Hitachi H8/300 Software

H8/300 Hitachi Switches

Application Note

Jennifer Ediyanto

HITACHI®

AE-0059
August 1994

Introduction

Hitachi America provides the Hitachi H8/300 Software Tools to program the Hitachi H8/300 series, H8/300L series, and H8/300H series microprocessors. These software tools consists of C compiler, assembler, linker, and debugger.

This paper will provide detailed descriptions on the command line switches of the Hitachi H8/300 software tools.

This paper uses the following software tools:

• CH38	Cross Compiler
• ASM38	Cross Assembler H8/300
• LNK	Linker H8/300

C Cross Compiler

On invocation, the Hitachi C compiler will preprocess and compile the C program. The default command to invoke the compiler is:

ch38 ***source_filename***

Example: **ch38 test.c**

Result:

- Source listing file with .lst extension
- Object file with .obj extension

Rules of the command line:

- Only one C source file with any file extension can be compiled at one time.
- The switch is not case sensitive, can be abbreviated, and has to be preceded by a slash (/).
- If a switch is set to more than one option, then the list of options has to be separated by comma and surrounded by a parenthesis. For example:
  /show=(source,object,allocation)

The following command invokes the compiler with switches:

ch38 [/switch] ***source_filename***

where:

ch38	The name of the compiler. If the compiler is invoked without any command line switches, it displays all the available switches.
/switch	Any of the command line switches. Each switch must be preceded by a slash (/) and is not case sensitive.
source_filename	The name of the input file which has to be a C source file with any file extension. Only one file can be compiled at one time.

The following are the compiler switches: (The options in bold characters are the default options and the underline specifies the abbreviation that can be used for that option).

H8/300 Hitachi Switches

1. optimize
Syntax: /optimize=<level>
<level>: 0 | **1**

Turning on the /optimize switch causes the compiler to optimize the source code by reducing the code size and execution time. The following is the value of the optimize switch:

/op=0	Turns the optimization OFF.
/op=1	Turns the optimization ON (default).

Example: **ch38 /op=1 test.c**

The above command line turns on the optimization. Please see *Listing 1, Listing 2, Listing 3* for test.c, test1.src, and test2.src files. Test.c is a simple c program to increment a variable 'Hitachi' for five times. Test1.src is a source program which results from turning off the optimization and Test2.src is a source program which results from turning on the optimization. In order to create a source program, the /op switch need to be accompanied by /c=a switch.

2. code
Syntax: /code=(<suboption>)
<suboption>:
machinecode | asmcode

Setting the /c switch to asmcode, the compiler will produce assembly source file that takes the C filename with .src file extension. By default, the compiler generates relocatable object file.

/c=a	Generates assembly source file.
/c=m	Generates relocatable object file (default).

Example: **ch38 /c=a test.c**

The above command line will produce assembly source file. By default, the compiler produces assembly source code using H8/300 instruction syntax. Please see *Listing 2* for Test1.src as a sample of assembly source file.

3. list
Syntax: **/list** [= <listing file name>]
/nolist

Specifying the /list switch will cause the compiler to produce a program listing file that contains the following:

- source listing
- section and program size
- symbol information
- cpu mode information

Example: **ch38 /l test.c**

The above command line will produce a listing file named Test.lst. By default, the listing filename is C filename with .lst extension. Please see *Listing 4* for Test.lst file.

4. show
Syntax: /show=(<suboption>, ...)
<suboption>:
source | nosource
object | **noobject**
statistics | nostatistics
allocation | **noallocation**
expansion | **noexpansion**
width = **132** | <numeric value>
length = **60** | <numeric value>

The /show switch controls the information in the source listing which is a file with .lst file extension.

The following are options to control the listing information:

• source	source list
• object	object list
• statistics	statistics information
• allocation	symbol allocation information
• expansion	include macro expansion

We can specify the negative forms by typing **no** before the name of the above options.

Example: **ch38 /l=tst.lst /sh=(noso,ob) test.c**

The above command line will generate a listing file called tst.lst with object listing and no source code. Please see *Listing 5* for Tst.lst file and compare it with *Listing 4* (Test.lst).

The following are options to control the listing format:

- width — maximum characters per line, 0 or 80-132. Default=132.
- length — maximum lines per page, 0, 20-255. Default=60.

5. cpu

Syntax: /cpu=<mode>
<mode>:
300stk | **300reg** | 300hn | 300ha

The /cpu switch is important to generate code for a specific H8/300 family microprocessor. The 300reg or 300stk options should be used for H8/300 and H8/300L family microprocessors. The 300hn or 300ha options are intended for H8/300H family microprocessors.

/cp=300stk	H8/300 stack parameter. Parameters are passed through the stack to the called function.
/cp=300reg	H8/300 register parameter. Parameters are passed through the register to the called function.
/cp=300hn	H8/300H in normal mode. The H8/300H microprocessor is set to the normal mode.
/cp=300ha	H8/300H in advanced mode. The H8/300H microprocessor is set to the advanced mode.

Example: **ch38 /cp=300ha test.c**

We can determine which mode the cpu is set to by looking at the listing file. To generate the listing file, we can use the /l switch.

6. debug

Syntax: /debug
/nodebug

The /debug switch will cause the compiler to produce object file with debugging information. This debugging information is important to be able to perform source level debugging during the debugging time.

/deb	Turns debug information ON.
/nodeb	Turns debug information OFF (default).

Example: **ch38 /deb test.c**

7. section

Syntax: /section=(<suboption>,...)
<suboption>:
program=<section-name> |
const=<section-name> |
data=<section-name> |
bss=<section-name>
Default: p=**P**, c=**C**, d=**D**, b=**B**

The /section switch will allow users to rename the default section name.

The following are the options for section names:

p	program section name is specified
c	constant section name is specified
d	data section name is specified
b	non-initialized data section name is specified.

Example: **ch38 /se=(p=myp,c=myc) test.c**

The above command line will rename the default program and const sections to myp and myc.

8. string

Syntax: /string=(<suboption>)
<suboption>:
const |
data

The /string switch will control the output area for strings in a program. If the string is not modified in the C program, then users can specify the compiler to output the string data in the constant area. When the string is modified in the C program, specify the data is to be output to the initialized data area.

The following are the explanation of the options:

/st=const	output to constant area. (default)
/st=data	output to initialized data area.

Example: **ch38 /st=data test.c**

9. include
Syntax: /include=(<pathname>,...)

The /include switch will specify the path to the include file that is included using <> sign in user's program. If the include file is included using the quotes "", then the compiler will search the current directory for the include file.

Example: **ch38 /i=(c:\h83\ch38\include) test.c**

10. define
Syntax: /define=(<suboption>,...)

```
<suboption>:
        <macro-name>=<name>     |
        <macro-name>=<constant> |
        <macro-name>
```

Example: **ch38 /def=(mymacro=yours) test.c**

Table 1: The Define Switch Options

Item	Explanation
Macro name	A character string beginning with an alphabetic letter or an underscore followed by zero or more alphabetic letters, underscores, and numbers (0 to 9)
Name	A character string beginning with a letter or an underscore followed by zero or more alphabetic letters, underscores, and numbers.
Constant	A character string of one or more numbers, or a character string of one or more numbers followed by a period (.) and zero or more numbers.

Cross Assembler

On invocation, the Hitachi Assembler will assemble the assembly program. The default command to invoke the assembler is:

asm38 *source_filename*

Example: **asm38 test.src**

The above command will produce an object file with .obj extension only. By default, the list file is not produced.

The assembly method can be specified using command line switches when the assembler is invoked. The following command line invokes the assembler with switches:

asm38 <input file> [,<input file>...] [/<switch>...]

Rules of the command line:

- When two or more input files are specified, they are joined together in the order of input and then assembled as one source file.
- The switch is not case sensitive, can be abbreviated, and has to be preceded by a slash (/).

The following are the assembler switches: (The underlined section is the abbreviated form of the switches.)

1. cpu
Syntax: /cpu=<cpu type>
<cpu type>: {300HA | 300HN | **300** | 300L}

The CPU switch specifies the object CPU for the source program to be assembled.

The following are options for the cpu switch:

/cpu=300HA	H8/300H advanced mode.
/cpu=300HN	H8/300H normal mode.
/cpu=300	H8/300
/cpu=300L	H8/300L

Example: **asm38 test.src /cp=300**

The above command line will set the cpu to H8/300 mode because the test.src program which is produced by the compiler is in the H8/300 instruction syntax. By default, the assembler's cpu is set to H8/300H advance mode.

2. [no]object
Syntax: /**object**[=<file name>]
/noobject

This switch specifies either the output of an object module or the suppression of that output. By default, the assembler will produce the object module.

Example: **asm38 test.src /o=mytest.obj**

The above command line will cause the assembler to produce the object module that is called mytest with obj file extension. When the object file name is omitted, the object module is written to a file with the same name as the source module, but with the extension obj.

Example: **asm38 test.src /o**

The object file produced by the above command is called 'test.obj'.

3. [no]debug
Syntax: /debug
/nodebug

The debug switch specifies the output of debugging information. The debugging information is important because it will enable users to do symbolic debugging. By default the debug switch is off and debugging information is not generated.

Example: **asm38 test.src /debug**

4. br_relative
Syntax: /br_relative=<bit count>
<bit count>: {8 | 16}

The br switch specifies the default displacement size used when the branch displacement is a forward reference value. The br switch is only valid when the cpu is set to either H8/300H advanced mode or the H8/300H normal mode.

Example: **asm38 test.src /br=8 /cpu=300HA**

The above command line will cause the assembler to have 8 bits displacement size. The defaults are 16 bits for the H8/300H advanced mode, and 8 bits for the H8/300H normal mode.

5. [no]list
Syntax: /list[=<file name>]
/nolist

The list switch causes the assembler to produce the assembly listing file. By default, the assembler does not produce assembly listing file, but only displays lines that generated errors on the screen.

Example: **asm38 test.src /list=test.lis**

The above command line will produce an assembly listing file named test.lis. Please see *Listing 6* for test.lis file.

6. [no]source
Syntax: /**source**
/nosource

The source switch specifies the output of a source program listing to the assembly listing. The source and nosource switches are valid only when the list switch is on.

When the list switch is turned on, the assembler will place the source in the assembly listing by default. If we do not want the source program to be listed in the assembly listing file, we need to use nosource switch together with the list switch.

Example: **asm38 test.src /nos /list**

7. [no]cross_reference
Syntax: /**cross_reference**
/nocross_reference

The cross_reference switch specifies the output of a cross_reference information to the assembly listing. The cross_reference and nocross_reference switches are valid only when the list switch is on.

When the list switch is turned on, the assembler will place the cross_reference in the assembly listing by default. If we do not want the cross_reference to be listed in the assembly listing file, we need to use nocross_reference switch together with the list switch.

Example: **asm38 test.src /nocr /list**

8. [no]section
Syntax: /**section**
/nosection

The section switch specifies the output of section information to the assembly listing. The section and nosection switches are valid only when the list switch is on.

When the list switch is turned on, the assembler will place the section information in the assembly listing by default. If when we do not want the section information to be listed in the assembly listing file, we need to use nosection switch together with the list switch.

Example: **asm38 test.src /nose /list**

9. [no]show
Syntax: /show[=<option>[,<option> ...]]
/noshow[=<option>[,<option> ...]]
<option>: {conditionals | definitions | calls |
expansions | structured | code}

The show switch specifies the output of preprocessor function source statements to the source program listing. The show switch is only valid when the source switch is on.

The following are options for show:

Conditionals:	failed conditional expansions.
Definitions:	macro definitions.
Calls:	macro calls.
Expansions:	code from macro expansions.
Structured:	structured assembly function expansions.
Code:	object code display lines.

Example:
asm38 test.src /list=test2.lis /nosh=code

The above command line will assemble test.src and produce a listing file named test2.lis that does not contain the object code display lines.

10. lines
Syntax: /lines=<line count>

The lines switch specifies the number of lines per page in the assembly listing. A line count of between 20 and 255 lines can be specified. The lines switch is valid when an assembly listing is output.

Example: **asm38 test.src /list /lines=40**

Linker

The following command invokes the linker:

lnk [/sub=<filename>]

where

lnk	linker name.
/sub	linker switch for loading the linker command file.
<filename>	linker command file.

Example: **lnk /sub=test.cmd**

The linker command file contains all linker commands. The linker commands can also be entered interactively by invoking the linker without using the /sub switch. Please see *Listing 7* for a sample of linker command file called test.cmd.

The following are the linker commands:

1. debug

The debug command is necessary to include symbolic information in the absolute file.

2. form <file type>

The form command is to produce the specified output file type. The output file types are 'a' for absolute file and 'r' for relocatable object file. If no form command is specified, the linker will produce the absolute file.

Example: **form a**

The above command will produce an absolute file.

3. input <filename>

The input command is to load the object file produced by the compiler or assembler.

Example: **input test.obj**

4. start <section>(<start address>)

The start command sets the starting address of a specified section.

Example: **start P(1000)**

The above command sets the starting address of the code section to H'1000.

5. entry <symbol name>

The entry command sets the address value of a symbol as the execution start address of the program.

Example: **entry _main**

The above command sets main function as the start address of the program. The symbol is preceding by the underscore because of the C calling convention.

6. output <file name>

The output command produces the absolute file with the specified file name and .abs extension.

Example: **output test**

The above command will produce an absolute file called test.abs.

7. print <file name>

The print command produces the linker map file that contains the information about sections, symbols, and addresses.

Example: **print test**

Please see *Listing 8* for the generated map file called test.map.

8. exit

The exit command will exit from the linker.

Listing 1. Test.c (begin)

```
/* A simple example of a C program */
main()
{
   int i, Hitachi = 0;

   for (i = 0; i < 5; i++)
     Hitachi = Hitachi + i;
}
```

Listing 1. Test.c (end)

Listing 2. Test1.src (begin)

```
; This file is a sample of source program with the optimization OFF
; which result from the following command line:
;  ch38 /op=0 /c=a test.c
;
          .EXPORT     _main
          .SECTION    P,CODE,ALIGN=2
;*** File TEST.C    , Line 1     ; block
_main:                           ; function: main
          PUSH.W      R6
          MOV.W       SP,R6
          PUSH.W      R5
          PUSH.W      R4
          SUBS.W      #2,SP
          SUBS.W      #2,SP
;*** File TEST.C    , Line 2     ; block
;*** File TEST.C    , Line 3     ; expression statement
          SUB.W       R5,R5
          MOV.W       R5,@(-8:16,R6)
;*** File TEST.C    , Line 5     ; for
          SUB.W       R5,R5
          MOV.W       R5,@(-6:16,R6)
          BRA         L6
L5:
;*** File TEST.C    , Line 6     ; expression statement
          MOV.W       @(-8:16,R6),R5
          MOV.W       @(-6:16,R6),R4
          ADDS.W      R4,R5
          MOV.W       R5,@(-8:16,R6)
          MOV.W       @(-6:16,R6),R5
          ADDS.W      #1,R5
          MOV.W       R5,@(-6:16,R6)
L6:
          MOV.W       @(-6:16,R6),R5
          CMP.B       #5:8,R5L
          SUBX.B      #0:8,R5H
          BLT         L5
;*** File TEST.C    , Line 7     ; block
          ADDS.W      #2,SP
          ADDS.W      #2,SP
          POP.W       R4
          POP.W       R5
          POP.W       R6
          RTS
          .END
```

Listing 2. Test1.src (end)

Listing 3. Test2.src (begin)

```
; This file is a sample of source program with the optimization ON
; which result from the following command line:
;    ch38 /op=1 /c=a test.c
;
          .EXPORT     _main
          .SECTION    P,CODE,ALIGN=2
;*** File TEST.C    , Line 1     ; block
_main:                           ; function: main
;*** File TEST.C    , Line 2     ; block
;*** File TEST.C    , Line 5     ; expression statement
          SUB.W       R0,R0
;*** File TEST.C    , Line 5     ; do
L5:
;*** File TEST.C    , Line 5     ; expression statement
          ADDS.W      #1,R0
          MOV.W       R0,R1
          CMP.B       #5:8,R1L
          SUBX.B      #0:8,R1H
          BLT         L5
;*** File TEST.C    , Line 7     ; block
          RTS
          .END
```

Listing 3. Test2.src (end)

Listing 4. Test.lst (begin)

```
; This file is a source listing which result from the following command:
;    ch38 /l test.c
;
H8/300 SERIES C COMPILER (Ver. 2.0B)
11-Aug-1994  10:36:50  PAGE   1

************ SOURCE LISTING ************

FILE NAME: TEST.C

  Seq File        Line Pi 0----+----1----+----2----+----3----+----4----+----
    1 TEST.C         1     main()
    2 TEST.C         2     {
    3 TEST.C         3        int i, Hitachi = 0;
    4 TEST.C         4
    5 TEST.C         5        for (i = 0; i < 5; i++)
    6 TEST.C         6          Hitachi = Hitachi + i;
    7 TEST.C         7     }

H8/300 SERIES C COMPILER (Ver. 2.0B)
11-Aug-1994  10:36:58  PAGE   1

******* SECTION SIZE INFORMATION *******

PROGRAM  SECTION(P): 0x00000E Byte(s)
CONSTANT SECTION(C): 0x000000 Byte(s)
DATA     SECTION(D): 0x000000 Byte(s)
BSS      SECTION(B): 0x000000 Byte(s)

 TOTAL PROGRAM SIZE: 0x00000E Byte(s)

** ASSEMBLER/LINKAGE EDITOR LIMITS INFORMATION **

NUMBER OF EXTERNAL REFERENCE  SYMBOLS:        0
NUMBER OF EXTERNAL DEFINITION SYMBOLS:        1
NUMBER OF INTERNAL/EXTERNAL SYMBOLS:          2

********* CPU MODE INFORMATION *********
cpu=300reg
```

Listing 4. Test.lst (end)

Listing 5. Tst.lst (begin)

```
; This file is a source listing result from combination of listing
; and show switch. The following is the command line:
;    ch38 /l=tst.lst /sh=(noso,ob) test.c
;
H8/300 SERIES C COMPILER (Ver. 2.0B)
30-Aug-1994  11:06:56  PAGE   1

************ OBJECT LISTING ************

FILE NAME: TEST.C

SCT OFFSET CODE       C LABEL     INSTRUCTION OPERAND    COMMENT

  P                     ;*** File TEST.C    , Line 1     ; block
     0000               _main:                           ; function: main
                        ;*** File TEST.C    , Line 2     ; block
                        ;*** File TEST.C    , Line 5     ; expr statement
     0000  1900                   SUB.W       R0,R0
                        ;*** File TEST.C    , Line 5     ; do
     0002               L5:
                        ;*** File TEST.C    , Line 5     ; expr statement
     0002  0B00                   ADDS.W      #1,R0
     0004  0D01                   MOV.W       R0,R1
     0006  A905                   CMP.B       #5:8,R1L
     0008  B100                   SUBX.B      #0:8,R1H
     000A  4DF6                   BLT         L5
                        ;*** File TEST.C    , Line 7     ; block
     000C  5470                   RTS


H8/300 SERIES C COMPILER (Ver. 2.0B)
30-Aug-1994  11:06:56  PAGE   1

******* SECTION SIZE INFORMATION *******

PROGRAM  SECTION(P): 0x00000E Byte(s)
CONSTANT SECTION(C): 0x000000 Byte(s)
DATA     SECTION(D): 0x000000 Byte(s)
BSS      SECTION(B): 0x000000 Byte(s)
```

```
 TOTAL PROGRAM SIZE: 0x00000E Byte(s)



** ASSEMBLER/LINKAGE EDITOR LIMITS INFORMATION **

NUMBER OF EXTERNAL REFERENCE  SYMBOLS:        0
NUMBER OF EXTERNAL DEFINITION SYMBOLS:        1
NUMBER OF INTERNAL/EXTERNAL SYMBOLS:          2



********* CPU MODE INFORMATION *********
cpu=300reg
```

Listing 5. Tst.lst (end)

Listing 6. Test.lis (begin)

```
; This file is an example of listing file produced by the assembler.
; The following is the command line:
;    asm38 test.src /list=test.lis
;
*** H8/300 ASSEMBLER Ver.3.2E ***   10/07/94 11:24:57
PAGE     1
PROGRAM NAME =
    1                 1              .EXPORT     _main
    2   0000          2              .SECTION    P,CODE,ALIGN=2
    3                 3    ;*** File TEST.C    , Line 1   ;block
    4   0000          4    _main:              ;function: main
    5                 5    ;*** File TEST.C    , Line 2     ; block
    6                 6    ;*** File TEST.C    , Line 5     ; expr statement
    7   0000 1900     7              SUB.W       R0,R0
    8                 8    ;*** File TEST.C    , Line 5     ; do
    9   0002          9    L5:
   10                 10    ;*** File TEST.C    , Line 5     ;expr statement
   11   0002 0B00     11              ADDS.W      #1,R0
   12   0004 0D01     12              MOV.W       R0,R1
   13   0006 A905     13              CMP.B       #5:8,R1L
   14   0008 B100     14              SUBX.B      #0:8,R1H
   15   000A 4DF6     15              BLT         L5
   16                 16    ;*** File TEST.C    , Line 7     ; block
   17   000C 5470     17              RTS
   18                 18              .END
  *****TOTAL ERRORS       0
  *****TOTAL WARNINGS     0
*** H8/300 ASSEMBLER Ver.3.2E ***   10/07/94 11:24:57
PAGE     2
*** CROSS REFERENCE LIST
NAME                             SECTION  ATTR VALUE             SEQUENCE
L5                               P             00000002      9*   15
P                                P        SCT  00000000      2*
_main                            P        EXPT 00000000      1     4*
*** H8/300 ASSEMBLER Ver.3.2E ***   10/07/94 11:24:57
PAGE     3
*** SECTION DATA LIST
SECTION                          ATTRIBUTE    SIZE             START
P                                REL-CODE    0000E
```

Listing 6. Test.lis (end)

Listing 7. Test.cmd (begin)

```
debug
form a
input test.obj
start P(1000)
entry _main
output test
print test
exit
```

Listing 7. Test.cmd (end)

Listing 8. Test.map (begin)

```
; This file is the linker map file produced by the linker.
; The following is the command line:
;     lnk /sub=test.cmd
;
                     H SERIES LINKAGE EDITOR Ver. 5.1
LINK COMMAND LINE
LNK /sub=test.cmd
LINK SUBCOMMANDS

debug
form a
input test.obj
start P(1000)
output test
print test
exit

H SERIES LINKAGE EDITOR Ver. 5.1                      PAGE :     1
***     LINKAGE EDITOR LINK MAP LIST      ***

SECTION   NAME    START - END          LENGTH      UNIT NAME    MODULE NAME
ATTRIBUTE  :   CODE  NOSHR
P           H'00001000  -  H'0000100D   H'0000000E  test         test

* TOTAL ADDRESS *     H'00001000    -    H'0000100D   H'0000000E

H SERIES LINKAGE EDITOR Ver. 5.1                      PAGE :     1
***  LINKAGE EDITOR EXTERNALLY DEFINED SYMBOLS LIST  ***

SYMBOL  NAME                        ADDR                TYPE
_main                               H'00001000          DAT
```

Listing 8. Test.map (end)