# SH7268/SH7269 Group

## E10A-USB Flash Memory Download Function
(Download to the NOR Flash Memory)

### Abstract

E10A-USB emulator has the function to download a load module to the flash memory. This function requires a download program to access the flash memory (hereinafter called "FMTOOL"). This application note describes how to download a load module to the NOR flash memory (JEDEC standard compatible command method) to which the FMTOOL is applied.

### Target Device

SH7268/SH7269 Group (hereinafter called "SH7269")

When using this application note with other Renesas MCUs, careful evaluation is recommended after making modifications to comply with the alternate MCU.
Contents

1. Specifications ........................................................................................................................................... 3
2. Operation Confirmation Conditions ........................................................................................................ 4
3. Reference Application Note .................................................................................................................. 4
4. Hardware ................................................................................................................................................ 5
   4.1 Pins used ........................................................................................................................................... 5
   4.2 Reference Circuit ............................................................................................................................... 6
5. Software .................................................................................................................................................. 7
   5.1 Operation Overview .......................................................................................................................... 7
   5.1.1 Batch File ....................................................................................................................................... 7
   5.1.2 Erase Module .............................................................................................................................. 7
   5.1.3 Write Module .............................................................................................................................. 8
   5.2 File Composition ............................................................................................................................. 9
   5.3 Constants ......................................................................................................................................... 10
   5.4 Variable .......................................................................................................................................... 13
   5.5 Functions ........................................................................................................................................ 13
   5.6 Function Specifications .................................................................................................................... 14
   5.7 Flowcharts ....................................................................................................................................... 17
   5.7.1 Batch File ...................................................................................................................................... 17
   5.7.2 Erase Module ............................................................................................................................. 18
   5.7.3 Write module ............................................................................................................................. 19
6. Application Example ............................................................................................................................... 20
   6.1 Procedure of User Program Download .......................................................................................... 20
   6.1.1 Prepare for Download Environment ......................................................................................... 20
   6.1.2 Registering Batch File .............................................................................................................. 21
   6.1.3 Setting Configuration Dialog Box ............................................................................................ 22
   6.1.4 Adding Download Module ........................................................................................................ 23
   6.1.5 Downloading User Programs .................................................................................................... 23
6.2 Customizing FMTOOL ......................................................................................................................... 24
   6.2.1 Device Specification for Sample Code ....................................................................................... 24
   6.2.2 Applicable Flash Memory Boot Types ....................................................................................... 25
   6.2.3 Contents of Customization ......................................................................................................... 26
   6.2.4 Customizing Uniform Type ........................................................................................................ 28
   6.2.5 Customizing to Bottom Boot Type ............................................................................................ 29
   6.2.6 Customizing to Top Boot Type ................................................................................................... 30
   6.2.7 Customizing to Dual Boot Type ................................................................................................. 31
   6.2.8 Definitions of Command ........................................................................................................... 32
7. Sample Code .......................................................................................................................................... 33
8. Reference Documents .......................................................................................................................... 33
1. Specifications

Download the load module which is allocated in the external address space (CS0 space) to the flash memory using the FMTOOL. The FMTOOL allows the flash memory accessed when it is connected to the CS0 space (16-bit bus).

Table 1.1 lists the Peripheral Functions and Their Applications and Figure 1.1 shows the Procedure of Download Using FMTOOL.

<table>
<thead>
<tr>
<th>Peripheral Function</th>
<th>Application</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bus state controller (CS0)</td>
<td>Downloads to the flash memory</td>
</tr>
<tr>
<td>On-chip high-speed RAM</td>
<td>Work memory</td>
</tr>
<tr>
<td>H-UDI</td>
<td>Connects the E10A-USB emulator</td>
</tr>
</tbody>
</table>

Figure 1.1 Procedure of Download Using FMTOOL
2. Operation Confirmation Conditions

The sample code accompanying this application note has been run and confirmed under the conditions below.

Table 2.1 Operation Confirmation Conditions

<table>
<thead>
<tr>
<th>Item</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>MCU used</td>
<td>SH7269</td>
</tr>
<tr>
<td>Device used</td>
<td>Flash memory (connected by 16-bit bus)</td>
</tr>
<tr>
<td></td>
<td>Manufacturer: Spansion</td>
</tr>
<tr>
<td></td>
<td>Model: S29GL256P90TFI01</td>
</tr>
<tr>
<td>Operating frequency</td>
<td>CPU internal clock (Iφ): 266.67MHz</td>
</tr>
<tr>
<td></td>
<td>Internal clock (Bφ): 133.33MHz</td>
</tr>
<tr>
<td></td>
<td>Peripheral clock 1 (P1φ): 66.67MHz</td>
</tr>
<tr>
<td></td>
<td>Peripheral clock 0 (P0φ): 33.33MHz</td>
</tr>
<tr>
<td>Operating voltage</td>
<td>Source power (I/O): 3.3V</td>
</tr>
<tr>
<td></td>
<td>Source power (internal): 1.25V</td>
</tr>
<tr>
<td>Integrated development</td>
<td>Renesas Electronics</td>
</tr>
<tr>
<td>environment</td>
<td>High-performance Embedded Workshop Ver.4.07.00</td>
</tr>
<tr>
<td>C compiler</td>
<td>Renesas Electronics</td>
</tr>
<tr>
<td></td>
<td>SuperH RISC engine Family C/C++ Compiler Package Ver.9.03 Release02</td>
</tr>
<tr>
<td>Compiler option</td>
<td>-cpu=sh2afpu -fpu=single -object=&quot;${CONFIGDIR}${FILELEAF}.obj&quot;</td>
</tr>
<tr>
<td></td>
<td>-debug -gbr=auto -chngincpath -errorpath -global_volatile=0</td>
</tr>
<tr>
<td></td>
<td>-opt_range=all -infinite_loop=0 -del_vacant_loop=0 -struct_alloc=1</td>
</tr>
<tr>
<td></td>
<td>-nologo</td>
</tr>
<tr>
<td></td>
<td>(with default setting in the integrated development environment)</td>
</tr>
<tr>
<td>Board used</td>
<td>R0K572690C000BR</td>
</tr>
</tbody>
</table>

3. Reference Application Note

For additional information associated with this document, refer to the following application note.

- Flash Memory Download Program for the E10A-USB Emulator Application Note (document No. REJ10J1221)
4. Hardware

4.1 Pins used

Table 4.1 lists the Used Pins and Their Functions.

<table>
<thead>
<tr>
<th>Pin Name</th>
<th>Input/Output</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>A24 to A1</td>
<td>Output</td>
<td>Address bus</td>
</tr>
<tr>
<td>D15 to D0</td>
<td>Input/output</td>
<td>Data bus</td>
</tr>
<tr>
<td>CS0#</td>
<td>Output</td>
<td>Output device selection signal to the flash memory</td>
</tr>
<tr>
<td>RD#</td>
<td>Output</td>
<td>Output read control signal to the flash memory</td>
</tr>
<tr>
<td>WE0#</td>
<td>Output</td>
<td>Output write enable control signal to the flash memory</td>
</tr>
<tr>
<td>MD_BOOT1, MD_BOOT0</td>
<td>Input</td>
<td>Selection of boot mode</td>
</tr>
<tr>
<td>AUDCK</td>
<td>Output</td>
<td>Clock output to the E10A-USB emulator (38-pin)</td>
</tr>
<tr>
<td>AUDATA3 to AUDATA0</td>
<td>Output</td>
<td>Address output to the E10A-USB emulator (38-pin)</td>
</tr>
<tr>
<td>AUDSYNC#</td>
<td>Output</td>
<td>Synchronous signal output to the E10A-USB emulator (38-pin)</td>
</tr>
<tr>
<td>TCK</td>
<td>Input</td>
<td>Clock input from the E10A-USB emulator</td>
</tr>
<tr>
<td>TMS</td>
<td>Input</td>
<td>Mode selection from the E10A-USB emulator</td>
</tr>
<tr>
<td>TRST#</td>
<td>Input</td>
<td>Reset input from the E10A-USB emulator</td>
</tr>
<tr>
<td>TDI</td>
<td>Input</td>
<td>Data input from the E10A-USB emulator</td>
</tr>
<tr>
<td>TDO</td>
<td>Output</td>
<td>Data output to the E10A-USB emulator</td>
</tr>
<tr>
<td>ASEBRKAK#/ASEBRK#</td>
<td>Input/output</td>
<td>Break request and response</td>
</tr>
<tr>
<td>RES#</td>
<td>Input</td>
<td>System reset signal</td>
</tr>
<tr>
<td>ASEMD#</td>
<td>Input</td>
<td>Selection of ASE mode</td>
</tr>
</tbody>
</table>

Note: The symbol "#" indicates a negative-true logic or an active low.
4.2 Reference Circuit

Figure 4.1 shows the Connection Example when downloading to the NOR flash memory.

Note: The symbol "#" indicates a negative-true logic or an active low.
5. Software

5.1 Operation Overview

The FMTOOL consists of two programs; the erase module and the write module. The E10A-USB emulator writes program data in the flash memory using these programs. For details on these modules, refer to the section "6.22 Download Function to the Flash Memory" in the "Super HTM Family E10A-USB Emulator User's Manual".

In the sample code, without using a chip erase, only necessary sectors are erased to speed up the download process. The erase processing for the sectors is executed in the write module.

5.1.1 Batch File

Executes a reset command to initialize the SH7269 using the batch file which is executed before downloading the load module. For details on the batch file and the reset command, refer to the manual listed in the integrated development environment on our web site.

5.1.2 Erase Module

When commencing download of the load module, the FMTOOL is transferred to the on-chip high-speed RAM on the SH7269. The erase module is executed only once after the transfer.

Although the primary purpose of the erase module operation is to implement a chip erase processing, initialization of the internal information to be used in the write module is executed in the sample code.
5.1.3 **Write Module**

The write module is executed repeatedly when downloading the load module. The program data divided into the access size is received as the argument.

Figure 5.1 shows the Write Module Diagram. The write module is executed on the on-chip high-speed RAM of the SH7269, receives the program data address and word data as the argument, and writes into the flash memory. When the destination address is in the sector that has not been erased, writes after erasing the sector.

![Write Module Diagram](image-url)
5.2 File Composition

Table 5.1 lists the File Composition. Files generated by the integrated development environment should not be listed in this table.

<table>
<thead>
<tr>
<th>File Name</th>
<th>Outline</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>fmtool_entry.src</td>
<td>Entry module of the FMTOOL</td>
<td>Entry of erase module and write module</td>
</tr>
<tr>
<td>fmtool_main.c</td>
<td>Main module of the FMTOOL</td>
<td></td>
</tr>
<tr>
<td>flash_cfg.h</td>
<td>Configuration of the flash memory</td>
<td>Configuration file of the flash memory (Spansion, S29GL256P) Customizing is required when the above flash memory does not comply with the device specification</td>
</tr>
<tr>
<td>sh7269_fmtool.hdc</td>
<td>Batch file</td>
<td>Registers in the integrated development environment</td>
</tr>
</tbody>
</table>
### 5.3 Constants

Table 5.2 lists the Constants Used in the Sample Code (1/3).

<table>
<thead>
<tr>
<th>Constant Name</th>
<th>Setting Value</th>
<th>Contents</th>
<th>Need for change</th>
</tr>
</thead>
<tbody>
<tr>
<td>FM_UNIFORM</td>
<td>0</td>
<td>Uniform type</td>
<td></td>
</tr>
<tr>
<td>FM_TOP_BOOT</td>
<td>1</td>
<td>Top boot type</td>
<td></td>
</tr>
<tr>
<td>FM_BOTTOM_BOOT</td>
<td>2</td>
<td>Bottom boot type</td>
<td></td>
</tr>
<tr>
<td>FM_DUAL_BOOT</td>
<td>3</td>
<td>Dual boot type</td>
<td></td>
</tr>
<tr>
<td>FM_BOOT_TYPE</td>
<td>FM_UNIFORM</td>
<td>Type of sector structure</td>
<td>x</td>
</tr>
<tr>
<td>FM_B_BOOT_SECTOR_START</td>
<td>0x00000000UL</td>
<td>Start address of the bottom boot sector (by the word)</td>
<td>x</td>
</tr>
<tr>
<td>FM_B_BOOT_SECTOR_SIZE</td>
<td>0x00000000UL</td>
<td>Size per sector for bottom boot sector (by the word)</td>
<td>x</td>
</tr>
<tr>
<td>FM_B_BOOT_SECTOR_NUM</td>
<td>0</td>
<td>Number of bottom boot sectors</td>
<td>x</td>
</tr>
<tr>
<td>FM_NORMAL_SECTOR_START</td>
<td>0x00000000UL</td>
<td>Start address of the area except for the boot sector (by the word)</td>
<td>x</td>
</tr>
<tr>
<td>FM_NORMAL_SECTOR_SIZE</td>
<td>0x00010000UL</td>
<td>Size per sector for the area except for the boot sector (by the word)</td>
<td>x</td>
</tr>
<tr>
<td>FM_NORMAL_SECTOR_NUM</td>
<td>256</td>
<td>Number of sectors in the area except for the boot sector</td>
<td>x</td>
</tr>
<tr>
<td>FM_T_BOOT_SECTOR_START</td>
<td>0x00000000UL</td>
<td>Start address of the top boot sector (by the word)</td>
<td>x</td>
</tr>
<tr>
<td>FM_T_BOOT_SECTOR_SIZE</td>
<td>0x00000000UL</td>
<td>Size per sector for top boot sector (by the word)</td>
<td>x</td>
</tr>
<tr>
<td>FM_T_BOOT_SECTOR_NUM</td>
<td>0</td>
<td>Number of top boot sectors</td>
<td>x</td>
</tr>
<tr>
<td>FM_END_ADDRESS</td>
<td>0x00FFFFFFUL</td>
<td>End address of the flash memory (by the word)</td>
<td>x</td>
</tr>
<tr>
<td>FM_CMD_RESET</td>
<td>0x00F0</td>
<td>Reset command</td>
<td>x</td>
</tr>
<tr>
<td>FM_CMD_S_ERASE_ADDR_1</td>
<td>0x0555</td>
<td>Address of 1st cycle sector erase command (address for word command)</td>
<td>x</td>
</tr>
<tr>
<td>FM_CMD_S_ERASE_ADDR_2</td>
<td>0x02AA</td>
<td>Address of 2nd cycle sector erase command (address for word command)</td>
<td>x</td>
</tr>
<tr>
<td>FM_CMD_S_ERASE_ADDR_3</td>
<td>0x0555</td>
<td>Address of 3rd cycle sector erase command (address for word command)</td>
<td>x</td>
</tr>
<tr>
<td>FM_CMD_S_ERASE_ADDR_4</td>
<td>0x0555</td>
<td>Address of 4th cycle sector erase command (address for word command)</td>
<td>x</td>
</tr>
<tr>
<td>FM_CMD_S_ERASE_ADDR_5</td>
<td>0x02AA</td>
<td>Address of 5th cycle sector erase command (address for word command)</td>
<td>x</td>
</tr>
</tbody>
</table>

Note: The constants marked with "x" in the "need for change" depend on the flash memory specification to be used. "6.2 Customizing FMTOOL" describes customizing method.
Table 5.3  Constants Used in the Sample Code (2/3)

<table>
<thead>
<tr>
<th>Constant Name</th>
<th>Setting Value</th>
<th>Contents</th>
<th>Need for change</th>
</tr>
</thead>
<tbody>
<tr>
<td>FM_CMD_S_ERASE_DATA_1</td>
<td>0x00AA</td>
<td>Data of 1st cycle sector erase command</td>
<td>x</td>
</tr>
<tr>
<td>FM_CMD_S_ERASE_DATA_2</td>
<td>0x0055</td>
<td>Data of 2nd cycle sector erase command</td>
<td>x</td>
</tr>
<tr>
<td>FM_CMD_S_ERASE_DATA_3</td>
<td>0x0080</td>
<td>Data of 3rd cycle sector erase command</td>
<td>x</td>
</tr>
<tr>
<td>FM_CMD_S_ERASE_DATA_4</td>
<td>0x00AA</td>
<td>Data of 4th cycle sector erase command</td>
<td>x</td>
</tr>
<tr>
<td>FM_CMD_S_ERASE_DATA_5</td>
<td>0x0055</td>
<td>Data of 5th cycle sector erase command</td>
<td>x</td>
</tr>
<tr>
<td>FM_CMD_SECTOR_ERASE</td>
<td>0x0030</td>
<td>Data of 6th cycle sector erase command</td>
<td>x</td>
</tr>
<tr>
<td>FM_CMD_PROGRAM_ADDR_1</td>
<td>0x0555</td>
<td>Address of 1st cycle single word program command</td>
<td>x</td>
</tr>
<tr>
<td>FM_CMD_PROGRAM_ADDR_2</td>
<td>0x02AA</td>
<td>Address of 2nd cycle single word program command</td>
<td>x</td>
</tr>
<tr>
<td>FM_CMD_PROGRAM_ADDR_3</td>
<td>0x0555</td>
<td>Address of 3rd cycle single word program command</td>
<td>x</td>
</tr>
<tr>
<td>FM_CMD_PROGRAM_DATA_1</td>
<td>0x00AA</td>
<td>Data of 1st cycle single word program command</td>
<td>x</td>
</tr>
<tr>
<td>FM_CMD_PROGRAM_DATA_2</td>
<td>0x0055</td>
<td>Data of 2nd cycle single word program command</td>
<td>x</td>
</tr>
<tr>
<td>FM_CMD_PROGRAM_DATA_3</td>
<td>0x00A0</td>
<td>Data of 3rd cycle single word program command</td>
<td>x</td>
</tr>
<tr>
<td>FM_CHK_DQ5</td>
<td>0x0020</td>
<td>Mask information of bit for excess of timing limit (DQ5)</td>
<td></td>
</tr>
<tr>
<td>FM_CHK_DQ6</td>
<td>0x0040</td>
<td>Mask information of toggle bit (DQ6)</td>
<td></td>
</tr>
<tr>
<td>FM_CHK_DQ7</td>
<td>0x0080</td>
<td>Mask information of data polling bit (DQ7)</td>
<td></td>
</tr>
<tr>
<td>FM_CS0_NON_CACHE_START</td>
<td>0x20000000UL</td>
<td>Start address of the SH7269 CS0 space (cache-disabled space)</td>
<td></td>
</tr>
<tr>
<td>FM_TYPE_BYTE</td>
<td>0x4220</td>
<td>R5 parameter of write module (access the flash memory by the byte-size)</td>
<td></td>
</tr>
<tr>
<td>FM_TYPE_WORD</td>
<td>0x5720</td>
<td>R5 parameter of write module (access the flash memory by the word-size)</td>
<td></td>
</tr>
<tr>
<td>FM_TYPE_LONG</td>
<td>0x4C20</td>
<td>R5 parameter of write module (access the flash memory by the long-size)</td>
<td></td>
</tr>
</tbody>
</table>

Note: The constants marked with “x” in the "need for change" depend on the flash memory specification to be used. “6.2 Customizing FMTOOL” describes customizing method.
Table 5.4 Constants Used in the Sample Code (3/3)

<table>
<thead>
<tr>
<th>Constant Name</th>
<th>Setting Value</th>
<th>Contents</th>
<th>Need for change</th>
</tr>
</thead>
<tbody>
<tr>
<td>FM_TOOL_OK</td>
<td>0</td>
<td>Return value: no error</td>
<td></td>
</tr>
<tr>
<td>FM_TOOL_E_ERASE</td>
<td>-1</td>
<td>Return value: erase error</td>
<td></td>
</tr>
<tr>
<td>FM_TOOL_E_WRITE</td>
<td>-2</td>
<td>Return value: write error</td>
<td></td>
</tr>
<tr>
<td>FM_TOOL_E_VERIFY</td>
<td>0x4254</td>
<td>Return value: verify error</td>
<td></td>
</tr>
<tr>
<td>FM_TOOL_E_ARGUMENT</td>
<td>-16</td>
<td>Return value: argument error</td>
<td></td>
</tr>
<tr>
<td>FM_WRITE_TOP</td>
<td>H'FFF80000</td>
<td>Start address of write module</td>
<td></td>
</tr>
<tr>
<td>FM_ERASE_TOP</td>
<td>H'FFF81000</td>
<td>Start address of erase module</td>
<td></td>
</tr>
<tr>
<td>FM_STACK_TOP</td>
<td>H'FFF90000</td>
<td>Stack pointer the FMTOOL uses</td>
<td></td>
</tr>
<tr>
<td>FM_SECTOR_NUM</td>
<td>Total number of sectors Number of all sectors (FM_T_BOOT_SECTOR_NUM+ FM_B_BOOT_SECTOR_NUM+ FM_NORMAL_SECTOR_NUM)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Note: The constants marked with "x" in the "need for change" depend on the flash memory specification to be used. "6.2 Customizing FMTOOL" describes customizing method.
5.4 Variable

Table 5.5 lists the Static Variable

Table 5.5  Static Variable

<table>
<thead>
<tr>
<th>Type</th>
<th>Variable Name</th>
<th>Contents</th>
<th>Function Used</th>
</tr>
</thead>
<tbody>
<tr>
<td>uint32_t</td>
<td>fmtool_pre_erase_sctno[(FM_SECTOR_NUM/32)+1]</td>
<td>Management information of the erased sectors</td>
<td>FmtoolInit, FmtoolWrite, FmCheckPreErase</td>
</tr>
</tbody>
</table>

5.5 Functions

Table 5.6 lists the Functions.

Table 5.6  Functions

<table>
<thead>
<tr>
<th>Function Name</th>
<th>Outline</th>
</tr>
</thead>
<tbody>
<tr>
<td>_FMTOOL_ERASE</td>
<td>Entry processing for the erase module</td>
</tr>
<tr>
<td>_FMTOOL_WRITE</td>
<td>Entry processing for the write module</td>
</tr>
<tr>
<td>FmtoolInit</td>
<td>Main processing for the erase module (Initialization)</td>
</tr>
<tr>
<td>FmtoolWrite</td>
<td>Main processing for the write module (erasing/writing)</td>
</tr>
<tr>
<td>FmResetCmd</td>
<td>Reset processing for the flash memory</td>
</tr>
<tr>
<td>FmSectorEraseCmd</td>
<td>Sector erase processing</td>
</tr>
<tr>
<td>FmiWordProgramCmd</td>
<td>Single word programming</td>
</tr>
<tr>
<td>FmCheckPreErase</td>
<td>Checking the erased sectors</td>
</tr>
</tbody>
</table>
5.6 Function Specifications

The following tables list the function specifications of the sample code.

_FMTool_ERASE

Outline: Entry processing for the erase module
Header: None
Declaration: _FMTool_ERASE:
Description: Allocates in the address H'FFF8 1000 in the entry section of the erase module which is commenced by the E10A-USB flash memory download function. This module saves the register to execute the function FmtoolLimit after setting the stack pointer. Then it restores the register and places the control back to the E10A-USB emulator according to the RTS instruction.
Argument: R4 register: Access size
(Byte: H'4220, Word: H'5720, Long: H'4C20)
Return Value: None
Remarks: Described in the assembly language

_FMTool_WRITE

Outline: Entry processing for the write module
Header: None
Declaration: _FMTool_WRITE:
Description: Allocates in the address H'FFF8 0000 in the entry section of the write module which is commenced by the E10A-USB flash memory download function. This module saves the register to execute the function FmtoolWrite after setting the stack pointer. Then it restores the register and places the control back to the E10A-USB emulator according to the RTS instruction.
Argument: R4 register: Write address
R5 register: Access size
(Byte: H'4220, Word: H'5720, Long: H'4C20)
R6 register: Write data
Setting Value: R0 register = 0: Normal end
Except for R0 resister = 0: Error end
Remarks: Described in the assembly language

FmtoolInit

Outline: Main processing for the erase mod (initialization)
Header: "flashCfg.h"
Declaration: void FmtoolInit(uint32_t access_size);
Description: Initializes the variable used in main processing for write module. This function is executed from the entry point of the FMTOOL (_FMTool_ERASE)
Argument: First argument: access_size: Access size (unused)
(Byte: H'4220, Word: H'5720, Long: H'4C20)
Return Value: None
Remarks: None
### FmtoolWrite

**Outline**
Main processing for the write module (erasing/writing)

**Header**
`flash_cfg.h`

**Declaration**

```
int32_t FmtoolWrite(uint32_t addr, uint32_t access_size, uint32_t data);
```

**Description**
Executes erasing and writing of the flash memory. The flash memory is accessed by the sector for erasing and by the 16-bit for writing. When the destination address specified by the argument `addr` is not erased, writes after erasing the sector. When the sector has been erased, executes only writing. This function is executed from the entry point of the FMTOOL (`_FMTOOL_WRITE`)

**Argument**
- First argument: `addr` : Write address
- Second argument: `access_size` : Access size 
  (Byte: 0x4220, Word: 0x5720, Long: 0x4C20)
- Third argument: `data` : Write data

**Return Value**
- 0: Normal end
- -1: Error end (erase error)
- -2: Error end (write error)
- 0x4254: Error end (verify error)
- -16: Error end (argument error)

**Remarks**
Only word size is available for the access size.
When any size except for the word size is specified, end this function with argument error.

### FmResetCmd

**Outline**
Reset processing for the flash memory

**Header**
`flash_cfg.h`

**Declaration**

```
void FmResetCmd(volatile uint16_t *pResetAddr);
```

**Description**
Executes a reset command to the flash memory

**Argument**
- First argument: `*pResetAddr` : Pointer to the address for reset

**Return Value**
None

**Remarks**

### FmSectorEraseCmd

**Outline**
Sector erase processing

**Header**
`flash_cfg.h`

**Declaration**

```
int32_t FmSectorEraseCmd(volatile uint16_t *pEraseBlock);
```

**Description**
Executes the sector erase to the specified sector. Checks the toggle bit and the polling bit after executing the erase command, and ends this function with error when the erase error comes up.

**Argument**
- First argument: `*pEraseBlock` : Pointer to start address of the sector for erase

**Return Value**
- 0: Normal end
- -1: Error end (erase error)

**Remarks**

---
## FmWordProgramCmd

**Outline**

Single word programming

**Header**

"flash_cfg.h"

**Declaration**

```c
int32_t FmWordProgramCmd(volatile uint16_t *pWriteAddr, uint16_t Data);
```

**Description**

Writes the specified data into the specified address. Checks the toggle bit and the polling bit after executing the write command and ends this function with error when the write error comes up. When the data can be written without error, executes verification, and ends this function with error if the verify error comes up.

**Argument**

- First argument: *pWriteAddr : Pointer to the address for write
- Second argument: Data : Write data

**Return Value**

- 0: Normal end
- -2: Error end (write error)
- H'4254: error end (verify error)

**Remarks**

FmCheckPreErase

**Outline**

Checking erased sectors

**Header**

"flash_cfg.h"

**Declaration**

```c
int32_t FmCheckPreErase(uint32_t sect_no);
```

**Description**

Checks whether or not the address specified write address has been erased, and returns the result

**Argument**

First Argument: sect_no : Sector number

**Return Value**

- 0: The sector of write address is not erased
- 1: The sector of write address is erased

**Remarks**
5.7 Flowcharts
This section describes the procedure of the batch file processing and the procedures of the major functions used in the sample code.

5.7.1 Batch File
Figure 5.2 shows the procedure of the Batch File Processing.

![Flowchart Diagram]

Figure 5.2 Batch File Processing

Note: On the SH7269, the pins of A25 to A21 have port functions in the initial state. When using these pins, it is required to set the address bus function in the batch file.
5.7.2 Erase Module

Figure 5.3 shows the Procedure of Erase Module.

- Set the stack pointer
- Save the register
- Execute main processing of the erase module: FmtoolInit()
- Restore the register
- Return

Described in the assembly language

Saves PR, R1 to R14 registers in the stack area.

Initializes the variable to be used in the main processing of write module.
5.7.3 Write module

Figure 5.4 shows the Procedure of Write Module.

---

**Figure 5.4 Procedure of Write Module**

1. **_FM_TOOL_WRITE**
   - Described in the assembly language

2. Set the stack pointer
3. Save the register
4. Execute main processing of the module FmtoolWrite()
5. Restore the register
6. return (return value of FmtoolWrite)

- **FmtoolWrite**
  - First argument: write address
  - Second argument: access size (byte/word/long)
  - Third argument: write data

- **Word access?**
  - Yes
    - Mask write address with the end address of the flash memory
    - inapplicable access size (error)
    - return (-16)
  - No
    - Execute single word program FmWordProgramCmd()
    - Execute reset FmResetCmd()
    - Program error or verify error
    - return (-1)
    - Verify error
    - return (H'4254)
    - return (-2)

- **Erased?**
  - Yes
    - Execute sector erase FmSectorEraseCmd()
    - Execute reset FmResetCmd()
    - Erase error?
      - No error
        - Update the erased sector information
        - return (0)
      - Erase error
        - No error
        - Program error
        - Verify error
        - return (H'4254)
        - return (-2)
    - No
      - Execute sector erase command Check toggle bit and polling bit
      - Erase error?
        - No error
          - Program error or verify error
          - return (-1)
        - Verify error
          - Program error
          - return (H'4254)
          - return (-2)
      - No error
        - Program error or verify error
        - return (-1)
        - Verify error
        - Program error
        - return (H'4254)
        - return (-2)

- **Calculate start address of the sector from the write address**

- **Execute reset command**

- **Check the erased sector FmCheckPreErase()**

- **Execute the single word program FmWordProgramCmd()**

- **Execute reset FmResetCmd()**

- **Check the toggle bit and polling bit**

- **Program error or verify error**

- **Verify error**

- **Program error**
6. Application Example

6.1 Procedure of User Program Download

This section describes the procedure of downloading user programs to the flash memory using the created FMTOOL (sh7269_fmtool.mot).

6.1.1 Prepare for Download Environment

1. Connect user's system with the E10A-USB emulator connected to PC.
2. Start the High-performance Embedded Workshop to open the workspace for user programs.
3. The Device Select Dialog Box is open as shown in Figure 6.1. Select the CPU in use in the drop-down listbox for Device. Click the OK button.

![Device Select Dialog Box](image)

Note: The shown window is an example adopting the SH72691

Figure 6.1 Device Select Dialog Box

4. The Connecting box is displayed and emulator connection starts. The RESET Signal Request Dialog Box shown in Figure 6.2 is displayed.

![RESET Signal Request Dialog Box](image)

Figure 6.2 RESET Signal Request Dialog Box

5. Turn on the user's system.

Having received the RESET signal from the user's system, click the OK button.

When "connected" is displayed on the Output Window in the High-performance Embedded Workshop, the E10A-USB emulator successfully started.
6.1.2 Registering Batch File

1. Select in the menu; [Debug] → [Debug Settings]
2. The Window for Debug Setting shown in Figure 6.3 opens.
3. Select "Before download modules" in the pull-down menu for the "Command batch file load timing".
4. Click the "Add" at "Command line batch processing" to add a batch file.
5. Click the OK button, and registration is completed.

![Figure 6.3 Window for Debug Setting](image)
6.1.3 Setting Configuration Dialog Box

1. Select in the menu; [Setup] → [Emulator] → [System]

2. Figure 6.4 shows the Configuration Dialog Box (in the page of loading flash memory) for setting to download a user program to the external flash memory using the E10A-USB emulator.

![Configuration Dialog Box (in the page of loading flash memory)](image)

Table 6.1 lists the setting in the items in the configuration dialog box. Finish setting and click the OK button, configuration is completed.

**Table 6.1 Setting Value in the Configuration Dialog Box**

<table>
<thead>
<tr>
<th>Item</th>
<th>Setting Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Loading flash memory</td>
<td>Enable</td>
</tr>
<tr>
<td>Erasing flash memory</td>
<td>Enable</td>
</tr>
<tr>
<td>File Name</td>
<td>(Directory stores the FMTOOL) \sh7269_fmtool.mot</td>
</tr>
<tr>
<td>Bus width of flash memory</td>
<td>16-bit bus width</td>
</tr>
<tr>
<td>All erasing module address</td>
<td>Assign start address of the erase module (H'FFF8 1000)</td>
</tr>
<tr>
<td>Writing module address</td>
<td>Assign start address of the write module (H'FFF8 0000)</td>
</tr>
</tbody>
</table>
6.1.4 Adding Download Module
Open the debug setting window from the debug menu. Click "Add". In the Download Module Window shown in Figure 6.5, add user programs to be loaded in the serial flash memory to the download module.

6.1.5 Downloading User Programs
Using the download function shown in Figure 6.6, download the user programs.
6.2 Customizing FMTOOL

The sample code is dependent on the specification of the device in the serial flash memory. Customization of the program may be necessary when altering the device.

The sample code corresponds to the JEDEC standard compatible command method. When using the flash memory of the method, the sample code can be applied.

When using the flash memory of the CUI (Command User interface) command method, the sample code cannot be applied. In this case, the user needs to produce a new download program.

Notes:
1. JEDEC standard compatible command method is a writing method which requires to issue a command to the predetermined address (H'555, H'2AA, and etc.).
2. CUI command method is the method which requires issuing the write command (H'40) and the erase command (H'20) to the CUI.

6.2.1 Device Specification for Sample Code

Table 6.2 and Table 6.3 list the Specification of Used Device and the Commands Used in Sample Code respectively.

<table>
<thead>
<tr>
<th>Table 6.2 Specification of Used Device</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Item</strong></td>
</tr>
<tr>
<td>Manufacturer</td>
</tr>
<tr>
<td>Model</td>
</tr>
<tr>
<td>Capacity</td>
</tr>
<tr>
<td>Data bus width</td>
</tr>
<tr>
<td>Access time</td>
</tr>
<tr>
<td>Sector structure</td>
</tr>
<tr>
<td>Sector size</td>
</tr>
<tr>
<td>Number of sector</td>
</tr>
<tr>
<td>Writing method</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Table 6.3 Commands Used in Sample Code</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Item</strong></td>
</tr>
<tr>
<td>Erase command (Sector erase)</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>Program command</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>Reset command</td>
</tr>
</tbody>
</table>

Notes:
*1. SA indicates the sector address. It assigns sector address to be erased.
*2. PA indicates the program address. It assigns address to be written.
*3. PD indicates the program data. It assigns data to be written.
*4. The symbol "-" indicates the address of the space where the flash memory is allocated. Any address can be assigned in the space where the flash memory is allocated.
6.2.2 Applicable Flash Memory Boot Types

The following four flash memory boot types are applicable to the SH7269 by customizing the sample code.

1) Uniform type
2) Bottom boot type
3) Top boot type
4) Dual boot type

Figure 6.7 shows the Memory Map for Flash Memory Boot Types.

![Memory Map for Flash Memory Boot Types](image_url)
### 6.2.3 Contents of Customization

Table 6.4 lists the Cases for Customization and their Descriptions (1/2).

#### Table 6.4 Cases for Customization and their Descriptions (1/2)

<table>
<thead>
<tr>
<th>case</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>When the boot type is different</td>
<td>The sample code corresponds to erasing and writing of the flash memory with Uniform type. When using the flash memory with Top boot, Bottom boot and Dual boot type, change the macro definition FM_BOOT_TYPE as follows: Top boot type: FM_TOP_BOOT Bottom boot type: FM_BOTTOM_BOOT Dual boot type: FM_DUAL_BOOT (The Uniform type FM_UNIFORM is defined as the initial value.)</td>
</tr>
<tr>
<td>Change the boot type to: (1) Bottom boot type</td>
<td>Change the sector definition -FM_B_BOOT_SECTOR_START -FM_B_BOOT_SECTOR_SIZE -FM_B_BOOT_SECTOR_NUM -FM_NORMAL_SECTOR_START -FM_NORMAL_SECTOR_SIZE -FM_NORMAL_SECTOR_NUM Refer to the section 6.2.5 to change the definition. The initial values of the following sector definitions are defined as 0. These are applicable without change. -FM_T_BOOT_SECTOR_START -FM_T_BOOT_SECTOR_SIZE -FM_T_BOOT_SECTOR_NUM</td>
</tr>
<tr>
<td>Change the boot type to: (2) Top boot type</td>
<td>Change the sector definition -FM_NORMAL_SECTOR_START -FM_NORMAL_SECTOR_SIZE -FM_NORMAL_SECTOR_NUM -FM_T_BOOT_SECTOR_START -FM_T_BOOT_SECTOR_SIZE -FM_T_BOOT_SECTOR_NUM Refer to the section 6.2.6 to change the definition. The initial values of the following sector definitions are defined as 0. These are applicable without change. -FM_B_BOOT_SECTOR_START -FM_B_BOOT_SECTOR_SIZE -FM_B_BOOT_SECTOR_NUM</td>
</tr>
</tbody>
</table>

Note: The FMTOOL depends on the flash memory specification. Note that the cases listed in Table 6.4 and Table 6.5 may not be applicable for every situation. Check the device data sheet and modify the FMTOOL according to the device specification.
### Table 6.5 Necessary Customization and their Descriptions (2/2)

<table>
<thead>
<tr>
<th>Case</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Change the boot type to:</td>
<td></td>
</tr>
<tr>
<td>(3) Dual boot type</td>
<td>Change the sector definition</td>
</tr>
<tr>
<td></td>
<td>-FM_B_BOOT_SECTOR_START</td>
</tr>
<tr>
<td></td>
<td>-FM_B_BOOT_SECTOR_SIZE</td>
</tr>
<tr>
<td></td>
<td>-FM_B_BOOT_SECTOR_NUM</td>
</tr>
<tr>
<td></td>
<td>-FM_NORMAL_SECTOR_START</td>
</tr>
<tr>
<td></td>
<td>-FM_NORMAL_SECTOR_SIZE</td>
</tr>
<tr>
<td></td>
<td>-FM_NORMAL_SECTOR_NUM</td>
</tr>
<tr>
<td></td>
<td>-FM_T_BOOT_SECTOR_START</td>
</tr>
<tr>
<td></td>
<td>-FM_T_BOOT_SECTOR_SIZE</td>
</tr>
<tr>
<td></td>
<td>-FM_T_BOOT_SECTOR_NUM</td>
</tr>
<tr>
<td></td>
<td>Refer to the section 6.2.7 to change the definition.</td>
</tr>
<tr>
<td>When the sector size or the number of sectors are different</td>
<td>Change the sector definition</td>
</tr>
<tr>
<td></td>
<td>-FM_NORMAL_SECTOR_START</td>
</tr>
<tr>
<td></td>
<td>-FM_NORMAL_SECTOR_SIZE</td>
</tr>
<tr>
<td></td>
<td>-FM_NORMAL_SECTOR_NUM</td>
</tr>
<tr>
<td></td>
<td>Refer to the section 6.2.4 to change the definition.</td>
</tr>
<tr>
<td></td>
<td>The initial values of the following sector definitions are defined as 0. These are applicable</td>
</tr>
<tr>
<td></td>
<td>when the sector size or the number of sectors are different.</td>
</tr>
<tr>
<td></td>
<td>-FM_B_BOOT_SECTOR_START</td>
</tr>
<tr>
<td></td>
<td>-FM_B_BOOT_SECTOR_SIZE</td>
</tr>
<tr>
<td></td>
<td>-FM_B_BOOT_SECTOR_NUM</td>
</tr>
<tr>
<td></td>
<td>-FM_T_BOOT_SECTOR_START</td>
</tr>
<tr>
<td></td>
<td>-FM_T_BOOT_SECTOR_SIZE</td>
</tr>
<tr>
<td></td>
<td>-FM_T_BOOT_SECTOR_NUM</td>
</tr>
<tr>
<td>When the memory capacity is different</td>
<td>Change the setting value for the macro FM_END_ADDRESS</td>
</tr>
<tr>
<td>When the boot type is different from the types listed in Figure 6.7</td>
<td>The operation function of the flash memory needs to be customized. Refer to the sample code</td>
</tr>
<tr>
<td></td>
<td>for details.</td>
</tr>
<tr>
<td>When the command specifications for erasing, writing and reset are</td>
<td></td>
</tr>
<tr>
<td>different</td>
<td></td>
</tr>
<tr>
<td>When the write command requires the CUI command method</td>
<td></td>
</tr>
<tr>
<td>When the SH7269 and the flash memory are connected by the bus</td>
<td></td>
</tr>
<tr>
<td>except for 16-bit</td>
<td></td>
</tr>
</tbody>
</table>

Note: The FMTOOL depends on the flash memory specification. Note that the cases listed in Table 6.4 and Table 6.5 may not be applicable for every situation. Check the device data sheet and modify the FMTOOL according to the device specification.
6.2.4 Customizing Uniform Type

Figure 6.8 shows the Method to Customize Sector Size and Number of Sectors with Uniform Type. When using the flash memory with Uniform type which has different sector size and number of sectors from the S29GL256P90TFIR1, change the setting value for macro of the Uniform type sector information. When the different flash memory capacity is provided, change the setting value for macro of the end address.

![Figure 6.8 Method to Customize Sector Size and Number of Sectors with Uniform Type](image-url)
6.2.5 Customizing to Bottom Boot Type

Figure 6.9 shows the Method to Customize to Bottom Boot Type. When using the flash memory with Bottom boot type, change the setting value for macro of the sector information which relates to the bottom boot sector area and other areas except for boot sectors. When the different flash memory capacity is provided, change the setting value for macro of the end address.

![Diagram showing the method to customize to Bottom Boot Type](image)

**Figure 6.9 Method to Customize to Bottom Boot Type**

```c
/* ---- Specify boot sector type ---- */
enum ENUM_BOOT_TYPE {
    FM_UNIFORM, /* Uniform sector type     */
    FM_TOP_BOOT, /* Top boot sector type    */
    FM_BOTTOM_BOOT, /* Bottom boot sector type */
    FM_DUAL_BOOT /* Dual boot sector type   */
};
#define FM_BOOT_TYPE FM_BOTTOM_BOOT /* Boot type setting */

/* ---- Bottom boot sector ---- */
#define FM_B_BOOT_SECTOR_START 0x00000000UL
#define FM_B_BOOT_SECTOR_SIZE 0x00001000UL
#define FM_B_BOOT_SECTOR_NUM 8

/* ---- Sector except boot sector ---- */
#define FM_NORMAL_SECTOR_START 0x00008000UL
#define FM_NORMAL_SECTOR_SIZE 0x00008000UL
#define FM_NORMAL_SECTOR_NUM 63

/* ---- Top boot sector ---- */
#define FM_T_BOOT_SECTOR_START 0x00000000UL
#define FM_T_BOOT_SECTOR_SIZE 0x00000000UL
#define FM_T_BOOT_SECTOR_NUM 0

/* ---- End address of the all sector ---- */
#define FM_END_ADDRESS 0x001FFFFFUL
```

Method for definition

*File name: flash_cfg.h*

**Define start address of the Top boot sector,**

0x00000000 for the sector size and 0 for the number of sectors when the boot type is Bottom.

**Define FM_BOTTOM_BOOT when the boot type is Bottom**

**Define start address (word address)**

**Define sector size (word size)**

**Define the number of sectors**

**Define start address (word address)**

**Define sector size (word size)**

**Define the number of sectors**

**Define start address of the Top boot sector.**

0x00000000 for the sector size and 0 for the number of sectors when the boot type is Bottom.

**Define end address of the flash memory**

0x01FFFFFFUL
6.2.6 Customizing to Top Boot Type

Figure 6.10 shows the Method to Customize to Top Boot Type. When using the flash memory with Top boot type, change the setting value for macro of sector information which relates to the top boot sector area and other areas except for the boot sector. When the different flash memory capacity is provided, change the setting value for macro of the end address.

```
/* ---- Specify boot sector type ---- */
enum ENUM_BOOT_TYPE {
    FM_UNIFORM, /* Uniform sector type */
    FM_TOP_BOOT, /* Top boot sector type */
    FM_BOTTOM_BOOT, /* Bottom boot sector type */
    FM_DUAL_BOOT /* Dual boot sector type */
};

#define FM_BOOT_TYPE FM_TOP_BOOT /* Boot type setting */

/* ---- Bottom boot sector ---- */
#define FM_B_BOOT_SECTOR_START 0x00000000UL
#define FM_B_BOOT_SECTOR_SIZE 0x00000000UL
#define FM_B_BOOT_SECTOR_NUM 0

/* ---- Sector except boot sector ---- */
#define FM_NORMAL_SECTOR_START 0x00000000UL
#define FM_NORMAL_SECTOR_SIZE 0x00008000UL
#define FM_NORMAL_SECTOR_NUM 63

/* ---- Top boot sector ---- */
#define FM_T_BOOT_SECTOR_START 0x001F8000UL
#define FM_T_BOOT_SECTOR_SIZE 0x00001000UL
#define FM_T_BOOT_SECTOR_NUM 8

/* ---- End address of the all sector ---- */
#define FM_END_ADDRESS 0x001FFFFFUL
```

Figure 6.10 Method to Customize to Top Boot Type
6.2.7 Customizing to Dual Boot Type

Figure 6.11 shows the Method to Customize to Dual Boot Type. When using the flash memory with Dual boot type, change the setting value for macro of the sector information which relates to the top boot sector area, the bottom boot sector area and other areas except for the boot sectors. When the different flash memory capacity is provided, change the setting value for macro of the end address.

![Diagram of Dual Boot Type Customization]

**Figure 6.11 Method to Customize to Dual Boot Type**
6.2.8 Definitions of Command

Figure 6.12 shows the Definitions of Command. The sample code corresponds to the JEDEC standard compatible command, and defines the command when connects the SH7269 and the flash memory by 16-bit. Refer to the word mode (x16-bit mode) command on the flash memory data sheet to verify that no difference exists in the command contents between the reset command, the sector erase command and program command (write command). Note that the sample code does not correspond to page program (page write).

* File name: flash_cfg.h

```c
/* ==== Specify command ==== */
/* ---- Reset command ---- */
#define FM_CMD_RESET 0x00F0 /* Data (First cycle) */
/* ---- Sector erase command ---- */
#define FM_CMD_S_ERASE_ADDR_1 0x0555 /* Address (First cycle) */
#define FM_CMD_S_ERASE_ADDR_2 0x02AA /* Address (Second cycle) */
#define FM_CMD_S_ERASE_ADDR_3 0x0555 /* Address (Third cycle) */
#define FM_CMD_S_ERASE_ADDR_4 0x0555 /* Address (Fourth cycle) */
#define FM_CMD_S_ERASE_ADDR_5 0x02AA /* Address (Fifth cycle) */
#define FM_CMD_S_ERASE_DATA_1 0x00AA /* Data (First cycle) */
#define FM_CMD_S_ERASE_DATA_2 0x0055 /* Data (Second cycle) */
#define FM_CMD_S_ERASE_DATA_3 0x0080 /* Data (Third cycle) */
#define FM_CMD_S_ERASE_DATA_4 0x00AA /* Data (Fourth cycle) */
#define FM_CMD_S_ERASE_DATA_5 0x0055 /* Data (Fifth cycle) */
#define FM_CMD_SECTOR_ERASE 0x0030 /* Data (Sixth cycle) */
/* ---- Single word program command ---- */
#define FM_CMD_PROGRAM_ADDR_1 0x0555 /* Address (First cycle) */
#define FM_CMD_PROGRAM_ADDR_2 0x02AA /* Address (Second cycle) */
#define FM_CMD_PROGRAM_ADDR_3 0x00AA /* Address (Third cycle) */
#define FM_CMD_PROGRAM_DATA_1 0x00AA /* Data (First cycle) */
#define FM_CMD_PROGRAM_DATA_2 0x0055 /* Data (Second cycle) */
#define FM_CMD_PROGRAM_DATA_3 0x00A0 /* Data (Third cycle) */
```

Figure 6.12 Definitions of Command
7. Sample Code

Sample code can be downloaded from the Renesas Electronics website.

8. Reference Documents

User's Manual: Hardware
   SH7268/SH7269 Group User's Manual: Hardware Rev. 1.00
   The latest version can be downloaded from the Renesas Electronics website.

Technical Update/Technical News
   The latest information can be downloaded from the Renesas Electronics website.

Development Tool Manual
   SuperH RISC Engine Family C/C++ Compiler Package V.9.04
   C compiler User's Manual Rev. 1.01
   The latest version can be downloaded from the Renesas Electronics website.

   SuperH Family E10A-USB Emulator User's Manual Rev. 9.00
   The latest version can be downloaded from the Renesas Electronics website.

   Supplementary Information on Using the SH7264, SH7262, SH7266, SH7267, SH7268 and SH7269
   The latest version can be downloaded from the Renesas Electronics website.

Website and Support

Renesas Electronics website
   http://www.renesas.com

Inquiries
   http://www.renesas.com/contact/
### REVISION HISTORY

SH7268/SH7269 Group E10A-USB Flash Memory Download Function  
(Download to the NOR Flash Memory)

<table>
<thead>
<tr>
<th>Rev.</th>
<th>Date</th>
<th>Description</th>
<th>Page</th>
<th>Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.00</td>
<td>Jul. 18, 2012</td>
<td>First edition issued</td>
<td>-</td>
<td></td>
</tr>
</tbody>
</table>

All trademarks and registered trademarks are the property of their respective owners.
General Precautions in the Handling of MPU/MCU Products

The following usage notes are applicable to all MPU/MCU products from Renesas. For detailed usage notes on the products covered by this manual, refer to the relevant sections of the manual. If the descriptions under General Precautions in the Handling of MPU/MCU Products and in the body of the manual differ from each other, the description in the body of the manual takes precedence.

1. Handling of Unused Pins
   Handle unused pins in accord with the directions given under Handling of Unused Pins in the manual. The input pins of CMOS products are generally in the high-impedance state. In operation with an unused pin in the open-circuit state, extra electromagnetic noise is induced in the vicinity of LSI, an associated shoot-through current flows internally, and malfunctions occur due to the false recognition of the pin state as an input signal become possible. Unused pins should be handled as described under Handling of Unused Pins in the manual.

2. Processing at Power-on
   The state of the product is undefined at the moment when power is supplied.
   The states of internal circuits in the LSI are indeterminate and the states of register settings and pins are undefined at the moment when power is supplied.
   In a finished product where the reset signal is applied to the external reset pin, the states of pins are not guaranteed from the moment when power is supplied until the reset process is completed.
   In a similar way, the states of pins in a product that is reset by an on-chip power-on reset function are not guaranteed from the moment when power is supplied until the power reaches the level at which resetting has been specified.

3. Prohibition of Access to Reserved Addresses
   Access to reserved addresses is prohibited.
   The reserved addresses are provided for the possible future expansion of functions. Do not access these addresses; the correct operation of LSI is not guaranteed if they are accessed.

4. Clock Signals
   After applying a reset, only release the reset line after the operating clock signal has become stable.
   When switching the clock signal during program execution, wait until the target clock signal has stabilized.
   When the clock signal is generated with an external resonator (or from an external oscillator) during a reset, ensure that the reset line is only released after full stabilization of the clock signal. Moreover, when switching to a clock signal produced with an external resonator (or by an external oscillator) while program execution is in progress, wait until the target clock signal is stable.

5. Differences between Products
   Before changing from one product to another, i.e. to one with a different type number, confirm that the change will not lead to problems.
   The characteristics of MPU/MCU in the same group but having different type numbers may differ because of the differences in internal memory capacity and layout pattern. When changing to products of different type numbers, implement a system-evaluation test for each of the products.
Notice

1. Descriptions of circuits, software and other related information in this document are provided only to illustrate the operation of semiconductor products and application examples. You are fully responsible for the incorporation of these circuits, software, and information in the design of your equipment. Renesas Electronics assumes no responsibility for any losses incurred by you or third parties arising from the use of these circuits, software, or information.

2. Renesas Electronics has used reasonable care in preparing the information included in this document, but Renesas Electronics does not warrant that such information is error free. Renesas Electronics assumes no liability whatsoever for any damages incurred by you resulting from errors in or omissions from the information included herein.

3. Renesas Electronics does not assume any liability for infringement of patents, copyrights, or other intellectual property rights of third parties by or arising from the use of Renesas Electronics products or technical information described in this document. No license, express, implied or otherwise, is granted hereby under any patents, copyrights or other intellectual property rights of Renesas Electronics or others.

4. You should not alter, modify, copy, or otherwise misappropriate any Renesas Electronics product, whether in whole or in part. Renesas Electronics assumes no responsibility for any losses incurred by you or third parties arising from such alteration, modification, copy or otherwise misappropriation of Renesas Electronics product.

5. Renesas Electronics products are classified according to the following two quality grades: “Standard” and “High Quality.” The recommended applications for each Renesas Electronics product depends on the product’s quality grade, as indicated below.

   “Standard”: Computers, office equipment, communications equipment, test and measurement equipment, audio and visual equipment, home electronics, appliance, machine tools, personal electronic equipment, and industrial robots etc.

   “High Quality”: Transportation equipment (automobiles, trains, ships, etc.), traffic control systems, anti-disaster systems, anti-crime systems, and safety equipment etc.

Renesas Electronics products are neither intended nor authorized for use in products or systems that may pose a direct threat to human life or bodily injury (artificial life support devices or systems, surgical implantations etc.), or may cause serious property damages (nuclear reactor control systems, military equipment etc.). You must check the quality grade of each Renesas Electronics product before using it in a particular application. You may not use any Renesas Electronics product for any application for which it is not intended. Renesas Electronics shall not be in any way liable for any damages or losses incurred by you or third parties arising from the use of any Renesas Electronics product for which the product is not intended by Renesas Electronics.

6. You should use the Renesas Electronics products described in this document within the range specified by Renesas Electronics, especially with respect to the maximum rating, operating supply voltage range, movement power voltage range, heat radiation characteristics, installation and other product characteristics. Renesas Electronics shall have no liability for malfunctions or damages arising out of the use of Renesas Electronics products beyond such specified ranges.

7. Although Renesas Electronics endeavors to improve the quality and reliability of its products, semiconductor products have specific characteristics such as the occurrence of failure at a certain rate and malfunctions under certain use conditions. Further, Renesas Electronics products are not subject to radiation resistance design. Please be sure to implement safety measures to guard against the possibility of physical injury, and injury or damage caused by fire in the event of the failure of a Renesas Electronics product; such as safety design for hardware and software including but not limited to redundancy, the control and malfunction prevention, appropriate treatment for aging degradation or any other appropriate measures. Because the evaluation of micropcircuit software alone is very difficult, please evaluate the safety of the final products or systems manufactured by you.

8. Please contact a Renesas Electronics sales office for details as to environmental matters such as the environmental compatibility of each Renesas Electronics product. Please use Renesas Electronics products in compliance with all applicable laws and regulations that regulate the inclusion or use of controlled substances, including without limitation, the EU RoHS Directive. Renesas Electronics assumes no liability for damages or losses occurring as a result of your noncompliance with applicable laws and regulations.

9. Renesas Electronics products and technology may not be used for or incorporated into any products or systems whose manufacture, use, or sale is prohibited under any applicable domestic or foreign laws or regulations. You should not use Renesas Electronics products or technology described in this document for any purposes relating to military applications or use by the military, including but not limited to the development of weapons of mass destruction. When exporting the Renesas Electronics products or technology described in this document, you should comply with the applicable export control laws and regulations and follow the procedures required by such laws and regulations.

10. It is the responsibility of the buyer or distributor of Renesas Electronics products, who distribute, dispose of, or otherwise places the product with a third party, to notify such third party in advance of the contents and condition set forth in this document. Renesas Electronics assumes no responsibility for any losses incurred by you or third parties as a result of unauthorized use of Renesas Electronics products.

11. This document may not be reproduced or duplicated in any form, in whole or in part, without prior written consent of Renesas Electronics.

12. Please contact a Renesas Electronics sales office if you have any questions regarding the information contained in this document or Renesas Electronics products, or if you have any other inquiries.

13. Notice 2) “Renesas Electronics” as used in this document means Renesas Electronics Corporation and also includes its majority-owned subsidiaries.

14. Notice 2) “Renesas Electronics product” means any product developed or manufactured by or for Renesas Electronics.

SALES OFFICES
Renesas Electronics Corporation

http://www.renesas.com

Reneses Electronics America Inc.
2880 Scott Boulevard Santa Clara, CA 95050-2554, U.S.A.
Tel: +1-408-588-6000, Fax: +1-408-588-6130

Renesas Electronics Canada Limited
1131  Nicholson Road, Mississauga, Ontario L5Y 9C3, Canada
Tel: +1-905-898-5441, Fax: +1-905-898-3220

Renesas Electronics Europe Limited
Dukes Meadow, Millboard Road, Bourne End, Buckinghamshire, SL8 5FH, U.K
Tel: +44-1628-585-100, Fax: +44-1628-585-900

Reneses Electronics Europe GmbH
Arcadialinie 10, 40472 Düsseldorf, Germany
Tel: +49-211-65030, Fax: +49-211-6503-130

Reneses Electronics (China) Co., Ltd.
7th Flor, Quantum Plaza, No.27 Zhongfuju Haidian District, Beijing 100083, P.R.China
Tel: +86-10-8235-1155, Fax: +86-10-8235-7678

Reneses Electronics (Shanghai) Co., Ltd.
Unit 204, 205, A2/A3 Culmer, No.1231 Luqiao Rd., Pudong District, Shanghai 200120, China
Tel: +86-21-5871-1616, Fax: +86-21-5871-7853 / -7838

Reneses Electronics Hong Kong Limited
Unit 1901/1903, 19/F., Tower 2, Grand Century Place, 193 Prince Edward Road West, Mongkok, Kowloon, Hong Kong
Tel: +852-2886-9318, Fax: +852-2886-9022/9044

Reneses Electronics Taiwan Co., Ltd.
11F, No. 265, Fu Shing North Road, Taipei, Taiwan
Tel: +886-2-8175-9600, Fax: +886-2-8175-9670

Reneses Electronics Singapore Pte. Ltd.
1 harbouFront Avenue, 406-10, leppel Bay Tower, Singapore 098632
Tel: +65-6213-0000, Fax: +65-6278-8001

Reneses Electronics Malaysia Sdn.Bhd.
Unit 908, Block B, Menara Amcorp, Amcorp Trade Centre, No. 18, Jln Persiaran Barat, 46050 Petaling Jaya, Selangor Darul Ehsan, Malaysia
Tel: +60-3-7955-9930, Fax: +60-3-7955-9931

Reneses Electronics Korea Co., Ltd.
11F., Samil Lanni or 99rd., 720-2 Yeoksam-Dong, Gangnam-Ku, Seoul 135-080, Korea
Tel: +82-2-558-5137, Fax: +82-2-558-5141

© 2012 Renesas Electronics Corporation. All rights reserved.
Colophon 2.0