flag, the program either executes the next instruction or branches to another instruction. These flags are reset by the

next keyboard instruction.

10's BR (Tens Branch Key)

This key is used with branch keys 0-9 to produce branch options 10-19. This key may be pressed any time before pressing a branch key 0-9. The tens branch indicator lamp indicates the ten's branch key has been pressed. The depression of the Clear key, the Enter Bar, the Resume Bar, or any branch key resets the ten's branch indicator lamp.

SUMMARY

Source level 399 programs are written in mnemonics by the programmer on three types of worksheets. The Assembler Specification Worksheet entries establish the parameters for assembly. The Data Layout Worksheet contains the information needed by the assembler to produce the data definitions used in the program. The Coding Worksheet contains the user instruction statements, which the assembler transforms into object level instructions.

The data entered on the previously mentioned worksheets is transferred to punched cards or paper tape in a definite sequence. The Assembler Specification Worksheet, page and line number 000 000,is first; then comes the Data Layout Worksheets with page and line numbers in ascending order; the Coding Worksheets, with page and line numbers in ascending order, follow. The resulting deck or reel is the complete source level user program, ready for assembly. The source level program is input to the assembler program, which produces the object level program and, optionally, a printout of the program. At the object level, the program is represented by bit strings, which are encoded serially on a magnetic cassette.

The data fields are encoded on the cassette in ASCII for alphanumeric fields and in packed ASCII, using only the low order four bits, for numeric fields. Each data field is defined by a 16-bit header, which immediately precedes the field it defines; the header defines such things as field length, decimal position, and sign.

The bit string of a user instruction consists of the Q code and mode bits and in most instructions, mode digits and/or operand addresses. The Q code identifies the instruction, the mode bits and mode digits modify the command, and the operand addresses specify the data fields and/or user instruction to be used in the execution of the instruction. The program in this form is loaded into the processor memory.

The 399 object level language is not the machine language of the M05 processor. To make use of the 399 language, the interpretive mode of operation is employed.

A group of routines, in M05 language, is used to execute the user program; this package of routines is called an interpreter, and it is stored in M05 memory with the user program. The executive routine controls the program flow and sets up the user instruction for execution. One of several internal routines then takes over and executes the instruction, unless a peripheral is involved.

When a peripheral is to be dealt with, the internal routine performs preliminary functions and then transfers control to a peripheral driver routine, which handles the transfer of data.

After the instruction is accomplished, control is returned to the executive, which sets up the next user instruction.

UTILITY ROUTINES

DEBUG/FIXER

Debug and Fixer are mutually exclusive optional software tools provided to aid in debugging user programs. The routines are activated by pressing the MANUAL key, then indexing a legal two-digit function code on the numeric keyboard and pressing ENTER. Any GET, PUT, TYPE, or WAIT instruction must be terminated before Debug or Fixer can be activated.

The difference between the two routines is functional capabilities; more functions are available using Fixer. Table C-12 lists the functions that can be performed by Debug, and table C-13 lists functions available with Fixer.

DEBUG

CODE FUNCTION	FUNCTION
11	Print sequence control register
12	Print user instruction
13	Print user data
14	Print last branch return address
53	Replace numeric data
61	Load sequence control register

Table C-12 Debug function codes

Any entries other than those listed in table C-12 are illegal; if entered, the illegal code is printed and ignored.

FUNCTION CODE 11

The contents of register B0 are printed in hexadecimal, starting at column 7. Register B0 contains the address of the next user instruction.

FUNCTION CODE 12

After the function code is printed and the print ball is positioned to column 7, the alpha keyboard is enabled, and the 4-digit hexadecimal address of the desired instruction must be typed. When this address entry is terminated with the Resume Bar, 12 instruction digits are printed, starting at column 15. The first digit printed represents the contents of the instruction address entered.

FUNCTION CODE 13

After the function code is printed and the print ball is positioned to column 7, the alpha keyboard is enabled, and the 4-digit hexadecimal address of the desired user data field must be typed. When this address entry is terminated with the Resume Bar, the header and print position, if any, of the specified field are printed, starting at column 15. The data is printed, starting at column 23. If the field is numeric, the entire field is printed out.

If the field is alphanumeric, only the first 24 characters are

printed. The next 12 characters can be printed with a second function code 13 by incrementing the address entry by 13. If more characters are needed, another function code 13, with the address entry incremented by 12, must be used. This procedure is repeated, in increments of 12, until the entire field has been printed.

A packed REDEF area should be unpacked before an attempt is made to examine the headers.

FUNCTION CODE 14

The contents of YBRLNK are printed, starting at column 7. YBRLNK contains the address of the user instruction immediately following the last branch taken.

FUNCTION CODE 53

After the function code is printed and the print ball is positioned to column 7, the alpha keyboard is enabled, and the 4-digit hexadecimal address of the desired user data must be typed.

After the address is typed, the data in that field can be changed by typing one space and up to 16 numeric characters before pressing the Resume Bar. This function can only be used with numeric data fields.

FUNCTION CODE 61

After the function code is printed and the print ball is positioned to column 7, the alpha keyboard is enabled. The address of the instruction desired is typed. The entry is terminated with the Resume Bar, and the Step key must be pressed to execute the instruction.

Error Conditions

If an address greater than 7FFF is entered, the system halts in an M05 WAIT with ALERT 1, 2, and 16 lights ON. The function can be re-entered after the Compute key is pressed.

If an alphanumeric data field address is specified in a 53 function (replace numeric data), ALERT 1, 2, 4, and 16 lights are set ON. The Compute key must be pressed to recover.

If an address greater than the available data area is entered, the system halts with the Memory Address indicator ON. The Halt, Reset, and Compute keys must be pressed in sequence to return to the start of the user program.

FIXER

FUNCTION CODE	FUNCTION
00	Repeat print
11	Print sequence control register
12	Print user instruction
13	Print user data
14	Print last branch return address
52	Replace instruction
53	Replace data
61	Load sequence control register
72	Dump user instructions
73	Dump user data

Table C-13 Fixer function codes

FUNCTION CODE 00

Following execution of function code 12, 13, 72, or 73, an entry of 00 and depression of Enter results in the printing of the next 399 instruction or data field and its address. Repeated depression of Enter prints contiguous data.

FUNCTION CODE 11

Same as Debug.

FUNCTION CODE 12

Same as Debug, with the following exceptions:

1. Only the digits containing the instruction are printed, rather than a fixed 12 digits.
2. Following the printing of the instruction specified, depression of the Enter Bar causes the next instruction and its address to be printed; this may be done repeatedly.
3. A display of hexadecimal 004C on TRUNK 8 through MEDIA lights indicates an addressing error or the printing of the last instruction in memory.

FUNCTION CODE 13

Same as Debug, with the following exceptions:

1. When printing an alpha field, a maximum of 25, rather than 24, characters are printed on each line until the entire field has been printed. It is not necessary to re-enter the function code for each continuation line.
2. Following the printing of the specified data field, depression of the Enter Bar causes the next header and data field to be printed; this may be done repeatedly.
3. If an illegal data header is accessed by entering an incorrect address or by calling for print past the end of data, a line of 25 hexadecimal characters is printed. Print past the end of data also sets hexadecimal 004C on TRUNK 8 through MEDIA lights.

FUNCTION CODE 14

Same as Debug.

FUNCTION CODE 52

After the function code is printed and the print ball is positioned to column 7, the alpha keyboard is enabled, and the 4-digit hexadecimal address of the instruction data to be changed must be typed; the specified address does have to be the beginning of the instruction.

After the address is typed, the contents of the specified instruction can be changed by typing one space and up to 20 hexadecimal characters before pressing the Resume Bar.

If the function is terminated by a Resume Bar depression without entering data after the address, no memory is changed.

If the specified address is beyond the instruction area or if the entered data exceeds the instruction area, hexadecimal 004C is displayed on TRUNK 8 through MEDIA lights.

FUNCTION CODE 53

Same as Debug, with the following exceptions:

1. Either alpha or numeric data may be changed.
2. Twenty hexadecimal characters may be typed.
3. If the specified address is beyond the user data area or if the replaced data exceeds the user data area, hexadecimal 004C is displayed on TRUNK 8 through MEDIA lights.

FUNCTION CODE 72

After the function code is printed and the print ball is positioned to column 7, the alpha keyboard is enabled. The starting address for the program dump is entered on the alpha keyboard, followed by depression of the Resume Bar.

The program is printed by the serial printer, one instruction and its address per line. The print is terminated in one of two ways.

1. Pressing the Enter Bar stops the print. Additional Enter Bar depressions cause instructions to be printed one at a time, in sequence.
2. When the last instruction has been printed, printing stops and hexadecimal 004C is displayed by TRUNK 8 through MEDIA lights, signifying the end of the dump.

FUNCTION CODE 73

After the function code is printed and the print ball is positioned to column 7, the alpha keyboard is enabled. The starting address for the user data dump is entered on the alpha keyboard, followed by depression of the Resume Bar.

The user data is printed by the serial printer, one data field, with header, and its address per line, unless the data value is larger than 25 characters. In that case, multiple lines are used. The print is terminated in one of three ways.

1. Pressing the Enter Bar stops the print. Additional Enter Bar depressions cause data fields to be printed one at a time, in sequence.
2. If an illegal header is encountered, 25 hexadecimal digits are printed and the print stops.
3. When the last data field has been printed, printing stops and hexadecimal 004C is displayed by TRUNK 8 through MEDIA lights, signifying the end of the dump.

If a REDEF header area is encountered, the data printout is not predictable. The dump must be restarted at the next legal address following the REDEF area to maintain sync.

CASSETTE PRINT (CSTPRNT)

GENERAL

Cassette Print provides a method of printing all or selected portions of a program cassette, Master Software Cassette (MSC), Site Software Cassette (SSC), or user-created data cassette. The contents of the cassette are printed in both hexadecimal and decoded ASCII.

The operator can specify that all data on the cassette be printed (max. 2048 bytes) or specify the cassette block at which printing is to begin and the block at which it is to end.

START PROCEDURES

1. Processor switch ON.
2. Console switch ON.
3. Place MSC on transport 1 and close lid.

LOAD PROCEDURES

1. Press Load to load Linking Loader.
2. When ENA light and PROG STAT lights 1, 4, and 16 are ON, press Clear.
3. When ALPHA light and PROG STAT lights 1, 4, and 16 are ON, enter CSTPRNT through alpha keyboard.
4. Press Resume to load Cassette Print routine.
5. When PROG STAT lights 1, 2, 8, and 16 are ON, press Resume. At this point, the NUMERIC light comes ON, and the serial printer prints LOAD CASSETTE AND ENTER BLK NUMBERS.

INPUT PROCEDURES

1. If a beginning block number is to be specified, enter the block number (4 digits) through the numeric keyboard; the serial printer prints the entry. If the entire cassette is to be printed, this entry is unnecessary (see step 4).
2. If an ending block number is to be specified, enter the block number (4 digits) through the numeric keyboard; the serial printer prints the entry. If the print is to be to the end of the cassette, this entry is not necessary.
3. If an incorrect block number entry is made, press Clear and make entries again.
4. After block entries made are found to be correct, or if the entire cassette is to be printed, press Enter.

The serial printer begins printing the contents of the cassette.

A block header is printed at the beginning of each block of data. BLK NUM specifies the 4-digit number representing the ascending sequential position of the block, relative to 0001. BLK SIZE specifies the 4-digit number representing the number of data words in the block. REL LOC specifies

the 4-digit number representing the location of the first word of data of each line, relative to 0000.

The cassette data in hexadecimal form is printed below the numbers 0 through 9; the characters to the right on each line are the same data in decoded ASCII form. Unprintable ASCII characters are represented as spaces in the printout; this includes lower-case letters and any hex configuration with the high-order 8 bit ON.

OUTPUT PROCEDURES

Certain options may be exercised by the operator after the output begins. An end-of-program WAIT is indicated by PROG STAT light 1 ON.

1. The routine may be allowed to complete its run, at the end of which an end-of-program WAIT is entered.
2. The printing of any block may be terminated by pressing BK01; the line being printed is completed, and the next block started.
3. All printing may be terminated by pressing BK02; output ceases immediately, and the program enters an end-of-program WAIT.
4. When in an end-of-program WAIT, the program may be restarted by pressing Compute.

RESTRICTIONS

The maximum data block size is 2048 bytes; of a block exceeding this size, only the first 2048 bytes are printed.

This routine does not terminate after reading two contiguous tape marks, but the operator can terminate the routine when two consecutive blocks of zero length are recognized.

Cassette hardware cannot differentiate between a recorded tape mark and 19 inches of blank tape; if either is encountered, a block of zero length is output.

A line containing all zeroes is not printed, unless it is the first line of a block.

If the EOT hole is encountered during a read operation, the block being read is printed; the routine then goes to an end-of-program WAIT.

ERROR CONDITIONS

CAS1 light ON, ALERT lights 1 and 2 ON, and MEDIA light ON indicates that cassette 1 is not ready. Mount the cassette on transport 1, close the lid, and press Compute.

CAS1 light ON, DATA light ON, and WAIT light ON indicates that there has been an unrecoverable cassette read error. Press Reset, then Compute to go to an end-of-program WAIT.

PRT light ON and DATA light ON indicates that the last character output by the serial printer is in error. Mark the erroneous printed character and press Compute to continue printing.

CASSETTE-TO-CASSETTE COPY (CCOPY)

GENERAL

This utility provides a means of copying the contents of one cassette onto another. After two cassette marks have been read from the originating cassette and written to the destination cassette, the program goes to an end-of-program WAIT (PROG STAT light 1 ON). The program also provides for verification of correct copy.

START PROCEDURE

1. Processor switch ON.
2. Console switch ON.
3. Place MSC on transport 1 and close lid.
4. Place blank cassette with write tab enabled on transport 2 and close lid.

LOAD PROCEDURE

1. Press Load to load Linking Loader.
2. When ENA light and PROG STAT lights 1, 4, and 16 are ON, press Clear.
3. When ALPHA light and PROG STAT lights 1, 4, and 16 are ON, enter CCOPY through alpha keyboard.
4. Press Resume to load Cassette Copy routine.
5. When PROG STAT lights 1, 2, 8, and 16 are ON, press Resume.

INPUT PROCEDURE

1. When PROG STAT lights 1, 8, and 16 are ON:
   a. If the copy option is to be performed, press P1.
   b. If the verification option is to be performed, press P2.
2. If the wrong key has been pressed, press Reset and press correct key.
3. After correct key has been pressed, press Compute.
4. When CAS1 light, CAS2 light, and PROG STAT lights 1 and 2 are ON, mount the source cassette on transport 1 and the destination cassette (write tab enabled) on transport 2. Press Compute to start the program.
5. PROG STAT light 1 ON indicates an end-of-program WAIT;the program is complete, and both cassettes have been rewound.
6. To restart the program, press Reset and then Compute.

RESTRICTIONS

The copy and verification options are mutually exclusive; only one of the two may be performed at a time.

Maximum data block size for this routine is 2048 bytes. If a block exceeds 2048 bytes, only the first 2048 are written to the destination cassette.

The routine goes to an end-of-program WAIT after having read and written two tape marks. Cassette hardware cannot differentiate between a tape mark and 19 inches of blank tape; two tape marks are written to the destination tape, even though the data on the source cassette is not terminated by a tape mark.

Cassette Copy recognizes only two tape marks as end-of-data, so files or labels followed by a single tape mark are copied to the destination cassette as they were read from the source cassette.

If an EOT hole is detected in either cassette prior to completion of the copy, the routine halts in a mandatory abort condition.

ERROR CONDITIONS (COPY)

CAS1 or CAS2 light ON and ALERT light 1 ON indicates that clear trailer has been detected on cassette 1 or 2. The program must be aborted.

CAS1 or CAS2 light ON, ALERT lights 1 and 2 ON, and MEDIA light ON indicates that transport 1 is not ready to read, or that transport 2 is not ready to write. Prepare the unit for operation and press Compute.

CAS1 or CAS2 light ON, ALERT lights 1, 2, and 4 ON, and MEDIA light ON indicates that an attempt has been made to write to the cassette on transport 1 or 2, but the write tab has not been enabled. Enable the write tab and press Compute.

CAS1 or CAS2 light ON, ALERT light 16 ON, and MEDIA light ON indicates that the EOT hole has been detected on cassette 1 or 2. The run must be aborted.

CAS1 or CAS2 light ON, ALERT light 16 ON, and DATA light ON indicate that blank tape has been read from cassette 1 or 2. The run must be aborted.

CAS1 or CAS2 light ON and ALERT lights 1, 2, 4, 8, and 16 ON indicate that the lid to transport 1 or 2 was raised during operation. The run must be aborted.

CAS1 or CAS2 light ON, DATA light ON, and WAIT light ON indicates that an unrecoverable read error or an uncorrectable write error has occurred on cassette 1 or 2. In the case of a read error, the run must be aborted; for write error, change cassette and press Reset and then Compute.

ERROR CONDITIONS (VERIFY)

PROG STAT lights 1, 2, and 4 ON indicate that blocks of unequal length have been detected between the source and destination cassettes. Sixteen lights (TRUNK 8 through MEDIA) display the block number in BCD (four digits). Press Compute to go to end-of-program WAIT.

PROG STAT lights 1, 2, and 4 ON indicates that a character on the destination cassette does not match its counterpart on the source cassette. Sixteen lights (TRUNK 8 through MEDIA) display the block number in BCD; press Compute. The 16 lights display the relative location, within the block, of the word containing the error; this location is displayed in hexadecimal. Press Compute to go to end-of-program WAIT.

399 SOURCE/OBJECT PAPER TAPE TO CASSETTE (PTCT)

GENERAL

When a 399 program is assembled on an NCR Century system, it is necessary to convert the assembled program to cassette from the NCR Century output. The common media between the two systems is punched paper tape. The assembler output may be either source level (S) or object level (O); the tape format for each follows.

SOURCE

The source program is output from the assembler onto tape in 80-column card format. The first 80 frames contain the program control, as described by the Assembler Specification Worksheet.

Following program control are the data definitions and the 399 instructions, each one of which occupies 80 frames. If the data definitions and instructions total more than 500, a second reel must be used (500 is maximum on a 399 tape reader reel).

The program name and type are the first punches on each tape reel, followed by leader. Immediately following the leader, an S is punched to identify the program format as source; the reel number follows the S. The end of each reel has an 80-character control record, followed by END$.

OBJECT

The object program is output from the assembler onto tape in the following format.

The first 44 bytes (eight bits per byte) make up a header containing descriptive information pertinent to the program. The first 10 bytes of the 44 make up the program name.

Following the header is leader, followed by an O to identify the program format as object. After the O comes the unpacked hexadecimal object program.

OPERATIONAL PROCEDURE

1. Mount the paper tape on the 399 reader with the read head positioned on the leader.
2. Mount the Master Software Cassette (MSC) on cassette transport 1.
3. Press Load to read the Linking Loader into memory. When this has successfully been done, a hexadecimal wait code of 15 is displayed on the program status lights on the console; if this wait code does not appear, the procedure must be aborted.
4. Type program name PTCT and press Resume. When hexadecimal wait code 1B appears on the program

status lights, press Resume again. If 1B does not appear, the procedure must be aborted.

5. When hexadecimal wait code 0A appears on the program status lights, dismount the MSC from transport 1 and replace it with a scratch cassette; press Compute.

6. When the 399 WAIT light comes ON, the cassette on transport 1 contains the 399 program. To rewind the paper tape, press Resume; when the rewind is finished, a hex wait code of 0F is displayed on the PROG STAT lights. If no rewind is required, the program is finished when the WAIT comes on.

RESTART PROCEDURE

After a 399 program has been transferred from paper tape to cassette, other programs may be transferred by using the following restart procedure.

1. Mount new paper tape reel.
2. Mount MSC on transport 1.
3. Press Reset.
4. Press Compute.
5. Follow Operational Procedure, starting with step 5.

PROGRAM WAIT DISPLAYS

TRUNK 4 light ON indicates that the mounted paper tape is not a source or object tape; place the proper tape on the reader and press Reset, then Compute.

CAS1 light ON indicates that the cassette on transport 1 is not an MSC; place MSC on transport 1 and reset paper tape to the start position. Press Reset, then Compute.

PROG STAT lights 2 and 16 ON and TRUNK 4 light ON indicates that the next reel of tape is to be mounted (multi-reel program only). Mount the next reel of tape (maximum of four reels) and press Compute.

PROG STAT lights 1, 2, and 16 ON and TRUNK 4 light ON indicates that too many (more than four) reels of tape are required to contain the program. No recovery; cassette cannot accept program.

ERROR CONDITIONS (CASSETTE)

CAS1 light ON, ALERT lights 1, 2, and 4 ON, and MEDIA light ON indicates that cassette is not ready to write. Enable cassette to write and press Compute.

ALERT lights 1 and 2 ON and MEDIA light ON indicates same condition as previously described display.

CAS1 and 2 lights ON and ALERT lights 1, 2, 4, 8, and 16 ON indicates that the transport lid was opened during operation; restart, using restart procedure.

CAS1 light ON, DATA light ON, and WAIT light ON indicates a write error; restart, using restart procedure.

CAS1 light ON, ALERT light 16 ON, and MEDIA light ON indicates end of tape; no recovery.

ERROR CONDITIONS (PAPER TAPE)

TRUNK 4 light ON and DATA light ON indicates parity error (three retries). Restart, using restart procedure.

TRUNK 4 light ON and MEDIA light ON indicates that 16 contiguous delete characters have been encountered; restart, using restart procedure.

TRUNK 4 light ON and ALERT lights 1 and 2 ON indicates that the tape reader is not ON. Turn tape reader ON, press Compute.

SITE CONFIGURATOR

GENERAL

The software configuration on the Master Software Cassette (MSC) is designed for the standard U.S. domestic site. Sites which deviate from this standard use the Site Configurator utility routine, resident on the MSC, to create a Site Software Cassette (SSC) tailored to the requirements of the site.

Variations which require special routines are:

1. A non-U.S. standard character code set for the following devices:
   a. Serial Printer
   b. Keyboard
   c. Line Printer
   d. Card Reader
   e. Card Punch
2. Other than U.S. standard data edit symbols ($, . ◊ CR).

Special routines to compensate for these variations are present on the MSC, but they are not accessed on a standard site. The Site Configurator routine creates an SSC which contains all necessary software, including information defined by the user, to cause the appropriate special routines to be accessed and loaded by the Linking Loader during program load.

RESTRICTIONS

A 399 with two cassette handlers is required.

The character set of the system running the Site Configurator must be the same as that of the system which is to use the resulting SSC.

No more than one character can be used to replace any standard U.S. edit symbol.

PROCEDURE

Fig. C-44 presents a step-by-step operating procedure for this routine.

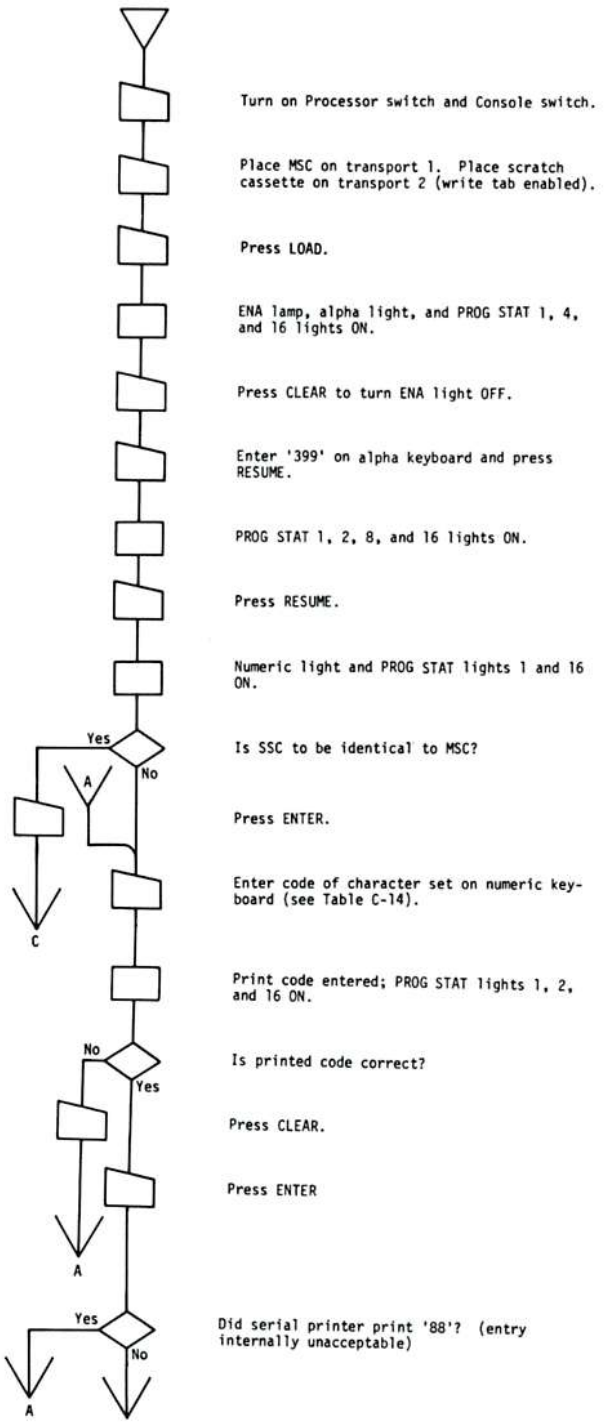

Fig. C-44 Site configurator

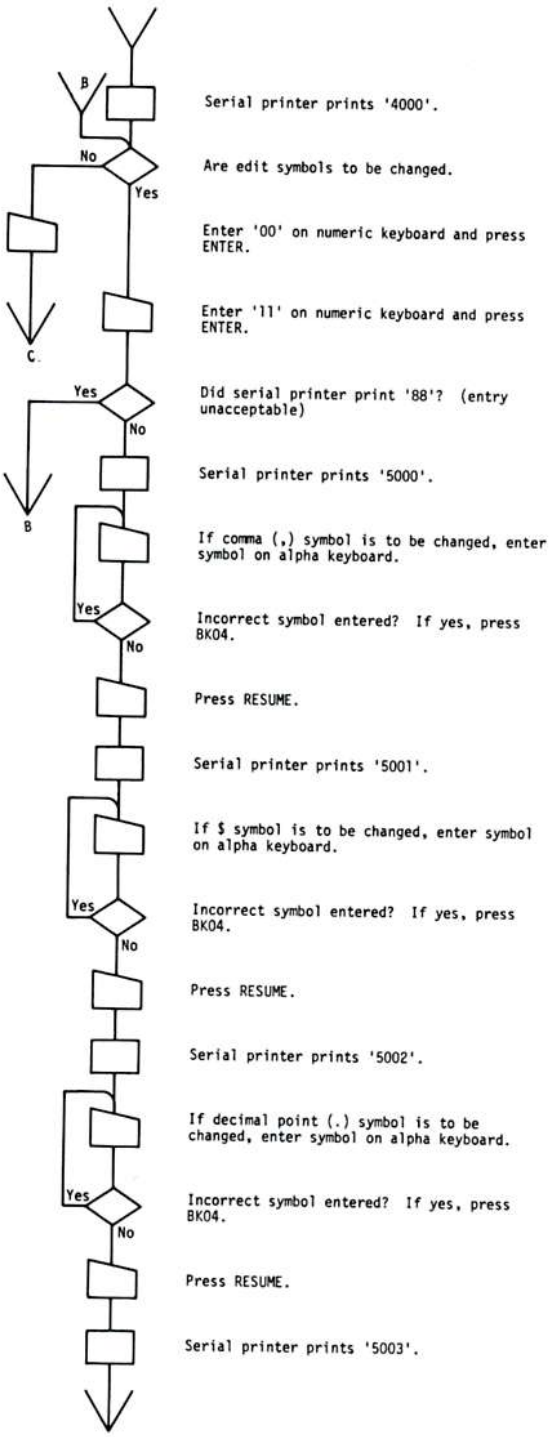

Fig. C-44 Continued

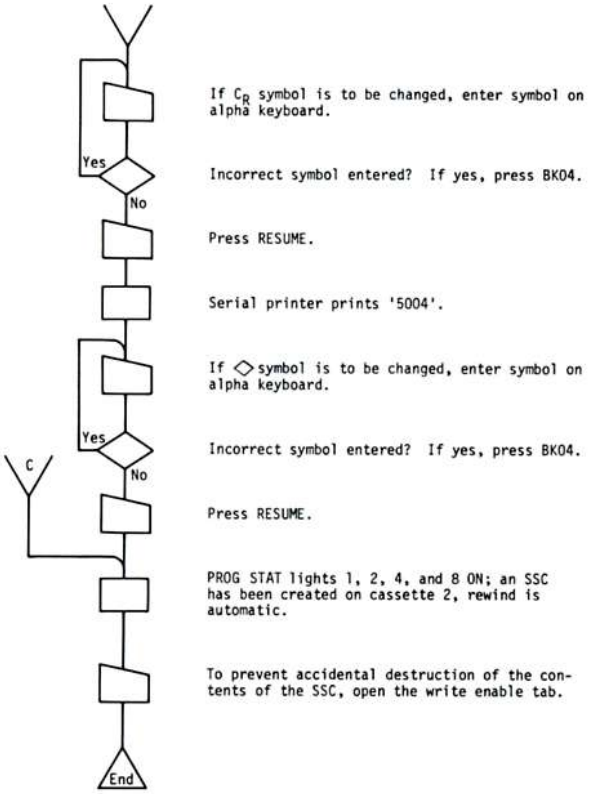

Fig. C-44 Continued

CHARACTER SET	CODE
U.S. Standard	00
France	01
Belgium	01
Italy	01
Katakana	02
Spain	03
Latin America	04
Germany	05
Austria	05
Switzerland	05
Portugal	06
Sweden	07
Finland	07
Denmark	08
Norway	08
Brazil	09
United Kingdom	10

Table C-14 Character set codes

ERROR CONDITIONS

A printed 99 indicates that the appropriate encode/decode tables cannot be found on the MSC. Retry; if 99 is printed again, replace MSC and press Load.

PROG STAT lights 8 and 16 ON indicates that MSC is not mounted on transport 1. Place MSC on transport 1 and press Compute.

PROG STAT lights 4 and 16 ON indicates unrecoverable errors on the cassette on transport 2. Mount new scratch cassette on transport 2 and press Compute.

CAS1 or CAS2 light ON and ALERT light 1 ON indicates that clear trailer has been detected on the indicated cassette. Abort the program.

CAS1 or CAS2 light ON, ALERT lights 1 and 2 ON, and MEDIA light ON indicates that the indicated cassette is not ready to read or write. Prepare unit for operation and press Compute.

CAS1 or CAS2 light ON, ALERT lights 1, 2, and 4 ON, and MEDIA light ON indicates that the write tab is not enabled on the cassette indicated. Enable write tab and press Compute.

CAS1 or CAS2 light ON, ALERT light 16 ON, and MEDIA light ON indicates that EOT hole has been detected on the cassette indicated. Abort the program.

CAS1 or CAS2 light ON, ALERT light 16 ON, and DATA light ON indicates that blank tape has been read from the cassette indicated. Abort the program.

CAS1 or CAS2 light ON and ALERT lights 1, 2, 4, 8, and 16 ON indicates that the lid of the indicated cassette was raised during operation. Abort the program.

CAS1 or CAS2 light ON, DATA light ON, and WAIT light ON indicates an unrecoverable read (cassette 1) or write (cassette 2) error occurred. Abort the run.

USER PROGRAM DUMP

GENERAL

The User Program Dump utility provides a printout, in source format, of a user program in memory. This can simplify the task of debugging, since it is not necessary to decode hexadecimal or binary data; the printout is almost identical to the program source listing.

This routine can be used only with 399 object programs; it is not designed to decompile M05 object programs.

User Program Dump is made up of two programs. One, UDUMP1, is used to write the user program to a scratch cassette in a specific format. The second routine, UDUMP2, uses the cassette encoded by UDUMP1 for input to carry out the dump.

UDUMP1

At any point during execution of user program, UDUMP1 may be employed. The user program must be halted, and a

software cassette containing UDUMP1 must be mounted on transport 1. UDUMP1 is then called in, using the standard procedure employed by Linking Loader.

After UDUMP1 has been loaded into memory, the NUMERIC light and PROG STAT lights 1 and 16 are ON, indicating that a blank cassette is to be mounted on transport 1 or 2. If transport 1 is to be used, the software cassette should be rewound before it is removed.

After the blank cassette is mounted, a branch key must be pressed to indicate which transport contains the cassette; BK01 indicates transport 1, and BK02 indicates transport 2.

When the appropriate branch key is pressed, the program writes the user program to the destination cassette; on completion, PROG STAT lights 1, 2, 4, and 8 ON indicate the end of UDUMP1.

UDUMP2

In order to print out the user program previously encoded on cassette by UDUMP1, the software cassette containing UDUMP2 is mounted on transport 1; UDUMP2 is then called in, using the standard procedure employed by Linking Loader.

After UDUMP2 has been loaded into memory, the ALPHA light and PROG STAT lights 1 and 16 are ON, indicating that an entry specifying serial printer (S) or line printer (L) must be made on the alpha keyboard. A Resume key depression completes this entry.

The line printer option is not included in early versions of this routine. If an L is entered on an early version, line printer inoperative (TRUNK 1 light and ALERT lights 1, 2, and 4 ON) is displayed; recovery is accomplished by pressing Compute, which recycles the program so an S may be entered.

After an S or an L entry has been validated, PROG STAT lights 2 and 16 are ON, requesting that the cassette to be dumped be mounted on transport 1 or 2. If transport 1 is used, the software cassette should be rewound before it is removed. After the cassette is mounted, an entry of 1 or 2, to specify the transport, must be made, followed by depression of the Resume key.

The program then verifies that the cassette mounted is a valid user dump cassette; if it is not, the cassette is rewound, and PROG STAT lights 4 and 16 are displayed, in addition to the appropriate CAS light. Recovery is accomplished by replacing the cassette with the proper one and pressing Compute. This recycles the program for re-entry of 1 or 2.

When the cassette has been validated, PROG STAT lights 1, 2, and 16 are displayed, requesting an entry concerning the type of dump desired, followed by depression of the Resume key. These entries are listed in the following text.

ALL – This entry results in a dump of the entire user program, starting at data location 0000 and terminating with the last user instruction.

DATA – This entry results in a dump of the user data area, starting at data location 0000 and terminating with the last data field.

CODE – This entry results in a dump of the user instruction area, starting at instruction location 0000 and terminating with the last user instruction.

AREA – This entry causes two additional messages, requiring responses, to be printed. The first of these messages is START AD ––, the reply to which is XXXXD or XXXXC. XXXXD represents the hexadecimal word address of the header of first data field to be dumped. XXXXC represents the hexadecimal digit address of the high order digit of the first user instruction to be dumped. Either entry is followed by a Resume key depression.

Validation is performed on the characters entered to ensure that the high order four are legal hexadecimal (O-F) and that the low order character is C or D; an error causes PROG STAT lights 1, 2, and 16 to be redisplayed, so that AREA may be entered again.

If the address entered is accepted, FINISH AD –– is printed, the reply to which is XXXXD or XXXXC, the ending address of the dump. The entry follows the same format as START AD ––, and the D or C entry must be the same in both START AD –– and FINISH AD ––. Depression of Resume causes the program to continue.

NEW – This entry causes the cassette which has been dumped to be rewound and displays CAS1 or CAS2 light, in addition to PROG STAT light 2. A new cassette is then mounted, followed by depression of the Compute key. The program recycles to the starting light display, PROG STAT lights 1 and 16 ON, where the sequence to dump the new cassette may be started; any number of cassettes may be dumped in this manner.

END – This entry causes the cassette in use to be rewound; PROG STAT lights 1, 2, 4, and 8 are displayed, indicating end of run.

If an error is made in entering any of the responses required, and it is detected before the Resume key is pressed, depression of BK04 causes a carriage return and line feed to occur, allowing the response to be re-entered. The same correction can be accomplished by back-spacing to the error and re-entering the response.

After a dump has been completed, the program recycles to the start (PROG STAT lights 1, 2, and 16 ON). At that point, any of the previously listed options may be exercised.

During the printing of the dump, the NUMERIC light is ON, abling the branch keys. Depression of any branch key causes the printing to stop, and the program recycles to the start (PROG STAT lights 1, 2, and 16 ON). Any of the previously listed options may then be exercised.

RESTRICTIONS

UDUMP cannot determine the presence of a data table in memory, because no indication is carried in the data header

to signify the presence of a table. Therefore, UDUMP does not output any data table definitions; the data items within the table are printed as normal data definitions.

A Redefine area cannot be printed, because there is no information in memory specifying the redefinition address. The word REDEFINITION is printed in the space occupied by the redefined data name.

The slew character, if any, in a data header is printed as a 1, 2, or 3. The user must determine whether the serial printer or line printer is involved; if it is determined that the slew is for the serial printer, 1 = R, 2 = L, and 3 = B.

ERROR CONDITIONS

If the logical contents of the object program have been destroyed in memory or erroneously patched (data header or Q code not in proper position), the dump prints one of the following two messages: ****** DATA HEADER IN ERROR – DATA ITEM MISSED OUT or ****** CODING IN ERROR – CODE ITEM MISSED OUT. UDUMP attempts to recover the correct sequence by scanning for the next valid data header or Q code in memory. There is no guarantee that what is detected as valid is in fact valid within the object program.

Various error conditions within the user program are printed as ERR or ERROR when detection is possible.

OUTPUT

The output of UDUMP2 is a printed listing closely resembling the source program listing. Each data definition or instruction occupies one line of the listing, and a heading is output every 60 lines to permit a tally roll listing to be folded in a fashion similar to continuous forms.

No tags are generated for data fields or for A, B, or C operands in instructions. The hexadecimal address is used to define them, making it necessary to correlate the address with the address printed on the source program listing.

Data Format

Each data definition is printed from left to right, as follows.

LINE – Page and line number – 6 columns

CD – Type of statement (D) – 1 column

WORD AD – Hexadecimal word address of header – 4 columns

T – Type of data definition – 1 column

LEN – Length of field – 3 columns

DP – Decimal point position – 2 columns

PRT – Print position – 3 columns

S – Slew – 1 column

S – Sign – 1 column

VALUE – Data field contents – 25 columns (larger fields use additional lines)

HEADER – Hexadecimal object bit string of header and print position – 6 columns

Instruction Format

Each instruction is printed from left to right, as follows.

LINE – Page and line number – 6 columns

CD – Type of statement (C) – 1 column

DIGIT AD – Hexadecimal digit address of Q code – 4 columns

OPCODE – Operation mnemonic – 6 columns

A-OPND – Hexadecimal word address of A operand – 4 columns

B-OPND – Hexadecimal word address of B operand – 4 columns

C-OPND – Hexadecimal word or digit address of C operand – 4 columns

ACTION – Action code – 6 columns

N-FLD – N-FIELDS entry – 6 columns

OBJECT – Hexadecimal object bit string of instruction – 12 columns

The headings (LINE, CD, etc.) described for data and instructions are printed once every 60 lines.

CASSETTE MESSAGES (LIGHT DISPLAY)

CAS1 or CAS2 light and DATA light ON indicates that the cassette indicated has had uncorrectable read errors; abort the run.

CAS1 or CAS2 light and DATA light ON indicates that the cassette indicated has had uncorrectable write errors; press Start after installing new cassette.

CAS1 or CAS2 light, ALERT light 1 and 2, and MEDIA light ON indicates that the cassette indicated is not ready; correct the situation and press Compute.

CAS1 or CAS2 light, ALERT lights 1, 2, and 4, and MEDIA light ON indicates that the cassette indicated does not have write tab enabled; enable write tab and press Compute.

CAS1 or CAS2 light and ALERT light 1 ON indicates that the cassette indicated is in trailer; abort the run.

CAS1 or CAS2 light, ALERT light 16, and MEDIA light ON indicates that the cassette indicated is at EOT; abort the run.

CAS1 or CAS2 light, ALERT light 16, and DATA light ON indicates that the cassette indicated is in blank tape; abort the run.

CAS1 or CAS2 light and ALERT lights 1, 2, 4, 8, and 16 ON indicates that the lid of the cassette indicated was opened; abort the run.

QUICK FIX

The following procedure can be used to implement Quick Fix to obtain a printout of the module headers and the volume header from either the MSC or SSC.

START PROCEDURES

1. Processor switch ON.
2. Console switch ON.
3. Mount MSC on transport 1 and close the lid.
4. Check serial printer for adequate supply of paper.
5. Turn OFF the CFF's.

LOAD PROCEDURES

1. Press LOAD. The ENA light turns ON following the load; press CLEAR.
2. When the ALPHA light and PROG STAT lights 1, 4, and 16 turn ON, enter QUIKFIX through the alpha keyboard and press RESUME.
3. When PROG STAT lights 1, 2, 8, and 16 turn ON, press RESUME.

INPUT PROCEDURES

1. The message MOUNT MSC CASSETTE ON TRANSPORT 1... is printed. Replace, if necessary, the rewound MSC on tranport 1 with the MSC or SSC whose headers are to be printed.
2. Enter Y through the alpha keyboard. If an error is made in this entry, press BK04 and re enter.
3. Press RESUME. If an incorrect entry was made in step 2 and was not corrected, the message is reprinted.
4. The message IS MODULE HEADER PRINT REQUIRED... is printed. To print the cassette module headers, enter Y and press RESUME; to bypass printing module headers, enter N and press RESUME.

OUTPUT

1. The serial printer prints the module headers, if called for in step 4, and the volume header, followed by the message END OF RUN...
2. Enter Y and press RESUME.
3. PROG STAT lights 1, 2, 4, and 8 ON indicate end-of-job.