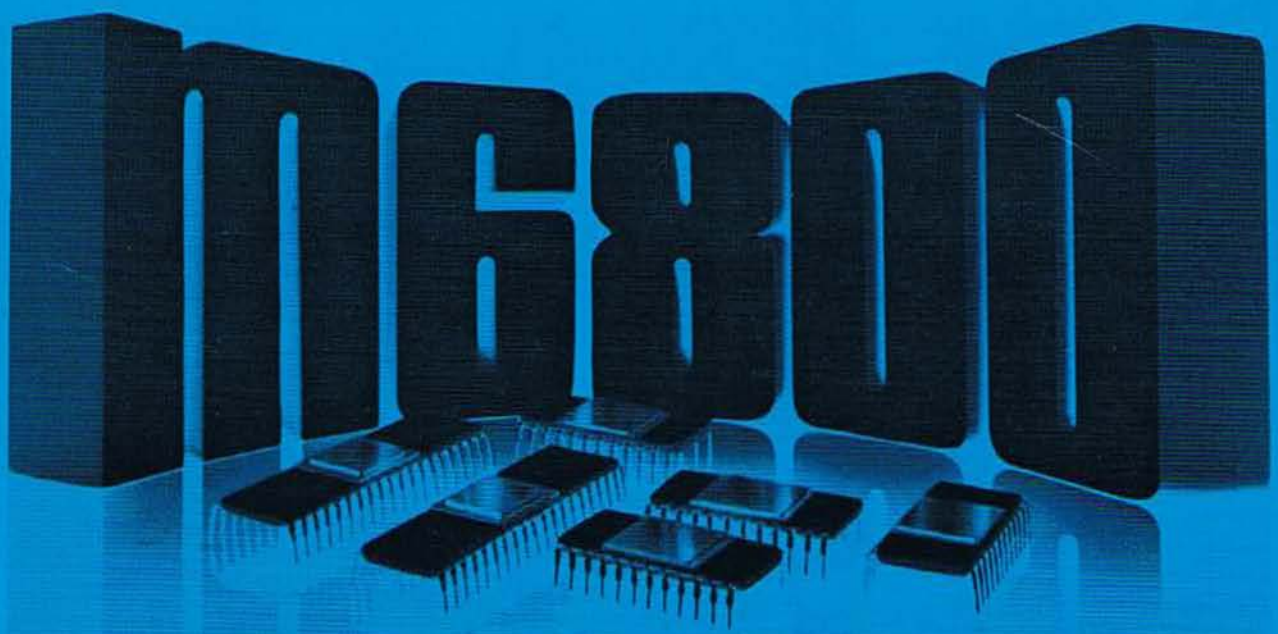
M68CRE(D)

M6800
CO-RESIDENT EDITOR REFERENCE MANUAL



MOTOROLA

MICROSYSTEMS

M68CRE(D)
Jan., 1977

M6800

CO-RESIDENT EDITOR
REFERENCE MANUAL

First Edition

TABLE OF CONTENTS

CHAPTER 1

GENERAL INFORMATION

1.1 INTRODUCTION

The M6800 Co-Resident Editor is an interactive program used to create and modify source program statements written in the M6800 Assembler Language. It offers character, line and character string commands and can co-reside in system memory with the M6800 Assembler. The process of translating these source program statements into an object program compatible with M6800 firmware loaders is described in the M6800 Co-Resident Assembler Manual.

1.2 EDITOR FEATURES

The features of the M6800 Co-Resident Editor include:

Memory sizing capability

Line deletion or insertion

Character string deletion, insertion or modification

May co-reside with the M6800 Assembler or reside alone

1.3 GENERAL DESCRIPTION

A source program is written for the purpose of defining the sequence of instructions which must be communicated to a microprocessor in order for it to perform the function desired by the user. Such a program is comprised of an ordered collection of statements in the language and format which can be processed by the system's assembler program into machine code.

A source statement is simply one or more assembler language elements expressed according to rules of program format. The language elements themselves are construction of ASCII (American Standard Code For Information Interchange) characters. Those characters recognized by the M6800 Assembler are listed in Appendix A. For a detailed description of the M6800 Co-Resident Assembler language, refer to the M6800 Co-Resident Assembler Reference Manual.

The time and effort required of the programmer to write his program can be greatly reduced by the use of an interactive editor. The Co-Resident Editor provides a buffer in system memory into which source statements may be read for expansion, deletion or modification by means of the editor commands. All or portions of the edited text may then be output on the desired system medium.

CHAPTER 2

CO-RESIDENT EDITOR

2.1 INTRODUCTION

The M6800 Co-Resident Editor accepts input text from either the System Console Device or the System Reader Device and accepts edit commands from the System Console Device. During a typical edit operation, successive portions of input text are transferred to the edit buffer. After editing, information in the buffer is transferred to the System Punch Device.

2.2 EDITOR INPUT

The Co-Resident Editor accepts input text from the System Reader Device (paper tape, cassette or diskette or the System Console Device (terminal keyboard). Commands to the editor are supplied from the System Console Device.

2.3 EDITOR OUTPUT

The Co-Resident Editor produces an output text on the System Output Device (paper tape, cassette or diskette). In addition, the Editor may be used to print selected portions of the edited text on the System Printer Device (terminal printer).

2.4 EDITOR OPERATION

Editing may be performed on either characters or lines. A character, to the editor, is any ASCII character. A line is any collection of characters bounded by Carriage Return characters. Non-printing characters such as CR and EOT are treated as characters by the editor and may be manipulated accordingly. Since the editor automatically outputs a Line Feed character on recognition of a Carriage Return, these need not be supplied with text input from either the System Console Device or System Reader Device. In the examples to follow, Carriage Return and Line Feed characters are symbolized as (CR) and (LF), respectively.

NOTE

The ASCII characters LF, NUL, DC1, DC2, DC3, DC4 and RUBOUT are deleted on input to the edit buffer.

Edit operations are performed on portions of the text held in an edit buffer in the EXORciser memory. A buffer pointer is maintained to specify a character location within the edit buffer. Certain of the

edit operations are performed on lines or characters located with respect to the buffer pointer. It is convenient to think of this pointer as being located between two characters. As shown in the example below, the buffer pointer is located between the D and E of RESIDENT.

```
                    ↓
EXAMPLE:    MOTOROLA M6800 RESIDENT EDITOR
```

2.5 LOADING THE EDITOR (TAPE OR CASSETTE)

If the Resident Editor is not already in memory, it may be loaded using the following procedure:

(a) Place the editor object tape (paper tape or cassette) into the System Reader Device.

(b) Enter the EXbug command "LOAD". EXbug will respond with "SGL/CONT".

(c) Type "S" after SGL/CONT to load the single file containing the editor. After the header record from the tape is printed, the file will be loaded. Upon completion, control is returned to EXbug.

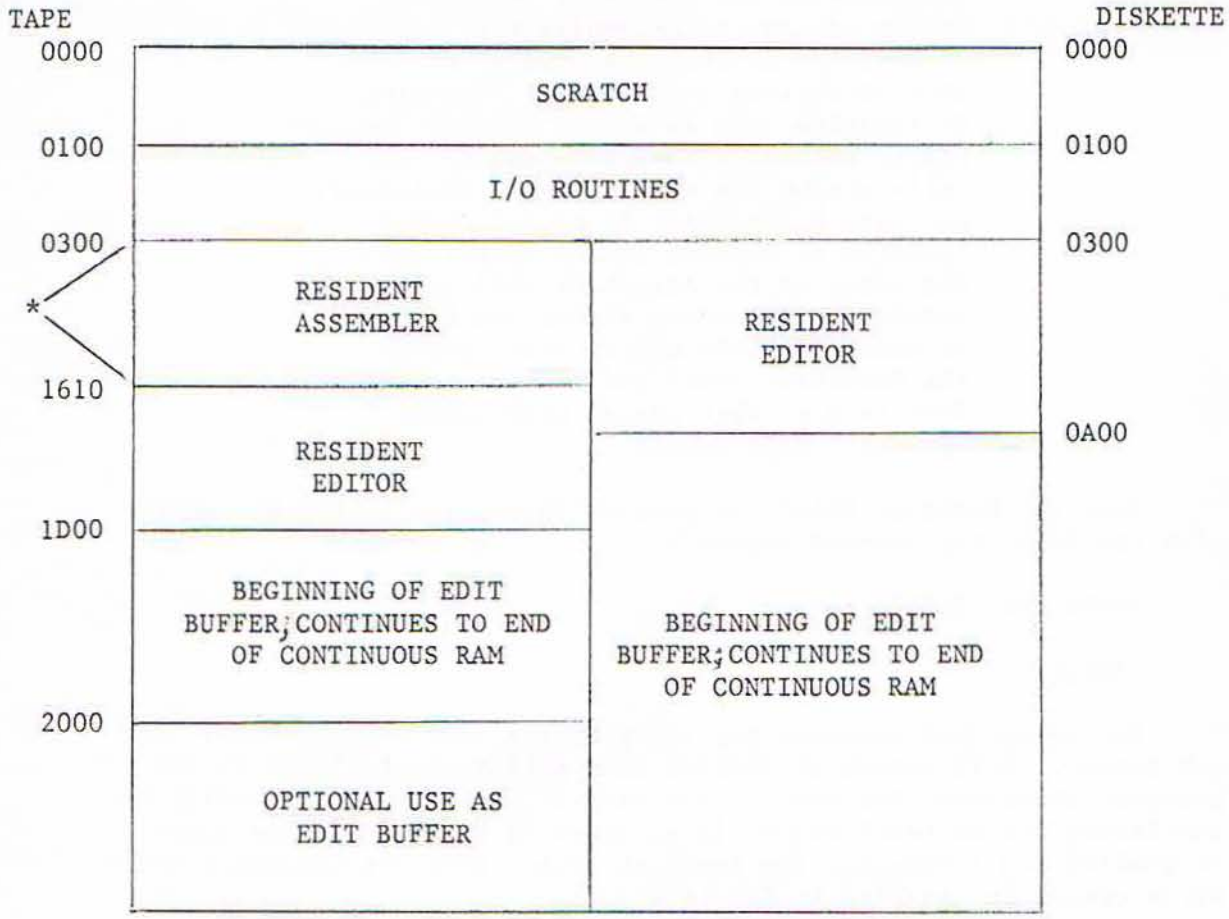
2.6 LOADING THE EDITOR (DISKETTE)

Instructions for loading, initiating and using the text editing capability of the disc operating system should be obtained from the operator's manual for that system.

2.7 INITIATING THE TAPE EDITOR

In normal operation, the memory region between the end of the Editor and the end of contiguous read-write storage is used by the Editor as the edit buffer. An 8K RAM system provides an edit buffer capacity of about 740 bytes. If a larger buffer is desired and additional memory is not available, it is possible to overwrite the Resident Assembler. This option provides an edit buffer of approximately 4900 bytes. These edit buffer options are illustrated in the memory map in Figure 2-1.

Selection of the Assembler overwrite option must be made after the Editor has been loaded, and before it is initiated. The overwrite option is enabled by using MAID to change the contents of memory location $300 to $FF. It is not necessary to load the Assembler to select the Assembler overwrite option.



* The Assembler overwrite flag is at $300. If its value is zero, the assembler area will not be used as the edit buffer. If non-zero, the assembler area will be used as the edit buffer instead of the memory following the editor.

Figure 2-1. Memory Map of Co-Resident Assembler and Editor.

NOTE

Selection of the Assembler overwrite option causes the Assembler entry to be modified so that EXbug is entered when attempting to enter the Assembler. If location $300 is set to 0 after the Editor has run with the Assembler overwrite option and the Editor is restarted, the edit buffer will be at its normal location at the end of the Editor but the entry to the Assembler will remain modified. Reloading either the Editor or Assembler into memory will restore the Assembler entry and reset location $300 to 0 so that normal edit buffer operation will result.

When the Resident Editor is present in memory, it may be entered with the following command sequence:

EXBUG n.n. MAID

*103;G

The underlined commands are input by the user causing entry into the editor. This method of showing user actions is followed in the examples throughout the rest of the manual. An example of loading and initiating the Resident Editor is provided in Figure 2-2. An example of loading and initiating the Resident Editor with the Assembler overwrite option is provided in Figure 2-3.

```
%
EXBUG 1.1 LOAD
SGL/CONT S
% EDT 1.3
EXBUG 1.1 MAID
*103;G
M6800 RESIDENT EDITOR 1.3
@
```

FIGURE 2-2. Loading and Initiating Resident Editor

```
%
EXBUG 1.1 LOAD
SGL/CONT S
% EDT 1.3
EXBUG 1.1 MAID
*300/00 FF
*103;G
M6800 RESIDENT EDITOR 1.3
@
```

FIGURE 2-3. Loading and Initiating Resident Editor with Assembler Overwrite Option

2.8 EDITOR COMMANDS

The Resident Editor prints "@" in the left margin as a prompt whenever it is waiting for a command. Commands to the editor are single characters entered from the System Console Device. Some commands have arguments associated with them. All commands must be terminated by two ESC ($1B) characters which cause the Editor to begin execution. Because CR is a legal text character, it could not be used for command termination. Note that since it is a non-printing character, the Editor echoes "$" whenever ESC is entered.

The Resident Editor commands are described in the following paragraphs and summarized in Table 2-1. Editor commands may be grouped into four categories:

Input/Output Operations	Edit Operations
Buffer Pointer Operations	Exit Editor

2.8.1 INPUT/OUTPUT OPERATIONS

Input/output operations control the transfer of information between the edit buffer and the System Reader Device, System Punch Device and the System Printer Device.

2.8.1.1 A -- APPEND

FORMAT: A

DESCRIPTION: The append command causes input text to be transferred from the System Reader Device and appended to the edit buffer. The transfer is terminated by one of the following conditions:

(1) End of tape	(3) Workspace full
(2) End of file character ($1A)	(4) 50 lines read

Null, Rubout, LF, Readeron, Punchon, Punchoff, and SUB ($1A) characters are deleted from the input text.

If the editor is unable to read the input text (because the reader device is off, etc.), it will time out and print "@" as a prompt for a new command.

If the workspace allows, more than 50 lines may be entered by repeating the A command.

EXAMPLE: Assume that the Editor has been loaded into memory and is running. Assume, further, that a tape containing the following information has been loaded into the System Reader Device:

```
10 NAM PGM(CR)(LF)
20  OTP M MEMORY FILE OPTION(CR)(LF)
30  OPT O OUTPUT OBJECT TAPES(CR)(LF)
40  OPT S SELECT PRINTING SYMBOLS(CR)(LF)
50  ORG 8192(CR)(LF)
60  LDA B ADDR(CR)(LF)
70 COUNT EQU @8  @ INDICATES OCTAL(CR)(LF)
80 START LDS #STACK INZ STACK POINTER(CR)(LF)
90  LDX ADDR(CR)(LF)
100  LDA B #COUNT IMMEDIATE ADDRESSING(CR)(LF)
110 BACK LDA A 10 DIRECT ADDRESSING(CR)(LF)
120  CMP A 2,X INDEXED ADDRESSING(CR)(LF)
130  BEQ FOUND RELATIVE ADDRESSING(CR)(LF)
140  DEX IMPLIED ADDRESSING(CR)(LF)
150  DEC B ACCUMULATOR ONLY ADDRESSING(CR)(LF)
160  BNE BACK(CR)(LF)
170  WAI  WAIT FOR INTERRUPT(CR)(LF)
180  SPC 1(CR)(LF)
190 FOUND JSR SUBRTN    JUMP TO SUBROUTINE(CR)(LF)
200  JMP START EXTENDED ADDRESSING(CR)(LF)
210 * COMMENT STATEMENT NOTE TRUNCATION 01234567890123456789(CR)(LF)
220 SUBRTN TAB COMMENT FIELD TRUNCATION0123456789(CR)(LF)
230  ORA A BYTE   SET MOST SIGNIFICANT BIT(CR)(LF)
240  RTS RETURN FROM SUBROUTINE(CR)(LF)
250  SPC 2(CR)(LF)
260  RMB 20 SCRATCH AREA FOR STACK(CR)(LF)
270 STACK RMB 1 START OF STACK(CR)(LF)
280 BYTE FCB $80 FORM CONSTANT BYTE(CR)(LF)
290  FCB $10,$4  $ INDICATES HEXADECIMAL(CR)(LF)
300 ADDR FDB DATA FORM CONSTANT DOUBLE BYTE(CR)(LF)
310 DATA FCC 'SET' FORM CONSTANT DATA STRING (ASCII)(CR)(LF)
320  END(CR)(LF)
*
```

Entering the A command loads the information on the tape into the edit buffer.

```
@A$$
@
```

If the Editor was just entered, the edit buffer is empty prior to the A command. Then, following the A command, the contents of the tape are loaded into the edit buffer until the buffer is full or the end of the tape is reached.

If the edit buffer was not empty prior to the A command, the amount of text entered into the edit buffer is dependent upon the four conditions listed above.

NOTE

The following examples assume that the contents of the tape listed above are the only contents of the edit buffer. Remember that Line Feeds are not entered into the edit buffer and that $ indicates ESC. Notice that a minimum of two spaces are required between a line number and an instruction without a label and only one space between a line number and a label. User entries are underlined.

2.8.1.2 E -- END EDIT OPERATION

FORMAT: E

DESCRIPTION: The end command terminates the edit operation and causes the contents of the edit buffer to be transferred to the System Punch Device. The remaining text is then copied from the System Reader Device to the System Punch Device. When an end of file character ($1A) is encountered on the input tape, or the tape ends, an end of file character ($1A) and blank trailer are written to the output tape. Upon completion, the Editor prints "@" and awaits further commands. The contents of the edit buffer are still available for editing. The buffer pointer remains at its previous position. The E command does not cause leader to be punched.

2.8.1.3 F -- TAPE LEADER/TRAILER

FORMAT: F

DESCRIPTION: The Tape Leader/Trailer command writes fifty NULL characters to the System Punch Device. This command may be used to produce Leader/Trailer for paper tapes.

2.8.1.4 P -- PUNCH

FORMAT: nP
n is a positive decimal integer less than 256. If omitted, the value is assumed to be 1.

DESCRIPTION: The Punch command causes a specified number of lines (n), beginning with the line specified by the buffer pointer, to be written from the edit buffer to the System Punch Device. The lines are deleted from the edit buffer. If a negative number n is entered, the negative sign is ignored and a positive n number of lines are punched. The command 0P (zero P) is ignored.

2.8.1.5 T -- TYPE

FORMAT: nT
n is a decimal integer in the range -254 n 255. If omitted, the value is assumed to be 1.

DESCRIPTION: The type command causes a specified number of lines (n) to be printed on the System Console Device. If n is positive, the first line typed is the current line (the line indicated by the current position of the buffer pointer). If n is negative, the n lines preceding the current line are typed. If fewer than n lines exist following (preceding) the end (beginning) of the edit buffer, only these lines are typed.

EXAMPLE:

```
$5T$$
40  OPT S SELECT PRINTING SYMBOLS
50  ORG 8192
60  LDA B ADDR
70 COUNT EQU 58  $ INDICATES OCTAL
80 START LDS #STACK INZ STACK POINTER

$0T$$

$-10T$$
10 NAM PGM
20  OTP M MEMORY FILE OPTION
30  OPT O OUTPUT OBJECT TAPES

$1T$$
40  OPT S SELECT PRINTING SYMBOLS

$
```

The 5T types the five lines following the buffer pointer location. Note that the buffer pointer is currently at the beginning of the fourth line of the edit buffer.

No lines are printed by the 0T command. If the buffer pointer is inside a line, the 0T command will print the characters from the beginning of the line to the buffer pointer.

The - 10T command prints the lines from the current buffer pointer location to the beginning of the edit buffer since there are less than 10 lines from the buffer pointer to the beginning of the edit buffer.

The command T prints the single line following the buffer pointer since the Editor assumes a one precedes the T.

2.8.2 BUFFER POINTER OPERATIONS

Buffer pointer operations are used to manipulate the position of the edit buffer pointer.

2.8.2.1 B -- BEGINNING

FORMAT: B

DESCRIPTION: The Beginning command moves the edit buffer pointer to the beginning of the edit buffer.

EXAMPLE:

```
$T$$
40  OPT S SELECT PRINTING SYMBOLS

$B$$

$T$$
10 NAM PGM

$
```

The line printed by the first T command is the fourth line in the edit buffer indicating that the buffer pointer is at the beginning of the fourth line.

The B command moves the buffer pointer to the beginning of the edit buffer.

The final T command causes the first line to be printed out (a non-specified n defaults to 1) confirming that the buffer pointer is at the beginning of the edit buffer.

2.8.2.2 Z -- END OF BUFFER

FORMAT: Z

DESCRIPTION: The End of Buffer command moves the edit buffer pointer to the end of the edit buffer.

EXAMPLE:

```
$T$$
10 NAM PGM

$Z$$

$-1T$$
330   MON

$
```

Initially the buffer pointer is at the beginning of the edit buffer as the first T command indicates by printing the first line.

The Z command moves the buffer pointer to the end of the edit buffer.

The - 1T command prints the line immediately preceding the buffer pointer location, in this case the last line of the edit buffer.

2.8.2.3 M -- MOVE CHARACTER POINTER

FORMAT: nM
n is a decimal integer in the range -254 n 255. If omitted, the value is assumed to be 1.

DESCRIPTION: The Move Character Pointer command moves the edit buffer pointer according to the number of characters specified by n. If n is positive, the pointer is moved forward n characters. If negative, the pointer is moved back n characters. If fewer than n characters are present between the initial buffer pointer and the end (beginning) of the edit buffer, the pointer is moved to the end (beginning) of the buffer.

EXAMPLE:

```
$4T$$
10 NAM PGM
20  OTP M MEMORY FILE OPTION
30  OPT O OUTPUT OBJECT TAPES
40  OPT S SELECT PRINTING SYMBOLS

$3M$$

$2T$$
NAM PGM
20  OTP M MEMORY FILE OPTION

$-1T$$
10
$8M$$

$T$$
20  OTP M MEMORY FILE OPTION

$-1M$$

$T$$


$
```

The 4T command prints the first 4 lines of the edit buffer indicating that the buffer pointer is at the beginning of the edit buffer.

The 3M command moves the buffer pointer forward 3 characters.

The 2T command demonstrates this by printing the current line, beginning at the buffer pointer location, and the next line.

The - 1T command prints the previous line, which in this case is the information from the beginning of the edit buffer to the current buffer pointer location.

The buffer pointer is moved forward eight characters by the 8M command. Now the buffer pointer is at the beginning of the second line in the edit buffer as the T command demonstrates.

The - 1M command moves the buffer pointer back one character to just before the Carriage Return at the end of the first line of the edit buffer. Now the T command prints only the Carriage Return, which is the remainder of the line it is in, and a Line Feed.

2.8.2.4 L -- LINE

FORMAT: nL
n is a decimal integer in the range -254 n 255. If omitted, the value is assumed to be 1.

DESCRIPTION: The Line command moves the edit buffer pointer according to the number of lines specified by n. If n is positive, the pointer is moved forward n lines. If negative, the pointer is moved backward n lines. A value of 0 causes the pointer to be moved to the beginning of the current line. If fewer than n lines are present between the initial buffer pointer and the end (beginning) of the edit buffer, the pointer is moved to the end (beginning) of the buffer.

NOTE

The M6800 Text Editor considers a line to be a sequence of characters delimited by Carriage Returns.

EXAMPLE:

```
$4T$$
10 NAM PGM
20  OTP M MEMORY FILE OPTION
30  OPT O OUTPUT OBJECT TAPES
40  OPT S SELECT PRINTING SYMBOLS

$3L$$

$T$$
40  OPT S SELECT PRINTING SYMBOLS

$-2L$$

$T$$
20  OTP M MEMORY FILE OPTION

$8M$$

$T$$
M MEMORY FILE OPTION

$0L$$

$T$$
20  OTP M MEMORY FILE OPTION

$
```

The 4T command prints the first four lines of the edit buffer indicating that the buffer pointer is at the beginning of the edit buffer.

The 3L command moves the buffer pointer forward three lines.

The T command then prints the fourth line of the edit buffer.

The -2L command moves the buffer pointer back two lines.

Now, the T command prints the second line of the edit buffer.

The 8M command moves the buffer pointer forward eight characters.

This is confirmed by the T command which prints only a portion of the second edit buffer line.

The command 0L causes the buffer pointer to move to the beginning of the line that it is currently in.

The final T command illustrates that the buffer pointer did move to the beginning of the second line.

2.8.2.5 S -- SEARCH

FORMAT: Sstring
The "string" argument is a string of 16 characters or less, made up of any ASCII characters except ESC, BS, CAN, and the characters deleted on input.

DESCRIPTION: The Search command causes a search of the edit buffer for the first occurrence of the specified string. The search begins at the buffer location specified by the current position of the buffer pointer. The search may be terminated in two ways:

(1) A match with the specified string is found. In this case, the buffer pointer is positioned immediately after the last character of the matched string.

(2) The search reaches the end of the edit buffer. In this case, a message

CANT'T FIND "string"

is printed. When no match is found, the buffer pointer resumes its position prior to the search.

EXAMPLE:

```
@B$$

@S40$$

@T$$
  OPT S SELECT PRINTING SYMBOLS

@SOTP$$

CAN'T FIND "OTP"


@B$$

@SOTP$$

@0L$$

@T$$
20   OTP M MEMORY FILE OPTION

@
```

The B command sets the buffer pointer at the beginning of the edit buffer.

The S40 command searches for 40.

Since the Editor came back with @, it found 40 and positioned the buffer pointer immediately after it. The T command demonstrates this by printing the portion of the line numbered 40 which follows the 40. Note that if line numbers are included in the program being edited, the Search command can be used to easily move the buffer pointer to any given line by searching for the appropriate line number.

The SOTP command searches for OTP.

The Editor printed CAN'T FIND "OTP" indicating that OTP does not occur between the current buffer pointer location and the end of the edit buffer.

Then the B command moves the buffer pointer to the beginning of the edit buffer.

Now OTP is searched for again by re-entering the command SOTP.

This time the Editor prints only @, indicating that it found OTP. The 0L command moves the buffer pointer to the beginning of the line containing OTP.

The T command prints the line containing OTP.

2.8.2.6 N -- SEARCH FILE

FORMAT: Nstring

The "string" argument is a string of 16 characters or less, made up of any ASCII characters except ESC, CAN and BS.

DESCRIPTION: The search file command causes a search of the edit buffer and file for the first occurrence of the specified string. The search begins at the buffer location specified by the current position of the buffer. If the specified string is not found in the buffer, the command effectively does a B50PZASstring command which, in order: 1) punches, starting at the beginning of the buffer a maximum of 50 lines to the output file; 2) moves the buffer pointer to the end of the buffer; 3) if sufficient buffer room exists, appends to the end of the buffer a maximum of 50 lines from the input file then 4) searches the appended lines for the first occurrence of the specified string. This procedure is repeated until search file is terminated in one of two ways: 1) When the specified string is located, the buffer pointer is positioned immediately after the last character of the string. 2) If the specified string cannot be found within the input file, a "CAN'T FIND string" message is printed. Following this, the E command is essentially executed i.e., text remaining in the edit buffer, an end of file character (the ASCII character SUB), and a blank trailer are written to the system punch device. The Editor then prints "@" and awaits further commands.

2.8.3 INSERT/DELETE/CHANGE OPERATIONS

These operations permit the insertion of text into the edit buffer, deletion of text in the buffer, or replacement of an existing character string with another string.

2.8.3.1 I -- INSERT

FORMAT: Istring

The "text" argument may include any ASCII characters except ESC, CANCEL and BS.

DESCRIPTION: The Insert command is used to insert lines or characters of text into the edit buffer. Text is inserted at the location specified by the current buffer pointer. Following the Insert operation, the pointer is positioned after the last character of inserted text. The ASCII characters Null, Rubout, Linefeed, Readeron, Punchon, Readeroff, and Punchoff are deleted from the inserted text by the Editor.

EXAMPLE:

```
$B$$

$2T$$
10 NAM PGM
20  OTP M MEMORY FILE OPTION

$2M$$

$I $$

$L$$

$I15  * REVISION 1
$$

$B$$

$3T$$
10  NAM PGM
15  * REVISION 1
20  OTP M MEMORY FILE OPTION

$
```

The B command sets the buffer pointer at the beginning of the edit buffer.

The 2T command prints the first two lines of the edit buffer.

The 2M command moves the buffer pointer to immediately past the 0 to 10.

The I (Space) command inserts a space at the current location of the buffer pointer.

The L command moves the buffer pointer to the beginning of the next line so that a line can be inserted between the lines numbered 10 and 20.

The I15 *REVISION 1 (Carriage Return) command then inserts the line numbered 15 between the lines numbered 10 and 20.

The B command sets the buffer pointer at the beginning of the edit buffer.

The 3T command then prints the first three lines of the edit buffer and confirms that the information was inserted.

2.8.3.2 D -- DELETE CHARACTERS

FORMAT: nD

n is a decimal integer in the range -254 n 255. If omitted, the value is assumed to be 1.

DESCRIPTION: The Delete Characters command deletes n characters from the edit buffer, beginning at the current position of the edit buffer pointer. If n is positive, n characters following the current pointer position are deleted. If negative, n characters preceding the current pointer position are deleted. If there are less than n characters between the edit buffer pointer and the end (beginning) of the edit buffer, then the characters will be deleted and the buffer pointer will point to the end (beginning) of the edit buffer.

EXAMPLE:

```
*B$$

*4T$$
10  NAM PGM
15  * REVISION 1
20  OTP M MEMORY FILE OPTION
30  OPT O OUTPUT OBJECT TAPES

*S15$$

*D$$

*STAPES$$

*-1D$$

*-4T$$
10  NAM PGM
15 * REVISION 1
20  OTP M MEMORY FILE OPTION
30  OPT O OUTPUT OBJECT TAPE
*
```

The B command sets the buffer pointer at the beginning of the edit buffer.

The 4T command prints the first four lines of the edit buffer.

The S15 command searches for 15 and locates the buffer pointer immediately after it.

The D command deletes the character after the buffer pointer, in this case a space.

The STAPES command searches for TAPES and positions the buffer pointer immediately after it.

The - 1D command deletes the character before the buffer pointer, which is the S of TAPES.

The - 4T command prints the four lines preceding the buffer pointer and confirms that the changes were made.

2.8.3.3 K -- KILL (DELETE) LINES

FORMAT: nK

n is a decimal integer in the range -254 n 255. If omitted, the value is assumed to be 1.

DESCRIPTION: The Kill Lines command is similar to the Delete Characters command, except that n specifies a number of lines to be deleted from the edit buffer, rather than a number of characters. If n is positive, n lines following the current pointer position are deleted. If negative, n lines preceding the current position are deleted. If fewer than n lines remain between the current pointer position and the end (beginning) of the edit buffer, then the lines are deleted and the buffer pointer will point to the end (beginning) of the edit buffer.

If n is zero, the characters between the buffer pointer and the immediately preceding Carriage Return will be deleted.

EXAMPLE:

```
@B$$

@7T$$
10  NAM PGM
15 * REVISION 1
20  OTP M MEMORY FILE OPTION
30  OPT O OUTPUT OBJECT TAPE
40  OPT S SELECT PRINTING SYMBOLS
50  ORG 8192
60  LDA B ADDR

@S60$$

@0L$$

@K$$

@B$$

@7T$$
10  NAM PGM
15 * REVISION 1
20  OTP M MEMORY FILE OPTION
30  OPT O OUTPUT OBJECT TAPE
40  OPT S SELECT PRINTING SYMBOLS
50  ORG 8192
70 COUNT EQU @8  @ INDICATES OCTAL

@
```

The B command sets the buffer pointer to the beginning of the edit buffer.

The 7T command prints the first seven lines of the edit buffer.

The S60 command searches for 60 and sets the buffer pointer immediately after it.

The 0L command moves the buffer pointer to the beginning of the line numbered 60.

The K command deletes the line that the buffer pointer is currently at. If the buffer pointer were somewhere else besides the beginning of a line, the K command would delete the characters from the buffer pointer through the following Carriage Return.

The B command sets the buffer pointer to the beginning of the edit buffer.

The 7T command prints the first seven lines of the edit buffer confirming that the line numbered 60 was deleted.

2.8.3.4 C -- CHANGE

FORMAT: Cstring1$string2

"string 1" is a string of 16 ASCII characters or less. The size of "string 2" is not limited. These strings may include any ASCII characters except ESC, BS, and CAN, and the characters deleted on input. The two strings need not be of the same length.

DESCRIPTION: The Change command searches the edit buffer from the current buffer pointer position. When the first occurrence of "string 1" is found, those characters are changed to "string 2". The buffer pointer will be moved to the end of "string 2".

If "string 1" cannot be found, the message

CAN'T FIND "string 1"

is printed, and the position of the buffer pointer is unchanged.

EXAMPLE:

```
)B$$

)5T$$
10  NAM PGM
15 * REVISION 1
20  OTP M MEMORY FILE OPTION
30  OPT O OUTPUT OBJECT TAPE
40  OPT S SELECT PRINTING SYMBOLS

)CSYMBOLS$OF SYMBOLS$$

)COTP$OPT$$

CAN'T FIND "OTP"

)B$$

)COTP$OPT$$

)B$$

)5T$$
10  NAM PGM
15 * REVISION 1
20  OPT M MEMORY FILE OPTION
30  OPT O OUTPUT OBJECT TAPE
40  OPT S SELECT PRINTING OF SYMBOLS

)
```

The B command sets the buffer pointer at the beginning of the edit buffer.

The 5T command then prints the first five lines of the edit buffer.

The CSYMBOLS$OF SYMBOLS command searches for SYMBOLS and substitutes for it OF SYMBOLS. The COTP$OPT command searches for OTP. However, the Editor prints CAN'T FIND "OTP" indicating that OTP does not occur between the current buffer pointer location and the end of the edit buffer.

The B command moves the buffer pointer to the beginning of the edit buffer so that the complete edit buffer can be searched for OTP.

This time the command COTP$OPT locates OTP and substitutes OPT for it.

The B command sets the buffer pointer at the beginning of the edit buffer.

The 5T command prints the first five lines of the edit buffer and confirms that the changes were made.

2.8.4 EXIT EDITOR

This operation allows exit from the Resident Editor program.

2.8.4.1 X -- EXbug

FORMAT: X

DESCRIPTION: The EXbug command causes control to be returned to the EXbug program.

2.9 EDITOR COMMAND CHAINING

The M6800 Resident Editor can accept sequences of edit commands. A command sequence consists of a sequence of editor commands and associated arguments, terminated by two ESC characters. Commands with arguments which follow them must be separated from subsequent commands with a single ESC character.

Two ESC characters mark the end of a command string and cause the Editor to begin execution. Commands in a string are executed from left-to-right, in the order in which they were entered. All commands preceding an illegal command in the command chain are executed.

EDITOR COMMAND CHAINING EXAMPLES

The following example assumes that the information contained on the tape in the Append example is the only content of the edit buffer.

EXAMPLE: (editor command chaining)

```
@BST$$
10 NAM PGM
20  OTP M MEMORY FILE OPTION
30  OPT O OUTPUT OBJECT TAPES
40  OPT S SELECT PRINTING SYMBOLS
50  ORG 8192
60  LDA B ADDR
70 COUNT EQU @8  @ INDICATES OCTAL
80 START LDS #STACK INZ STACK POINTER

@2MI $0LTCS$$
10  NAM PGM

@S60$0LKS@$DI993$0LTCOTP$OPT$$
70 COUNT EQU @3  @ INDICATES OCTAL

CAN'T FIND "OTP"


@BTSOTP$OPT

@BCOTP$OPT$BLI15 * REVISION 1
$SSYMBOLS$-7MIOF $$

@BSTE$$
10  NAM PGM
15 * REVISION 1
20  OPT M MEMORY FILE OPTION
30  OPT O OUTPUT OBJECT TAPE
40  OPT S SELECT PRINTING OF SYMBOLS
50  ORG 8192
70 COUNT EQU @3  @ INDICATES OCTAL
80 START LDS #STACK INZ STACK POINTER

@
```

The B8T command chain sets the buffer pointer at the beginning of the edit buffer and then prints the first eight lines of the buffer.

The next command chain, 2MI $OLTCS$$, does the following:

1. Moves the buffer pointer forward two characters (2M).

2. Inserts a space (I $).

3. Moves the buffer pointer to the beginning of the current line (OL).

4. Prints the line (T).

5. Moves the buffer pointer to just after the next S in the buffer, which is in TAPES, and then deletes the S (CS$$). Note that the C command can be used to delete character by not including a string 2 in the command. However, when used in this manner, the C command can only occur as an individual command or at the end of a command chain since two ESC characters must occur together due to the deletion of string 2.

The command chain

S60$OLKS@$DI993$OLTCOTP$OPT$$

performs the following actions:

1. Searches for 60 and moves the buffer pointer to just after it (S60$).

2. Moves the buffer pointer to the beginning of the line it is in (OL).

3. Deletes the line (K).

4. Searches for @ and moves the buffer pointer to just after it (S@$).

5. Deletes the next character, which is 8 (D).

6. Inserts a 3 (I9(backspace)3$). Note that a 9 was erroneously entered and was deleted using a backspace (Control H) character. The backspace character may be used to delete as many previous characters in a command as required. The Editor prints the character deleted by each backspace.

7. Moves the buffer pointer to the beginning of the line the buffer pointer is currently in (OL).

8. Types the line the buffer pointer is currently at (T).

9. Searches for OTP (COTP$OPT$$). But the Editor does not find it between the current buffer pointer location and the end of the buffer as it indicates by printing CAN'T FIND "OTP".

The command chain BTSOTP$OPT was not executed since a Control X character terminated the command chain. The Control X character deletes all commands up to the last prompt and prints another prompt.

The command chain BCOTPOPTBLI15* REVISION 1(CR)$SSYMBOLS$-7MIOF $$ does the following:

1. Moves the buffer pointer to the beginning of the edit buffer (B).

2. Changes OTP to OPT and moves the buffer pointer to just after OPT (COTPOPT).

3. Moves the buffer pointer back to the beginning of the edit buffer (B).

4. Moves the buffer pointer to the beginning of the next line (L).

5. Inserts the line 15 * REVISION 1 at the current buffer pointer location (I15*REVISION 1 (CR)$). Note that to insert a line the buffer pointer is moved to the beginning of the line that is to follow the inserted line. Then the line is inserted using the I command. A Carriage Return should be the last character of the inserted line.

6. Searches for SYMBOLS and moves the buffer pointer to just after it (SSYMBOLS$).

7. Moves the buffer pointer back seven characters (-7M) which puts it at the beginning of SYMBOLS.

8. Inserts OF(space) at the current buffer pointer location (OF $$).

The following actions are performed by the command chain B8TE:

1. Moves the buffer pointer to the beginning of the edit buffer (B).

2. Prints the eight lines following the current buffer pointer location (8T).

3. Ends editing on the contents of the edit buffer by punching a tape of the buffer contents and any remaining tape in the System Reader Device (E).

2.10 RESIDENT EDITOR MESSAGES

Table 2-2 lists and identifies the Resident Editor messages.

TABLE 2-1 EDITOR COMMAND SUMMARY

COMMAND	DESCRIPTION
A	Append. Appends input text from the System Reader Device to the edit buffer.
B	Beginning. Moves the edit buffer pointer to the beginning of the edit buffer.
Cstring1$ string2	Change. Replaces the first occurrence of "string 1" with "string 2".
nD	Delete. Deletes n characters from the edit buffer.
E (tape)	End. Terminates an edit operation by writing the contents of the edit buffer to the output tape and copying the remainder of the input tape to the output tape. Returns control to the editor.
E (disc)	End. Terminates an edit operation by writing the contents of the edit buffer to the output file and copying the remainder of the input file to the output file. Returns control to the disc operating system.
F (tape)	Tape Leader/Trailer. Writes 50 NULL characters into the system punch device.
F (disc)	The F command is ignored.
Istring	Insert. Inserts characters or lines of text into the edit buffer.
nK	Kill lines. Deletes n lines from the edit buffer.
nL	Line. Moves the edit buffer point n lines.
nM	Move character pointer. Moves the edit buffer pointer n characters.
Nstring (tape)	Search File. Searches file for first occurrence of "string".
Nstring (disc)	Search File. Searches file for first occurrence of "string". If "string" is not found, returns control to the disc operating system.
nP	Punch. Punches n lines from the edit buffer to the System Punch Device.
Sstring	Search. Searches the edit buffer for the first occurrence of "string".

TABLE 2-1 EDITOR COMMAND SUMMARY
(continued)

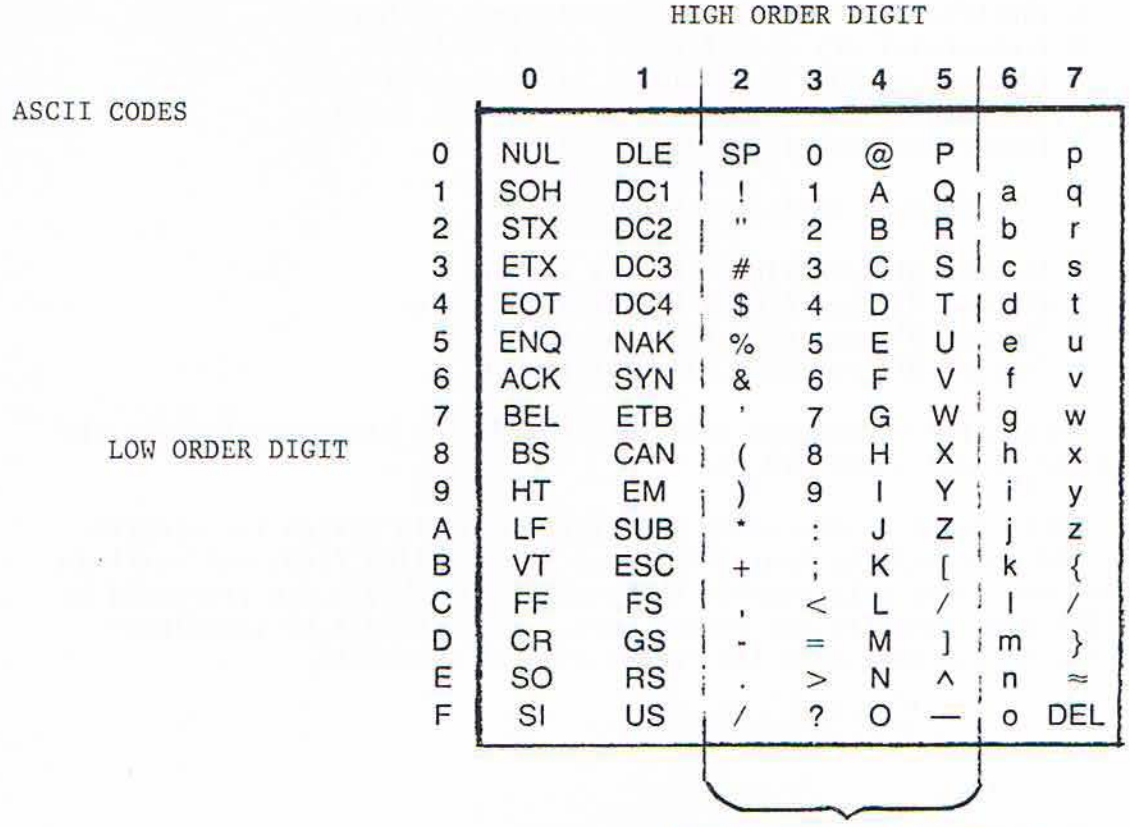
COMMAND	DESCRIPTION
nT	Type. Types n lines from the edit buffer to the System Console Device.
X (tape)	EXbug. Returns control to EXbug.
X (disc)	The X command is an illegal command in the disc version of the editor.
Z	End of edit buffer. Moves the edit buffer pointer to the end of the edit buffer.
Control H	Backspace. Causes the last character entered in the command mode to be typed on the System Console Device and deleted from the command.
Control X	Cancel. Causes all commands following the last prompt to be deleted and another prompt to be typed.

TABLE 2-2 EDITOR MESSAGES

MESSAGE	DESCRIPTION
M6800 RESIDENT EDITOR n.n	Printed upon initiation of editor. Revision is specified by n.n.
@	Prompt. Editor is waiting for a command.
????	Illegal command.
CAN'T FIND "string"	Editor cannot find the string specified by Search or Change command.
BELL	The editor rings the bell in the System Console Device when the user attempts to enter further commands into a full command buffer. The user must delete (backspace) two characters in order to terminate the command with two ESC characters.

APPENDIX A

THE M6800 ASSEMBLER CHARACTER SET



ASCII CODES RECOGNIZED BY
THE M6800 ASSEMBLER

The hexadecimal ASCII code for a character is obtained from the headings of the row and column that intersect at that character. For example, the hexadecimal code for the alphabetical character 'K', from the intersection of column 4 and row B, is '4B'.

The character set recognized by the Motorola M6800 Co-Resident Assembler is a subset of ASCII (American Standard Code For Information Interchange, 1968). Characters in this subset are those included in columns 2 through 5 in the above table - i.e., any ASCII character having a hexadecimal value from 20 (SP) through 5F (_).

Any character of this subset may be used in a source statement. Characters retain their conventional meanings in the label or comment fields, but certain ones acquire special meanings when used as prefixes or suffixes to operands, as follows:

Operand Prefix Denotations

(pound sign) specifies the immediate mode of addressing
$ (dollar sign) specifies a hexadecimal number
@ (commercial at) specifies an octal number
% (percent symbol) specifies a binary number
' (apostrophe) specifies an ASCII literal number
& (ampersand) specifies a decimal number

Operand Suffix Denotations

B (letter B) specifies a binary number
H (letter H) specifies a hexadecimal number
O (letter O) specifies an octal number
Q (letter Q) specifies an octal number

Separating characters used within a source statement include the space (20_{16}) and comma ($2D_{16}$).

ASCII codes generated by the system console device for program entry and text editing operations such as tab, line feed, and carriage return, and which fall outside the recognized subset, are processed by the EXORciser firmware and echoed back to the console or translated into the appropriate code for recognition at assembly.

APPENDIX Z

CO-RESIDENT ASSEMBLER AND EDITOR SUMMARY

The detachable reference card provided below summarizes the loading procedures, instructions, and error messages for the Co-Resident Assembler and Editor programs.

CO-RESIDENT ASSEMBLER AND EDITOR REFERENCE CARD

TEAR OUT FOR HANDY POCKET REFERENCE

CO-RESIDENT ASSEMBLER

LOADING AND INITIATING (User Responses Underlined)

CASSETTE/PAPER TAPE

EXbug n.n LOAD
SGL/CONT S
X ASMn.n
EXbug n.n MAID
*100;G
M6800 RESIDENT ASSEMBLER n.n
COPYRIGHT MOTOROLA 1976
ENTER PASS: 1P, 1S, 2P, 2L, 2T

DISKETTE

EXbug n.n LOAD
SGL/CONT S
X ASMn.n
EXbug n.n MAID
*E800;G
M6800 EDOS VER n.n
!ASM,2,PGMOT,PGM
M6800 RESIDENT ASSEMBLER n.n
COPYRIGHT MOTOROLA 1976

PASS CONTROLS AND OPTIONS

CASSETTE/PAPER TAPE

1P	Pass 1, clears symbols
1S	Pass 1, retains symbols
2P	Pass 2, generates assembly listing and object tape
2L	Pass 2, assembly listing only
2T	Pass 2, object tape only

DISKETTE

ASM,2	Generates assembly listing and object file
ASM,3	Generates assembly listing
ASM,4	Generates object file

ASSEMBLER DIRECTIVES

ASSEMBLY CONTROL

NAM	Program name
ORG	Origin
END	Program end

LISTING CONTROL

PAGE	Top of page
SPC n	Skip "n" lines
OPT NOO	Generate no object tape
OPT O	*Generate object tape
OPT M	Write object code to memory
OPT NOM	*Write no object code to memory
OPT S	Print symbols
OPT NOS	*Print no symbols
OPT L	*List assembled date
OPT NOL	List no assembled data
OPT P	*List assembled data in page format
OPT NOP	List assembled data in unpaged format
OPT G	*List data generated by FCC, FCB, and FDB
OPT NOG	List first line of data generated by FCC, FCB, and FDB

*Selected by default

DATA DEFINITION/STORAGE ALLOCATION

FCC	Form constant character
FCB	Form constant byte
FDB	Form double byte

SYMBOL DEFINITION

EQU	Assign permanent value

ASSEMBLER ERROR MESSAGES

NO.	MESSAGE
201	NAM directive used in other than first source statement. NAM used twice in same program (EXbug 1.2 only).
202	EQU directive syntax requires label (EXbug 1.2 only).
204	Source statement syntax incorrect.
205	Label not allowed. Label syntax incorrect.
206	Symbol previously defined.
207	Invalid directive or op code mnemonic.
208	Destination beyond relative branch range.
209	Address mode unallowed with op code.
210	Byte overflow. One-byte expression converts to value $>255_{10}$ or $<-128_{10}$.
211	Undefined symbol.
213	EQU directive requires label. (EXbug 1.2, error 213 = redefined symbol error.)
216	Directive operand error.
218	Overwrite attempted into non-existent memory.
220	Redefined label field symbol — pass 2 value differs from pass 1 value.
221	Symbol table overflow.

CO-RESIDENT EDITOR

LOADING AND INITIATING

```
X
EXbug n.n LOAD
SGL/CONT S
X EDTn.n
EXbug n.n MAID
*103:G
M6800 RESIDENT EDITOR n.n
@
```

LOADING AND INITIATING USING ASSEMBLER OVERWRITE

```
X
EXbug n.n LOAD
SGL/CONT S
X EDTn.n
EXbug n.n MAID
*300/00 FF
*103:G
M6800 RESIDENT EDITOR n.n
@
```

EDITOR COMMANDS

A (Append)
B (Beginning)
Cstring1$string2 (Change)
nD (Delete n characters)
E (End)
F (Punch 50 nulls)
Istring (Insert)
nK (Kill n lines)
nL (Move pointer n characters)
Nstring (Search file for "string")
nP (Punch n lines)
Sstring (Search edit buffer for "string")
nT (Type n lines)
X (Return to EXbug, tape only)
Z (End of buffer)
Control H (Backspace)
Control X (Cancel)

EDITOR MESSAGES

M6800 RESIDENT EDITOR n.n
@ (n.n version of the editor has been called)
@ (Prompt, editor is asking for a command)
???? (Illegal command)
CAN'T FIND "string" (N or S command error)
BELL RINGS (Command buffer full. Type 2 backspace, 2 ESC characters to terminate)

MOTOROLA Integrated Circuit Division

microsystems • 3102 North 56th Street • Phoenix, Arizona 85018