LAPIS SEMICONDUCTOR

SQ003099E051

HTU8
User's Manual

Program Development Support Software

Issue Date: Oct. 2011

NOTICE

No copying or reproduction of this document, in part or in whole, is permitted without the consent of LAPIS Semiconductor Co., Ltd.

The content specified herein is subject to change for improvement without notice.

The content specified herein is for the purpose of introducing LAPIS Semiconductor's products (hereinafter "Products"). If you wish to use any such Product, please be sure to refer to the specifications, which can be obtained from LAPIS Semiconductor upon request.

Examples of application circuits, circuit constants and any other information contained herein illustrate the standard usage and operations of the Products. The peripheral conditions must be taken into account when designing circuits for mass production.

Great care was taken in ensuring the accuracy of the information specified in this document. However, should you incur any damage arising from any inaccuracy or misprint of such information, LAPIS Semiconductor shall bear no responsibility for such damage.

The technical information specified herein is intended only to show the typical functions of and examples of application circuits for the Products. LAPIS Semiconductor does not grant you, explicitly or implicitly, any license to use or exercise intellectual property or other rights held by LAPIS Semiconductor and other parties. LAPIS Semiconductor shall bear no responsibility whatsoever for any dispute arising from the use of such technical information.

The Products specified in this document are intended to be used with general-use electronic equipment or devices (such as audio visual equipment, office-automation equipment, communication devices, electronic appliances and amusement devices).

The Products specified in this document are not designed to be radiation tolerant.

While LAPIS Semiconductor always makes efforts to enhance the quality and reliability of its Products, a Product may fail or malfunction for a variety of reasons.

Please be sure to implement in your equipment using the Products safety measures to guard against the possibility of physical injury, fire or any other damage caused in the event of the failure of any Product, such as derating, redundancy, fire control and fail-safe designs. LAPIS Semiconductor shall bear no responsibility whatsoever for your use of any Product outside of the prescribed scope or not in accordance with the instruction manual.

The Products are not designed or manufactured to be used with any equipment, device or system which requires an extremely high level of reliability the failure or malfunction of which may result in a direct threat to human life or create a risk of human injury (such as a medical instrument, transportation equipment, aerospace machinery, nuclear-reactor controller, fuel-controller or other safety device). LAPIS Semiconductor shall bear no responsibility in any way for use of any of the Products for the above special purposes. If a Product is intended to be used for any such special purpose, please contact a ROHM sales representative before purchasing.

If you intend to export or ship overseas any Product or technology specified herein that may be controlled under the Foreign Exchange and the Foreign Trade Law, you will be required to obtain a license or permit under the Law.

Windows is a registered trademark of Microsoft Corporation (USA) in USA and other countries, and other product names and company names are trademarks or registered trademarks.

Contents

1. Before You Begin

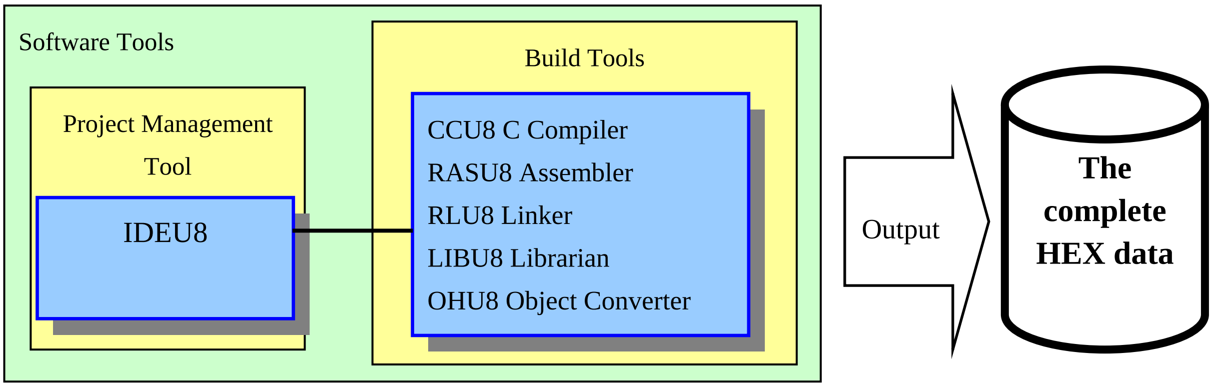
This manual describes the procedures for generating the data for writing the program data to the mask ROM or embedded flash ROM (here-in-after, called the ROM code data) in the target microcontroller which have nX-U8 core (here-in-after, called U8 microcontroller).

Please read this manual when you order us to write the ROM code data.

2. About the ROM code data

2.1 About the complete HEX data and the ROM code data

The complete HEX data is the data which generated by the tools of “Figure 2-1”.



* For details of above tools, please refer the user's manual of each the tool.

Figure 2-1 Tools that is used when you develop

The ROM code data is the data which generated by the tools of “Figure 2-2”, and we can write it to the flash ROM or the mask ROM in our factory.

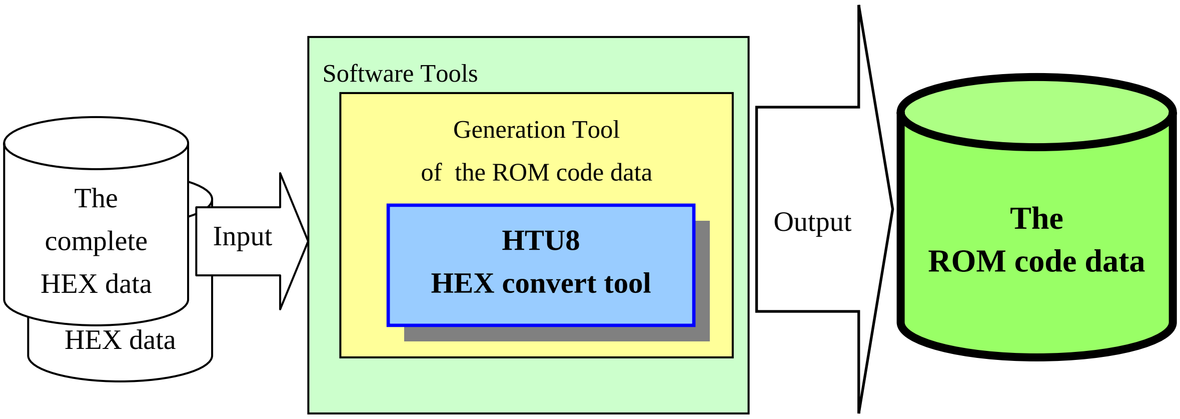


Figure 2-2 Generation Tool of the ROM code data

2.2 The ROM code data

Before writing the ROM code data to the mask ROM or the flash ROM, we recommend filling with “0FFFFH” (*) in the areas of which instructions do not exist (We call the area that is “the blank area”).

Because the contents of the blank area are undefined, if U8 microcontroller executes “the blank area”, the behavior of U8 microcontroller is unexpected.

You can generate the ROM code data which is filled with “0FFFFH” for “the blank area” by using HTU8. HTU8 assists to generate the ROM code data.

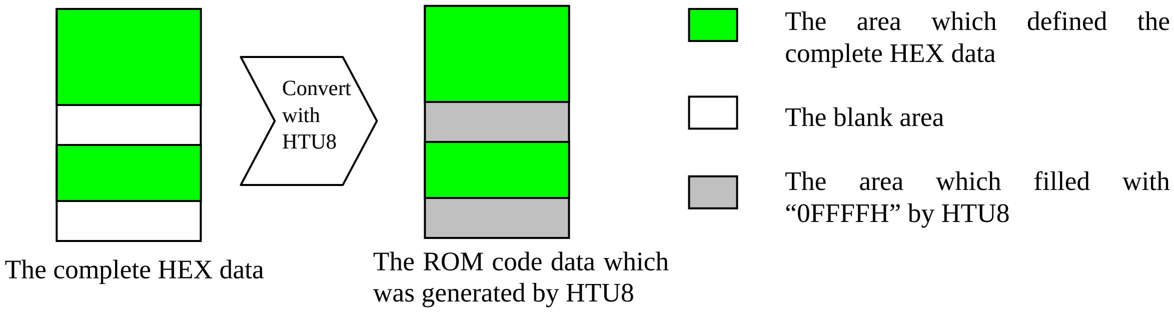

Convert with HTU8
The complete HEX data
The ROM code data which was generated by HTU8
🟩 The area which defined the complete HEX data
□ The blank area
▨ The area which filled with “0FFFFH” by HTU8

Figure 2-3 ROM code data image

*Note

“0FFFFH” corresponds with the break instruction of U8 microcontroller.

2.3 The recommendation of program development

We recommend as follows avoiding the trouble on the ROM code data.

(1) We recommend implementing the "Break Reset Routine".

For further details, refer "2.5.6 Break Reset Routine" in the "CCU8 Programming Guide".

(2) We recommend setting to generate the complete HEX data every time on the IDEU8 option settings.

For further details, refer "4.1 Convert ABS file to the complete HEX data".

(3) We recommend filling with 0FFH the blank area when debugging by DTU8.

When loading the program file by DTU8, filling the blank area with 0FFH, it can make the same state as the ROM code data which HTU8 generates.

For further details, refer "3.4.1 Reading in Program Code" in "DTU8 User's Manual".

(4) We recommend that you use the ROM code data as the final evaluation when you use the flash microcontroller.

Please write the ROM code data to the flash ROM and evaluate it on your systems finally.

For further details, refer "4.3 Evaluate the ROM code data".

3. Send files and information to us

You have to send files and information to us as following, and these are generated by HTU8.

(1)You have to send files to us.

No.	The kind of the file	The file name that is generated by HTU8	Description	Note
1	The ROM code data file	*TargetName_nnn*RA.hex or *TargetName_nnn*RA.s	The HEX file which filled 0FFFFH for the blank area	The contents of two (2) files (No.1 and No.2) are the same. We check that the contents of these files are the same for checking that the files have not broken when we received them.
2	(Same as above)	*TargetName_nnn*RB.hex or *TargetName_nnn*RB.s	The HEX file which filled 0FFFFH for the blank area	
3	The log file	*TargetName_nnn*RA.log	The text file witch is included the information about the generation	This file is included CHECKSUM

TargetName_nnn indicates the base name of output file name.

About the output file name

Please specify the output file name according to the following rule when you generate the above files by using HTU8.

“TargetName_nnn”

TargetName is the microcontroller name. For example, ML610Q340, ML610Q431 etc. please specify the name of microcontroller.

nnn is the three digit number which is informed by us (here-in-after, called the ROM code number).

Insert the underscore ‘_’ between *TargetName* and *nnn*.

For example, when the microcontroller name is ML610Q431 and ROM code number is 010, specify ML610Q431_010 as output file name.

(2)You have to send information

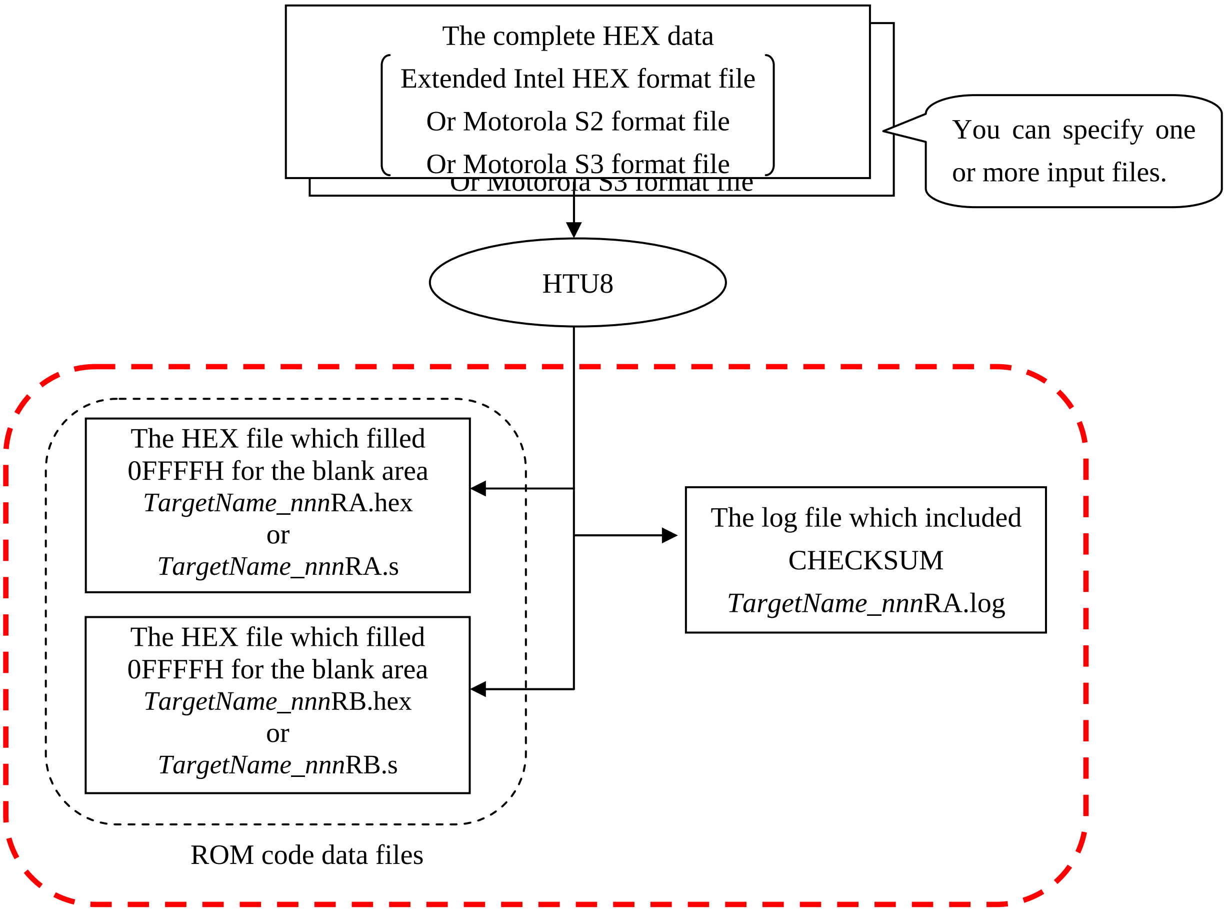
No.	The kind of the information	Description	Note
1	The CHECKSUM	The CHECKSUM is the value of the ROM code data.	There is the CHECKSUM in the above log file.

3.1 Overview of HTU8 data

The following figure shows HTU8 data flow.

Input files are the complete HEX data (the extended Intel HEX format, Motorola S2 format or Motorola S3 format). And the HTU8 generates two (2) ROM code data files (the extended Intel HEX format or Motorola S2 format) which filled ‘0FFFFH’ for “the blank area” and a log file.

The contents of two (2) ROM code data files are same. The log file is a text file which includes the CHECKSUM.


The complete HEX data
Extended Intel HEX format file
Or Motorola S2 format file
Or Motorola S3 format file
You can specify one or more input files.
HTU8
The HEX file which filled 0FFFFH for the blank area *TargetName_nnn*RA.hex or *TargetName_nnn*RA.s
The HEX file which filled 0FFFFH for the blank area *TargetName_nnn*RB.hex or *TargetName_nnn*RB.s
The log file which included CHECKSUM *TargetName_nnn*RA.log
ROM code data files

Please send the above two (2) ROM code data files and the log file to us.

Following Chapter, it describes how to generate the ROM code data and the log file by using HTU8.

4. The generating procedure of the ROM code data

You have to use the tools of "Figure 2-1" and HTU8 to generate the ROM code data.

The following chapter shows the procedure.

4.1 Convert ABS file to "the complete HEX data"

You have to convert an object file that is included the debugging information to the complete HEX data that are input files of HTU8. (Here-in-after, the object file is called the ABS file.)

"The complete HEX data" is usually converted by OHU8. It shows the way to convert the ABS file to "the complete HEX data" as following.

We recommend setting up the following settings at the time of the beginning of program development stage.

As shown in the figure 4-1, select [Project][Options][Target] from the IDEU8 menu.

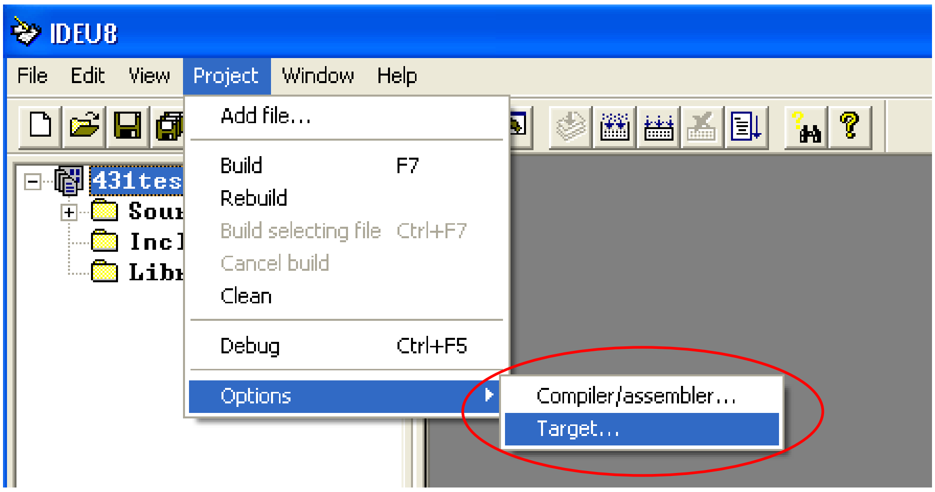
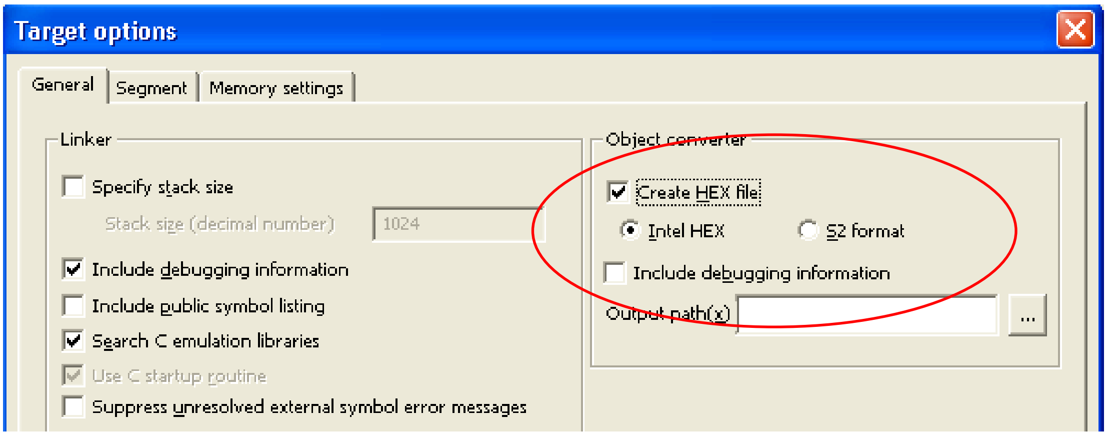


Figure 4-1 Selection for Target option dialogue

As shown in the figure 4-2, select [General] tab from [Target options] dialogue.

Check on [Create HEX file] in the [Object converter] group, select [Intel HEX] and check off [Include debugging information].



Linker

□ Specify stack size

Stack size (decimal number) 1024

☑ Include debugging information

□ Include public symbol listing

☑ Search C emulation libraries

☑ Use C startup routine

□ Suppress unresolved external symbol error messages

Object converter

☑ Create HEX file

● Intel HEX ○ S2 format

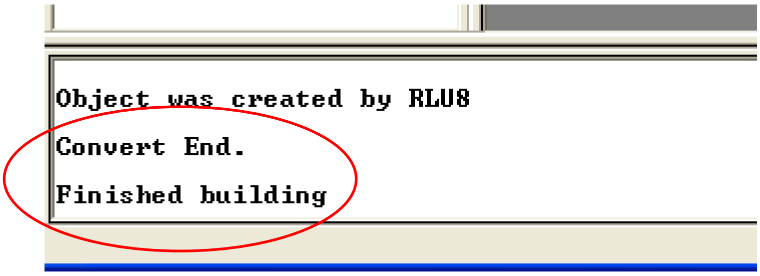
□ Include debugging information

Output path(x)

Figure 4-2 Target options dialogue

Build your program. After the building process is finished, confirm two messages "Convert End.", "Finished building" are displayed in the output window as shown in the figure 4-3.

"The complete HEX data" will be output in the specified folder.



```
Object was created by RLU8
Convert End.
Finished building
```

Figure 4-3 Conversion finishing message

4.2 Generate “the ROM code data” from “the complete HEX data” by HTU8

4.2.1 Open U8 command prompt

From the Start menu, select”All programs” >”U8 Tools”>”nX-U8”>”Command-line environment” icon to open the U8 command prompt.

4.2.2 Move to the folder which “the complete HEX data” is located in

On the U8 command prompt, by using CD command, move to the folder which “the complete HEX data” is located in.

For example, if “the complete HEX data” is located in “C:\Test\Sample\Hex”, input the command as follows.

Example:

```
 CD C:\Test\Sample\Hex
```

4.2.3 Generate “the ROM code data” from “the complete HEX data”

On the U8 command prompt, execute HTU8, please.

Example 1:

```
 HTU8 Test01.hex /TM610431 /FML610Q431_010 /OH
```

In the above example, specifies the input file is Test01.hex, to generate “the ROM code data”.

/T option is an option for specifying the file for initialization prepared according to the target device.

/F option is an option for specifying the output file name.

/OH option is an option for specifying the output format.

For further details, refer “5.3 Options”.

In the above example case, HTU8 generates two (2) “the ROM code data” files (ML610Q431_010RA.hex, ML610Q431_010RB.hex) and a log file (ML610Q431_010RA.log).

Example 2:

```
 HTU8 @Sample.res
```

The above example shows the case of using response file. The response file is a text file which specified the input files and the options. It is convenient when you specify multiple input files or specify complex operation by options.

The following example shows the sample of response file.

```
// Sample of the response file (Sample.res)
Test01.hex                          // Input file 1
Data01.s /AL(0, 7FFFH, 1:0000H)     // Input file 2
Data01.s /AL(8000H, 0FFFFH, 4000H)  // Input file 3
/TM610340                           // Target
/FML610Q340_020                     // Output file name
/OS                                 // Output file format
```

In the above example, it is specified that three input files are merged and it generates the ROM code data. All the data of the input file 1 (Test01.hex) is output to the ROM code data without changing address. By using /AL option, the data in the range of 0 - 7FFFH in the input file 2 (Data01.s) is allocated to 1:0000H, when output to the ROM code data. Also the data in the range of 8000H - FFFFH in the input file 3 (Data01.s) is allocated to 4000H, when output to the ROM code data.

For further details, refer “5.3 Options”.

In the above example case, HTU8 generates two (2) “the ROM code data” files (ML610Q340_020RA.s, ML610Q340_020RB.s) and a log file (ML610Q340_020RA.log).

The following figure shows the relationship of input files and ROM code data file.

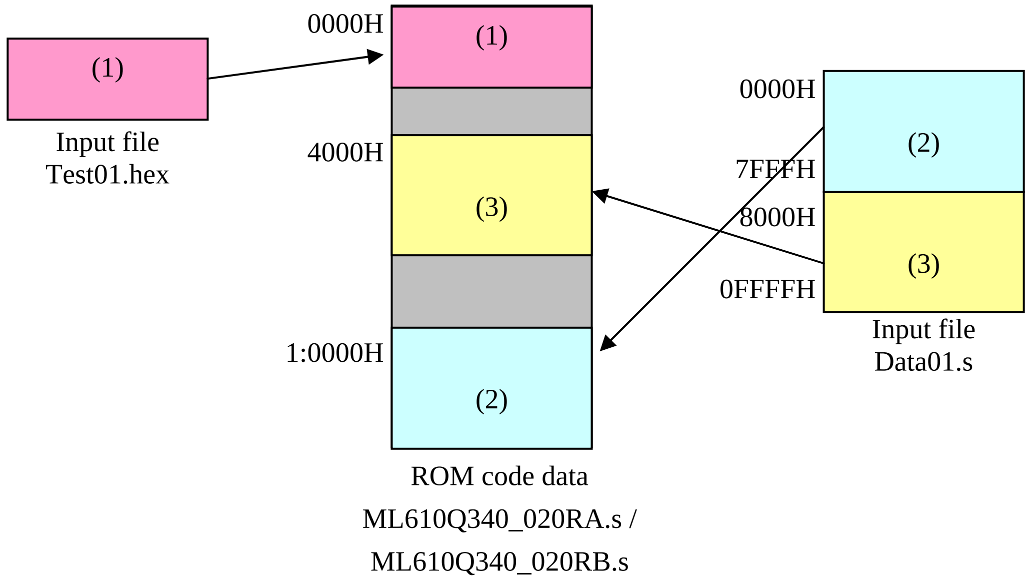


Figure 4-4 The relationship of input files and ROM code data file

If the conversion process is finished normally, HTU8 displays the following ending message.

```
*** The Result of HTU8 ***
Status            : OK
Total checksum    : XXXXXXXXXX
Target Name       : target
```

XXXXXXXXXX is a CHECKSUM for the generated “the ROM code data”.

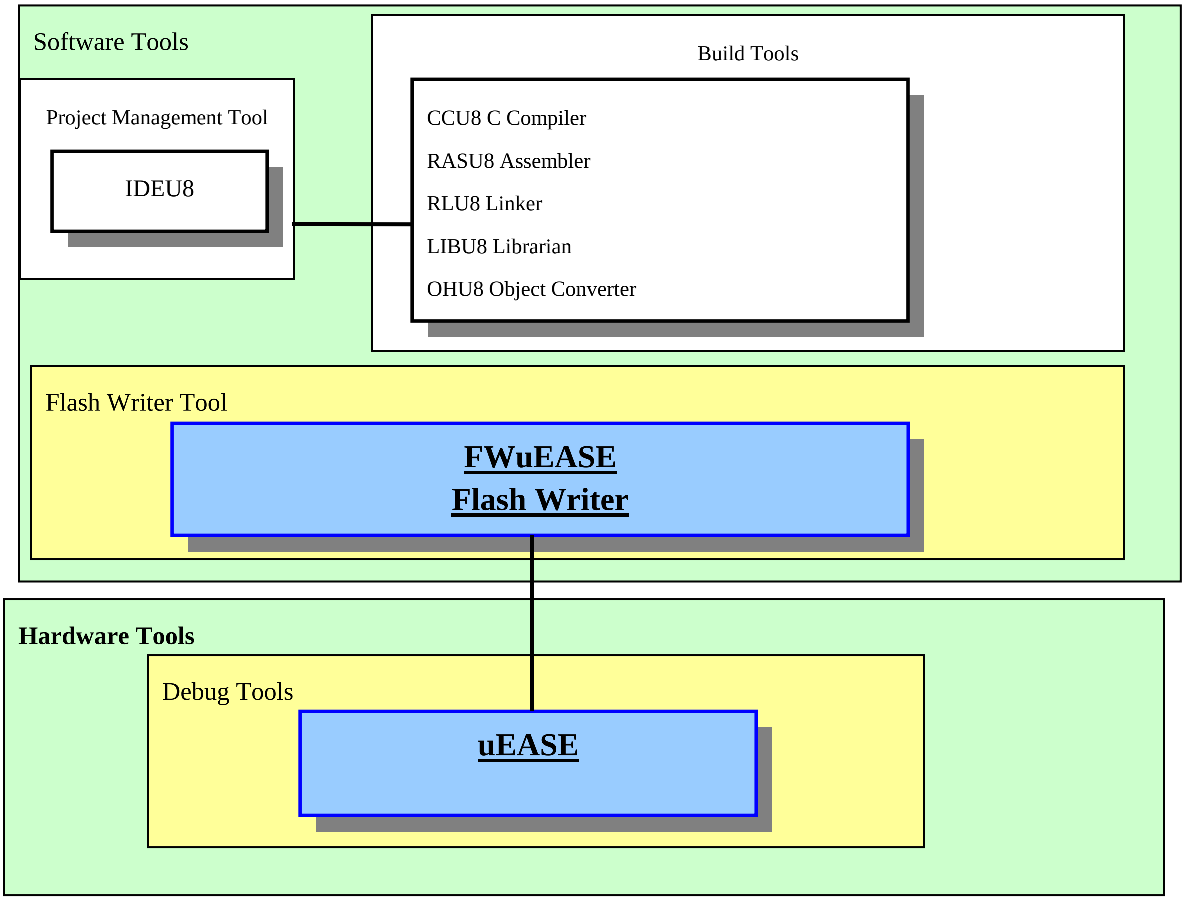
target is a target name specified by /T option.

Since the “Status: OK” means the conversion process is successful.

4.3 Evaluate “the ROM code data” (Only the case of Flash ROM microcontroller)

The case of Flash ROM microcontroller, please write “the ROM code data” to the flash ROM and finally evaluate it on your systems.

When you write “the ROM code data” to the flash ROM, you use uEASE of Hardware Tool and FWuEASE Flash Writer of Software Tool of the “Figure 4-5”.



* For details of above tools, please refer the user's manual of each the tool.

Figure 4-5 Tools that is used when you evaluate

4.4 Send “the ROM code data” to us

Please send the ROM code data files (*TargetName_nnn*RA.hex and *TargetName_nnn*RB.hex) and the log file (*TargetName_nnn*RA.log) to us. And please inform us the CHECKSUM, too.

5. Details of HTU8

5.1 Command line Syntax

HTU8 has the following command line syntax.

HTU8 *inputfiles options*

HTU8 @*response_file*

inputfiles means input files. Specify “The complete HEX data” to *inputfiles*. You can specify one or more input files. About the available input file, please see “5.2.1 Input file”.

options means options. You can specify options anywhere. About the details of the options, please see “5.3 Options”.

response_file following ‘@’ mark is treated as a response file. HTU8 reads the contents of the response file as command line input. About the details of the response file, please see “5.4 Response File”.

5.2 Input / Output Files

5.2.1 Input Files

The available input files of HTU8 are the extended Intel HEX format files, the Motorola S2 format files or the Motorola S3 format files.

You can specify one or more input files. The maximum number of input files which is available to specify is 16.

HTU8 does not accept the input file which includes the debugging information. About how to generate the file which does not include the debugging information see “4.1 Convert ABS file to the complete HEX data”.

If the input file which includes the debugging information is specified, HTU8 issues an error message.

5.2.2 Output Files

HTU8 generates two (2) ROM code data files and a log file which recorded the result of the conversion status. The contents of these two (2) ROM code data files are same. But the file names are a little bit different.

HTU8 gives output file name based on the output file name specified by /F option and the specified output file format options.

The number of characters of the base name of the output file has to be less than 30.

The naming rule of output file is as follows.

Output file	Output file	ROM code data file name	Log file name

base name	format option			
		Output file A	Output file B	
TargetName_nnn	/OH	*TargetName_nnn*RA.hex	*TargetName_nnn*RB.hex	*TargetName_nnn*RA.log
	/OS	*TargetName_nnn*RA.s	*TargetName_nnn*RB.s	*TargetName_nnn*RA.log

TargetName_nnn indicates the base name of output file name.

5.2.2.1 ROM code data Files

The ROM code data files are corresponding with the output file A and B of the above table. The contents of the output file A and B are same.

The valid address range excluding the content of input file will be filled with 0FFFFH.

5.2.2.2 Log File

The format of the log file is as follows.

(1) When the conversion process succeeded

Contents of log file	Description
HTU8, Ver.1.20	(1) The version of HTU8
Copyright 2009 - 2011 LAPIS Semiconductor Co., Ltd.	-
Tue Feb 12 23:11:31 2009	(2) The HTU8 execution date and time
*** The Result of HTU8 ***	-
Status : OK	(3) The result of conversion of HTU8
Total checksum : *0000FE5A8D*	(4) The check sum value The digit of the check sum is 10 digits.
Target Name : M610340	(5) The specified target name by /T option
*** COMMAND LINE ***	
test1.hex /AL(1000H, 3FFFH, 1:0000H) /FML610Q340_020 /M610340 /OS	(6) The contents of command line input
*** INPUT FILES ***	
File Name : *TEST1.HEX*	(7) Input file name
Format : *Extended Intel Hex*	(8) Input file format
Address Range : *00H:0000H 00H:5FFFH*	(9) Address range of contents of input
>Allocation Range : *00H:1000H 00H:3FFFH*	(10) Range of reading data of input file
Destination : *01H:0000H*	(11) Allocated address
*** OUTPUT FILES ***	
File Name : *ML610Q340_020RA.s* *ML610Q340_020RB.s*	(12) Output file name
Format : *Motorola S*	(13) Output file format
Address Range : *00H:0000 00H:0FFFFH* *01H:0000 01H:07FFFH*	(14) Address range of output
<End of Log>	(15) End of log

(2) When the conversion process failed

Contents of log file	Description
HTU8, Ver.1.20	(1) The version of HTU8
Copyright 2009 - 2011 LAPIS Semiconductor Co., Ltd.	-
Tue Feb 12 23:11:31 2009	(2) The HTU8 execution date and time
*** The Result of HTU8 ***	-
Status : NG	(3) The result of conversion of HTU8
Error No : *xxx*	(4) The error number

5.3 Options

The options of HTU8 are as follows.

/OH

This option specifies to the extended Intel HEX format for the output. The output range is from 0:0000H to 0FH:0FFFFH when this option is specified.

/OS

This option specifies to the Motorola S format for the output. The output range is from 0:0000H to 0FH:0FFFFH when this option is specified.

/OH and /OS can not specify concurrently. These options can omit to specify. Default is /OH.

/T*target*

This option specifies the target device. Please specify to *target* the string which removed ‘L’ and ‘Q’ from microcontroller name. In the case of ML610Q431, please specify it as /TM610431.

HTU8 searches *target*.tmi (here-in-after, called the TMI file), and get the target memory information from the TMI file. And HTU8 treats as output range the memory information which is gotten from the TMI file.

Usually, the TMI file is stored in the Tmi folder under the installation folder of U8 development tools.

/R(*start_addr*, *end_addr*)

This option specifies the range that it is specified as an output range.

/R can be specified to 32 sets. Usually, it does not need to specify /R option. When the TMI file for the microcontroller does not exist, it uses /R option.

/T option and the /R option have to specify either one. These options can not be omitted.

/P*path_name*

This option specifies the search path for TMI file.

Usually, it does not need to specify this option. When HTU8 can not find the TMI file, please specify this option.

/F*output_name*

This option specifies the base name of output file as *output_name*. *output_name* can include path. If *output_name* does not include path, the ROM code data files and the log file will be stored to the current directory. If *output_name* includes path, they will be stored to the specified directory.

If this option is not specified, then the base name of the first input file is treated as output file base name.

/AL(*start_addr*, *end_addr*, *allocation*)

This option specifies reading the data in the range of *start_addr*-*end_addr* in the specified input file just before this option, and allocating the data to the specified address by *allocation*.

It can specify decimal number or hexadecimal number to *start_addr*, *end_addr*, *allocation*. If the prefix ‘0x’ or the suffix ‘H’ is specified to the number, the number is treated as hexadecimal number. In the other case, the number is treated as decimal number.

In specifying a physical segment and offset by addressing, please divide with a colon ‘:’.

```
Data01.s /AL(0, 7FFFH, 1:0000H)
```

When it specifies as mentioned above, HTU8 reads the data of the range of 0-7FFFH of input file Data01.s, and it allocates the read data to 1:0000H.

The number of the /AL option which can be specified to one input file is one. When you specify reading the data of two or more ranges from one input file, divide into plurality and specify as follows.

```
Data01.s /AL(0, 7FFFH, 1:0000H) Data01.s /AL(8000H, 0FFFFH, 4000H)
```

5.4 Response File

The response file is a text file which specified the input files and the options. It is convenient when you specify multiple input files or specify complex operation by options.

It can describe a command line input over two or more lines in a response file. And it can describe the comments in a response file. The comments start with ‘#’ or ‘//’. HTU8 skips that and all characters from there to the line feed (0AH).

The response file which can be specified is only one. It is not possible to describe the response file in the response file.

The following example shows the sample of response file.

```
#########################################################
# Sample of the response file (Sample.res)
# It is considered as a comment from '#' or '//' to
# the end of the line
#########################################################
Test01.hex                          // Input file 1
Data01.s /AL(0, 7FFFH, 1:0000H)     // Input file 2
Data01.s /AL(8000H, 0FFFFH, 4000H)  // Input file 3
/TM610340                           // Target
/FML610Q340_020                     // Output file name
/OS                                 // Output file format
```

The following is an example which uses the response file.

```
HTU8 @Sample.res
```

It can describe a part of command line input.

```
// Sample of the response file (Sample2.res)
// These contents are a part of Sample.res
Data01.s /AL(0, 7FFFH, 1:0000H)     // Input file 2
Data01.s /AL(8000H, 0FFFFH, 4000H)  // Input file 3
/OS                                 // Output file format
```

```
HTU8 Test01.hex @Sample2.res /TM610340 /FML610Q340_020
```

The result of the above is the same as “HTU8 @Sample.res”.

5.5 HTU8 Output Messages

HTU8 output invoking message, the final summary, and any error messages arising during processing.

5.5.1 Invoking Message

HTU8 displays the following message on the screen.

```
HTU8 HEX Converter, Ver.1.20
Copyright  2009 - 2011 LAPIS Semiconductor Co., Ltd.
```

5.5.2 Final Summary

When the conversion process is successful, HTU8 displays the following messages.

```
*** The Result of HTU8 ***
Status           : OK
Total checksum   : XXXXXXXXXX
```

XXXXXXXXXX is a CHECKSUM for the generated “the ROM code data”.

When any error is occurred, HTU8 displays the following message.

```
Error nnn : error_message
```

nnn is an error code number, *error_message* indicates the nature of the error.

About the table of error messages, see “5.7 Error Message”.

5.6 Return Codes

HTU8 exits with one of the following return codes.

Return Code	Description
0	Success
3	HTU8 encountered a fatal error.

5.7 Error Messages

The error messages output by HTU8 are as follows.

Error code	Error Message	Description
000	Ok	Conversion process finished normally.
102	insufficient memory	There is not enough memory to continue.
153	File name too long (255 characters)	The length of the file name which includes path exceeds 255 characters.
154	Unable to open Input_file	HTU8 can not open the input file. Check for invalid character in the input file name.
158	Bad syntax on command line	The specified command line is wrong.
159	Invalid conversion range	The specified output range with /R(*start_addr*, *end_addr*) option is invalid.
160	/Ttarget option or /R option is not specified	It has to specify either one of /T*target* or /R(*start_addr*, *end_addr*) option.
162	Path name too long (255 characters)	The specified path name with /P option exceeds 255 characters.
163	Invalid Path name	The path name specified by /P option is illegal.
164	/R (start_addr, end_addr) too much	The number of /R(*start_addr*, *end_addr*) option which can be specified is to 32 sets.

Error code	Error Message	Description
166	address overlap	The area specified by /R(*start_addr*, *end_addr*) option overlaps.
168	Number of input files exceed maximum (16 files)	Number of input files exceeds 16.
170	Output file name not given.	Output file name is not given to /F option.
171	Output file name too long (255 characters)	The length of the output file name which includes the path and "RA" exceeds 255 characters.
172	Output file name base too long (32 characters)	The length of the output file name which includes "RA" exceeds 32 characters.
173	Invalid output file name	The output file name specified with /F option is invalid.
174	Response file name too long (255 characters)	The length of the response file name exceeds 255 characters.
175	Invalid response file name	The response file name is invalid.
176	Bad syntax on command line	Two or more response files are specified. The response file which can be specified is only one.
177	Unable to open response file	It can not open the response file.
178	No input files before a /AL option.	It has to specify an input file just before the /AL option.
179	Bad syntax on command line	The number of /AL option for an input file is only one.
180	Input files not given.	The input file is not specified at all.
181	XNV/SNV files are not supporting.	The file with the extension ".XNV" / ".SNV" is not supported.
182	Command line buffer overflow.	The command line buffer overflowed. The size of command line buffer is 32K bytes.
201	file not found : TMI file	*target*.tmi specified with /T*target* is not found.
207	Unable to close TMI file	Failed to close TMI file.
302	Unable to read Extended Intel HEX	Failed to read the extended Intel HEX input file.
305	Unable to read Motorola S	Failed to read the Motorola S input file.

Error code	Error Message	Description
309	Invalid Extended Intel HEX format.	The input file is not a valid extended Intel HEX file.
310	Invalid Motorola S format.	The input file is not a valid Motorola S file.
311	Invalid Extended Intel HEX format.	The file end record does not exist in the input file. The file might not be generated by OHU8.
312	Invalid Motorola S format.	S0 record or S8 record does not exist in the input file. The file might not be generated by OHU8.
313	Checksum failure	HTU8 has detected a checksum error in the current record. The input file might be corrupted.
351	Unable to open Extended Intel HEX	Failed to open the extended Intel HEX output file.
352	Unable to read Extended Intel HEX	Failed to read the extended Intel HEX output file.
353	Unable to open Motorola S	Failed to open the Motorola S output file.
354	Unable to read Motorola S	Failed to read the Motorola S output file.
355	Unable to write to Extended Intel HEX	Failed to write the extended Intel HEX output file.
356	Unable to write to Motorola S	Failed to write the Motorola S output file.
357	Unable to close Extended Intel HEX	Failed to close the extended Intel HEX output file.
358	Unable to close Motorola S	Failed to close the Motorola S output file.
360	Unable to write to *.LOG	Failed to write to the log file.
361	Unable to close *.LOG	Failed to close the log file.
369	Unable to open *.LOG	Failed to open the log file.
371	Out of Address Range	The address in input file exceeds the address range of output file. The target microcontroller might be different.
372	Duplicate Address Detection.	The data in which the address overlaps has been detected.
9*xx*	Internal_Error_9*xx*	HTU8 has detected an internal error. 9*xx* is an error code number. Please let me know the occurrence conditions of that error to us when this error occurs.

5.8 Restrictions

HTU8 has the following restriction:

- Don’t invoke HTU8 simultaneously from two or more command prompts. When HTU8 is invoked simultaneously, it does not run correctly.