SWTPC 6800 CO-RESIDENT EDITOR/ASSEMBLER

The CO-RES editor/assembler package allows the user to considerably reduce the amount of time involved in writing programs. The editor allows the composition and modification of assembler level programs written in mneumonics to be stored in memory. The assembler then changes this source program into hexadecimal machine code that the processor can understand. Once a source program has been written it can be saved and re-loaded using either paper or cassette tape.

This manual is designed to supplement the information given in chapters 5 and 6 of the MP-68 Microprocessor Programming Manual or the MP6800 Co-Resident Assembler Reference Manual. Some of the information in these chapters pertains to a time sharing editor/assembler and is not relevant to the CO-RES editor/assembler.

The CO-RES package consists of two separate programs-an editor and an assembler. To use the package set the program counter to Ø1ØØ and type G. The computer will respond: READY. At this point the editor is the current program in operation. The editor section of this manual describes the appropriate editor commands.

TEXT EDITOR

The text editor resident within this editor/assembler package accepts numbered lines of text and stores them in computer memory in sequence of line numbers. It allows for the insertion, replacement and deletion of text lines. The program uses memory within the MIKBUG® ROM and its associated RAM.

PROCEDURES FOR USE

GENERAL

- * A carriage return (C/R) signifies the completion of entry of data
- * Line feeds are superflous, as a C/R will initiate a linefeed.
- * A "CONTROL X" typed anytime during data entry (prior to a C/R) will delete the entire input line and allow for its reentry.
- * A "CONTROL O" or a "CONTROL H" typed will backspace to last character entered (the computer will echo a _ to signify the backspace in the case of the CTRL O)
- * Line numbers can consist of one to four digits separated by a space from the rest of the line.
- * Hitting the "BREAK" key at anytime will terminate the current computer operation and cause a READY response. (Rapid multiple hitting of the CTRL X key on terminals without a BREAK key will cause a DELETED message and return control to the READY mode.

ENTRY OF LINES

- * Type in a line number, followed by a space, followed by text. Terminate with a C/R.
- * The input buffer is 72 characters long. Extra characters are ignored.
- * Entry of more than four digits in a line number will cause the last four digits entered to be used.

MIKBUG® is a registered trademark of Motorola, Inc.

INSERTION OF LINES

* To insert a line between two lines, merely use a line number in between the two line numbers involved. (for example, to insert a line between lines 60 and 70, use any line number between 61 and 69)

REPLACEMENT OF A PREVIOUSLY ENTERED LINE

* Merely retype the line using the same line number as the line you wish to replace.

DELETION OF A PREVIOUSLY ENTERED LINE

* Merely retype the line using the same line number as the line you wish to replace.

DELETION OF A PREVIOUSLY ENTERED LINE

* Type the line number, followed immediately by a C/R (No intervening space).

EDITOR COMMANDS

LISTING THE FILE

* LIST - Lists the entire file.

* LIST N1-N2 - Lists the file, starting at the line N1 and ending at the line N2.
List N1 - lists line N1

RESEQUENCING THE FILE

* RESEQ - renumbers the file, starting with 0010 and incrementing by 10

* RESEQ N1 - renumbers the file, starting with the line number specified and incrementing by 10. Entering a starting line number that would cause a 4 digit overflow causes the command to be treated as a RESEQ without a line number.

DESEQUENCING THE FILE

* DESEQ - causes the removal of all line numbers.
CAUTION - The file must be resequenced prior to entering additional lines.

SAVING THE FILE

* SAVE - causes a specilly formatted output of the file with leader and trailer.

LOADING A FILE

* LOAD - Inputs a file form a tape previously created by a SAVE command. See Appendix B for the entry of source files generated by other editors.

SEARCHING FOR A CHARACTER STRING

* SEARCH N1 "STRING" - searches the edit buffer for the character string STRING starting from line number N1. The line containing the first occurance of STRING will be output on the terminal. STRING can be any combination of letters, numbers and spaces, but not C/R's.

* SEARCH "STRING" - searches the edit buffer from the beginning and returns the first occurance of STRING.

OTHER COMMANDS

* NEW - clears the file buffer and reinitializes all pointers and counters. A clear is automatically done the first time the editor is accessed.

* AUTO - causes the computer to type sequential (by 10) line numbers. Allows for data entry without operator entry of line numbers.
* AUTO N1- same as AUTO but with N1 specifying the starting line number. (the AUTO mode is exited by hitting the break key or by hitting CONTROL X to delete the computer generated line number and then entering any other command besides AUTO).
* PATCH - branches the program to MIKBUG(R) control. Typing a G will re-enter the editor without destroying the text file as long as the RESET button has not been pressed.
* ASSEMBLE - jumps control to the resident assembler. All necessary editor pointers and the file buffer are retained for future re-editing without loading the source program in through tapes.
* SIZE - outputs, in hexadecimal, the current address of the next available memory location on top of the edit buffer. This command can be used to keep track of how much memory is available. The SIZE command does not include the symbol table.
* PRINTER - transfers all output to a parallel interface located at $801C. Parallel output is PR-40 compatible.
* TERMINAL - transfers all output back to the CRT terminal or TTY.

EDITOR MESSAGES

* READY - Computer response upon the start of the editor program and upon completion of each editor command.
* WHAT? - Improper command, or improper line number (maybe no space following the number, or the command was misspelled).
* CORE FULL - The text buffer has now taken all available RAM memory. Lines must be deleted or the file cleared. The last line was not stored. Remember that enough room must be left in memory for the assembler to build a symbol table if an assembly will be done.

RESIDENT ASSEMBLER

The SWTPC 6800 assembler may be used with the computer to translate source programs for the 6800 microprocessor to object form. The M6800 Microprocessor Programming Manual describes the common features of the assembler and error messages. This discription emphasizes features and procedures unique to this assembler.

Standard input to the assembler is source language stored in the computer's memory. The area assembled is the editor's file to the first END command. The assembler will do two passes over this source file. During the first pass the assembler generates a record of the location of each symbol used in the program forming what is called a symbol table. Instruction syntax is also checked during this pass. The second pass actually generates the object output in the form of an object tape, a source-object listing, or both simultaneously. When the object code is punched to tape, up to eight characters from the address feild of the NAM directive will be inserted in the header record of the object tape.

ASSEMBLER OPERATION

Since the assembler is a two pass assembler, the source file must be read twice to generate the required object output. Upon entering an ASSEMBLE command in the editor the computer will respond:

```
    ENTER PASS:  1P, 2P, 2L, 2T
```

At this time the appropriate pass selection must be entered. If you accidently type in erroneous characters the input message will repeat.

PASS 1 OPTION

* 1P - Pass 1 produces a table of symbols which appear in the program and the corresponding memory addresses to which they are assigned. This table is used in Pass 2 to determine the address field for instructions which reference memory symbolically. Program syntax is also checked in Pass 1 and errors listed.

NOTE: The 1S pass associated with some other assemblers is not supported in this version of CO-RES.

PASS 2 OPTIONS

* 2P - Pass 2 rereads the source file and uses information in the symbol table to produce the assembled output. Pass 2 can produce both an object tape and an assembly listing, but the actual listing of the program will be sent to the punch device. Normally Pass 2 should be repeated to generate both output forms. (Using a 2L and a 2T).

* 2L - The L option for pass 2 is used to generate only an assembly listing (no object tape).

* 2T - The T option for pass 2 is used to generate only an object tape (no assembly listing).

NOTE: One pass operation - For source programs which have so symbolic forward references, pass 1 may be omitted. For short programs with only a few forward references, it is also possible to omit pass 1. In this case forward references will be flagged with an ERROR 211 and assembled with an address field of FFFF. The correct address can be patched after the symbol table is printed at the completion of the assembly.

* CTRL X - Entering a CONTROL X will return control to the editor.

Output from the assembler can be generated or altered by using the following assembler options. Multiple comma-separated options may be specified with a single statement.

*	OPT O	The assembler will generate object tape.
*	OPT NOO	No object tape (selected by default).
*	OPT S	The assembler will print the symbols at the end of pass 2.
*	OPT NOS	No printing of symbols (selected by default).
*	OPT NOL	The assembler will not print a listing of the assembled data.
*	OPT L	The listing of assembled data will be printed (selected by default).

*	OPT NOP	The assembler will inhibit format paging of the listing.
*	OPT P	The listing will be paged (selected by default).
*	OPT NOG	Causes only one line of data to be listed from the assembler directives FCC, FCB, FDB.
*	OPT G	All data generated by the FCC, FCB and FDB directives will be printed (selected by default).

More detailed information on the assembler directives can be found in Appendix B of the M6800 Microprocessor Programming Manual, or in the M6800 CO-RESIDENT EDITOR/ASSEMBLER manual. The OPT M option is not a part of this assembler. Appendix C lists the error messages that can be encountered during an assembly operation. Chapters 5 and 6 of the programming manual also contain much useful information on assembler use and operation. The G.E.Timeshare examples do not pertain to the SWTPC 6800 resident assembler.

The assembler has a few "design features" which may prove important to the user:

1.) All forward references to data are resolved by using extended addresses, even if the data can be addressed directly. The code produced works, but wastes space. Any data that can be addressed directly should be defined (used as a label) before it is referenced.

2.) The assembler will not reduce JMP and JSR to BRA and BSR, even if the target address is within range.

3.) Conversely, the assembler can not substitute JSR or JMP for BSR or BRA if the target address is outside the range of relative addressing.

4.) Duplicate symbol definitions confuse the assembler. It will lose both definitions and resolve addresses incorrectly.

5.) DO NOT mis-spell or leave out the END directive. If you accidently spell the directive EMD, ENF, etc. or put the directive in the wrong field, the assembler will enter an infinite loop of error messages and must be reset.

6.) The assembler does not write out the starting address of your program - you will have to get this yourself when you load the program.

7.) The assembler will not punch object tape when either the 2T of 2P pass is selected unless the source program contains an OPT O directive.

APPENDIX A
Use of the Control Interface for Read/Punch On/Off Decoding

The MP-EA software contains subroutines to send out pulses to unused pins of the PIA integrated circuit on the serial control interface that can be used for automatic reader/punch controls. These pulses can be used if you are using a SWTPC AC-30 cassette interface and a 300 baud terminal in which access to the control command decoding is denied. These subroutines work for both the editor and assembler and contain the necessary delays for use on cassette tape interfaces.

If you intend to use the read/punch control logic put out on the control interface board, make the following connections from the indicated pins of IC1 on the MP-C control interface board to the specified pins of a twelve pin male connector shell. The connector pinning shown below is correct for a SWTPC AC-30 cassette interface and will need modification for other units. Be sure to make the wires long enough to reach your AC-30 where the connector will be plugged. If you have access to your terminal's 16X baud rate clock, the terminal's clock buss should be broken and the 16X clock OUT and 16X clock IN lines brought out to the same connector:

FROM	TO
MP-C IC1 pin 7 (read on)	12 pin male shell female pin #1
MP-C IC1 pin 4 (punch on)	12 pin male shell female pin #2
MP-C IC1 pin 6 (read off)	12 pin male shell female pin #3
MP-C IC1 pin 5 (punch off)	12 pin male shell female pin #4
Terminal's 16X clock OUT	12 pin male shell female pin #5
Terminal's 16X clock IN	12 pin male shell female pin #6
MP-C ground	12 pin male shell female pin #12

Further instructions are given in the AC-30 instruction set.

APPENDIX B
Loading Source Programs From Other Editors.

In order for the LOAD function of the editor to operate, the data on the tape must be formatted in a special way. The format is as follows:

(02_{16}) (four digit line number and 1 space) (source line) (04_{16})

(four digit line number and 1 space) (source line) (04_{16})

(four digit line number and 1 space) (source line) (04_{16}) (03_{16})

If the source input tape does not conform to the above format the file will not load into memory and/or the editor's line numbering/finding/sequencing subroutines will not work correctly. If your present editor does not format source tapes as above you should write a small program to send out a 02_{16} at the beginning of the tape, an incrementing <u>four</u> digit line number and space (four nulls and 1 space can be used if the file is re-sequenced after it is loaded), the source line, and a 04_{16} to serve as an end of line character. The tape should end with a 03_{16}.

APPENDIX C
Notes on Memory Conservation

Since the CO-RES Editor/Assembler will be used on a microprocessor and not on a 1 Megabyte computer, care should be taken to conserve memory on long programs. Remember, each letter, number and space in the source program takes 1 byte of memory, so keep comments and spaces to a minimum. To help conserve memory type in the source program as:

```
0010  NAM TEST
0020  ORG 0
0030 LINE LDAA #$34
0040  END
```

not as

```
0010       NAM TEST
0020       ORG 0
0030 LINE  LDA A #$34
0040       END
```

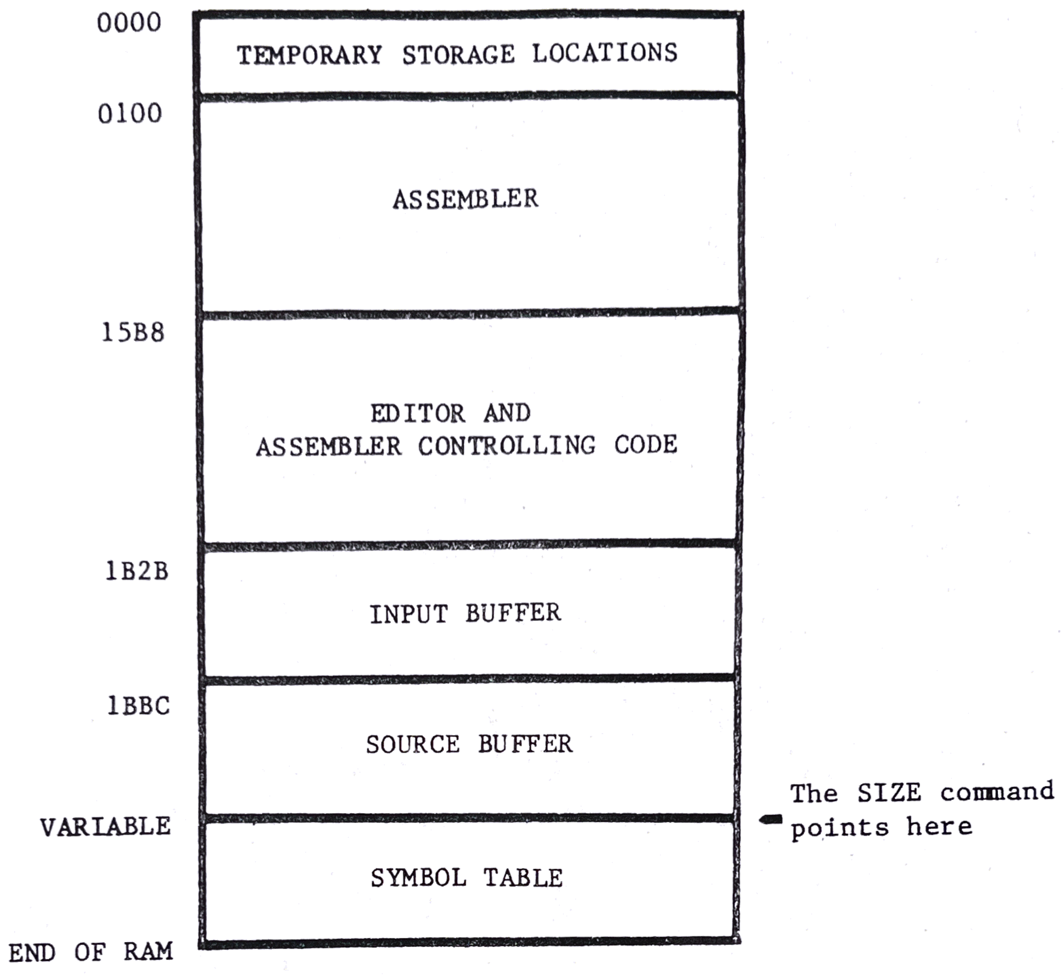
Extra spaces between the line number and the operator makes the source program look prettier, but can waste up to 5 extra bytes per line. Also typing LDA A rather than LDAA wastes 1 extra byte.

Remember also that each line flagged with a label will cause the assembler to allocate 8 bytes for that label in the symbol table, so use labels only where necessary.

APPENDIX D
Memory Map and Locations of Interest

0100	Hard start address (source buffer cleared)
0103	Soft start address (buffer left intact)
1BBC	Start of source buffer
17A3 and 1A7F	Contains the PIA address of the printer
060B - 060C	Sets the delay time between a PUNCH ON and the punching of data.

Memory Map


0000
TEMPORARY STORAGE LOCATIONS
0100
ASSEMBLER
15B8
EDITOR AND ASSEMBLER CONTROLLING CODE
1B2B
INPUT BUFFER
1BBC
SOURCE BUFFER
VARIABLE
SYMBOL TABLE
END OF RAM
The SIZE command points here

In Case of Problems

In the event that the CO-RES editor/assembler will not load or work properly, go back and verify that all memory in the system is 100% operational by running the MEMCON and ROBIT diagnostics. If you are using cassette tapes try both sides. Remember, one side (with the shorter leader) is binary and must be loaded with the reader locked in the ON position; the other side is regular ASCII.

Example Program

```
 ┌──── LINE NUMBER
 │    ┌─LABEL FIELD (1 space after line number)
 │    │┌─OPERATOR FIELD (at least one space beyond label)
 │    ││    ┌─OPERAND FIELD (1 sp after operator field)
 │    ││    │
0010 |NAM EXAMPLE
0020 *THIS PROGRAM WRITES THE WORDS
0030 *EDITOR-ASSEMBLER TEST ON A
0040 *CT-1024 OR TTY
0050  OPT 0,NOG
0060  SPC 1        ┌──────NOTE ONLY ONE SPACE
0070 PDATA1 EQU $E07E
0080 CONTRL EQU $E0E3
0090  ORG 0 ─────────────THERE MAY BE MORE THAN ONE SPACE HERE
0100 START LDX #MSG
0110  JSR PDATA1
0120  JMP CONTRL     /COMMENT FIELD (at least 1 space beyond operand)
0130 MSG FDB $1016 HOMEUP,EOF
0140  FCC /EDITOR-ASSEMBLER TEST/
0150  FCB 04
0160  SPC 1
0170  ORG $A048
0180  FDB START
0190  END
      └─ BE SURE THE END COMMAND IS IN THE CORRECT FIELD
```

```
PAGE  001    EXAMPLE                ASSEMBLED OUTPUT


00010                            NAM    EXAMPLE
00020                     *THIS PROGRAM WRITES THE WORDS
00030                     *EDITOR-ASSEMBLER TEST ON A
00040                     *CT-1024 OR TTY
00050                            OPT    0,NOG

00070           E07E      PDATA1 EQU    $E07E
00080           E0E3      CONTRL EQU    $E0E3
00090 0000                       ORG    0
00100 0000 CE 0009        START  LDX    #MSG
00110 0003 BD E07E               JSR    PDATA1
00120 0006 7E E0E3               JMP    CONTRL
00130 0009 1016           MSG    FDB    $1016    HOMEUP,EOF
00140 000B 45                    FCC    /EDITOR-ASSEMBLER TEST/
00150 0020 04                    FCB    04

00170 A048                       ORG    $A048
00180 A048 0000                  FDB    START
00190                            END

TOTAL ERRORS 00000
```