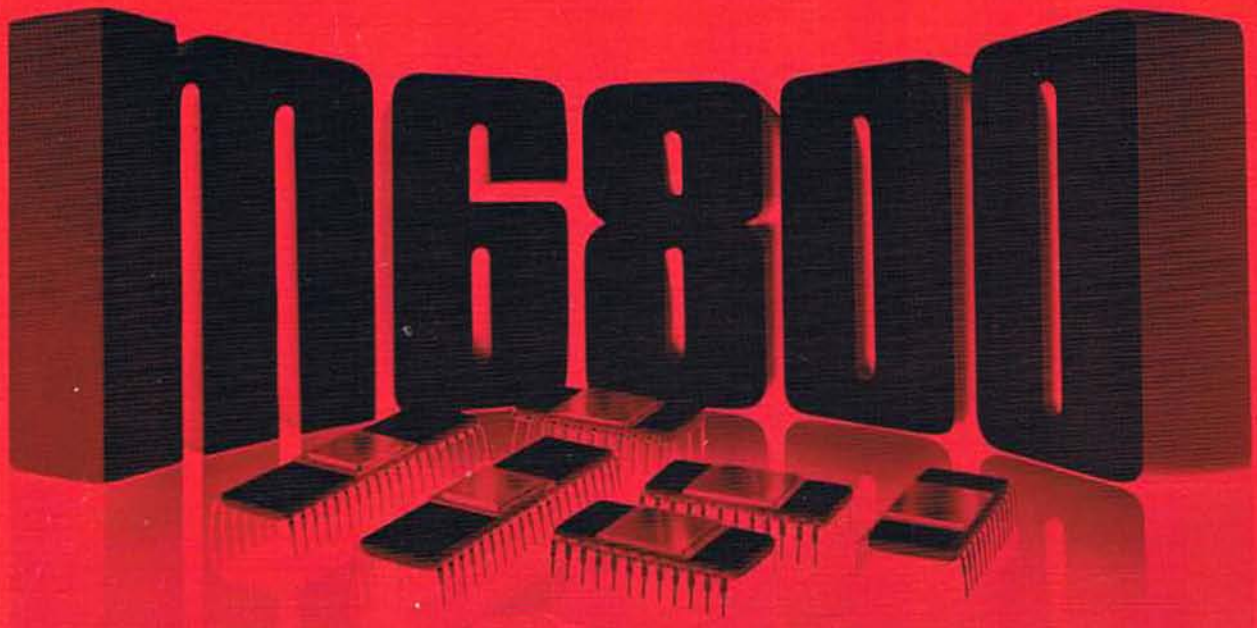
M68CRA(D)

M6800
CO-RESIDENT ASSEMBLER REFERENCE MANUAL



MOTOROLA MICROSYSTEMS

M68CRA(D)
Nov., 1976

M6800

CO-RESIDENT ASSEMBLER

REFERENCE MANUAL

First Edition
Motorola, Inc., 1976

TABLE OF CONTENTS

CHAPTER 1:	GENERAL INFORMATION	1-1
1.1	Introduction	1-1
1.2	M6800 Co-Resident Assembler Language	1-1
1.2.1	Machine Operation Codes	1-1
1.2.2	Directives	1-1
1.3	M6800 Co-Resident Assembler	1-2
1.3.1	Assembler Aims	1-2
1.3.2	Assembler Operation	1-2
1.4	Ordering Information	1-2
1.5	Operating Environments	1-2
1.5.1	Equipment Requirements	1-2
1.5.2	Software Requirements	1-3
CHAPTER 2:	CODING M6800 CO-RESIDENT ASSEMBLER LANGUAGE PROGRAMS	2-1
2.1	Source Statement Format	2-1
2.1.1	Sequence Numbers	2-1
2.1.2	Label Field	2-1
2.1.3	Operation Field	2-2
2.1.4	Operand Field	2-3
2.1.5	Comment Field	2-3
2.2	Expressions	2-3
2.2.1	Constants	2-4
2.2.2	ASCII Literals	2-4
2.3	Symbols	2-4
2.4	M6800 Addressing Modes	2-4
2.4.1	Inherent and Accumulator Addressing Mode	2-4
2.4.2	Immediate Addressing Mode	2-4
2.4.3	Relative Addressing Mode	2-5
2.4.4	Indexed Addressing Mode	2-5
2.4.5	Direct and Extended Addressing Mode	2-5
2.5	Assembler Listing	2-5
2.5.1	Assembly Listing	2-5
2.5.2	Object Program	2-6
CHAPTER 3:	ASSEMBLER DIRECTIVES	3-1
3.1	Introduction	3-1
3.2	End	3-4
3.3	EQU -- Equate Symbol Value	3-4
3.4	FCB -- Form Constant Byte	3-5
3.5	FCC -- Form Constant Character	3-6
3.6	FDB -- Form Double Constant Byte	3-7

TABLE OF CONTENTS (CONTINUED)

3.7	NAM -- Program Name	3-8
3.8	OPT -- Output Option	3-8
3.9	ORG -- Origin	3-9
3.10	PAGE -- Top of Page	3-11
3.11	RMB -- Reserve Memory Bytes	3-11
3.12	SPC -- Space	3-12
CHAPTER 4:	ASSEMBLY INFORMATION	4-1
4.1	General Information	4-1
4.2	Co-Resident Assembler Tape/Cassette Operating Procedures	4-1
4.2.1	Loading Co-Resident Assembler From Tape/Cassette	4-1
4.2.1.1	Loading Tape/Cassette Into EXORciser	4-1
4.2.1.2	Loading Tape/Cassette Into Evaluation Module Memory	4-1
4.2.2	Assembly Initiation	4-2
4.2.3	Tape/Cassette Co-Resident Assembler Operation	4-4
4.3	Co-Resident Assembler Diskette Operating Procedures	
4.3.1	Diskette Co-Resident Assembler Operating Characteristics	
4.3.2	Diskette Co-Resident Assembler Operation	
APPENDIX A:	CHARACTER SET	A-1
APPENDIX B:	SUMMARY OF M6800 INSTRUCTIONS	B-1
APPENDIX C:	M6800 CO-RESIDENT ASSEMBLY DIRECTIVES SUMMARY	C-1
APPENDIX D:	ASSEMBLER ERROR MESSAGES	D-1
APPENDIX E:	ABSOLUTE OBJECT RECORD FORMAT	E-1
APPENDIX F:	SAMPLE PROGRAM	F-1
APPENDIX G:	USING MIKBUG VERSION OF THE M6800 CO-RESIDENT SOFTWARE	G-1
APPENDIX H:	USE OF OTHER PERIPHERALS WITH THE CO-RESIDENT SOFTWARE	H-1
APPENDIX I:	PROM VERSION OF CO-RESIDENT ASSEMBLER/EDITOR	I-1

CHAPTER 1

GENERAL INFORMATION

1.1 INTRODUCTION

The M6800 Co-Resident Assembler is a program that processes source program statements written in M6800 Assembly Language, translates these source statements into object programs compatible with the M6800 Firmware loaders, and produces a formatted listing of the source program. The M6800 Co-Resident Assembler is compatible with the MPCASM and M68SAM cross-assemblers. This Assembler can co-reside in memory with the M6800 Co-Resident Editor. The editor is described in the M6800 Co-Resident Editor Manual.

1.2 M6800 CO-RESIDENT ASSEMBLER LANGUAGE

The symbolic language used to code source programs to be processed by the assembler is called the M6800 Co-Resident Assembler Language.

The language is a collection of mnemonic symbols representing:

- Operations
  - M6800 machine-instruction operation codes
  - M6800 Co-Resident Assembler directives
- Symbolic names (labels)
- Operators
- Special symbols

1.2.1 Machine Operation Codes

The assembly language provides mnemonic machine-instruction operation codes for all machine instructions in the M6800 instruction set. The M6800 instructions are described in detail in the M6800 Programming Reference Manual. Refer to Appendix B for a summary of the M6800 instructions.

1.2.2 Directives

The assembly language also includes mnemonic directives which specify auxiliary actions to be performed by the assembler. Directives are not always translated into machine language. (Directives are described in Chapter 3 and a summary of directives is included in Appendix C.)

1.3 M6800 CO-RESIDENT ASSEMBLER

The M6800 Co-Resident Assembler translates source statements written in M6800 Assembly Language into machine language, assigns storage locations to instructions and data, and performs auxiliary assembler actions designated by the programmer.

1.3.1 Assembler Aims

The two basic aims of the M6800 Co-Resident Assembler are:

- To translate source programs into object code in the format required by the M6800 resident loaders or an EXORciser-compatible loader.
- To provide a printed listing containing the source language input, assembler object code, and additional information (such as error codes, if any) useful in program analysis.

1.3.2 Assembler Operation

The assembler reads the source program twice: first, to develop the symbol table; second, to assemble the object program with reference to the symbol table developed in Pass 1. During Pass 2, the object code and the assembly listing are generated. Each source language line is processed before the next line is read.

As each line is processed, the assembler examines the location, operation, and operand fields. The operation code table is scanned for a match with the operation field. If a standard machine operation code is being processed, the proper data is inserted into the object code. If a directive is specified, the proper action is taken. The object code and the assembly listing are formed for output, with any detected actual or potential errors flagged before the line containing the error is printed.

1.4 ORDERING INFORMATION

The M6800 Co-Resident Assembler may be used with the M6800 EXORciser, Evaluation Module I, Evaluation Module II and Evaluation Kit. Table 1-1 identifies the options of the Assembler, their part numbers, and the hardware they are designed to work with.

1.5 OPERATING ENVIRONMENTS

1.5.1 Equipment Requirements

Minimum equipment requirements for the M6800 Co-Resident Assembler include:

- EXORciser, Evaluation Module I, Evaluation Module II, or Evaluation Kit
- 8k bytes of RAM
- Terminal with TTY (20m A neutral loop current) or RS-232C interface and equipped with an automatic reader/punch control.

1.5.2 Software Requirements

The M6800 Co-Resident Assembler operates with the EXbug Firmware, the MIKBUG Firmware, and the MINIBUG Firmware. This Assembler also may be used with EXORdisk and the EDOSII software operating system.

NOTE:

When using the Co-Resident software with Evaluation Module I or the Evaluation Kit modify this hardware in accordance with Appendix G.

TABLE 1-1. Co-Resident Assembler Packages

HARDWARE	SOFTWARE PACKAGE NAME	SOFTWARE PACKAGE PART NUMBER*
1. EXORciser (EXbug)	Co-Resident Assembler	M68ASMR013 A, B, D
2. Evaluation Module I (MIKBUG)	Co-Resident Assembler/Editor	M68ASM6813 A, B
3. Evaluation Module II (MINIBUG II)	Co-Resident Assembler	M68ASMR213 A, B

*A = Cassette, B = Paper Tape, D = Diskette

CHAPTER 2

CODING M6800 CO-RESIDENT ASSEMBLER LANGUAGE PROGRAMS

2.1 SOURCE STATEMENT FORMAT

Programs written in assembly language consist of a sequence of source statements. Each source statement consists of a sequence of ASCII characters ending with a carriage return. Refer to Appendix A for a listing of the supported ASCII character set.

Each source statement may include up to five fields:

. Sequence number

. Label (or "*" implying a comment)

. Operation

. Operand

. Comment

2.1.1 Sequence Numbers

The sequence number field is an option provided as a programmer convenience. The sequence number field starts at the beginning of a source line and consists of up to five decimal digits (the value must be less than 65,536). Sequence numbers must be followed by a space.

Although sequence numbers are optional, they must be consistently used or not used for an entire program. If the first source statement includes a sequence number, then every succeeding statement must also include a sequence number. If the first source statement is unnumbered, then no other statement may be numbered. In this case the Assembler will provide sequential line numbers on the assembly listing.

2.1.2 Label Field

The label field occurs directly after the sequence number field (if there is one) or as the first field of a source line. The label field may take one of the following forms:

(1) An asterisk (*) as the first character indicates that the rest of the source line is a comment and should be ignored (except for listing purposes) by the assembler.

(2) A blank (b) as the first character indicates that the label field is empty (the line is not a comment and does not have a label).

(3) A symbol.

The attributes of a symbol are:

- consists of 1 to 6 characters
- valid characters in a symbol are A through Z and 0 through 9.
- the first character of a symbol must be alphabetic.
- the symbols "A", "B", and "X" are special symbols used by the assembler and should never be used in the label field.

A symbol may occur only once in the label field. If a symbol does occur in more than one label field, then each reference to that symbol will cause an error.

A label (symbol in the label field) is normally assigned the value of the program location counter of the first byte of the instruction or data being assembled.

The label of an EQU directive is assigned the value of the expression in the operand field.

Some directives must not have a label in the label field. These directives include: ORG, NAM, END, OPT, PAGE, and SPC.

Each symbol in a program is allocated an eight byte block in the symbol table.

2.1.3 Operation Field

The operation field occurs directly after the label field in an assembly language source statement. This field consists of an operation code of three or four characters. The rules governing symbols also apply to entries in the operation code field.

Entries in the operation code field may be one of two types:

- machine mnemonic operation code - these correspond directly to M6800 machine instructions. This operation code field includes the "A" or "B" character for the "dual" or "accumulator" addressing modes. For compatibility with other M6800 assemblers, a space may separate the operator from the accumulator designation (i.e., LDA A is the same as LDAA).
- directive - special operation codes known to the assembler which control the assembly process rather than being translated directly to machine language.

The assembler searches for operation codes in the table of machine operation codes and directives. If not found, an error message is printed.

2.1.4 Operand Field

Interpretation of the operand field is dependent on the operation field. For the M6800 machine instructions, the operand field must specify the addressing mode. The operand field formats and the corresponding addressing modes are as follows:

Operand Format		M6800 Machine Instruction Addressing Mode
no operand	-	inherent and accumulator
expression	-	direct or extended (direct will be used if possible)
#< expression >	-	immediate
< expression >,X	-	indexed

Addressing modes and expressions are described in the M6800 Programming Manual. Assembler directives can take on another form. These directives are described in Chapter 3.

2.1.5 Comment Field

The last field of an M6800 Assembly Language source line is the comment field. This field is optional and is ignored by the assembler except for being included in the listing. The comment field is separated from the operand field (or the operator field if there is no operand) by one or more blanks and may consist of any ASCII character. This field is important in documenting the operation of a program.

2.2 EXPRESSIONS

An expression is a combination of symbols and/or numbers separated by one of the arithmetic operators (+, -, *, or /).

The assembler evaluates expressions algebraically from left to right without parenthetical grouping. There is no precedence hierarchy among the arithmetic operators. A fractional result, or intermediate result obtained during the evaluation of an expression, will be truncated to an integer value.

2.2.1 Constants

Decimal: < number >

Hexidecimal: $ < number > or < number > H
(first digit in latter case must be 0 - 9)

Octal: @ < number > or < number > O or < number > Q

Binary: % < number > or < number > B

2.2.2 ASCII Literals

'< character > (apostrophe followed by an ASCII character)
The result is the numeric value for the ASCII character.

2.3 SYMBOLS

A symbol in an expression is similar to a symbol in the label field except that the value of the symbol is referenced instead of defined. An asterisk "*" is a special symbol recognized by the assembler and represents the value of the current location counter (first byte of an instruction, when used in the context of the symbol.

A 16-bit integer value is associated with each symbol. This value is used in place of the symbol during expression evaluation.

The M6800 Co-Resident Assembler is a two-pass assembler. The symbol table is built on the first pass. Object records and listing are produced on the second pass. Certain expressions cannot be fully evaluated during the first pass because they may contain (forward) references to symbols which have not yet been defined. In some cases, a symbol may not be defined before being used in the second pass. Since the assembler cannot evaluate such symbols, these cases are treated as errors. Only one level of forward referencing is allowed.

2.4 M6800 ADDRESSING MODES

2.4.1 Inherent and Accumulator Addressing Mode

The M6800 includes some instructions which require only an operation code byte. These self-contained instructions employ inherent or accumulator addressing and do not require the operand field when written in the M6800 assembly language.

2.4.2 Immediate Addressing Mode

Immediate addressing refers to the use of one or two bytes immediately following the instruction operation code as the instruction operand. Immediate addressing is selected by preceding the operand field in the source line with the character "#". The expression following the "#" may require one or two bytes, depending on the instruction.

2.4.3 Relative Addressing Mode

Relative addressing is used by the branch instructions. Branches can be made only within the range -126 to 129 relative to the first byte of the branch instruction:

$$(PC+2)-128 \leq D \leq (PC+2)+127$$

PC = address of first byte of branch instruction

D = address of the destination of the branch

The actual branch offset put into the second byte of the branch instruction is the two's complement representation of the difference between the location of the byte immediately following the branch instruction and the location of the destination.

2.4.4 Indexed Addressing Mode

Indexed addresses are relative to the M6800 index register. The address is calculated at the time of instruction execution by adding the one-byte displacement in the second instruction byte to the current contents of the 16-bit index register. Since no sign extension is performed, the offset cannot be negative.

Indexed addressing is normally indicated by the characters ",X" following the expression in the operand field. (Special cases of ",X" or "X" alone are the same as "0,X".)

2.4.5 Direct and Extended Addressing Mode

Direct and extended addressing utilize one (direct) or two (extended) bytes to form the address of the operand desired. Direct addressing is limited to the first 256 bytes of memory, 0-255. Direct and extended addressing are selected by simply putting an expression in the operand field of the source line. Direct addressing is used if possible. An error results if a directly-addressable variable is referenced before it is defined in a source program since this can cause a phasing error. To avoid phasing problems, directly addressable variables should always be defined before any reference to the variable.

2.5 ASSEMBLER LISTING

Assembler outputs include an assembly listing and an object program.

2.5.1 Assembly Listing

The assembly listing includes the source program as well as additional information generated by the assembler. Most lines in the listing correspond directly to a source statement. Lines which do not correspond directly to a source line include:

. page header lines

. error lines (see Appendix D for a listing of error numbers)

. expansion lines for the FCC, FDB, FCB directives

Most listing lines follow the standard format shown in Table 2-1.

TABLE 2-1. Standard Format

(Special cases may not use exactly the same format.)

COLUMN	CONTENTS
1-5	Source line # - 5 digit decimal counter kept by assembler
7-10	Current Location Counter value (in hex)
12-13	Machine Operation Code (hex)
15-16	First byte of operand (hex)
17-18	Second byte of operand (if there is one)
20-25	Label Field
27-31	Operation Field
34-41	Operand Field (longer operand extends into comment field)
43-Last Column	Comment Field

2.5.2 Object Program

Detailed descriptions of the absolute and relocatable object format is included in Appendix E.

CHAPTER 3

ASSEMBLER DIRECTIVES

3.1 INTRODUCTION

Assembler directives are instructions to the assembler rather than instructions to be directly translated into object code. This section describes the directives recognized by the M6800 Co-Resident Assembler.

In Table 3-1 the directives are grouped by function performed. Detailed descriptions of each directive are arranged alphabetically.

TABLE 3-1. Assembly Directives

DIRECTIVE	FUNCTION
ASSEMBLY CONTROL	
NAM	Program name
ORG	Origin
END	Program end
LISTING CONTROL	
PAGE	Top of page
SPC	Skip "n" lines
OPT NOO	No object tape
OPT O (Object Tape)	The Assembler will generate an object tape (selected by default).
OPT M (Memory File)	The Assembler will write machine code to memory.
OPT NOM	No memory (selected by default).
OPT S (Print Symbols)	The Assembler will print the symbols at the end of Pass 2.
OPT NOS	No printing of symbols (selected by default).
OPT NOL (No Listing)	The Assembler will not print a listing of the assembler data.
OPT L	The listing of assembled data will be printed (selected by default).
OPT NOP (No Page)	The Assembler will inhibit format paging of the assembly listing.
OPT P	The listing will be paged (selected by default).
OPT NOG (No Generate)	Causes only 1 line of data to be listed from the assembler directions FCC, FCB, and FDB.

TABLE 3-1. Assembly Directives (Continued)

DIRECTIVE	FUNCTION
OPT G	All data generated by the FCC, FCB, and FDB directions will be printed (selected by default).
DATA DEFINITION/STORAGE ALLOCATION	
FCC	Character string data
FCB	One byte data
FDB	Double byte data
RMB	Reserve memory bytes
SYMBOL DEFINITION	
EQU	Assign permanent value

3.2 END

FORMAT: END

DESCRIPTION: The END directive indicates to the Assembler that the source is finished. Subsequent source statements are ignored. The END directive encountered at the end of the first pass through the source program causes the Assembler to start the second pass.

3.3 EQU - Equate Symbol Value

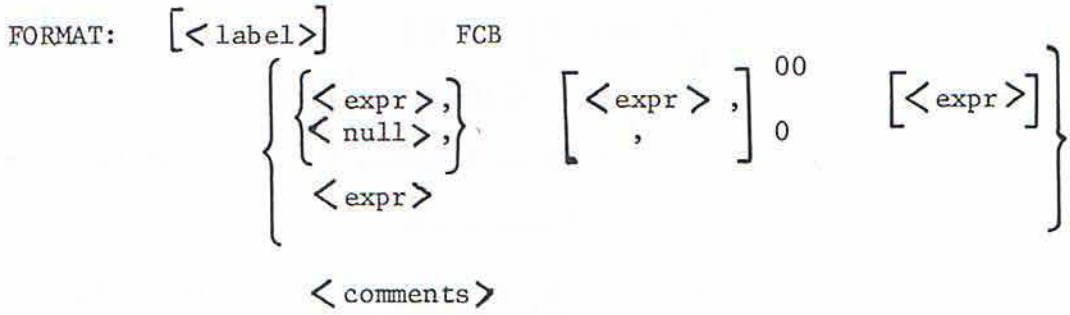
FORMAT: <label> EQU <expression> [<comments>]

DESCRIPTION: The EQU directive assigns the value of the expression in the operand field to the symbol in the label field. The label and expression follow the rules given in a previous section. Note that EQU is one operator that assigns a value other than the program location counter to the label. The label and operand fields are both required and the label cannot be defined anywhere else in the program.

The expression in the operand field of an EQU cannot include a symbol that is undefined or not yet defined (no forward references are allowed).

3.4 FCB - Form Constant Byte

FORMAT:



DESCRIPTION: The FCB directive may have one or more operands, separated by commas. An 8-bit unsigned binary number corresponding to the value of each operand is stored in a byte of the object program. If there is more than one operand, they are stored in successive bytes. The operand field may contain the actual value (decimal, hexadecimal, octal, or binary). Alternatively, the operand may be a symbol or an expression which can be assigned a numerical value by the Assembler.

An FCB directive followed by one or more null operands separated by commas will store zeros for the null operands.

3.5 FCC - Form Constant Character

FORMAT:

```
[<label>]   FCC

{ d   <ASCII string>   d                  }
{ <decimal number>  ,  <ASCII string>     }

   <comments>
```

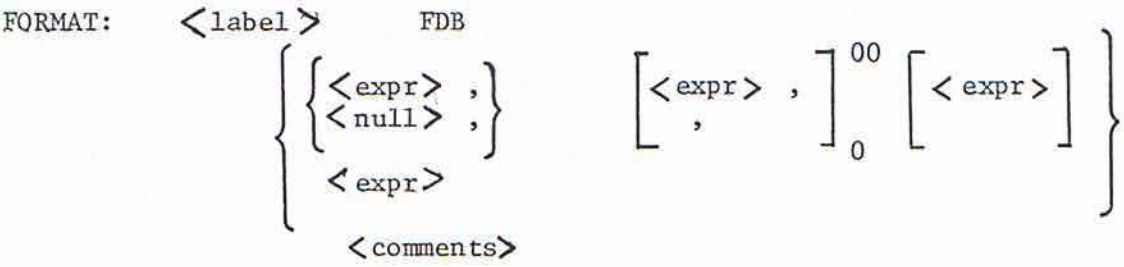
NOTE:
1. "d" is any non-numeric character (used as a delimiter).
2. ASCII string may not include a carriage return.

DESCRIPTION: The FCC directive translates strings of characters into their 7-bit ASCII codes. Any of the characters which correspond to ASCII hexadecimal codes 20 (SP) through 5F (_) can be processed by this directive.

1. Count, comma, text. Where the count specifies how many ASCII characters to generate and the text begins following the first comma of the operand. Should the count be longer than the text, spaces will be inserted to fill the count. Maximum count is 255.
2. Text enclosed between identical delimiters, each being any single character. (If the delimiters are numbers, the text must not begin with a comma.)

3.6 FDB - Form Double Constant Byte

FORMAT:



DESCRIPTION: The FDB directive may have one or more operands separated by commas. The 16-bit unsigned binary number corresponding to the value of each operand is stored in two bytes of the object program. If there is more than one operand, they are stored in successive bytes. The operand field may contain the actual value (decimal, hexadecimal, octal, or binary). Alternatively, the operand may be a symbol or an expression which can be assigned a numerical value by the Assembler.

An FDB directive followed by one or more null operands separated by commas will store zeros for the null operands.

The label is optional.

```
00001                          NAM     FDB
00002                  *
00003                  *   PROGRAM TO ILLUSTRATE USE OF FORM DOUBLE
00004                  *   BYTE CONSTANT DIRECTIVE
00005                  *

00007 0000 0002                FDB     2
00008 0002 0000        LABEL   FDB     ,$F,$FF,$FFF,,$FFFF
      0004 000F
      0005 00FF
      0008 0FFF
      000A 0000
      000C FFFF
00009 000E 000C                FDB     LABEL+10,LABEL+5,LABEL
      0010 0007
      0012 0002
00010                          END

TOTAL ERRORS 00000
```

3.7 NAM - Program Name

FORMAT: NAM <program name> [<comments>]

DESCRIPTION: The NAM directive must be the first statement of a M6800 Co-Resident Assembler source program. The NAM directive does not allow a label, but it does require an operand -- a program name (one-eight characters).

The program name from the NAM directive is printed on the header line for each listing page.

3.8 OPT - Output Option

FORMAT: OPT <option> [, <option>]

DESCRIPTION: The OPT directive is used to give the programmer optional control of the format of the Assembler output. The options are written in the operand field and are separated by commas. The options may have the character "NO" as a prefix which reverses their meaning.

OPTION	MEANING
OPT O (object tape)	The Assembler will generate an object tape. (selected by default)
OPT NOO	No object tape
OPT M (memory file)	The Assembler will write machine code into memory.
OPT NOM	No memory (selected by default).
OPT S (printed symbols)	The Assembler will print the symbols at the end of Pass 2.
OPT NOS	No printing of symbols (selected by default).
OPT L	The listing of assembled data will be printed (selected by default).
OPT NOL (no listing)	The Assembler will not print a listing of the assembled data.

OPT P	The listing will be paged (selected by default).
OPT NOP	The Assembler will inhibit format paging of the assembly listing.
OPT G	All data generated by the FCC, FCB, and FDB directions will be printed (selected by default).
OPT NOG (no generate)	Causes only one line of data to be listed from the assembler directions FCC, FCB, and FDB.

3.9 ORG - Origin

FORMAT: ORG <expression> [<comments>]

DESCRIPTION: The ORG directive changes the program counter to the value specified by the expression in its operand field. Subsequent statements are assigned memory locations starting with the new program counter value. If no ORG is specified, the program counter is initialized with a value of 0. The ORG directive may not include a label.

```
00001                           NAM     ORG
00002                   *
00003                   *   PROGRAM TO ILLUSTRATE USE OF THE ORIGIN
00004                   *   DIRECTIVE
00005                   *


00007 0000 0001     BILL    RMB     1           PC STARTS AT ZERO
00008      0001     JOHN    EQU     *
00009 0020                  ORG     $20         PC SET TO HEX 20
00010 0020 000A             RMB     10
00011 0001                  ORG     JOHN        PC SET TO VALUE OF JOHN
00012 0001 000A             RMB     10
00013                       END

TOTAL ERRORS 00000
```

3.10 PAGE - Top of Page

FORMAT: PAGE

DESCRIPTION: The PAGE directive causes the Assembler to advance the paper to the top of the next page. The PAGE directive does not appear on the program listing. No label or operand is used, and no machine code results.

3.11 RMB - Reserve Memory Bytes

FORMAT: [<label>] RMB <expression> [<comments>]

DESCRIPTION: The RMB directive causes the location counter to be increased by the value of the operand field. This reserves a block of memory whose length is equal to the value of the operand field. The operand field may contain the actual number (decimal, hexadecimal, octal or binary) equal to the number of bytes to be reserved. Alternatively, the operand may be a symbol or an expression which can be assigned a numerical value by the Assembler.

The block of memory which is reserved by the RMB directive is unchanged by that directive.

The expression must not contain symbols which are defined later in the program (forward references).

```
00001                    NAM     RMB
00002                *
00003                *   PROGRAM TO ILLUSTRATE USE OF THE RESERVE
00004                *   MEMORY BYTE DIRECTIVE
00005                *

00007 0000 0001  CLAB1   RMB     1          1 BYTE RESERVED FOR CLAB1
00008 0001 0002  CLAB2   RMB     2          2 BYTES RESERVED FOR CLAB2
00009 0003 0003          RMB     *-CLAB1    EXPRESSION DETERMINES SIZE
00010                    END

TOTAL ERRORS 00000
```

3.12 SPC - Space

FORMAT: SPC <expression>

DESCRIPTION: The SPC directive provides n vertical spaces for formatting the program listing. It does not itself appear in the listing. The number of lines to be left blank is stated by an operand in the operand field.

The operand would normally contain the actual number (decimal, hexadecimal, octal or binary) equal to the number of lines to be left blank. A symbol or an expression is also allowed.

When the SPC directives causes the listing to cross page boundries, only those blank lines required to get to the top of the next page will be generated.

CHAPTER 4

ASSEMBLER OPERATION

4.1 GENERAL INFORMATION

The user may have received the M6800 Co-Resident Assembler on cassette, paper tape, or diskette. The loading, initialization and operation of the Co-Resident Assembler in paper tape and cassette is discussed in Paragraph 4.2 while the loading and operation of the Co-Resident Assembler from diskette is discussed in Paragraph 4.3.

4.2 CO-RESIDENT ASSEMBLER TAPE/CASSETTE OPERATING PROCEDURES

4.2.1 Loading Co-Resident Assembler From Tape/Cassette

The Co-Resident Assembler must be present in the EXORciser or Evaluation Module memory prior to the initiation of the assembler operation. However, it is not always necessary to load the Assembler before each assembly operation. If several programs are assembled in succession, or if the programs are tested without modifying the memory locations used by the assembler, then the Assembler will remain intact in memory and available for subsequent uses without reloading.

4.2.2.1 LOADING TAPE/CASSETTE INTO EXORciser MEMORY. Load the Co-Resident Assembler into the EXORciser from tape/cassette as follows:

a. Place the Co-Resident Assembler object tape (paper tape or cassette) into the System Reader Device.

b. Enter the EXbug command "LOAD". The EXbug Firmware will respond with "SGL/CONT".

c. Type "S" after SGL/CONT to load the single file containing the Co-Resident Assembler. After the header record from the tape is printed, the file is loaded into memory. Upon completion, control is returned to EXbug.

4.2.2.2 LOADING TAPE/CASSETTE INTO EVALUATION MODULE MEMORY. Load the Co-Resident Assembler into the Evaluation Module from paper tape/ cassette as follows:

a. Load the Co-Resident Assembler object tape (paper tape or cassette) into the System Reader Device.

b. Enter the character L after the asterisk. This initiates the Evaluation Module loading procedure. The Evaluation Module loads the Co-Resident Assembler into memory and then prints an asterisk.

4.2.2 Assembler Initiation

In normal operation, the memory region between the end of the Co-Resident Editor and location $2000 is used by the Assembler for the symbol table. This table provides space for 90 symbols. If a larger symbol table is required, the symbol table area can be extended at either end.

By selecting the editor over-write feature, the area occupied by the Co-Resident Editor can be appended to the beginning of the symbol table. This increases the symbol table capacity to 312 symbols. The over-write option is enabled by using MAID to change the contents of memory location 303_{16} to FF_{16}.

```
EXBUG 1.2 MAID
*303/00 FF
*
```

If more than 8k bytes of read-write memory are available, additional memory can be appended to the end of the symbol table. This is accomplished by modifying the end-of-symbol-table address in memory locations 301_{16} and 302_{16}. Eight bytes of read-write memory are required for the storage of each symbol. Modifying locations 301_{16} and 302_{16} to contain 2400_{16} extends the symbol table by 1k bytes, or 128 symbols for a total of 218, assuming the editor over-write is not selected.

```
EXBUG 1.2 MAID
*301/20 24
0302/00 00
*
```

If the object code is to be written into memory (OPT M), the end-of-symbol-table address delimits the address. For example, if the symbol table ends at 2000_{16} (the default value), a program beginning at 2000_{16} or higher may have its output directed into EXORciser memory (assuming the memory is available). If, on the other hand, only 8k of memory is available and the programmer wishes to assemble into memory (OPT M), the symbol table can be shortened to make memory available for the object code. This is accomplished by changing the end-of-symbol-table address to a lower address. For example, assume $1F00_{16}$ is the new end-of-symbol-table address.

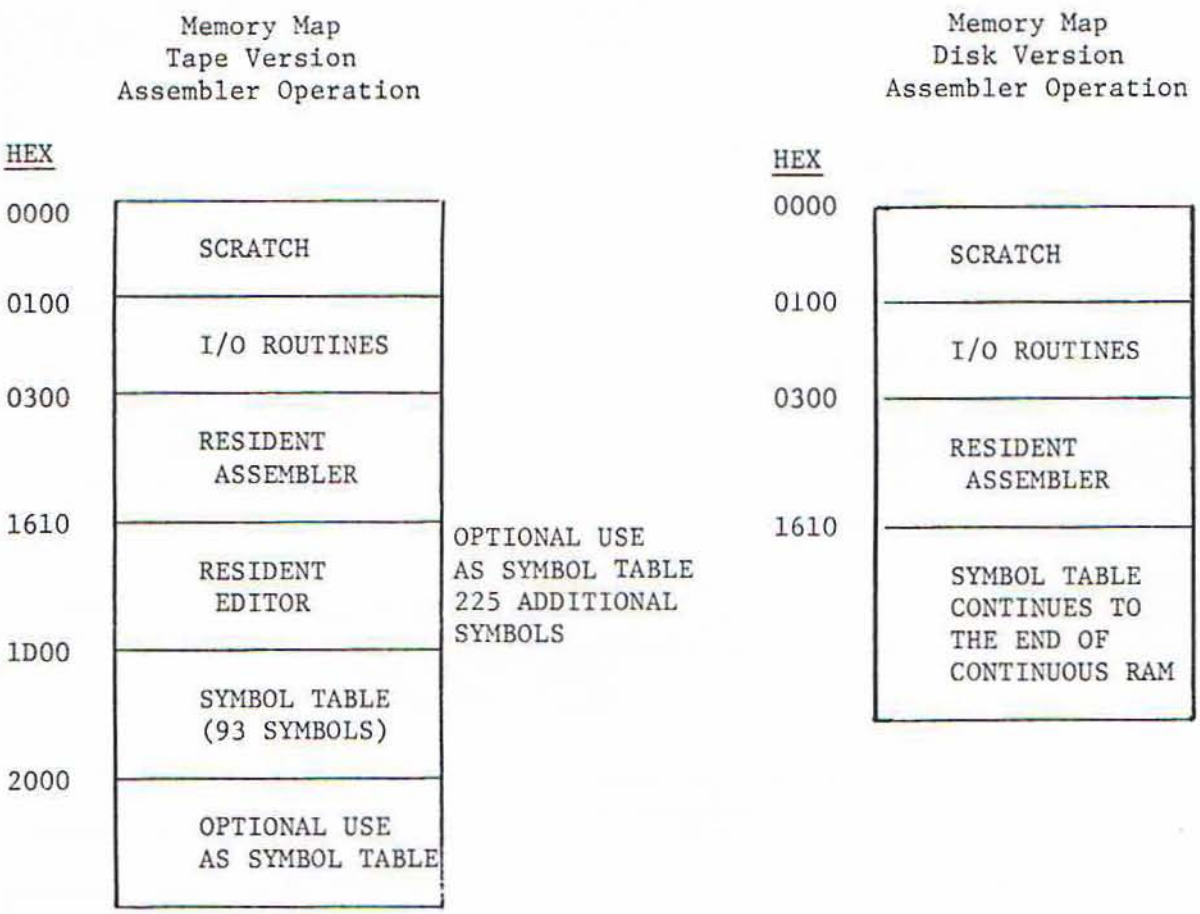
```
EXBUG 1.2 MAID
*301/20 1F
0302/00 00
*
```

A program beginning at $1F00_{16}$ now can be assembled into memory.

Should an end-of-symbol-table address be entered that is less than the start-of-symbol-table address, the Co-Resident Assembler uses the default address 2000_{16}.

A user's program may take advantage of the direct addressing mode and use the first 256 bytes ($0\text{-}100_{16}$) for scratch memory. However, no instructions that generate data; such as FCC, FDB, or FCB; may be assembled into this area because the Assembler and Editor also use this portion of memory for scratch storage.

Figure 4-1 depicts a memory map of the Co-Resident Assembler.



NOTE: (TAPE VERSION ONLY)
The editor overwrite flag is at $303. If it is zero the editor area will not be used as symbol table. If it is non-zero the editor area will be used as symbol table.
Locations $301-$302 contain the address of the end of the symbol table. The default value of $2000 may be changed by the user.

FIGURE 4-1. Memory Maps of Co-Resident Assembler.

Selection of the Editor over-write feature and modification of the end-of-symbol-table must be done after the Assembler has been loaded and before it is initiated. Figure 4-2 illustrates the procedure for loading the Assembler and initiating it without modification. Appendix F depicts the Program Assembling Procedures.

```
EXBUG 1.2 LOAD
SGL/CONT S
S ASM1.3
EXBUG 1.2 MAID
*100;G
M6800 RESIDENT ASSEMBLER 1.3
COPYRIGHT MOTOROLA 1976
ENTER PASS: 1P,1S,2P,2L,2T

1P
M6800 RESIDENT ASSEMBLER 1.3
COPYRIGHT MOTOROLA 1976
ENTER PASS: 1P,1S,2P,2L,2T

2P
```

FIGURE 4-2. Program Assembling Procedures

4.2.3 Tape/Cassette Co-Resident Assembler Operation

The Co-Resident Assembler is a two-pass assembler. That is, the Co-Resident Assembler must read a source program twice--once to build a symbol table and a second time to produce the assembled output. In response to the assembler prompt message.

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

Select the appropriate assembler pass. The Co-Resident Assembler Pass controls are described in the following paragraph and are summarized in Table 4-1.

TABLE 4-1. Co-Resident Assembler Pass Controls and Options

CONTROL	DESCRIPTION
1P	Pass 1, clears symbol table
1S	Pass 1, inhibits clearing of symbol table
2P	Pass 2, assembly listing and object tape output.
2L	Pass 2, assembly listing only
2T	Pass 2, object tape only.

PASS 1P -- Pass 1 produces a table of the 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 are listed.

PASS 1 Option 1S -- In the assembly of multiple source tapes, it may be advantageous to be known to each assembly. The S option for Pass 1 inhibits the clearing of the symbol table before the pass is started.

PASS 2P -- Pass 2 rereads the source tape and uses information in the symbol table to produce the assembled output. Using terminals which permit independent on/off control of the tape output and printer devices, Pass 2 can produce both an object tape and an assembly listing. A terminal without independent controls will permit the generation of either an object tape or an assembly listing (not both). In this case, Pass 2 may be repeated to generate both output forms.

PASS 2 OPTIONS

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 an object tape (no assembly listing).

NOTE:

One-Pass Operation. For source programs which have no 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, however, the forward references will be flagged with error 211 and the assembled program with an address field of FFFF. The correct address can be patched after the symbol table is printed at the completion of the assembly.

In combination with the options for entering a source program from the terminal keyboard and for assembling an object program in memory, short programs may be assembled and executed without the use of tapes.

4.3 CO-RESIDENT ASSEMBLER DISKETTE OPERATING PROCEDURES

4.3.1 Disk Co-Resident Assembler Operating Characteristics

The Co-Resident Assembler on diskette, when working with the EXORdisk with its EDOS Firmware, has several unique characteristics. In this application, the EDOSII Firmware automatically selects the Editor-overwrite option. Also the assembler searches the EXORciser for the end of its continuous memory to deter the end-of-symbol-table address.

If the user wishes to use the OPT M directive and insert the assembled output into memory he must provide a block of memory that is not continuous with the memory being used by the Co-Resident Assembler.

4.3.2 Diskette Co-Resident Assembler Operation

The Co-Resident Assembler is a two pass assembler that resides in the diskette file named ASMB. That is, in its assembly operation the Assembler reads the source program twice -- once to build a symbol table, and a second time to produce the assembled output. Unlike the two pass operation of the assembler on tape or diskette, this assembly automatically performs the two passes in sequence.

This assembler working with the EXORdisk's EDOS Firmware assembles the source file and directs the assembled object output (if selected) to the object file and the assembly listing (if selected) to the terminal device. In initiating the assembly process, the user instructs the EXORciser to run the EDOS Firmware. On receiving the EDOS prompt (!) the user enters the appropriate assembly command. The three assembly operations are described in Figure 4-3 and illustrated in Figure 4-4. In entering the assembly command, all three operands must be specified. In the case where no object file is to be created, any dummy file name may be entered in the operand field. In this case, no file entry will be created on the diskette.

Name: ASM

Format: ASM, passoption, objectfilename, sourcefilename

Purpose: To assemble the contents of the source file and to direct the assembled object output, if any to the object output file and the assembled listing, if any, to the list device.

Comments: All three operands must be specified. If no object file is to be created, any dummy file name (i.e. X or Y or Z etc.) may be entered in this operand field since no file directory entry will be created.

The pass option operand field may contain the number 2, 3, or 4.

2 = both an assembly listing and an object output are produced.

3 = only as assembly listing is generated to the list device.

4 = only an object output is generated to the output object file.

Example: ASM,4,JOEO,JOES

Produce an object file named JOEO from the source file named JOES.

FIGURE 4-3. Assemble (ASM) Command

```
+<BUG 1.2 M4ID
*E800;G
M6800 EDOS VER. 2.2

!ASM,3,PGMOT,PGM

M6800 RESIDENT ASSEMBLER 1.3
COPYRIGHT MOTOROLA 1976
```

FIGURE 4-4. Example of Disk Assembly Operation

APPENDIX A

CHARACTER SET

The character set recognized by the Motorola M6800 Co-Resident Assembler is a subset of ASCII (American Standard Code for Information Interchange, 1968). The ASCII Code is shown in the M6800 Programming Reference Manual. The following characters are recognized by the assembler.

1. The upper case letters A through Z

2. The integers Ø through 9

3. Four arithmetic operators:

   + - * /

4. Characters used as special prefixes:

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

5. Characters used as special suffixes:

   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

6. Three separating characters:

   SPACE
   CR (carriage return)
   , (comma)

7. A comment in a source statement may include any characters with ASCII hexadecimal values from 20 (SP) through 5F (__).

8. In addition to the above, the assembler has the capability of reading string of characters and of entering the corresponding 7-bit ASCII code into specified locations in the memory. This capability is provided by the assembler directive FCC (see Chapter 3). Any characters corresponding to ASCII hexadecimal values 20 (SP) through 5F (_) can be processed. This kind of processing can also be done, for a single ASCII character, by using the immediate mode of addressing with an operand in the form " 'C".

APPENDIX B
SUMMARY OF M6800 INSTRUCTIONS

	(Dual Operand)	ACCX	Immediate	Direct	Extended	Indexed	Implied	Relative
ABA		•	•	•	•	•	2	•
ADC	x	•	2	3	4	5	•	•
ADD	x	•	2	3	4	5	•	•
AND	x	•	2	3	4	5	•	•
ASL		2	•	•	6	7	•	•
ASR		2	•	•	6	7	•	•
BCC		•	•	•	•	•	•	4
BCS		•	•	•	•	•	•	4
BEA		•	•	•	•	•	•	4
BGE		•	•	•	•	•	•	4
BGT		•	•	•	•	•	•	4
BHI		•	•	•	•	•	•	4
BIT	x	•	2	3	4	5	•	•
BLE		•	•	•	•	•	•	4
BLS		•	•	•	•	•	•	4
BLT		•	•	•	•	•	•	4
BMI		•	•	•	•	•	•	4
BNE		•	•	•	•	•	•	4
BPL		•	•	•	•	•	•	4
BRA		•	•	•	•	•	•	4
BSR		•	•	•	•	•	•	8
BVC		•	•	•	•	•	•	4
BVS		•	•	•	•	•	•	4
CBA		•	•	•	•	•	2	•
CLC		•	•	•	•	•	2	•
CLI		•	•	•	•	•	2	•
CLR		2	•	•	6	7	•	•
CLV		•	•	•	•	•	2	•
CMP	x	•	2	3	4	5	•	•
COM		2	•	•	6	7	•	•
CPX		•	3	4	5	6	•	•
DAA		•	•	•	•	•	2	•
DEC		2	•	•	6	7	•	•
DES		•	•	•	•	•	4	•
DEX		•	•	•	•	•	4	•
EOR	x	•	2	3	4	5	•	•

	(Dual Operand)	ACCX	Immediate	Direct	Extended	Indexed	Implied
INC		2	•	•	6	7	•
INS		•	•	•	•	•	4
INX		•	•	•	•	•	4
JMP		•	•	•	3	4	•
JSR		•	•	•	9	8	•
LDA	x	•	2	3	4	5	•
LDS		•	3	4	5	6	•
LDX		•	3	4	5	6	•
LSR		2	•	•	6	7	•
NEG		2	•	•	6	7	•
NOP		•	•	•	•	•	2
ORA	x	•	2	3	4	5	•
PSH		•	•	•	•	•	4
PUL		•	•	•	•	•	4
ROL		2	•	•	6	7	•
ROR		2	•	•	6	7	•
RTI		•	•	•	•	•	10
RTS		•	•	•	•	•	5
SBA		•	•	•	•	•	2
SBC	x	•	2	3	4	5	•
SEC		•	•	•	•	•	2
SEI		•	•	•	•	•	2
SEV		•	•	•	•	•	2
STA	x	•	•	4	5	6	•
STS		•	•	5	6	7	•
STX		•	•	5	6	7	•
SUB	x	•	2	3	4	5	•
SWI		•	•	•	•	•	12
TAB		•	•	•	•	•	2
TAP		•	•	•	•	•	2
TBA		•	•	•	•	•	2
TPA		•	•	•	•	•	2
TST		2	•	•	6	7	•
TSX		•	•	•	•	•	4
TSX		•	•	•	•	•	4
WAI		•	•	•	•	•	9

NOTE: Interrupt time is 12 cycles from the end of the instruction being executed, except following a WAI instruction. Then it is 4 cycles.

INSTRUCTION ADDRESSING MODES AND EXECUTION TIMES (TIMES IN MACHINE CYCLES)

OPERATIONS	MNEMONIC	IMMED OP	IMMED ~	IMMED =	DIRECT OP	DIRECT ~	DIRECT =	INDEX OP	INDEX ~	INDEX =	EXTND OP	EXTND ~	EXTND =	IMPLIED OP	IMPLIED ~	IMPLIED =	BOOLEAN/ARITHMETIC OPERATION (All register labels refer to contents)	5 H	4 I	3 N	2 Z	1 V	0 C
Add	ADDA	3B	2	2	9B	3	2	AB	5	2	BB	4	3				A + M → A	↕	•	↕	↕	↕	↕
	ADDB	CB	2	2	DB	3	2	EB	5	2	FB	4	3				B + M → B	↕	•	↕	↕	↕	↕
Add Acmltrs	ABA													1B	2	1	A + B → A	↕	•	↕	↕	↕	↕
Add with Carry	ADCA	89	2	2	99	3	2	A9	5	2	B9	4	3				A + M + C → A	↕	•	↕	↕	↕	↕
	ADCB	C9	2	2	D9	3	2	E9	5	2	F9	4	3				B + M + C → B	↕	•	↕	↕	↕	↕
And	ANDA	84	2	2	94	3	2	A4	5	2	B4	4	3				A · M → A	•	•	↕	↕	R	•
	ANDB	C4	2	2	D4	3	2	E4	5	2	F4	4	3				B · M → B	•	•	↕	↕	R	•
Bit Test	BITA	85	2	2	95	3	2	A5	5	2	B5	4	3				A · M	•	•	↕	↕	R	•
	BITB	C5	2	2	D5	3	2	E5	5	2	F5	4	3				B · M	•	•	↕	↕	R	•
Clear	CLR							6F	7	2	7F	6	3				00 → M	•	•	R	S	R	R
	CLRA													4F	2	1	00 → A	•	•	R	S	R	R
	CLRB													5F	2	1	00 → B	•	•	R	S	R	R
Compare	CMPA	81	2	2	91	3	2	A1	5	2	B1	4	3				A − M	•	•	↕	↕	↕	↕
	CMPB	C1	2	2	D1	3	2	E1	5	2	F1	4	3				B − M	•	•	↕	↕	↕	↕
Compare Acmltrs	CBA													11	2	1	A − B	•	•	↕	↕	↕	↕
Complement, 1's	COM							63	7	2	73	6	3				M̄ → M	•	•	↕	↕	R	S
	COMA													43	2	1	Ā → A	•	•	↕	↕	R	S
	COMB													53	2	1	B̄ → B	•	•	↕	↕	R	S
Complement, 2's (Negate)	NEG							60	7	2	70	6	3				00 − M → M	•	•	↕	↕	①	②
	NEGA													40	2	1	00 − A → A	•	•	↕	↕	①	②
	NEGB													50	2	1	00 − B → B	•	•	↕	↕	①	②
Decimal Adjust, A	DAA													19	2	1	Converts Binary Add. of BCD Characters into BCD Format	•	•	↕	↕	↕	③
Decrement	DEC							6A	7	2	7A	6	3				M − 1 → M	•	•	↕	↕	4	•
	DECA													4A	2	1	A − 1 → A	•	•	↕	↕	4	•
	DECB													5A	2	1	B − 1 → B	•	•	↕	↕	4	•
Exclusive OR	EORA	88	2	2	98	3	2	A8	5	2	B8	4	3				A⊕M → A	•	•	↕	↕	R	•
	EORB	C8	2	2	D8	3	2	E8	5	2	F8	4	3				B⊕M → B	•	•	↕	↕	R	•
Increment	INC							6C	7	2	7C	6	3				M + 1 → M	•	•	↕	↕	⑤	•
	INCA													4C	2	1	A + 1 → A	•	•	↕	↕	⑤	•
	INCB													5C	2	1	B + 1 → B	•	•	↕	↕	⑤	•
Load Acmltr	LDAA	86	2	2	96	3	2	A6	5	2	B6	4	3				M → A	•	•	↕	↕	R	•
	LDAB	C6	2	2	D6	3	2	E6	5	2	F6	4	3				M → B	•	•	↕	↕	R	•
Or, Inclusive	ORAA	8A	2	2	9A	3	2	AA	5	2	BA	4	3				A + M → A	•	•	↕	↕	R	•
	ORAB	CA	2	2	DA	3	2	EA	5	2	FA	4	3				B + M → B	•	•	↕	↕	R	•
Push Data	PSHA													36	4	1	A → M_{SP}, SP − 1 → SP	•	•	•	•	•	•
	PSHB													37	4	1	B → M_{SP}, SP − 1 → SP	•	•	•	•	•	•
Pull Data	PULA													32	4	1	SP + 1 → SP, M_{SP} → A	•	•	•	•	•	•
	PULB													33	4	1	SP + 1 → SP, M_{SP} → B	•	•	•	•	•	•
Rotate Left	ROL							69	7	2	79	6	3				M	•	•	↕	↕	⑥	↕
	ROLA													49	2	1	A	•	•	↕	↕	⑥	↕
	ROLB													59	2	1	B	•	•	↕	↕	⑥	↕
Rotate Right	ROR							66	7	2	76	6	3				M	•	•	↕	↕	⑥	↕
	RORA													46	2	1	A	•	•	↕	↕	⑥	↕
	RORB													56	2	1	B	•	•	↕	↕	⑥	↕
Shift Left, Arithmetic	ASL							68	7	2	78	6	3				M	•	•	↕	↕	⑥	↕
	ASLA													48	2	1	A	•	•	↕	↕	⑥	↕
	ASLB													58	2	1	B	•	•	↕	↕	⑥	↕
Shift Right, Arithmetic	ASR							67	7	2	77	6	3				M	•	•	↕	↕	⑥	↕
	ASRA													47	2	1	A	•	•	↕	↕	⑥	↕
	ASRB													57	2	1	B	•	•	↕	↕	⑥	↕
Shift Right, Logic	LSR							64	7	2	74	6	3				M	•	•	R	↕	⑥	↕
	LSRA													44	2	1	A	•	•	R	↕	⑥	↕
	LSRB													54	2	1	B	•	•	R	↕	⑥	↕
Store Acmltr.	STAA				97	4	2	A7	6	2	B7	5	3				A → M	•	•	↕	↕	R	•
	STAB				D7	4	2	E7	6	2	F7	5	3				B → M	•	•	↕	↕	R	•
Subtract	SUBA	80	2	2	90	3	2	A0	5	2	B0	4	3				A − M → A	•	•	↕	↕	↕	↕
	SUBB	C0	2	2	D0	3	2	E0	5	2	F0	4	3				B − M → B	•	•	↕	↕	↕	↕
Subtract Acmltrs.	SBA													10	2	1	A − B → A	•	•	↕	↕	↕	↕
Subtr. with Carry	SBCA	82	2	2	92	3	2	A2	5	2	B2	4	3				A − M − C → A	•	•	↕	↕	↕	↕
	SBCB	C2	2	2	D2	3	2	E2	5	2	F2	4	3				B − M − C → B	•	•	↕	↕	↕	↕
Transfer Acmltrs	TAB													16	2	1	A → B	•	•	↕	↕	R	•
	TBA													17	2	1	B → A	•	•	↕	↕	R	•
Test, Zero or Minus	TST							6D	7	2	7D	6	3				M − 00	•	•	↕	↕	R	R
	TSTA													4D	2	1	A − 00	•	•	↕	↕	R	R
	TSTB													5D	2	1	B − 00	•	•	↕	↕	R	R
																		H	I	N	Z	V	C

ADDRESSING MODES

BOOLEAN/ARITHMETIC OPERATION

COND. CODE REG.

LEGEND:

OP Operation Code (Hexadecimal);
~ Number of MPU Cycles;
= Number of Program Bytes;
+ Arithmetic Plus;
− Arithmetic Minus;
· Boolean AND;
M_{SP} Contents of memory location pointed to be Stack Pointer;
+ Boolean Inclusive OR;
⊕ Boolean Exclusive OR;
M̄ Complement of M;
→ Transfer Into;
0 Bit = Zero;
00 Byte = Zero;

Note – Accumulator addressing mode instructions are included in the column for IMPLIED addressing

CONDITION CODE SYMBOLS:

H Half-carry from bit 3;
I Interrupt mask
N Negative (sign bit)
Z Zero (byte)
V Overflow, 2's complement
C Carry from bit 7
R Reset Always
S Set Always
↕ Test and set if true, cleared otherwise
• Not Affected

POINTER OPERATIONS	MNEMONIC	IMMED OP	IMMED ~	IMMED #	DIRECT OP	DIRECT ~	DIRECT #	INDEX OP	INDEX ~	INDEX #	EXTND OP	EXTND ~	EXTND #	IMPLIED OP	IMPLIED ~	IMPLIED #	BOOLEAN/ARITHMETIC OPERATION	COND. CODE REG. 5 H	4 I	3 N	2 Z	1 V	0 C
Compare Index Reg	CPX	8C	3	3	9C	4	2	AC	6	2	BC	5	3				$X_H - M, X_L - (M + 1)$	•	•	⑦	↕	⑧	•
Decrement Index Reg	DEX													09	4	1	$X - 1 \rightarrow X$	•	•	•	↕	•	•
Decrement Stack Pntr	DES													34	4	1	$SP - 1 \rightarrow SP$	•	•	•	•	•	•
Increment Index Reg	INX													08	4	1	$X + 1 \rightarrow X$	•	•	•	↕	•	•
Increment Stack Pntr	INS													31	4	1	$SP + 1 \rightarrow SP$	•	•	•	•	•	•
Load Index Reg	LDX	CE	3	3	DE	4	2	EE	6	2	FE	5	3				$M \rightarrow X_H, (M + 1) \rightarrow X_L$	•	•	⑨	↕	R	•
Load Stack Pntr	LDS	8E	3	3	9E	4	2	AE	6	2	BE	5	3				$M \rightarrow SP_H, (M + 1) \rightarrow SP_L$	•	•	⑨	↕	R	•
Store Index Reg	STX				DF	5	2	EF	7	2	FF	6	3				$X_H \rightarrow M, X_L \rightarrow (M + 1)$	•	•	⑨	↕	R	•
Store Stack Pntr	STS				9F	5	2	AF	7	2	BF	6	3				$SP_H \rightarrow M, SP_L \rightarrow (M + 1)$	•	•	⑨	↕	R	•
Indx Reg → Stack Pntr	TXS													35	4	1	$X - 1 \rightarrow SP$	•	•	•	•	•	•
Stack Pntr → Indx Reg	TSX													30	4	1	$SP + 1 \rightarrow X$	•	•	•	•	•	•

OPERATIONS	MNEMONIC	RELATIVE OP	RELATIVE ~	RELATIVE #	INDEX OP	INDEX ~	INDEX #	EXTND OP	EXTND ~	EXTND #	IMPLIED OP	IMPLIED ~	IMPLIED #	BRANCH TEST	COND. CODE REG. 5 H	4 I	3 N	2 Z	1 V	0 C
Branch Always	BRA	20	4	2										None	•	•	•	•	•	•
Branch If Carry Clear	BCC	24	4	2										C = 0	•	•	•	•	•	•
Branch If Carry Set	BCS	25	4	2										C = 1	•	•	•	•	•	•
Branch If = Zero	BEQ	27	4	2										Z = 1	•	•	•	•	•	•
Branch If ≥ Zero	BGE	2C	4	2										$N \oplus V = 0$	•	•	•	•	•	•
Branch If > Zero	BGT	2E	4	2										$Z + (N \oplus V) = 0$	•	•	•	•	•	•
Branch If Higher	BHI	22	4	2										C + Z = 0	•	•	•	•	•	•
Branch If ≤ Zero	BLE	2F	4	2										$Z + (N \oplus V) = 1$	•	•	•	•	•	•
Branch If Lower Or Same	BLS	23	4	2										C + Z = 1	•	•	•	•	•	•
Branch If < Zero	BLT	2D	4	2										$N \oplus V = 1$	•	•	•	•	•	•
Branch If Minus	BMI	2B	4	2										N = 1	•	•	•	•	•	•
Branch If Not Equal Zero	BNE	26	4	2										Z = 0	•	•	•	•	•	•
Branch If Overflow Clear	BVC	28	4	2										V = 0	•	•	•	•	•	•
Branch If Overflow Set	BVS	29	4	2										V = 1	•	•	•	•	•	•
Branch If Plus	BPL	2A	4	2										N = 0	•	•	•	•	•	•
Branch To Subroutine	BSR	8D	8	2											•	•	•	•	•	•
Jump	JMP				6E	4	2	7E	3	3				See Special Operations	•	•	•	•	•	•
Jump To Subroutine	JSR				AD	8	2	BD	9	3					•	•	•	•	•	•
No Operation	NOP										01	2	1	Advances Prog. Cntr. Only	•	•	•	•	•	•
Return From Interrupt	RTI										3B	10	1		⑩					
Return From Subroutine	RTS										39	5	1		•	•	•	•	•	•
Software Interrupt	SWI										3F	12	1	See Special Operations	•	•	•	•	•	•
Wait for Interrupt*	WAI										3E	9	1		•	⑪	•	•	•	•

*WAI puts Address Bus, R/W, and Data Bus in the three-state mode while VMA is held low.

OPERATIONS	MNEMONIC	IMPLIED OP	IMPLIED ~	IMPLIED #	BOOLEAN OPERATION	COND. CODE REG. 5 H	4 I	3 N	2 Z	1 V	0 C
Clear Carry	CLC	0C	2	1	$0 \rightarrow C$	•	•	•	•	•	R
Clear Interrupt Mask	CLI	0E	2	1	$0 \rightarrow I$	•	R	•	•	•	•
Clear Overflow	CLV	0A	2	1	$0 \rightarrow V$	•	•	•	•	R	•
Set Carry	SEC	0D	2	1	$1 \rightarrow C$	•	•	•	•	•	S
Set Interrupt Mask	SEI	0F	2	1	$1 \rightarrow I$	•	S	•	•	•	•
Set Overflow	SEV	0B	2	1	$1 \rightarrow V$	•	•	•	•	S	•
Acmltr A → CCR	TAP	06	2	1	$A \rightarrow CCR$	⑫					
CCR → Acmltr A	TPA	07	2	1	$CCR \rightarrow A$	•	•	•	•	•	•

CONDITION CODE REGISTER NOTES: (Bit set if test is true and cleared otherwise)

1 (Bit V) Test: Result = 10000000?

2 (Bit C) Test: Result = 00000000?

3 (Bit C) Test: Decimal value of most significant BCD Character greater than nine? (Not cleared if previously set.)

4 (Bit V) Test: Operand = 10000000 prior to execution?

5 (Bit V) Test: Operand = 01111111 prior to execution?

6 (Bit V) Test: Set equal to result of N⊕C after shift has occurred.

7 (Bit N) Test: Sign bit of most significant (MS) byte = 1?

8 (Bit V) Test: 2's complement overflow from subtraction of MS bytes?

9 (Bit N) Test: Result less than zero? (Bit 15 = 1)

10 (All) Load Condition Code Register from Stack. (See Special Operations)

11 (Bit I) Set when interrupt occurs. If previously set, a Non-Maskable Interrupt is required to exit the wait state.

12 (All) Set according to the contents of Accumulator A.

APPENDIX C

M6800 Co-Resident Assembly Directives
Summary

DIRECTIVE	FUNCTION
ASSEMBLY CONTROL	
NAM	Program name
ORG	Origin
END	Program End
LISTING CONTROL	
PAGE	Top of page
SPC	Skip "n" lines
OPT NOO	No object tape
OPT O (Object Tape)	The Assembler will generate object tapes (selected by default).
OPT M (Memory File)	The Assembler will write machine code to memory.
OPT NOM	No memory (selected by default).
OPT S (Print Symbols)	The Assembler will print the symbols at the end of Pass 2.
OPT NOS	No printing of symbols (selected by default).
OPT L	The listing of assembled data will be printed (selected by default).
OPT NOL (No Listing)	The Assembler will not print a listing of the assembled data.

M6800 Co-Resident Assembly Directives
Summary (Continued)

DIRECTIVE	FUNCTION
OPT P	The listing will be paged (selected by default).
OPT NOP	The Assembler will inhibit format paging of the assembly listing.
OPT G	All data generated by the FCC, FCB, and FDB directions will be printed (selected by default).
OPT NOG (No Generate)	Causes only 1 line of data to be listed from the assembly directions FCC, FCB, and FDB.
DATA DEFINITION/STORAGE ALLOCATION	
FCC	Character string data (Form constant character)
FCB	One byte data (Form constant byte)
FDB	Reserve memory bytes (Form double byte)
SYMBOL DEFINITION	
EQU	Assign permanent value

APPENDIX D

ASSEMBLER ERROR MESSAGES

201 NAM DIRECTIVE ERROR
MEANING: The NAM directive is not the first source statement, or it occurs more than once in the same source program (Applies only to version 1.2)

202 EQU DIRECTIVE SYNTAX ERROR
MEANING: The EQU directive requires a label (Applies only to version 1.2)

204 STATEMENT SYNTACTICALLY INCORRECT
MEANING: The source statement is syntactically incorrect

205 LABEL ERROR
MEANING: The statement may not have a label or the label is syntactically incorrect.

206 REDEFINED SYMBOL
MEANING: The symbol has been previously defined.

207 UNDEFINED OPCODE
MEANING: The symbol in the operation code field is not a valid operation code mnemonic or directive.

208 BRANCH ERROR
MEANING: The branch count is beyond the relative byte's range. The allowance is
(* +2) - 128 D (* +2) + 127
where D = address of the destination of the branch instruction.
* = address of the first byte of the branch instruction.

209 ILLEGAL ADDRESS MODE
MEANING: The mode of addressing is not allowed with the operation code type.

210 BYTE OVERFLOW
MEANING: A one byte expression has been converted to a value greater than 255_{10} or less than -128_{10}.

211 UNDEFINED SYMBOL
MEANING: The symbol does not appear in the label field.

213* EQU DIRECTIVE SYNTAX ERROR
MEANING: The EQU directive requires a label.

216 DIRECTIVE OPERAND ERROR
MEANING: The directive operand field is in error.

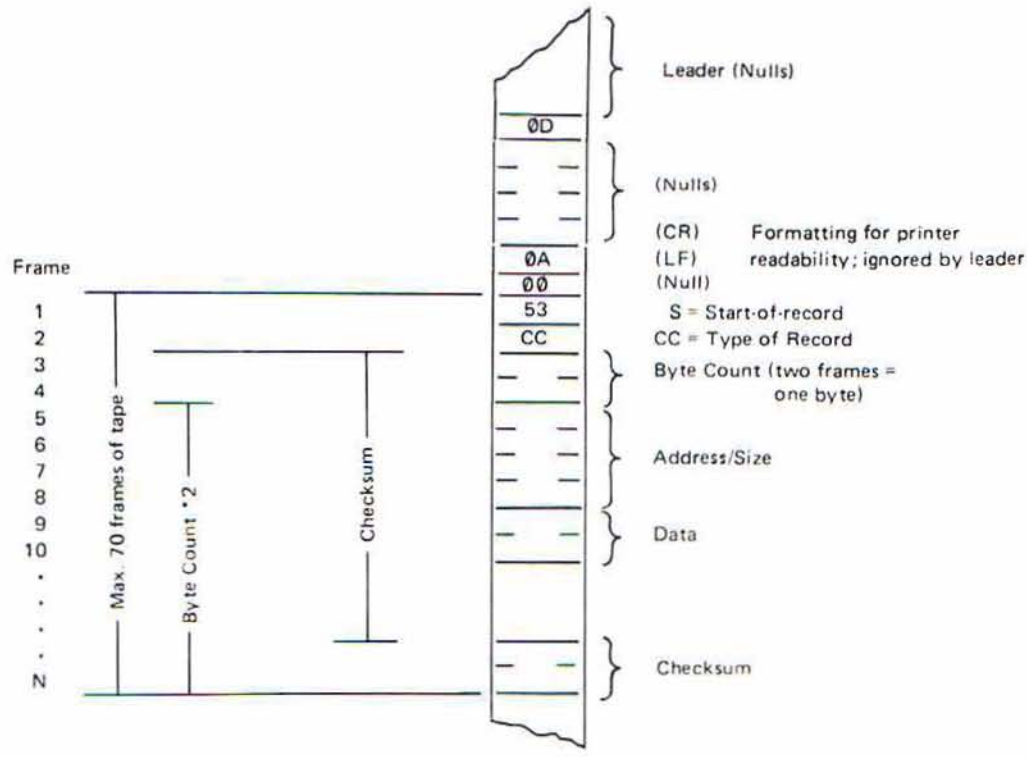
218 MEMORY ERROR
MEANING: The memory option was used and the object code was directed to overwrite the assembler/editor onto non-existent memory.

220 REDEFINED LABEL ERROR
MEANING: The symbol in the label field has been redefined and has a different value on Pass 2 than on Pass 1.

221 SYMBOL TABLE OVERFLOW
MEANING: The symbol table has overflowed. See assembler operation paragraph in Chapter 3 for extending the symbol table.

* In version 1.2 ERROR 213 is a redefined symbol error.

APPENDIX E
ABSOLUTE OBJECT RECORD FORMAT



Frames 3 through N are hexadecimal digits (in 7-bit ASCII) which are converted to BCD. Two BCD digits are combined to make one 8-bit byte.

The checksum is the one's complement of the summation of 8-bit bytes.

Frame	CC = 30 Header Record		CC = 31 Data Record		CC = 39 End-of-File Record	
1. Start-of-Record	53	S	53	S	53	S
2. Type of Record	30	0	31	1	39	9
3. Byte Count	31	12	31	16	30	03
4.	32		36		33	
5.	30		31		30	
6. Address/Size	30	0000	31	1100	30	0000
7.	30		30		30	
8.	30		30		30	
9. Data	34	48-H	39	98	46	FC
10.	38		38		43	
.	34	44-D	30	32		(Checksum)
.	34		32			
.	35	52-R				
.	32		41	A8 (Checksum)		
.			48			
.						
N. Checksum	39	9E				
	45					

APPENDIX F

SAMPLE PROGRAM

```
PAGE  001   PGM


00001                          NAM     PGM
00002                  * REVISION 1
00003                          OPT     O          OUTPUT OBJECT TAPE
00004                          OPT     S          SELECT PRINTING OF SYMBOLS
00005                          OPT     M
00006 2000                     ORG     $2000
00007       0003       COUNT   EQU     @3         @ INDICATES OCTAL
00008 2000 8E 2032     START   LDS     #STACK     INZ STACK POINTER
00009 2003 FE 2036             LDX     ADDR
00010 2006 C6 03               LDA B   #COUNT     IMMEDIATE ADDRESSING
00011 2008 96 0A       BACK    LDA A   10         DIRECT ADDRESSING
00012 200A A1 02               CMP A   2,X        INDEXED ADDRESSING
00013 200C 27 05               BEQ     FOUND      RELATIVE ADDRESSING
00014 200E 09                  DEX                IMPLIED ADDRESSING
00015 200F 5A                  DEC B              ACCUMULATOR ONLY ADDRESSING
00016 2010 26 F6               BNE     BACK
00017 2012 3E                  WAI                WAIT FOR INTERRUPT

00019 2013 BD 2019     FOUND   JSR     SUBRTN     JUMP TO SUBROUTINE
00020 2016 7E 2000             JMP     START      EXTENDED ADDRESSING
00021                  * COMMENT STATEMENT NOTE TRUNCATION 0123456789012345
00022 2019 16          SUBRTN  TAB                COMMENT FIELD TRUNCATION01234
00023 201A BA 2033             ORA A   BYTE       SET MOST SIGNIFICANT BIT
00024 201D 39                  RTS                RETURN FROM SUBROUTINE


00026 201E 0014                RMB     20         SCRATCH AREA FOR STACK
00027 2032 0001        STACK   RMB     1          START OF STACK
00028 2033 80          BYTE    FCB     $80        FORM CONSTANT BYTE
00029 2034 10                  FCB     $10,$4     $ INDICATES HEXADECIMAL
      2035 04
00030 2036 2038        ADDR    FDB     DATA       FORM CONSTANT DOUBLE BYTE
00031 2038 53          DATA    FCC     /SET/ FORM CONSTANT DATA STRING (ASCI
      2039 45
      203A 54
00032                          END
COUNT  0003 START  2000 BACK   2008 FOUND  2013 SUBRTN 2019
STACK  2032 BYTE   2033 ADDR   2036 DATA   2038
TOTAL ERRORS 00000
```

```
S00B000050474D202020202070
S11E20008E2032FE2036C6039609A1022705095A26F63EBD20197E200016BA34
S106201B20333933
S10B2033801004203853455409
S90300000FC
```

```
X
EXBUG 1.2 PRNT
BEG ADDR 0601 2000
END ADDR FFFF 203A
EXEC Y

2000 8E 20 32 FE 20 36 C6 03 96 09 A1 02 27 05 09 5A  . 2. 6F...!.'..Z
2010 26 F6 3E BD 20 19 7E 20 00 16 BA 20 33 39 55 55  &.>= .. ..: 39UU
2020 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55  UUUUUUUUUUUUUUUU
2030 55 55 55 80 10 04 20 38 53 45 54 55 55 55 55 55  UUU... 8SETUUUUU
BEG ADDR 2000
```

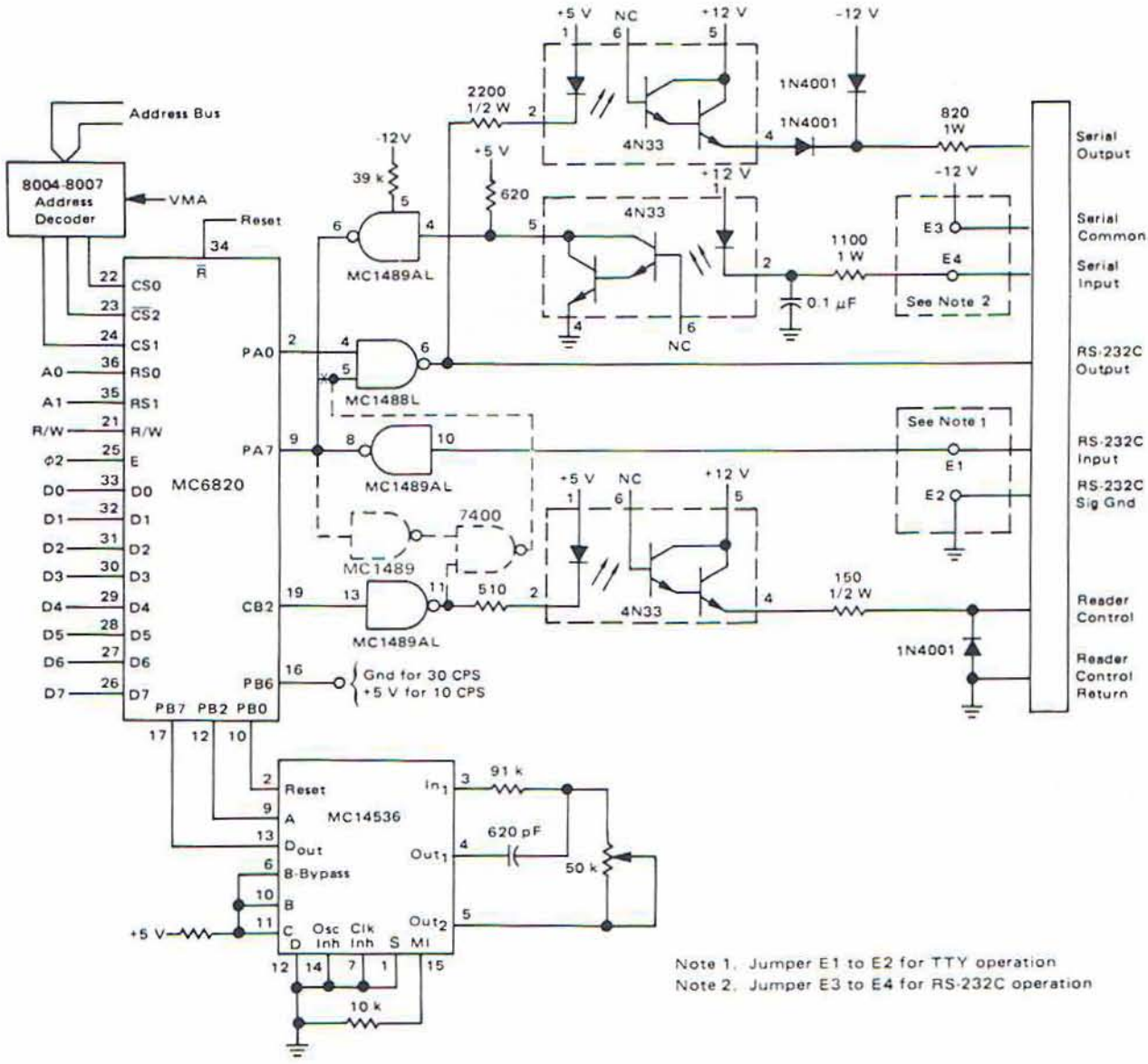
APPENDIX G

USING MIKBUG VERSION OF THE M6800 CO-RESIDENT SOFTWARE WITH THE MEX6800 D1 EVALUATION KIT

The MIKBUG version of the M6800 Co-Resident Software may be used with the MEK6800 D1 Evaluation Kit. As discussed in Engineering Note 100. The Evaluation Kit uses the MIKBUG Firmware stored in the MCM6830L7 ROM and interfaces with the selected data terminal via a MC6820 Peripheral Interface Adapter at addresses 8004 through 8007. Interfacing the Evaluation Kit with a RS-232C compatible or TTY (20 mA neutral current loop) terminal is depicted in Figure 3-4 of Engineering Note 100. The following changes are required to make the Evaluation Kit compatible with the M6800 Co-Resident Software.

a. Change the schematic in Figure 3-4 of Engineering Note 100 in accordance with Figure 1 of this document. The changes are depicted in dotted lines. Use the modified schematic to design and build your terminal interface.

b. Change the Control H character delete command in Table 2-1 of the M6800 RESIDENT SOFTWARE SUPPLEMENT to control H. (In Edition 1 of the supplement, add the Control A to Table 2-1.)



Note 1. Jumper E1 to E2 for TTY operation
Note 2. Jumper E3 to E4 for RS-232C operation

APPENDIX H

USE OF OTHER PERIPHERALS WITH THE CO-RESIDENT SOFTWARE

The Co-Resident Assembler/Editor has been designed to operate with TTY terminals equipped with automatic reader/punch control, or other compatible terminals such as Texas Instruments 733/ASR. Normally these console devices are also used for communication with the resident system's monitor program. Since other terminal types may offer advantages such as lower cost or higher performance, the Co-Resident software was designed to easily accommodate other peripherals. All Assembler/Editor input/output requests are processed by a common input/output program that resides in memory locations 0100_{16} -$02FF_{16}$.

Each input/output operation, such as punch record, print record, etc. is invoked by entering the input/output package through the appropriate jump vector. In the standard version, the input/output routine processes the input/output request and performs the input/output operations on the console device by calling the elementary input/output routines in the resident monitor. As a result, there are three versions of the common input/output program:

EXORciser -- Input/Output via EXbug

Evaluation -- Input/Output via MIKBUG
Module I

Evaluation -- Input/Output via MINIBUG II
Module II

In order to substitute the other peripheral devices, the user must supply the appropriate input/output drivers and patch the common I/O programs so that his drivers are called rather than the standard ones. To facilitate such modifications, source listings of the three common input/output programs are available through the M6800 User's Group Library.

LINE PRINTER INTERFACE

The input/output hardware and common input/output program modifications listed in the following paragraphs provide an example of the changes required to operate the Co-Resident Assembler/Editor with a line printer (Centronics type). The Disk Operating System includes the necessary commands and driver routines for implementing a line printer. The commands and driver routines required for paper tape and cassette are provided in the following paragraphs.

Hardware

The jumper connections listed in Table H1 must be performed if the MEX6820 Input/Output Module is used to interface the printer with the EXORciser. However, if the MEX68PI Printer Interface Module is used, these connections are not required. When using either of these modules, refer to the appropriate User's Guide Supplement.

Common Input/Output Program

A. Disk Assembler - ASMB (Version 1.3 Only)

In order to enable the printer patch included on the 1.3 version of the ASMB disk file, the disk file must be amended with the object file patch provided in Figure H1.

B. Disk Assembler - ASMB (Version 1.2 or 1.2A Only)

Version 1.2 of the disk assembler (ASMB) does not include a printer patch. However, the object file patch (ASMPATCH) provided in Figure H2 may be used to amend this version of the disk file to permit printer operation.

C. Paper Tape and Cassette Assemblers

Assembler software provided on either paper tape or cassette must be amended in the following manner to operate with a printer.

1. The object file (LPTDVR) provided in Figure H3 must be stored in memory at a location contiguous with the Co-Resident Assembler and Co-Resident Editor.

2. The current version of the assembler must be amended with the object file (ASMPATCH) listed in Figure H3.

TABLE H1 MEX6820 Input/Output Module Jumper

Requirements for Operating With a Line Printer

PIA SIGNAL	JUMPER CONNECTIONS*		PRINTER CONNECTOR PIN NUMBER
	PIA PIN NUMBER	CONNECTOR PIN NUMBER	
CA2	39	1	1
PA0	2	3	2
PA1	3	5	3
PA2	4	7	4
PA3	5	9	5
PA4	6	11	6
PA5	7	13	7
PA6	8	15	8
PA7	9	17	9
CA1	40	19	10

*Jumper connections to be performed between PIA1 (U13) or PIA2 (U15) on MEX6820 and connector P2 or P3 respectively.

NOTES:

1. The following pins on connector P2 or P3 (MEX6820) should be connected to ground.
   2, 4, 6, 8, 10, 12, 14, 16, 18 and 20

2. Printer connector pins 19 through 28 should be connected to ground.

```
!EDIT,,PAT

M6800 RESIDENT EDITOR 1.3
COPYRIGHT MOTOROLA 1976
$I NAM PAT
 ORG $20
 LDS #$FF8A
 END
$$

$BE$$

!ASM,2,PATO,PAT

M6800 RESIDENT ASSEMBLER 1.3
COPYRIGHT MOTOROLA 1976



---


PAGE  001   PAT


00001                           NAM     PAT
00002 0020                      ORG     $20
00003 0020 8E FF8A              LDS     #$FF8A
00004                           END

TOTAL ERRORS 00000

!RENAM,ASMB,ASMBX

!MERGE,ASMB,ASMBX,PATO
```

FIGURE H1. ASMB Version 1.3 Object File Printer Patch

```
PAGE  001   ASMPATCH


00001                          NAM     ASMPATCH
00002                          OPT     O,S
00003                  *THIS PROGRAM PATCHES THE M6800 RESIDENT ASSEMBLER
00004                  *TO REQUEST THE USER TO SELECT IF OUTPUT IS TO BE S
00005                  *TO THE PRINTER DEVICE USING THE PROM DRIVER
00006        0000      PASS    EQU     0
00007        FF8A      XSTACK  EQU     $FF8A
00008        FF53      AECHO   EQU     $FF53
00009        011E      XDATA   EQU     $11E
00010        011B      XCIE    EQU     $11B
00011        0133      XHEAD   EQU     $133
00012        0136      XLINE   EQU     $136
00013        EAD5      LDATA   EQU     $EAD5
00014        EADD      LDATA1  EQU     $EADD
00015        0100      ASMB    EQU     $100
00016  0020                    ORG     $20
00017  0020 8E FF8A            LDS     #XSTACK
00018  0023 96 00              LDA A   PASS
00019  0025 81 09              CMP A   #$9         OBJECT ONLY?
00020  0027 27 1A              BEQ     ASM         YES
00021  0029 CE 0054 TOP       LDX     #MSG
00022  002C BD 011E            JSR     XDATA
00023  002F CE 0000            LDX     #0
00024  0032 09        DELAY   DEX
00025  0033 26 FD              BNE     DELAY
00026  0035 7F FF53            CLR     AECHO
00027  0038 BD 011B            JSR     XCIE
00028  003B 81 59              CMP A   #'Y
00029  003D 27 07              BEQ     PRNTR
00030  003F 81 4E              CMP A   #'N
00031  0041 26 E6              BNE     TOP
00032  0043 7E 0100 ASM       JMP     ASMB
00033  0046 CE EAD5 PRNTR     LDX     #LDATA
00034  0049 FF 0134            STX     XHEAD+1
00035  004C CE EADD            LDX     #LDATA1
00036  004F FF 0137            STX     XLINE+1
00037  0052 20 EF              BRA     ASM
00038  0054 50        MSG     FCC     /PRINTER?/
       0055 52
       0056 49
       0057 4E
       0058 54
       0059 45
       005A 52
       005B 3F
00039  005C 04                 FCB     4
00040                          END
```

FIGURE H2. ASMB Versions 1.2 and 1.2A Object File Printer Patch

```
00001                           NAM LPTDVR
00002                           OPT    D
00003 EAB0                      ORG    $EAB0
00004                  *IN EDOS ASSEMBLER, SET THE FOLLOWING LOCATIONS
00005                  *     (PDATA) = JMP PDATA
00006                  *     (PDATA1) = JMP PDATA1
00007       EC11       CNTRL  EQU    $EC11     PIA ADDRESS
00008       EC10       DATA   EQU    $EC10     PIA ADDRESS
00009       EAB0       LIST   EQU    *
00010 EAB0 36           PSH A
00011 EAB1 7F EC11              CLR    CNTRL
00012 EAB4 86 FF                LDA A  #$FF
00013 EAB6 B7 EC10              STA A  DATA
00014 EAB9 86 3E                LDA A  #$3E
00015 EABB B7 EC11              STA A  CNTRL
00016 EABE 32                   PUL A
00017 EABF B7 EC10              STA A  DATA
00018 EAC2 86 36                LDA A  #$36
00019 EAC4 B7 EC11              STA A  CNTRL
00020 EAC7 86 3E                LDA A  #$3E
00021 EAC9 B7 EC11              STA A  CNTRL
00022 EACC B6 EC11 LIST1        LDA A  CNTRL
00023 EACF 2A FB                BPL    LIST1
00024 EAD1 B6 EC10              LDA A  DATA
00025 EAD4 39              LIST2  RTS
00026       EAD5           PDATA  EQU    *
00027 EAD5 86 0D                LDA A  #$D
00028 EAD7 8D D7                BSR    LIST
00029 EAD9 86 0A                LDA A  #$A
00030 EADB 8D D3                BSR    LIST
00031 EADD A6 00           PDATA1 LDA A  X
00032 EADF 81 04                CMP A  #4
00033 EAE1 27 F1                BEQ    LIST2
00034 EAE3 8D CB                BSR    LIST
00035 EAE5 08                   INX
00036 EAE6 20 F5                BRA    PDATA1
00037                           END
```

FIGURE H3. Paper Tape and Cassette Line Printer Driver Object File Patch

APPENDIX I

PROM VERSION OF CO-RESIDENT ASSEMBLER/EDITOR

The Co-Resident Assembler/Editor also is available to operate in the ROM environment. The Assembler's starting address is $C000_{16}$ and Editor's starting address is $C003_{16}$. The Co-Resident Assembler/Editor program uses 7k bytes of ROM and requires a minimum of 1k byte of RAM. This Assembler/Editor program resides in memory locations $C000_{16}$ through $DBFF_{16}$ and uses the RAM memory locations 0000_{16} through $01FF_{16}$ for scratch-pad memory. The symbol table starts at memory location 0200_{16} and ends at the default 1200_{16}. This provides a buffer for 500 symbols.

To change the size of the symbol table, the user enters into memory locations 0100_{16} and 0101_{16} the end-of-symbol-table address plus one. In this case, the user enters the Assembler at $C039_{16}$ rather than $C000_{16}$.

If the object code is to be written into memory (OPT M), the end-of-symbol-table address delimits the address. For example, if the symbol table ends at 1200_{16} (the default address), a program beginning at memory location 1200_{16} or may have its output directed into EXORciser memory (providing it is available).

It should be noted that the edit buffer starts at memory location 0200_{16} and extends to the end of continuous RAM memory.

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

M6345 PRINTED IN USA (11/76) 10PE 5M