TASM11

USER'S MANUAL

A Cross Assembler Program for Motorola 68HC11 Microcomputers

The Engineers Collaborative, Inc.
Webite at www.tec-i.com
Email support@tec-i.com

* * * IMPORTANT WARRANTY AND LIABILITY INFORMATION * * *

The Engineers Collaborative, Inc. makes no warranties, expressed or implied for this software. No warranty of fitness for a particular purpose is offered. The user is advised to test the software thoroughly before relying on it. The user assumes the entire risk of using the product. The total liability of The Engineers Collaborative, Inc. is limited to the purchase price of the product, and does not cover any lost profits, special, incidental or consequential damages, or any claim against the purchaser by any party.

* * * SOFTWARE LICENSE STATEMENT * * *

US copyright law and international treaty provisions protect this software. Therefore, you must treat this software *just like a book*, with the following exception. The Engineers Collaborative, Inc. authorizes you to make archival copies of the software for the sole purpose of backing-up our software and protecting your investment from loss. By saying, *just like a book*, The Engineers Collaborative, Inc. means that the software cannot be used by two different people in two different places at the same time.

* * * OTHER * * *

The information contained in this manual has been carefully checked and is believed to be accurate and complete at the time of printing. However, no responsibility is assumed for errors that might appear. The Engineers Collaborative, Inc. reserves the right to make changes to the product and/or the manual at any time without notice. Furthermore, The Engineers Collaborative, Inc. assumes no liability arising out of the use or application of any of its products. No part of this document may be copied or reproduced in any form or by any means without prior written consent of The Engineers Collaborative, Inc.

Web Site at www.tec-i.com
Email support@tec-i.com

TABLE OF CONTENTS

Table of Contents (cont.)

1.0 INTRODUCTION

TASM11 is a 68HC11 cross assembler program. It is designed to run on the IBM family of personal computers under the PC-DOS disk operating system Version 2.1 or higher. It also runs on true IBM compatible personal computers.

TASM11 takes its input from a PC-DOS text file and creates as output an object code file in either Intel HEX or Motorola S-record format and a listing file in PC-DOS text format.

When invoking TASM11 a number of option switches may be used to control the output generated. A variety of pseudo-operation codes may be included in the source code text file to control the format of the output generated.

This manual is intended as a reference document for the experienced 68HC11 assembly language programmer. It is assumed that the user is familiar with the 68HC11 instruction set and chip architure. Less experienced users should become familiar with the information contained in the following publicationsavailable from Motorola Semiconductor Products Inc. 3501 Ed Bluestein Blvd.,
Austin, Texas 78721

MOTOROLA SINGLE-CHIP MICROCOMPUTER DATA (DL132R1)
MC68HC11A8 PROGRAMMING REFERENCE GUIDE (MC68HC11A8RG/AD)

2.0 GETTING STARTED

Before using the TASM11 program for the first time the user should take the time to 1) make one or more copies of the distribution diskette, 2) make one or more bootable working copy diskettes and 3) READ THE MANUAL!

2.1 The Distribution Diskette

The Engineers Collaborative, Inc. TASM11 Cross Assembler Program is supplied on a 3 1/2 inch double sided double density floppy diskette.

The file supplied on this diskette is as follows:

TASM11.EXE This file contains the 68HC11 cross assembler program.

2.2 Backing up the distribution diskette

Use the DOS "DISKCOPY" command to make at least one backup copy of thedistribution diskette. When this is done, one of the backup copies should be used to create "working copy" diskettes as described below. At least one
backup copy and the original distribution diskette should then be stored in a safe place for use the inevitable day when disaster strikes!

2.3 Creating Working Copy Diskettes

Use the DOS "FORMAT" command with the /s option to format the number of working copy diskettes that you wish to create. The /s option copies the files required to make the working copy diskette a "bootable" or "system" diskette.

Use the DOS "COPY" command to copy all of the files supplied on the distribution diskette to the working copy diskettes.

Use the DOS "COPY" command to copy the DOS file ANSI.SYS to the working copy diskettes. This file must have been supplied with the disk operating system for your computer. Note: This file is not required if only TASM11 is to be used, but is required for all other The Engineers Collaborative, Inc. software development tools.

Use the DOS "COPY" command to copy the text editor program that you will be using to create 68HC11 source code files to the working copy diskettes.

At this point you have finished creating the working copy diskettes but to use the software tools you must make sure that the extended screen and keyboard device driver(ANSI.SYS) is loaded. The easiest way to do this is to boot the system with the working copy diskette in the boot drive.

3.0 RUNNING TASM11

To invoke TASM11, type at the DOS ready prompt:

[d:]TASM11 [d:]fname.asm [option 1] [option 2] . . .

[d:] denotes optional drive specifiers where the TASM11 program or the source file is to be found and [option] denotes optional output control switches. The output files will always be directed to the currently logged drive.

fname.asm is the source code filename. There is no default source file extension so the extension, if any, must be included when invoking TASM11.

Any number of output control options may follow the source file name and these options can be in any order. Options must be separated by one or more spaces.

Normally, TASM11 generates two output files, a listing file and an object code file. These files have the same name as the source code file with different file extensions. The listing file has extension '.PRN' and the object code file has extension '.MIK' or optionally '.HEX'. Object code files with extension '.MIK' are in Motorola S record format and object code files with extension '.HEX' are in Intel HEX format. The default format is '.MIK'

TASM11 is a two pass assembler, it must read the source file twice to complete the generation of object code. On the first pass through the source file statements TASM11 does only the work required to associate values with the symbols that appear in the source file. The output files are generated on the second pass. If errors are found on the first pass, error messages are displayed on the screen and TASM11 terminates before attempting the second pass.

3.1 OUTPUT CONTROL SWITCHES

The output control switches are 3 letter mnemonics that are typed on the command line following the source code filename when invoking TASM11. These switches control the output generated by TASM11 as follows:

NLF NO LISTING FILE - NLF can be used to prevent the generation of the listing file. This switch might be used when first assembling a source code file to see if the file assembles without errors.

NOF NO OBJECT FILE - NOF can be used to prevent the generation of the object code file. It might be used in the same manner as NLF.

NST NO SYMBOL TABLE - NST prevents the generation of the symbol table. The symbol table is an alphabetical listing of all symbols and their associated values. The symbol table is appended to the end of the listing file.

LTP LIST TO PRINTER - LTP can be used to send a copy of the listing file to the printer.

HEX Changes the object code file format to INTEL HEX.

WOE WAIT ON ERROR - WOE is used to prevent error messages from scrolling off the screen before they can be read. When used,

assembly is halted after an error message has been sent to the screen until the operator presses any key.

SPF SUPPRESS PAGE FORMATTING - This switch is used to suppress normal page formatting. TASM11 normally inserts a form feed character into the listing file after each 64 lines (or after the number of lines specified by the PGLEN pseudo-op) then it inserts 5 lines of page header information into the listing file. See section 5.5. The SPF switch is used to prevent this page formatting.

3.2 EXAMPLES

Some examples of invoking TASM11 follow. The examples assume that a source code file named WIDGET.ASM exists.

Type at the DOS ready prompt:

TASM11 WIDET.ASM or tasm11 widget.asm

In this example no output control switches are used therefore a listing file will be created on the currently logged drive named WIDGET.PRN and an object code file will be created in Motorola S record format named WIDGET.MIK.

If we had typed at the DOS ready prompt:

TASM11 WIDGET.ASM HEX LTP

a listing file named WIDGET.PRN and an object code file named WIDGET.HEX in Intel HEX format would have been created and a copy of the listing file would have been sent to the printer.

If we had typed:

TASM11 WIDGET.ASM NOF NLF WOE

no output files would be created and if errors were detected in the source

file the program would pause after displaying the error message until the operator pressed any key.

4.0 TASM11 SOURCE FILES

Source code files compatible with TASM11 are standard PC-DOS ASCII text files. The only control codes that can be present in the files are carriage return (CR), line feed (LF) and horizontal tab characters.

These files may be created by the EDLIN text editor supplied with DOS but it is recommended that a more sophisticated editor such as Personal Editor from IBM, Wordstar used in the non-document mode or the Sidekick notepad editor be used.

The reader is encouraged to study the example listing in Appendix A to acquire an understanding of source code syntax and format acceptable to TASM11.

Source files consist of a series of statements of one of the following forms where [] denotes optional statement fragments:

```
[*comment]
  or
[;comment]
                OPCODE          [arguments]     [comment]
symbol          OPCODE          [arguments]     [comment]
                PSEUDO-OP       [arguments]     [comment]
symbol          PSEUDO-OP       [arguments]     [comment]
```

PCODE is any standard 68HC11 operation code and must not start in the first column of a statement. Opcodes can be either upper or lower case characters or combinations of both. BRCLR and BrClr are considered the same by TASM11.

PSEUDO-OP is any TASM11 pseudo-operation and must not start in the first column of a statement. TASM11 pseudo-ops are described later in this document. They can be either upper or lower case characters or combinations of both.

Symbols, if present, must start in column 1 of the statement, consist of 1 to 15 ASCII alphanumeric characters and begin with a letter, a period, or an underscore. Capital letters are significant in symbol names. That is the assembler will treat 'Dog', 'dog' and 'doG' as unique symbols.

Comments must be preceded by a semicolon (;) or an asteric (*) when they appear on a line by themselves. Comments that appear as the last field of a line do not have to be preceded by a semicolon or asteric but must be preceded by a space or horizontal tab character and can not start with an expression

operator character (+ - * /....).
The fields of a statement must be delimited by at least one space or a horizontal tab character.

The arguments portion of a statement consists of from 1 to 4 symbols, numeric constants, character constants, expressions or combinations depending on the OPCODE or PSEUDO-OP.
Constants consist of a sequence of characters optionally preceded by a radix specifier. The default radix is ten. A radix of 16 may be specified by a leading $ or by a trailing H. If a trailing H is used the first digit of the constant must be a decimal digit.

As an example the numeric constant 15 may be specified as 15, $F, or as 0FH.

Constant examples:

```
One_hundred    EQU  100       a decimal constant
One_hundred    EQU  $64       a hexadecimal constant
One_hundred    EQU  @144      an octal constant
One_hundred    EQU  %1100100  a binary constant
Two_hundred    EQU  0C8H      another form of hex constant
```

Character constants are enclosed in single or double quotes (' or ").

As an example, the statement:

```
TEXT DB   "The quick brown fox."
```

is equivalent to the statement:

```
TEXT DB   'The quick brown fox.'
```

5.0 EXPRESSIONS

Expressions may consist of symbols, constants or the characters * or $ (* or $ denote the current value of the program counter) joined together by one of the operators: +-*/%&|^. The operators are the same as in C:

+	add
-	subtract
*	multiply

/	divide
%	remainder after division
&	bitwise logical and
|	bitwise logical or
^	bitwise logical exclusive or

Expressions are evaluated left to right and there is no provision for parenthesized expressions. Arithmetic is carried out in signed two's compliment integer precision (16 bits on the IBM PC).

Constants are constructed as follows:

'	followed by an ASCII character
$	followed by hexadecimal constant
@	followed by an octal constant
%	followed by a binary constant
digits	decimal constant

Refer to the listing in appendix a for examples of correct expression syntax.

6.0 TASM11 PSEUDO-OPERATIONS

6.1 Storage Definition Pseudo-Ops

DEFINE BYTES

```
            DB   Argument
            or
Symbol      DB   Argument
```

The define bytes pseudo-op is used to initialize memory locations to the value of the argument. It can also be used to associate a symbolic name to the memory location.
The argument must be either a numeric constant or a character string constant. Numeric constants must be a value between 0H and 0FFH.

Examples:

```
            DB   $FF
            DB   0B3H
            DB   255
            DB   'How now brown cow'
symbol      DB   "How now brown cow"
```

DEFINE WORD

```
            DW   Argument
            or
Symbol      DW   Argument
```

The define word pseudo-op is used to initialize two consecutive memory locations to the value of the argument. It can also be used to associate a symbolic name to the first of 2 memory locations.
The argument must be a numeric constant with value between 0H and 0FFFFH.

Examples:

```
            DW   0
            DW   257
            DW   0FC49H
symbol      DW   $FFFF
```

DEFINE STORAGE

```
            DS   Argument
            or
Symbol      DS   Argument
```

The define storage pseudo-op is used to reserve the number of consecutive memory locations specified by the argument. It can also be used to associate a symbolic name to the first of these memory locations.
The argument must be a numeric constant with value between 0 and 255.
The bytes reserved are not initialized.

Examples:

```
            DS   10   ;reserves 10 bytes of storage
symbol      DS   10H  ;reserves 16 bytes of storage
```

FORM CONSTANT BYTES

```
            FCB  Argument
            or
Symbol      FCB  Argument
```

The form constant bytes pseudo-op is used to initialize one or more memory locations to the values specified by the argument. It can also be used to associate a symbolic name to the memory location.

The values of the argument must be either numeric constants, character constants or symbol names that have been previously defined. The values of the argument must be separated by commas. Numeric constants must be a value between 0H and 0FFH.

Examples:

```
            FCB  $FF
            FCB  0B3H, %01110000
symbol      FCB  1, 2, 3, 4
```

FORM DOUBLE BYTES

```
            FDB  Argument
            or
Symbol      FDB  Argument
```

The form double bytes pseudo-op is used to initialize two or more consecutive memory locations to the values specified by the argument. It can also be used to associate a symbolic name to the memory location. The values of the argument must be either numeric constants, character constants or symbol names that have been previously defined. The values of the argument must be separated by commas. Numeric constants must be a value between 0H and 0FFFFH.

Examples:

```
            FDB  $FFFF
            FDB  0B333H, %1110111001110000
symbol      FDB  1, 2, 3, 4
```

RESERVE MEMORY BYTES

```
        RMB  Argument
        or
Symbol  RMB  Argument
```

The reserve memory bytes pseudo-op is used to reserve the number of consecutive memory locations specified by the argument. It can also be used to associate a symbolic name to the first of these memory locations. The argument must be a numeric constant with value between 0 and 255. The bytes reserved are not initialized.

Examples:

```
            RMB  10   ;reserves 10 bytes of storage
symbol      RMB  10H  ;reserves 16 bytes of storage
```

BLOCK STORE ZEROS

```
            BSZ  Argument
            or
Symbol      BSZ  Argument
```

The block store zeros pseudo-op is used to reserve the number of consecutive memory locations specified by the argument. It can also be used to associate a symbolic name to the first of these memory locations.
The argument must be a numeric constant with value between 0 and 255.
The bytes reserved are each initialized to zero.

Examples:

```
            BSZ  10   ;reserves 10 bytes of storage initialized to zero
symbol      BSZ  10H  ;reserves 16 bytes of storage initialized to zero
```

ZERO MEMORY BYTES

```
            ZMB  Argument
            or
Symbol      ZMB  Argument
```

The zero memory bytes pseudo-op is the same as the BSZ pseudo-op described above. It is included in TASM11 for compatibility with other assemblers.

FORM CONSTANT CHARACTERS

```
            FCC  Argument
            or
Symbol      FCC  Argument
```

The form constant characters pseudo-op is used to initialize one or more consecutive memory locations to the string characters specified by the argument. The characters of the string must be delimited by (') or (").

Examples:

```
            FCC  'Now is the time for all good men. . .'
Symbol      FCC  "How now brown cow."
```

FILL

```
            FILL Argument1,Argument2
            or
Symbol      FILL Argument1,Argument2
```

The fill pseudo-op is used to fill the number of consecutive memory locations specified by argument2 with the value specified by argument1. The arguments must be separated by a comma. The value of argument1 must be between 0 and 255.

Examples:

```
            FILL $FF, 10      fills ten memory locations with 0FFH
Symbol      FILL 1,3          fills three memory locations with one's.
```

6.2 ORG and END Pseudo-Ops

ORG argument

Set program origin. Sets the program counter to the value of the argument. This statement must precede the first code generating statement in a source file. Additional ORG pseudo-ops may be used throughout the source file to create separate program fragments.
The argument may be a numeric constant or an expression as shown in the examples below.

Examples:

```
            ORG  100H            ;sets origin to 256
```

```
            ORG  10 * 2          ;sets origin to 20
            ORG  10 / 2          ;sets origin to 5
            ORG  10 - 2          ;sets origin to 8
            ORG  10 + 2          ;sets origin to 12
            ORG  $ + 2           ;sets origin to current value of pc + 2
            ORG  * + 2           ;sets origin to current value of pc + 2
                                 Note that either $ or * may be used to
                                 specify current program counter value.
memsize     EQU  2 * 1024        ;memsize = 2048

            ORG  memsize - 8     ;sets origin to 2040
```

END argument

The END pseudo-op is an optional statement that can be included as the last statement of the primary source file(INCLUDE files must not have an END statement). If used, the value of the argument becomes the start address specified by the end record of the object code file.
The argument must be a numeric or symbolic constant with value between 0H and 0FFFFH.

Examples:

```
            END  80H

start       EQU  80H
            END  start
```

6.3 Symbolic Constants

symbol EQU argument

The EQU pseudo-op assigns the value of the argument to the symbol name. The argument must be a numeric constant or an expression. Once defined by EQU the symbolic name may be used throughout the source file in place of its value.
Once a symbol has been assigned a value by the EQU pseudo-op the same symbol can not be assigned another value elsewhere in the source file. Symbolic constants are constants!

Examples:

```
PORTA     EQU       $1000
memsize   EQU       2 * 1024

          ldx       #PORTA
          bset      0,x $01
          ORG       memsize - 10
```

6.4 Include Pseudo-OP

This pseudo-op causes the specified file to be processed as if its source code statements were present in the main source file at the location of the INCLUDE pseudo-op. Included files must not contain INCLUDE pseudo-ops. Include files can not be nested! The main source file can contain any number of INCLUDE pseudo-ops. Included files must not contain an END pseudo-op. If the file to be included does not reside on the currently logged drive, the filename must contain the drive specification.

Examples:

```
INCLUDE     B:MYFILE.ASM
INCLUDE     EQUATES.ASM
INCLUDE     A:TIMERISR.ASM
```

6.5 Listing Control Pseudo-Ops

PAGE

The PAGE (or optionally just PAG) pseudo-op causes a form feed character to be inserted into the listing output file. This causes the printed listing to skip to the top of the next page.

PGLEN argument

The PGLEN pseudo-op sets the number of lines that will be printed on each page of the listing to the value specified by the argument. The argument must be a numeric constant.

Examples:

```
PGLEN    55
PGLEN    66


TITLE    'text'
SBTTL    'text'
```

Each page of the listing file contains a heading at the top of the page that is of the following form:

blank line
The Engineers Collaborative, Inc. 68HC11 Cross Assembler V#.#
blank line
User specified title
User specified subtitle
blank line

The user specifies the text of the title and subtitle by means of the TITLE and SBTTL pseudo-ops as shown in the following examples.

```
TITLE    'Text of users title goes here!'
SBTTL    'The text of the users subtitle goes here!'



NOLIST
```

The NOLIST pseudo-op can be used to prevent certain sections of the source file from appearing in the listing file. All lines of the source file after the NOLIST statement in encountered will not appear in the listing file unless a LIST pseudo-op is encountered.

```
LIST
```

The LIST pseudo-op is used to resume the listing of source file lines in the listing file after listing has been stopped with the NOLIST pseudo-op.

6.6 OPT PSEUDO-OP

The option (OPT) pseudo-op is included for compatibility with other assemblers. Generally the same functions can be performed with other

pseudo-ops. The syntax of the OPT pseudo-op is as follows:

```
OPT  option
```

where the option is any of the following:

LIST or L	same as the LIST pseudo-op
NOLIST or NOL	same as the NOLIST pseudo-op
CYCLES or C	does nothing in TASM11
NOCYCLES or NOC	does nothing in TASM11
CONTCYCLES or CONTC	does nothing in TASM11

Examples:

```
OPT  LIST
OPT  NOLIST
```

6.7 NULL PSEUDO-OPS

Null pseudo-ops are included in TASM11 for compatibility with other assemblers. TASM11 recognizes them so that errors will not be generated but the program performs no functions when thay are encountered.

The NULL PSEUDO-OPS are:

```
NAME
NAM
SPC
CYCLES
NOCYCLES
CONTCYCLES
```

7.0 ERROR MESSAGES

Source files must assemble with no errors before the generated code can be trusted. When no errors are found, TASM11 prints the message

"Assembly complete . . . errors = 0, warnings = 0."

on the video screen and includes this message at the end of the listing file.

When errors are detected a message is displayed on the video screen and an error message is included in the listing file on the line above the line in which the error was detected. Error messages sent to the video screen include a description of the error, the line number of the line in error and the name of the file that contains the line in error. Only one error is detected per line. Before the program ends it prints a message that indicates how many lines contained errors.

TASM11 is a two pass assembler. Pass 1 must be completed without errors before pass 2 is started. Errors detected on pass 1 are displayed on the video screen only since the output files are generated on pass 2. If no errors are detected during pass 1 TASM11 prints the message "Starting Pass 2 . . ." on the video screen.

7.1 Fatal Errors

Fatal errors are errors serious enough that the assembler can not continue to assemble the source statements. When a fatal error is detected TASM11 prints the error message on the video screen then exits to DOS.

Fatal error messages and causes are listed below.

"Fatal error - Error - can't open fname"

This message is displayed when the main source file typed when invoking TASM11 can not be opened. The file name may have been typed incorrectly or the file may not reside on the currently logged drive or the specified drive.

"FATAL ERROR, couldn't open include file fname

This message is displayed when an include file can't be opened. The file name may have been typed incorrectly in the main source file or the file may not reside on the currently logged drive or the specified drive.

"Fatal error - Error - source file name must follow program name!"

This message is displayed when the operator fails to type the name of the source code file after TASM11 when invoking the assembler.

"Fatal error - error writing listing file, check disk space!"
"Fatal ERROR writing to disk - check disk space. . ."
"Fatal error - Can't open output listing file - check disk space"
"Fatal error - Can't open output object code file - check disk space"

These errors may occur part of the way through an assembly if all of the available disk space on the currently logged drive is used. To correct this error more space will have to be made available on the currently logged drive.

"Fatal error - No mem for fwd. refs
"Fatal error - No space for fwd. refs

These errors may occur if your machine does not have enough available memory to store the required information for the forward references in the source file being assembled. To correct the problem the number of forward references in the source file must be reduced. Currently, TASM11 allocates space for 500 forward references.

Fatal error - Error in Mnemonic table

If this error appears, the TASM11.EXE program file has been corrupted. A new copy of the program will have to be obtained from a backup copy or from The Engineers Collaborative, Inc..

7.2 Non-Fatal Errors

Non-fatal errors are errors that are detected but not serious enough to cause termination of the assembly. They most certainly are serious enough to prevent the program under development from performing its intended function however.

A list of non-fatal errors and probable causes are as follows:
These error messages will appear in the listing file on the line above the source statement in which the error was detected. When displayed on the video screen the line number in which the error occurred and the name of file containing the line in error will be printed in front of the error message.

"Unrecognized Mnemonic"
"Undefined Pseudo Operation"

Indicates an invalid opcode field in the source code statement. The opcode or pseudo-op was misspelled or not separated from other fields of the source statement by spaces or tab characters.

"Symbol Redefined"

An attempt was made to change the definition of an already defined symbol. This can happen if more than one source statement has been given the same label or symbol name.
"Illegal Symbol Name"

The symbol field contains a symbol that contains characters other than the alphanumeric ASCII characters or the symbol name starts with a character other than the alpha characters or the underscore or the period.

"Out of memory space for the symbol table."

Your machine does not contain enough available memory space to contain the symbol table. The number of symbols must be reduced or the amount of available memory increased.

"Symbol Undefined on Pass 2"

The most probable cause of this error is that a symbol name was referred to in the arguments field of a source statement but never appeared in the symbol field of another source statement.
As an example, if we had the source statement

```
JMP     WATCHDOG
```

and no source statement had WATCHDOG in the symbol field then we would get an undefined symbol error when TASM11 read the JMP statement on pass 2 of the assembly.

"Undefined operand on Pass One"

The operands for the EQU and ORG statements must be defined on pass 1 and must occur in the source file before they are used.

"Branch out of Range"

Target addresses for the relative branch instructions must be within -128 to +127 bytes of the first byte of the instruction following the branch instruction. This error indicates that the target address of the branch is outside these limits.

"Immediate Addressing Illegal"
"Extended Addressing not allowed"
"Unknown Addressing Mode"

These errors are generated when an illegal or unrecognizeable addressing mode is detected for certain operation codes.

"SYNTAX"

This is a catch all error message that indicates that the assembler found something that it did not understand but is so confused that it can not give a more specific error message.

"Phasing error"

Lets hope this error never occurs. It indicates that assumptions made by the assembler on pass 1 were proven invalid on pass 2. If this error occurs check for possible problems related to the size of forward references. Rearrange your code and try again!

"Missing Delimiter"
"Missing Delimiter Character"

These errors occur with the DB or FCC pseudo-ops when the delimiter character is missing from the end of a string constant.

"EQU requires lable"

An EQU pseudo-op was found that did not have a symbol field. EQU statements must have a symbol with which to equate the value specified by the operand

field.

"Bad fill"

A FILL pseudo-op was found that did not have the proper operands. The FILL spseudo-op requires two operands separated by a comma.

"Unrecognized OPT"

An OPT pseudo-op was found with an invalid argument or operand.

7.3 WARNING ERRORS

Warning errors are generated when the assembler finds unexpected values or statement syntax in the source file. The resulting code generated may or may not be correct. Source files should be corrected so that no warnings errors occur before the generated code is trusted.

A list of warning errors and possible causes are as follows:

"Value Truncated"

An operand was encountered that was larger than 255 when a byte value was expected. The value was truncated to fit in a byte.

Indexed Addressing Assumed"

Incorrect indexed addressing syntax was found and the indexed addressing mode was assumed.

"Continuation too long"

To eliminate this warning make sure that each source statement is complete on one line.

8.0 68HC11 ADDRESSING MODES

A summary of 68HC11 addressing modes and instruction syntax is provided in this section. The reader is referred to the publications listed in the introduction for additional information. Appendix A contains an actual assembly listing of all the 68HC11 instructions and the addressing modes available for each instruction. This listing should be used to provide more comprehensive examples of proper 68HC11 source code syntax.

8.1 Inherent Addressing

Inherent addressing instructions contain the operand information implicitly. These are one byte instructions such as:

```
ROLA        ;Rotate the A register left thru carry
ASLA        ;Arithmetic shift left the A register
```

8.2 Immediate Addressing

In immediate addressing the operand data is contained in the byte(s) following the opcode byte. Immediate addressing is designated by a leading "#".

```
LDA  #7         ;load the A register with the number 7.
LDX  #large     ;load the X register with the value of the symbol
                ;large
```

8.3 Direct Addressing

In direct addressing the address of the operand is contained in the byte following the instruction opcode. Since the address in contained in one byte, direct addressing can only be used to access memory locations in the first page of memory (addresses 0 - 255). The operand field must contain a numeric

or symbolic constant.

```
LDA     0           ;load the A register with the contents of
                    ;memory location 0
LDB     small       ;load the B register with the contents of
                    ;the memory location specified by small.
```

8.4 Extended Addressing

In extended addressing the address of the operand is contained in the two bytes following the instruction opcode. Extended addressing can be used to access any memory location. The operand field must contain a numeric or symbolic constant.

```
LDA     256         ;load the A register with the contents of
                    ;memory location 256
LDX     large       ;load the X register with the contents of
                    ;the memory location specified by large.
```

8.5 Indexed Addressing

In the indexed addressing mode, one of the index registers(x or y) is used in calculating the effective address. In this case the effective address is variable and depends on two factors. 1) the current contents of the index register(x or y) being used and 2) the the 8 bit unsigned offset contained in the instruction.

```
SMALL   EQU     20

        LDX     #SMALL
        LDA     0,X         ;load the A reg
                            ;with the contents of address 20
        LDA     7,X         ;load the A reg
                            ;with the contents of address 27
```

8.6 Relative Addressing

Relative addressing is used by the branch instructions and by the bit test and branch instructions BRCLR and BRSET. In relative addressing, the last byte of the instruction is added to the program counter to obtain the address of the next instruction if the branch is taken.
The target address of the branch must be within -128 to +127 bytes of the program counter value following the branch instruction. TASM11 calculates the

value of the byte to added to the pc value and generates a "branch out of range" error message if these limits are exceeded.

```
BCS     timer           ;branch to the inst labeled "timer" if the
                        ;carry flag is set

ldx   #$1000      ;load x with address of port a
brclr  0,x $81 timer      ;branch to the inst labeled "timer" if
                          ;bits 7 and 1 of port a are zero.
```

9.0 OUTPUT FILE FORMAT

9.1 Listing file format

Each line of the listing file generated by TASM11 can be a maximum of 132 characters long. The text from each line of the source code file is copied to the corresponding line of the listing file starting at column 16. To create neat appearing printed listing files the user must be aware that each listing file line will be 16 columns longer than the source file lines. If listing files will be printed with 80 column lines then the text of the source file lines should occupy 64 columns or less. If listing files will be printed with 132 column lines then the text of the source file lines should occupy 116 columns or less. It is up to the user to set up his printer for the most appropriate line length.

Columns 2 - 5 of a listing file line contain a 4 character hexadecimal number that corresponds to the current value of the program counter.
Columns 7 - 14 contain the hexadecimal representation of the code generated by TASM11.

Each page of the listing file contains a 6 line heading as explained in the description of the TITLE and SBTTL pseudo-ops.

At the end of the listing file is the alphabetical listing of all symbols and their associated values in hexadecimal representation.

9.2 Motorola S-record Object File Format

Each line of this file has the following format:

Bytes 1 & 2"S1
- 3 & 4 — The number of data bytes in this record + 3.
- 5 & 6 — Load Address - high byte
- 7 & 8 — Load Address - low byte
- 9 . . X — Data bytes, 2 characters each
- X+1 & X+2 checksum
- X+3 & X+4 carriage return and linefeed

The last record in the file is the same as above except that it starts with "S9 and the load address field contains the program starting address when specified by an END pseudo-op in the source file. This field contains zero otherwise.

The checksum is the ones-compliment of the 8 bit sum without carry of the byte count, the load address and the data bytes.

9.3 Intel HEX Object File Format

Each line of this file has the following format:

Bytes 1 — Colon (:)
- 2 & 3 — Number of data bytes in this record
- 4 & 5 — Load Address - high byte
- 6 & 7 — Load Address - low byte
- 8 & 9 — Unused - "00"
- 10 - X — Data bytes - 2 characters each
- X+1 & X+2 — checksum
- X+3 & x+4 — carriage return and linefeed

The last line of the file is similar to that described above except the number of data bytes is 0 and the load address is the program starting address if specified by an END pseudo-op in the source file. This field is zero otherwise.

The checksum is the twos compliment of the 8 bit sum without carry of all the data bytes, the load address and the byte count.

```
The Engineers Collaborative, Inc. 68HC11 Cross Assembler V1.0

APPENDIX A - TASM11 REFERENCE MANUAL
TASM11 SOURCE FILE SYNTAX, OPCODES & ADDRESSING MODE EXAMPLES

                ;*******************************************************************************
                ;**                          Test File - Fname = TEST11.ASM                   **
                ;**                                12/17/87                                   **
                ;*******************************************************************************
                ;
                ; This is a comment line!
                * This is another form of comment line!
                ; The following show the syntax of pseudo-ops whose source lines do not show
                ; up in the listing file. Use as shown without comment char to start the line.
                ;
                ;   INCLUDE A:MYFILE.ASM                    ;include MYFILE.ASM
                ;   TITLE'Title text'                       ;specify the page header title
                ;   SBTTL'Subtitle text'                    ;specify the page header subtitle
                ;   PGLEN64                                 ;print 64 lines per page
                ;   NOLIST                                  ;turn listing off
                ;   LIST                                    ;turn listing on
                ;   OPT  NOLIST                             ;another way to turn listing off
                ;   PAGE                                    ;force a page break
                ;
                                                            ;Equate examples
FFFF            memory_size   EQU     64 * 1024 - 1         ;EXPRESSION EXAMPLE
0000            ram_memory    equ     0                     ;start of ram memory
E000            rom_memory    equ     $e000                 ;start of rom memory
B600            eeprom_memory equ     $b600                 ;start of eeprom memory
FFC0            irq_vectors   equ     $ffc0                 ;

1122            WORD          EQU     1122H                 ;16 BIT CONSTANT
0003            BYTE          EQU     3                     ; 8 BIT CONSTANT
00C3            MASK          EQU     $c3                   ;
1000            REGISTERS     EQU     $1000
1000            PORTA         EQU     REGISTERS
1001            RESERVED      EQU     REGISTERS+1
1002            PIOC          EQU     REGISTERS+2
1003            PORTC         EQU     REGISTERS+3
1004            PORTB         EQU     REGISTERS+4
1005            PORTCL        EQU     REGISTERS+5
1006            RESERVED2     EQU     REGISTERS+6
1007            DDRC          EQU     REGISTERS+7
1008            PORTD         EQU     REGISTERS+8
1009            DDRD          EQU     REGISTERS+9
100A            PORTE         EQU     REGISTERS+10
100B            CFORC         EQU     REGISTERS+11
```

```
100C            OC1M            EQU     REGISTERS+12
100D            OC1D            EQU     REGISTERS+13
100E            TCNT            EQU     REGISTERS+14
100F            TCNT1           EQU     REGISTERS+15
1010            TIC1            EQU     REGISTERS+16
1011            TIC11           EQU     REGISTERS+17
1012            TIC2            EQU     REGISTERS+18
1013            TIC22           EQU     REGISTERS+19
1014            TIC3            EQU     REGISTERS+20
1015            TIC33           EQU     REGISTERS+21
1016            TOC1            EQU     REGISTERS+22
1017            TOC11           EQU     REGISTERS+23
1018            TOC2            EQU     REGISTERS+24
1019            TOC22           EQU     REGISTERS+25
101A            TOC3            EQU     REGISTERS+26
101B            TOC33           EQU     REGISTERS+27
101C            TOC4            EQU     REGISTERS+28
```

```
101D            TOC44         EQU     REGISTERS+29
101E            TOC5          EQU     REGISTERS+30
101F            TOC55         EQU     REGISTERS+31
1020            TCTL1         EQU     REGISTERS+32
1021            TCTL2         EQU     REGISTERS+33
1022            TMSK1         EQU     REGISTERS+34
1023            TFLG1         EQU     REGISTERS+35
1024            TMSK2         EQU     REGISTERS+36
1025            TFLG2         EQU     REGISTERS+37
1026            PACTL         EQU     REGISTERS+38
1027            PACNT         EQU     REGISTERS+39
1028            SPCR          EQU     REGISTERS+40
1029            SPSR          EQU     REGISTERS+41
102A            SPDR          EQU     REGISTERS+42
102B            BAUD          EQU     REGISTERS+43
102C            SCCR1         EQU     REGISTERS+44
102D            SCCR2         EQU     REGISTERS+45
102E            SCSR          EQU     REGISTERS+46
102F            SCDR          EQU     REGISTERS+47
1030            ADCTL         EQU     REGISTERS+48
1031            ADR1          EQU     REGISTERS+49
1032            ADR2          EQU     REGISTERS+50
1033            ADR3          EQU     REGISTERS+51
1034            ADR4          EQU     REGISTERS+52
1035            RESERVED3     EQU     REGISTERS+53
1036            RESERVED4     EQU     REGISTERS+54
1037            RESERVED5     EQU     REGISTERS+55
1038            RESERVED6     EQU     REGISTERS+56
1039            OPTION        EQU     REGISTERS+57
103A            COPRST        EQU     REGISTERS+58
103B            PPROG         EQU     REGISTERS+59
103C            HPRIO         EQU     REGISTERS+60
103D            INIT          EQU     REGISTERS+61
103E            TEST1         EQU     REGISTERS+62
103F            CONFIG        EQU     REGISTERS+63
```

```
The Engineers Collaborative, Inc. 68HC11 Cross Assembler V1.0

APPENDIX A - TASM11 REFERENCE MANUAL
TASM11 SOURCE FILE SYNTAX, OPCODES & ADDRESSING MODE EXAMPLES

                                          ;expression examples
0001            ONE           EQU  3-2       this is a comment
0001            ONE_1         EQU 3 - 2      another comment
0001            One           EQU  1 + 0     still another comment
0001            One_1         EQU $10 - 15   still another comment
0006            Six           EQU  3*2       multiply
0007            SEVEN         EQU Six+1      add
000A            TEN           EQU  20/2      divide
0002            TWO           EQU  8% 3      remainder after division
0080            _128          EQU  $FF & $80 bitwise AND
0005            FIVE          EQU  4 |1      bitwise OR
0007            SEVEN_1       EQU  5 ^ 2     bitwise XOR
0006            Complex       equ ONE+ONE_1*Six/TWO =6 (expressions are evaluated left to right)
                ;
E000                          ORG     rom_memory     ORIGINATE AT ADDRESS 0
                                                     ;EXAMPLES OF STORAGE DEFINITION PSEUDO-OPS
E000 00                       DB      0              define byte example
E001 0000                     DW      0              define word example
E003                          DS      2              define storage example, reserves 2 bytes
E005                          RMB     10             reserve 10 memory bytes
E00F 00010203                 FCB     0,1,2,3        form constant bytes
E013 00000001                 FDB     0,1,2,3        form double bytes
     00020003
E01B 48656C6C                 FCC     'Hello'        form constant characters
     6FE020 48656C6C          DB      'Hello'        another way to do it!
     6F
E025 00000000                 ZMB     8              zero 8 memory bytes
     00000000
E02D 00000000                 BSZ     8              another way to do it!
     00000000
E035 FFFFFFFF                 FILL    0FFH, 10       fill ten bytes with 0ffH
     FFFFFFFF
     FFFF
                ;
E049                    ORG     $ + 10               ;ORG WITH EXPRESSION
E053                          ORG     * + 10          another way to do it!
E053 4E6F7720   MESSAGEDB'Now is the time for all good men . . .'  a long message
     69732074
     68652074
     696D6520
     666F7220
     616C6C20
     676F6F64
     206D656E
     202E202E
     202E
```

```
The Engineers Collaborative, Inc. 68HC11 Cross Assembler V1.0

APPENDIX A - TASM11 REFERENCE MANUAL
TASM11 SOURCE FILE SYNTAX, OPCODES & ADDRESSING MODE EXAMPLES

                ;*************************************************************************
                ;**THE FOLLOWING IS AN ASSEMBLY OF ALL 68HC11 OPCODES AND ADDRESSING MODES   **
                ;*************************************************************************
                ;
E079 1B         START           ABA                     ;ADD ACCUMULATORS
E07A 3A                         ABX                     ;ADD B TO X
E07B 183A                       ABY                     ;ADD B TO Y
                                                        ;ADD WITH CARRY TO A
E07D 8903                       ADCA    #BYTE                   ;IMMEDIATE
E07F 9903                       ADCA    BYTE                    ;DIRECT
E081 B91122                     ADCA    WORD                    ;EXTENDED
E084 A900                       ADCA    0,X                     ;INDEXED BY X
E086 18A900                     ADCA    0,Y                     ;INDEXED BY Y
E089 A903                       ADCA    BYTE,X                  ;INDEXED BY X
E08B 18A903                     ADCA    BYTE,Y                  ;INDEXED BY Y

                                                        ;ADD WITH CARRY TO B
E08E C903                       ADCB    #BYTE                   ;IMMEDIATE
E090 D903                       ADCB    BYTE                    ;DIRECT
E092 F91122                     ADCB    WORD                    ;EXTENDED
E095 E900                       ADCB    0,X                     ;INDEXED BY X
E097 18E900                     ADCB    0,Y                     ;INDEXED BY Y
E09A E903                       ADCB    BYTE,X                  ;INDEXED BY X
E09C 18E903                     ADCB    BYTE,Y                  ;INDEXED BY Y

                                                        ;ADD MEMORY TO A
E09F 8B03                       ADDA    #BYTE                   ;IMMEDIATE
E0A1 9B03                       ADDA    BYTE                    ;DIRECT
E0A3 BB1122                     ADDA    WORD                    ;EXTENDED
E0A6 AB00                       ADDA    0,X                     ;INDEXED BY X
E0A8 18AB00                     ADDA    0,Y                     ;INDEXED BY Y
E0AB AB03                       ADDA    BYTE,X                  ;INDEXED BY X
E0AD 18AB03                     ADDA    BYTE,Y                  ;INDEXED BY Y

                                                        ;ADD MEMORY TO B
E0B0 CB03                       ADDB    #BYTE                   ;IMMEDIATE
E0B2 DB03                       ADDB    BYTE                    ;DIRECT
E0B4 FB1122                     ADDB    WORD                    ;EXTENDED
E0B7 EB00                       ADDB    0,X                     ;INDEXED BY X
E0B9 18EB00                     ADDB    0,Y                     ;INDEXED BY Y
E0BC EB03                       ADDB    BYTE,X                  ;INDEXED BY X
E0BE 18EB03                     ADDB    BYTE,Y                  ;INDEXED BY Y

                                                        ;ADD 16 BIT TO D
E0C1 C31122                     ADDD    #WORD                   ;IMMEDIATE
E0C4 D303                       ADDD    BYTE                    ;DIRECT
E0C6 F31122                     ADDD    WORD                    ;EXTENDED
E0C9 E300                       ADDD    0,X                     ;INDEXED BY X
E0CB 18E300                     ADDD    0,Y                     ;INDEXED BY Y
E0CE E303                       ADDD    BYTE,X                  ;INDEXED BY X
E0D0 18E303                     ADDD    BYTE,Y                  ;INDEXED BY Y
```

```
The Engineers Collaborative, Inc. 68HC11 Cross Assembler V1.0

APPENDIX A - TASM11 REFERENCE MANUAL
TASM11 SOURCE FILE SYNTAX, OPCODES & ADDRESSING MODE EXAMPLES

                                    ;AND A WITH MEMORY
E0D3 8403      ANDA #BYTE           ;IMMEDIATE
E0D5 9403      ANDA BYTE            ;DIRECT
E0D7 B41122    ANDA WORD            ;EXTENDED
E0DA A400      ANDA 0,X             ;INDEXED BY X
E0DC 18A400    ANDA 0,Y             ;INDEXED BY Y
E0DF A403      ANDA BYTE,X          ;INDEXED BY X
E0E1 18A403    ANDA BYTE,Y          ;INDEXED BY Y

                                    ;AND B WITH MEMORY
E0E4 C403      ANDB #BYTE           ;IMMEDIATE
E0E6 D403      ANDB BYTE            ;DIRECT
E0E8 F41122    ANDB WORD            ;EXTENDED
E0EB E400      ANDB 0,X             ;INDEXED BY X
E0ED 18E400    ANDB 0,Y             ;INDEXED BY Y
E0F0 E403      ANDB BYTE,X          ;INDEXED BY X
E0F2 18E403    ANDB BYTE,Y          ;INDEXED BY Y

                                    ;ARITHMETIC SHIFT LEFT
E0F5 48        ASLA                 ;A INHERENT
E0F6 58        ASLB                 ;B INHERENT
E0F7 05        ASLD                 ;D INHERENT
E0F8 781122    ASL  WORD            ;EXTENDED
E0FB 6800      ASL  0,X             ;INDEXED BY X
E0FD 186800    ASL  0,Y             ;INDEXED BY Y
E100 6803      ASL  BYTE,X          ;INDEXED BY X
E102 186803    ASL  BYTE,Y          ;INDEXED BY Y

                                    ;ARITHMETIC SHIFT RIGHT
E105 47        ASRA                 ;A INHERENT
E106 57        ASRB                 ;B INHERENT
E107 771122    ASR  WORD            ;EXTENDED
E10A 6700      ASR  0,X             ;INDEXED BY X
E10C 186700    ASR  0,Y             ;INDEXED BY Y
E10F 6703      ASR  BYTE,X          ;INDEXED BY X
E111 186703    ASR  BYTE,Y          ;INDEXED BY Y

E114 24FE      BCC  $               ;BRANCH IF CARRY CLEAR

                                    ;CLEAR BIT(S)
E116 1503C3    BCLR BYTE MASK       ;DIRECT
E119 1D00C3    BCLR 0,X MASK        ;INDEXED BY X
E11C 181D00C3  BCLR 0,Y MASK        ;INDEXED BY Y
E120 1D03C3    BCLR BYTE,X MASK     ;INDEXED BY X
E123 181D03C3  BCLR BYTE,Y MASK     ;INDEXED BY Y

E127 25FE      BCS  $               ;BRANCH IF CARRY SET
E129 27FE      BEQ  $               ;BRANCH IF ZERO
E12B 2CFE      BGE  $               ;BRANCH IF >= ZERO
E12D 2EFE      BGT  $               ;BRANCH IF > ZERO
E12F 22FE      BHI  $               ;BRANCH IF HIGHER
E131 24FE      BHS  $               ;BRANCH IF HIGHER OR SAME
```

```
The Engineers Collaborative, Inc. 68HC11 Cross Assembler V1.0

APPENDIX A - TASM11 REFERENCE MANUAL
TASM11 SOURCE FILE SYNTAX, OPCODES & ADDRESSING MODE EXAMPLES

                                                ;BIT(S) TEST A WITH MEMORY
E133 8503      BITA    #BYTE                    ;IMMEDIATE
E135 9503      BITA    BYTE                     ;DIRECT
E137 B51122    BITA    WORD                     ;EXTENDED
E13A A500      BITA    0,X                      ;INDEXED BY X
E13C 18A500    BITA    0,Y                      ;INDEXED BY Y
E13F A503      BITA    BYTE,X                   ;INDEXED BY X
E141 18A503    BITA    BYTE,Y                   ;INDEXED BY Y

                                                ;BIT(S) TEST B WITH MEMORY
E144 C503      BITB    #BYTE                    ;IMMEDIATE
E146 D503      BITB    BYTE                     ;DIRECT
E148 F51122    BITB    WORD                     ;EXTENDED
E14B E500      BITB    0,X                      ;INDEXED BY X
E14D 18E500    BITB    0,Y                      ;INDEXED BY Y
E150 E503      BITB    BYTE,X                   ;INDEXED BY X
E152 18E503    BITB    BYTE,Y                   ;INDEXED BY Y

E155 2FFE      BLE     $                        ;BRANCH IF <= ZERO
E157 25FE      BLO     $                        ;BRANCH IF LOWER
E159 23FE      BLS     $                        ;BRANCH IF LOWER OR SAME
E15B 2DFE      BLT     $                        ;BRANCH IF < ZERO
E15D 2BFE      BMI     $                        ;BRANCH IF MINUS
E15F 26FE      BNE     $                        ;BRANCH IF != ZERO
E161 2AFE      BPL     $                        ;BRANCH IF PLUS
E163 20FE      BRA     $                        ;BRANCH ALWAYS

                                                ;BRANCH IF BIT(S) CLEAR
E165 1303C3FC  BRCLR   BYTE MASK $              ;DIRECT
E169 1F00C3FC  BRCLR   0,X MASK $               ;INDEXED BY X
E16D 181F00C3  BRCLR   0,Y MASK $               ;INDEXED BY Y
     FB
E172 1F03C3FC  BRCLR   BYTE,X MASK $            ;INDEXED BY X
E176 181F03C3  BRCLR   BYTE,Y MASK $            ;INDEXED BY Y
     FB

E17B 21FE      BRN     $                        ;BRANCH NEVER

                                                ;BRANCH IF BIT(S) SET
E17D 1203C3FC  BRSET   BYTE MASK $              ;DIRECT
E181 1E00C3FC  BRSET   0,X MASK $               ;INDEXED BY X
E185 181E00C3  BRSET   0,Y MASK $               ;INDEXED BY Y
     FB
E18A 1E03C3FC  BRSET   BYTE,X MASK $            ;INDEXED BY X
E18E 181E03C3  BRSET   BYTE,Y MASK $            ;INDEXED BY Y
     FB

                                                ;SET BIT(S)
E193 1403C3    BSET    BYTE MASK                ;DIRECT
E196 1C00C3    BSET    0,X MASK                 ;INDEXED BY X
E199 181C00C3  BSET    0,Y MASK                 ;INDEXED BY Y
E19D 1C03C3    BSET    BYTE,X MASK              ;INDEXED BY X
E1A0 181C03C3  BSET    BYTE,Y MASK              ;INDEXED BY Y

E1A4 8DFE      BSR     $                        ;BRANCH TO SUBROUTINE
E1A6 28FE      BVC     $                        ;BRANCH IF OVERFLOW CLEAR
E1A8 29FE      BVS     $                        ;BRANCH IF OVERFLOW SET
```

```
The Engineers Collaborative, Inc. 68HC11 Cross Assembler V1.0

APPENDIX A - TASM11 REFERENCE MANUAL
TASM11 SOURCE FILE SYNTAX, OPCODES & ADDRESSING MODE EXAMPLES

E1AA 11        CBA                ;COMPARE A TO B
E1AB 0C        CLC                ;CLEAR CARRY BIT
E1AC 0E        CLI                ;CLEAR INTERRUPT MASK

                                  ;CLEAR MEMORY BYTE
E1AD 7F1122    CLR  WORD          ;EXTENDED
E1B0 6F00      CLR  0,X           ;INDEXED BY X
E1B2 186F00    CLR  0,Y           ;INDEXED BY Y
E1B5 6F03      CLR  BYTE,X        ;INDEXED BY X
E1B7 186F03    CLR  BYTE,Y        ;INDEXED BY Y

E1BA 4F        CLRA               ;CLEAR A
E1BB 5F        CLRB               ;CLEAR B

E1BC 0A        CLV                ;CLEAR OVERFLOW FLAG

                                  ;COMPARE A TO MEMORY
E1BD 8103      CMPA #BYTE         ;IMMEDIATE
E1BF 9103      CMPA BYTE          ;DIRECT
E1C1 B11122    CMPA WORD          ;EXTENDED
E1C4 A100      CMPA 0,X           ;INDEXED BY X
E1C6 18A100    CMPA 0,Y           ;INDEXED BY Y
E1C9 A103      CMPA BYTE,X        ;INDEXED BY X
E1CB 18A103    CMPA BYTE,Y        ;INDEXED BY Y

                                  ;COMPARE B TO MEMORY
E1CE C103      CMPB #BYTE         ;IMMEDIATE
E1D0 D103      CMPB BYTE          ;DIRECT
E1D2 F11122    CMPB WORD          ;EXTENDED
E1D5 E100      CMPB 0,X           ;INDEXED BY X
E1D7 18E100    CMPB 0,Y           ;INDEXED BY Y
E1DA E103      CMPB BYTE,X        ;INDEXED BY X
E1DC 18E103    CMPB BYTE,Y        ;INDEXED BY Y

                                  ;COMPLIMENT
E1DF 43        COMA               ;A
E1E0 53        COMB               ;B
E1E1 731122    COM  WORD          ;EXTENDED
E1E4 6300      COM  0,X           ;INDEXED BY X
E1E6 186300    COM  0,Y           ;INDEXED BY Y
E1E9 6303      COM  BYTE,X        ;INDEXED BY X
E1EB 186303    COM  BYTE,Y        ;INDEXED BY Y

                                  ;COMPARE D TO MEMORY
E1EE 1A831122  CPD  #WORD         ;IMMEDIATE
E1F2 1A9303    CPD  BYTE          ;DIRECT
E1F5 1AB31122  CPD  WORD          ;EXTENDED
E1F9 1AA300    CPD  0,X           ;INDEXED BY X
E1FC CDA300    CPD  0,Y           ;INDEXED BY Y
E1FF 1AA303    CPD  BYTE,X        ;INDEXED BY X
E202 CDA303    CPD  BYTE,Y        ;INDEXED BY Y
```

```
The Engineers Collaborative, Inc. 68HC11 Cross Assembler V1.0

APPENDIX A - TASM11 REFERENCE MANUAL
TASM11 SOURCE FILE SYNTAX, OPCODES & ADDRESSING MODE EXAMPLES

                                    ;COMPARE X TO MEMORY
E205 8C1122    CPX  #WORD           ;IMMEDIATE
E208 9C03      CPX  BYTE            ;DIRECT
E20A BC1122    CPX  WORD            ;EXTENDED
E20D AC00      CPX  0,X             ;INDEXED BY X
E20F CDAC00    CPX  0,Y             ;INDEXED BY Y
E212 AC03      CPX  BYTE,X          ;INDEXED BY X
E214 CDAC03    CPX  BYTE,Y          ;INDEXED BY Y

                                    ;COMPARE Y TO MEMORY
E217 188C1122  CPY  #WORD           ;IMMEDIATE
E21B 189C03    CPY  BYTE            ;DIRECT
E21E 18BC1122  CPY  WORD            ;EXTENDED
E222 1AAC00    CPY  0,X             ;INDEXED BY X
E225 18AC00    CPY  0,Y             ;INDEXED BY Y
E228 1AAC03    CPY  BYTE,X          ;INDEXED BY X
E22B 18AC03    CPY  BYTE,Y          ;INDEXED BY Y

E22E 19        DAA                  ;DECIMAL ADJUST A

                                    ;DECREMENT
E22F 4A        DECA                 ;A
E230 5A        DECB                 ;B
E231 7A1122    DEC  WORD            ;EXTENDED
E234 6A00      DEC  0,X             ;INDEXED BY X
E236 186A00    DEC  0,Y             ;INDEXED BY Y
E239 6A03      DEC  BYTE,X          ;INDEXED BY X
E23B 186A03    DEC  BYTE,Y          ;INDEXED BY Y

E23E 34        DES                  ;DECREMENT STACK POINTER
E23F 09        DEX                  ;DECREMENT X
E240 1809      DEY                  ;DECREMENT Y

                                    ;XOR A WITH MEMORY
E242 8803      EORA #BYTE           ;IMMEDIATE
E244 9803      EORA BYTE            ;DIRECT
E246 B81122    EORA WORD            ;EXTENDED
E249 A800      EORA 0,X             ;INDEXED BY X
E24B 18A800    EORA 0,Y             ;INDEXED BY Y
E24E A803      EORA BYTE,X          ;INDEXED BY X
E250 18A803    EORA BYTE,Y          ;INDEXED BY Y

                                    ;XOR B WITH MEMORY
E253 C803      EORB #BYTE           ;IMMEDIATE
E255 D803      EORB BYTE            ;DIRECT
E257 F81122    EORB WORD            ;EXTENDED
E25A E800      EORB 0,X             ;INDEXED BY X
E25C 18E800    EORB 0,Y             ;INDEXED BY Y
E25F E803      EORB BYTE,X          ;INDEXED BY X
E261 18E803    EORB BYTE,Y          ;INDEXED BY Y

E264 03        FDIV                 ;FRACTIONAL DIVIDE 16 BY 16
E265 02        IDIV                 ;INTEGER DIVIDE 16 BY 16
```

```
The Engineers Collaborative, Inc. 68HC11 Cross Assembler V1.0

APPENDIX A - TASM11 REFERENCE MANUAL
TASM11 SOURCE FILE SYNTAX, OPCODES & ADDRESSING MODE EXAMPLES

                                    ;INCREMENT
E266 4C         INCA                ;A
E267 5C         INCB                ;B
E268 7C1122     INC  WORD           ;EXTENDED
E26B 6C00       INC  0,X            ;INDEXED BY X
E26D 186C00     INC  0,Y            ;INDEXED BY Y
E270 6C03       INC  BYTE,X         ;INDEXED BY X
E272 186C03     INC  BYTE,Y         ;INDEXED BY Y

E275 31         INS                 ;INCREMENT STACK POINTER
E276 08         INX                 ;INCREMENT X
E277 1808       INY                 ;INCREMENT Y

                                    ;JUMP
E279 7E1122     JMP  WORD           ;EXTENDED
E27C 6E00       JMP  0,X            ;INDEXED BY X
E27E 186E00     JMP  0,Y            ;INDEXED BY Y
E281 6E03       JMP  BYTE,X         ;INDEXED BY X
E283 186E03     JMP  BYTE,Y         ;INDEXED BY Y

                                    ;JUMP TO SUBROUTINE
E286 9D03       JSR  BYTE           ;DIRECT
E288 BD1122     JSR  WORD           ;EXTENDED
E28B AD00       JSR  0,X            ;INDEXED BY X
E28D 18AD00     JSR  0,Y            ;INDEXED BY Y
E290 AD03       JSR  BYTE,X         ;INDEXED BY X
E292 18AD03     JSR  BYTE,Y         ;INDEXED BY Y

                                    ;LOAD ACCUM A
E295 8603       LDAA #BYTE          ;IMMEDIATE
E297 9603       LDAA BYTE           ;DIRECT
E299 B61122     LDAA WORD           ;EXTENDED
E29C A600       LDAA 0,X            ;INDEXED BY X
E29E 18A600     LDAA 0,Y            ;INDEXED BY Y
E2A1 A603       LDAA BYTE,X         ;INDEXED BY X
E2A3 18A603     LDAA BYTE,Y         ;INDEXED BY Y

                                    ;LOAD ACCUM B
E2A6 C603       LDAB #BYTE          ;IMMEDIATE
E2A8 D603       LDAB BYTE           ;DIRECT
E2AA F61122     LDAB WORD           ;EXTENDED
E2AD E600       LDAB 0,X            ;INDEXED BY X
E2AF 18E600     LDAB 0,Y            ;INDEXED BY Y
E2B2 E603       LDAB BYTE,X         ;INDEXED BY X
E2B4 18E603     LDAB BYTE,Y         ;INDEXED BY Y

                                    ;LOAD ACCUM D
E2B7 CC1122     LDD  #WORD          ;IMMEDIATE
E2BA DC03       LDD  BYTE           ;DIRECT
E2BC FC1122     LDD  WORD           ;EXTENDED
E2BF EC00       LDD  0,X            ;INDEXED BY X
E2C1 18EC00     LDD  0,Y            ;INDEXED BY Y
E2C4 EC03       LDD  BYTE,X         ;INDEXED BY X
E2C6 18EC03     LDD  BYTE,Y         ;INDEXED BY Y
```

```
The Engineers Collaborative, Inc. 68HC11 Cross Assembler V1.0

APPENDIX A - TASM11 REFERENCE MANUAL
TASM11 SOURCE FILE SYNTAX, OPCODES & ADDRESSING MODE EXAMPLES

                                    ;LOAD STACK POINTER
E2C9 8E1122    LDS  #WORD           ;IMMEDIATE
E2CC 9E03      LDS  BYTE            ;DIRECT
E2CE BE1122    LDS  WORD            ;EXTENDED
E2D1 AE00      LDS  0,X             ;INDEXED BY X
E2D3 18AE00    LDS  0,Y             ;INDEXED BY Y
E2D6 AE03      LDS  BYTE,X          ;INDEXED BY X
E2D8 18AE03    LDS  BYTE,Y          ;INDEXED BY Y

                                    ;LOAD X
E2DB CE1122    LDX  #WORD           ;IMMEDIATE
E2DE DE03      LDX  BYTE            ;DIRECT
E2E0 FE1122    LDX  WORD            ;EXTENDED
E2E3 EE00      LDX  0,X             ;INDEXED BY X
E2E5 CDEE00    LDX  0,Y             ;INDEXED BY Y
E2E8 EE03      LDX  BYTE,X          ;INDEXED BY X
E2EA CDEE03    LDX  BYTE,Y          ;INDEXED BY Y

                                    ;LOAD Y
E2ED 18CE1122  LDY  #WORD           ;IMMEDIATE
E2F1 18DE03    LDY  BYTE            ;DIRECT
E2F4 18FE1122  LDY  WORD            ;EXTENDED
E2F8 1AEE00    LDY  0,X             ;INDEXED BY X
E2FB 18EE00    LDY  0,Y             ;INDEXED BY Y
E2FE 1AEE03    LDY  BYTE,X          ;INDEXED BY X
E301 18EE03    LDY  BYTE,Y          ;INDEXED BY Y

                                    ;LOGICAL SHIFT LEFT
E304 48        LSLA                 ;A
E305 58        LSLB                 ;B
E306 05        LSLD                 ;D
E307 781122    LSL  WORD            ;EXTENDED
E30A 6800      LSL  0,X             ;INDEXED BY X
E30C 186800    LSL  0,Y             ;INDEXED BY Y
E30F 6803      LSL  BYTE,X          ;INDEXED BY X
E311 186803    LSL  BYTE,Y          ;INDEXED BY Y

                                    ;LOGICAL SHIFT RIGHT
E314 44        LSRA                 ;A
E315 54        LSRB                 ;B
E316 04        LSRD                 ;D
E317 741122    LSR  WORD            ;EXTENDED
E31A 6400      LSR  0,X             ;INDEXED BY X
E31C 186400    LSR  0,Y             ;INDEXED BY Y
E31F 6403      LSR  BYTE,X          ;INDEXED BY X
E321 186403    LSR  BYTE,Y          ;INDEXED BY Y

E324 3D        MUL                  ;MULTIPLY 8 X 8

                                    ;NEGATE(2'S COMPLIMENT)
E325 40        NEGA                 ;A
E326 50        NEGB                 ;B
E327 701122    NEG  WORD            ;EXTENDED
E32A 6000      NEG  0,X             ;INDEXED BY X
E32C 186000    NEG  0,Y             ;INDEXED BY Y
E32F 6003      NEG  BYTE,X          ;INDEXED BY X
E331 186003    NEG  BYTE,Y          ;INDEXED BY Y
```

```
The Engineers Collaborative, Inc. 68HC11 Cross Assembler V1.0

APPENDIX A - TASM11 REFERENCE MANUAL
TASM11 SOURCE FILE SYNTAX, OPCODES & ADDRESSING MODE EXAMPLES

E334 01        NOP                 ;NO OPERATION

                                   ;OR ACCUM A
E335 8A03      ORAA #BYTE          ;IMMEDIATE
E337 9A03      ORAA BYTE           ;DIRECT
E339 BA1122    ORAA WORD           ;EXTENDED
E33C AA00      ORAA 0,X            ;INDEXED BY X
E33E 18AA00    ORAA 0,Y            ;INDEXED BY Y
E341 AA03      ORAA BYTE,X         ;INDEXED BY X
E343 18AA03    ORAA BYTE,Y         ;INDEXED BY Y

                                   ;OR ACCUM B
E346 CA03      ORAB #BYTE          ;IMMEDIATE
E348 DA03      ORAB BYTE           ;DIRECT
E34A FA1122    ORAB WORD           ;EXTENDED
E34D EA00      ORAB 0,X            ;INDEXED BY X
E34F 18EA00    ORAB 0,Y            ;INDEXED BY Y
E352 EA03      ORAB BYTE,X         ;INDEXED BY X
E354 18EA03    ORAB BYTE,Y         ;INDEXED BY Y

E357 36        PSHA                ;PUSH A ONTO STACK
E358 37        PSHB                ;PUSH B ONTO STACK
E359 3C        PSHX                ;PUSH X ONTO STACK
E35A 183C      PSHY                ;PUSH Y ONTO STACK
E35C 32        PULA                ;PULL A FROM STACK
E35D 33        PULB                ;PULL B FROM STACK
E35E 38        PULX                ;PULL X FROM STACK
E35F 1838      PULY                ;PULL Y FROM STACK

                                   ;ROTATE LEFT
E361 49        ROLA                ;A
E362 59        ROLB                ;B
E363 791122    ROL  WORD           ;EXTENDED
E366 6900      ROL  0,X            ;INDEXED BY X
E368 186900    ROL  0,Y            ;INDEXED BY Y
E36B 6903      ROL  BYTE,X         ;INDEXED BY X
E36D 186903    ROL  BYTE,Y         ;INDEXED BY Y

                                   ;ROTATE RIGHT
E370 46        RORA                ;A
E371 56        RORB                ;B
E372 761122    ROR  WORD           ;EXTENDED
E375 6600      ROR  0,X            ;INDEXED BY X
E377 186600    ROR  0,Y            ;INDEXED BY Y
E37A 6603      ROR  BYTE,X         ;INDEXED BY X
E37C 186603    ROR  BYTE,Y         ;INDEXED BY Y

E37F 3B        RTI                 ;RETURN FROM INTERRUPT
E380 39        RTS                 ;RETURN FROM SUBROUTINE

E381 10        SBA                 ;SUBTRACT B FROM A
```

```
The Engineers Collaborative, Inc. 68HC11 Cross Assembler V1.0

APPENDIX A - TASM11 REFERENCE MANUAL
TASM11 SOURCE FILE SYNTAX, OPCODES & ADDRESSING MODE EXAMPLES

                                    ;SUBTRACT WITH CARRY FROM A
E382 8203      SBCA #BYTE           ;IMMEDIATE
E384 9203      SBCA BYTE            ;DIRECT
E386 B21122    SBCA WORD            ;EXTENDED
E389 A200      SBCA 0,X             ;INDEXED BY X
E38B 18A200    SBCA 0,Y             ;INDEXED BY Y
E38E A203      SBCA BYTE,X          ;INDEXED BY X
E390 18A203    SBCA BYTE,Y          ;INDEXED BY Y

                                    ;SUBTRACT WITH CARRY FROM B
E393 C203      SBCB #BYTE           ;IMMEDIATE
E395 D203      SBCB BYTE            ;DIRECT
E397 F21122    SBCB WORD            ;EXTENDED
E39A E200      SBCB 0,X             ;INDEXED BY X
E39C 18E200    SBCB 0,Y             ;INDEXED BY Y
E39F E203      SBCB BYTE,X          ;INDEXED BY X
E3A1 18E203    SBCB BYTE,Y          ;INDEXED BY Y

E3A4 0D        SEC                  ;SET CARRY FLAG
E3A5 0F        SEI                  ;SET INTERRUPT MASK
E3A6 0B        SEV                  ;SET OVERFLOW FLAG

                                    ;STORE ACCUM A
E3A7 9703      STAA BYTE            ;DIRECT
E3A9 B71122    STAA WORD            ;EXTENDED
E3AC A700      STAA 0,X             ;INDEXED BY X
E3AE 18A700    STAA 0,Y             ;INDEXED BY Y
E3B1 A703      STAA BYTE,X          ;INDEXED BY X
E3B3 18A703    STAA BYTE,Y          ;INDEXED BY Y

                                    ;STORE ACCUM B
E3B6 D703      STAB BYTE            ;DIRECT
E3B8 F71122    STAB WORD            ;EXTENDED
E3BB E700      STAB 0,X             ;INDEXED BY X
E3BD 18E700    STAB 0,Y             ;INDEXED BY Y
E3C0 E703      STAB BYTE,X          ;INDEXED BY X
E3C2 18E703    STAB BYTE,Y          ;INDEXED BY Y

                                    ;STORE ACCUM D
E3C5 DD03      STD  BYTE            ;DIRECT
E3C7 FD1122    STD  WORD            ;EXTENDED
E3CA ED00      STD  0,X             ;INDEXED BY X
E3CC 18ED00    STD  0,Y             ;INDEXED BY Y
E3CF ED03      STD  BYTE,X          ;INDEXED BY X
E3D1 18ED03    STD  BYTE,Y          ;INDEXED BY Y

E3D4 CF        STOP                 ;STOP INTERNAL CLOCKS

                                    ;STORE STACK POINTER
E3D5 9F03      STS  BYTE            ;DIRECT
E3D7 BF1122    STS  WORD            ;EXTENDED
E3DA AF00      STS  0,X             ;INDEXED BY X
E3DC 18AF00    STS  0,Y             ;INDEXED BY Y
E3DF AF03      STS  BYTE,X          ;INDEXED BY X
E3E1 18AF03    STS  BYTE,Y          ;INDEXED BY Y
```

```
The Engineers Collaborative, Inc. 68HC11 Cross Assembler V1.0

APPENDIX A - TASM11 REFERENCE MANUAL
TASM11 SOURCE FILE SYNTAX, OPCODES & ADDRESSING MODE EXAMPLES

                                    ;STORE X
E3E4 DF03      STX  BYTE            ;DIRECT
E3E6 FF1122    STX  WORD            ;EXTENDED
E3E9 EF00      STX  0,X             ;INDEXED BY X
E3EB CDEF00    STX  0,Y             ;INDEXED BY Y
E3EE EF03      STX  BYTE,X          ;INDEXED BY X
E3F0 CDEF03    STX  BYTE,Y          ;INDEXED BY Y

                                    ;STORE Y
E3F3 18DF03    STY  BYTE            ;DIRECT
E3F6 18FF1122  STY  WORD            ;EXTENDED
E3FA 1AEF00    STY  0,X             ;INDEXED BY X
E3FD 18EF00    STY  0,Y             ;INDEXED BY Y
E400 1AEF03    STY  BYTE,X          ;INDEXED BY X
E403 18EF03    STY  BYTE,Y          ;INDEXED BY Y

                                    ;SUBTRACT MEMORY FROM A
E406 8003      SUBA #BYTE           ;IMMEDIATE
E408 9003      SUBA BYTE            ;DIRECT
E40A B01122    SUBA WORD            ;EXTENDED
E40D A000      SUBA 0,X             ;INDEXED BY X
E40F 18A000    SUBA 0,Y             ;INDEXED BY Y
E412 A003      SUBA BYTE,X          ;INDEXED BY X
E414 18A003    SUBA BYTE,Y          ;INDEXED BY Y

                                    ;SUBTRACT MEMORY FROM B
E417 C003      SUBB #BYTE           ;IMMEDIATE
E419 D003      SUBB BYTE            ;DIRECT
E41B F01122    SUBB WORD            ;EXTENDED
E41E E000      SUBB 0,X             ;INDEXED BY X
E420 18E000    SUBB 0,Y             ;INDEXED BY Y
E423 E003      SUBB BYTE,X          ;INDEXED BY X
E425 18E003    SUBB BYTE,Y          ;INDEXED BY Y

                                    ;SUBTRACT MEMORY FROM D
E428 831122    SUBD #WORD           ;IMMEDIATE
E42B 9303      SUBD BYTE            ;DIRECT
E42D B31122    SUBD WORD            ;EXTENDED
E430 A300      SUBD 0,X             ;INDEXED BY X
E432 18A300    SUBD 0,Y             ;INDEXED BY Y
E435 A303      SUBD BYTE,X          ;INDEXED BY X
E437 18A303    SUBD BYTE,Y          ;INDEXED BY Y

E43A 3F        SWI                  ;SOFTWARE INTERRUPT

E43B 16        TAB                  ;TRANSFER A TO B
E43C 06        TAP                  ;TRANSFER A TO CC REGISTER
E43D 17        TBA                  ;TRANSFER B TO A
E43E 07        TPA                  ;TRANSFER CC REGISTER TO A
```

```
The Engineers Collaborative, Inc. 68HC11 Cross Assembler V1.0

APPENDIX A - TASM11 REFERENCE MANUAL
TASM11 SOURCE FILE SYNTAX, OPCODES & ADDRESSING MODE EXAMPLES

                                    ;TEST FOR ZERO OR MINUS
E43F 4D        TSTA                 ;A
E440 5D        TSTB                 ;B
E441 7D1122    TST  WORD            ;EXTENDED
E444 6D00      TST  0,X             ;INDEXED BY X
E446 186D00    TST  0,Y             ;INDEXED BY Y
E449 6D03      TST  BYTE,X          ;INDEXED BY X
E44B 186D03    TST  BYTE,Y          ;INDEXED BY Y

E44E 30        TSX                  ;TRANSFER STACK POINTER TO X
E44F 1830      TSY                  ;TRANSFER STACK POINTER TO Y
E451 35        TXS                  ;TRNASFER X TO STACK POINTER
E452 1835      TYS                  ;TRNASFER Y TO STACK POINTER

E454 3E        WAI                  ;WAIT FOR INTERRUPT

E455 8F        XGDX                 ;EXCHANGE D WITH X
E456 188F      XGDY                 ;EXCHANGE D WITH Y

FFC0           ORG  irq_vectors     ;VECTORS
FFC0 E079      DW   START           ;reserved
FFC2 E079      DW   START           ;reserved 1
FFC4 E079      DW   START           ;reserved 2
FFC6 E079      DW   START           ;reserved 3
FFC8 E079      DW   START           ;reserved 4
FFCA E079      DW   START           ;reserved 5
FFCC E079      DW   START           ;reserved 6
FFCE E079      DW   START           ;reserved 7
FFD0 E079      DW   START           ;reserved 8
FFD2 E079      DW   START           ;reserved 9
FFD4 E079      DW   START           ;reserved 10
FFD6 E079      DW   START           ;sci serial system
FFD8 E079      DW   START           ;spi serial transfer complete
FFDA E079      DW   START           ;pulse accumulator input edge
FFDC E079      DW   START           ;pulse accumulator overflow
FFDE E079      DW   START           ;timer overflow
FFE0 E079      DW   START           ;timer output compare 5
FFE2 E079      DW   START           ;timer output compare 4
FFE4 E079      DW   START           ;timer output compare 3
FFE6 E079      DW   START           ;timer output compare 2
FFE8 E079      DW   START           ;timer output compare 1
FFEA E079      DW   START           ;timer input capture 3
FFEC E079      DW   START           ;timer input capture 2
FFEE E079      DW   START           ;timer input capture 1
FFF0 E079      DW   START           ;real time interrupt
FFF2 E079      DW   START           ;!IRQ (external pin or parallel I/O)
FFF4 E079      DW   START           ;!xirq pin
FFF6 E079      DW   START           ;SWI
FFF8 E079      DW   START           ;illegal opcode trap
FFFA E079      DW   START           ;COP failure (reset)
FFFC E079      DW   START           ;COP clock monitor fail
FFFE E079      DW   START           ;!RESET
E079           END  START           ;

Assembly complete - Errors = 0 , Warnings = 0
```

```
  ---Symbol Table---
ADCTL           1030
ADR1            1031
ADR2            1032
ADR3            1033
ADR4            1034
BAUD            102B
BYTE            0003
CFORC           100B
CONFIG          103F
COPRST          103A
Complex         0006
DDRC            1007
DDRD            1009
FIVE            0005
HPRIO           103C
INIT            103D
MASK            00C3
MESSAGE         E053
OC1D            100D
OC1M            100C
ONE             0001
ONE_1           0001
OPTION          1039
One             0001
One_1           0001
PACNT           1027
PACTL           1026
PIOC            1002
PORTA           1000
PORTB           1004
PORTC           1003
PORTCL          1005
PORTD           1008
PORTE           100A
PPROG           103B
REGISTERS       1000
RESERVED        1001
RESERVED2       1006
RESERVED3       1035
RESERVED4       1036
RESERVED5       1037
RESERVED6       1038
SCCR1           102C
SCCR2           102D
SCDR            102F
SCSR            102E
SEVEN           0007
SEVEN_1         0007
SPCR            1028
SPDR            102A
SPSR            1029
START           E079
Six             0006
TCNT            100E
TCNT1           100F
TCTL1           1020
TCTL2           1021
TEN             000A
TEST1           103E
TFLG1           1023
TFLG2           1025
TIC1            1010
TIC11           1011
TIC2            1012
TIC22           1013
TIC3            1014
TIC33           1015
TMSK1           1022
TMSK2           1024
TOC1            1016
TOC11           1017
TOC2            1018
TOC22           1019
TOC3            101A
```

```
TOC33           101B
TOC4            101C
TOC44           101D
TOC5            101E
TOC55           101F
TWO             0002
WORD            1122
_128            0080
eeprom_memory   B600
irq_vectors     FFC0
memory_size     FFFF
ram_memory      0000
rom_memory      E000
```