To our customers,

---

**Old Company Name in Catalogs and Other Documents**

On April 1st, 2010, NEC Electronics Corporation merged with Renesas Technology Corporation, and Renesas Electronics Corporation took over all the business of both companies. Therefore, although the old company name remains in this document, it is a valid Renesas Electronics document. We appreciate your understanding.

Renesas Electronics website: [http://www.renesas.com](http://www.renesas.com)

---

April 1st, 2010
Renesas Electronics Corporation

---

Issued by: Renesas Electronics Corporation ([http://www.renesas.com](http://www.renesas.com))

Notice

1. All information included in this document is current as of the date this document is issued. Such information, however, is subject to change without any prior notice. Before purchasing or using any Renesas Electronics products listed herein, please confirm the latest product information with a Renesas Electronics sales office. Also, please pay regular and careful attention to additional and different information to be disclosed by Renesas Electronics such as that disclosed through our website.

2. 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.

3. You should not alter, modify, copy, or otherwise misappropriate any Renesas Electronics product, whether in whole or in part.

4. 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.

5. When exporting the 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. You should not use Renesas Electronics products or the technology described in this document for any purpose relating to military applications or use by the military, including but not limited to the development of weapons of mass destruction. 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.

6. 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.

7. Renesas Electronics products are classified according to the following three quality grades: “Standard”, “High Quality”, and “Specific”. The recommended applications for each Renesas Electronics product depends on the product’s quality grade, as indicated below. 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 categorized as “Specific” without the prior written consent of Renesas Electronics. Further, you may not use any Renesas Electronics product for any application for which it is not intended without the prior written consent of Renesas Electronics. 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 an application categorized as “Specific” or for which the product is not intended where you have failed to obtain the prior written consent of Renesas Electronics. The quality grade of each Renesas Electronics product is “Standard” unless otherwise expressly specified in a Renesas Electronics data sheets or data books, etc.

   “Standard”: Computers; office equipment; communications equipment; test and measurement equipment; audio and visual equipment; home electronic appliances; machine tools; personal electronic equipment; and industrial robots.

   “High Quality”: Transportation equipment (automobiles, trains, ships, etc.); traffic control systems; anti-disaster systems; anti-crime systems; safety equipment; and medical equipment not specifically designed for life support.

   “Specific”: Aircraft; aerospace equipment; submersible repeaters; nuclear reactor control systems; medical equipment or systems for life support (e.g. artificial life support devices or systems), surgical implantations, or healthcare intervention (e.g. excision, etc.), and any other applications or purposes that pose a direct threat to human life.

8. 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.

9. 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 them 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, fire control and malfunction prevention, appropriate treatment for aging degradation or any other appropriate measures. Because the evaluation of microcomputer software alone is very difficult, please evaluate the safety of the final products or system manufactured by you.

10. 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.

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.

(Note 1) “Renesas Electronics” as used in this document means Renesas Electronics Corporation and also includes its majority-owned subsidiaries.

(Note 2) “Renesas Electronics product(s)” means any product developed or manufactured by or for Renesas Electronics.
Flash Development Toolkit
Method for Using the User Program Mode (SH7086 Application)

Summary

This application note describes how to use the Flash Development Toolkit from Renesas, and explains how to use the SH7086 (SH family) user program mode using the Flash Development Toolkit, as follows:

• Boot mode 1 (write to the user boot area)
• Boot mode 2 (write to the user area)
• User boot mode
• User program mode

From the explanation given here, please understand the differences between the boot mode, user boot mode and user program mode, and learn how to use the user program mode.

In this application note, an explanation is made of a file created for use in user program mode. This file processes a write to and erase of the internal flash memory that is used in both user program mode and user boot mode in common. When writing to and erasing the flash memory in user program mode or user boot mode, refer to this file for user program mode.

In this document, we’ll use Flash Development Toolkit 4.01.
Table of Contents

1. SH7086 ................................................................................................................................. 3
   1.1 Flash Memory Configuration .......................................................................................... 3
   1.2 Operation Modes ............................................................................................................. 3
   1.3 On-board Programming Mode .......................................................................................... 4
2. Functionality of the Flash Development Toolkit ................................................................. 5
   2.1 Main Facilities .................................................................................................................. 5
3. Method for Using the Flash Development Toolkit in Each Mode ....................................... 6
   3.1 Connecting the E8a Emulator ......................................................................................... 7
   3.2 Setting Up the Flash Development Toolkit ....................................................................... 11
   3.3 Boot Mode 1 (Write to the User Boot Area) .................................................................. 24
   3.4 Boot Mode 2 (Write to the User Area) ........................................................................... 44
   3.5 User Boot Mode ............................................................................................................. 51
   3.6 User Program Mode ........................................................................................................... 67
4. Processes of the Flash Development Toolkit ........................................................................ 79
5. Files for User Program Mode ............................................................................................... 80
   5.1 Source File List ............................................................................................................... 80
   5.2 Module List ..................................................................................................................... 81
   5.3 Hierarchical Module Structure ....................................................................................... 82
   5.4 Program Flow ................................................................................................................. 85
6. Sources of the Files for User Program Mode ....................................................................... 88
   6.1 Header Files .................................................................................................................... 88
   6.2 Main Process and ROM Main Process ............................................................................ 92
   6.3 RAM Main Process .......................................................................................................... 93
7. Programming Guide ............................................................................................................ 99
   7.1 Functional Outline .......................................................................................................... 99
   7.2 Control Registers and Control Bits .............................................................................. 100
   7.3 Using the Libraries ......................................................................................................... 101
   7.4 List of Module Facilities ............................................................................................... 102
   7.5 Module Specifications .................................................................................................... 103
1. SH7086

1.1 Flash Memory Configuration

The flash memory of the SH7086 (SH family) has two types of memory mats: a user mat (user area) and a user boot mat (user boot area).

In addition to these, there is another area in which the flash memory write/erase control program is stored. This area is referred to as the boot mat (boot area). In this application note, they are referred to as the boot area, user area and user boot area, respectively.

<table>
<thead>
<tr>
<th>Area</th>
<th>Type</th>
<th>Size</th>
<th>Block(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>User area</td>
<td>Flash memory</td>
<td>512 Kbytes</td>
<td>16 blocks</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Eight 4-Kbyte blocks</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>One 32-Kbyte block</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Seven 64-Kbyte blocks</td>
</tr>
<tr>
<td>User boot area</td>
<td>Flash memory</td>
<td>8 Kbytes</td>
<td>1 block</td>
</tr>
<tr>
<td>Boot area</td>
<td>Control program</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

1.2 Operation Modes

The SH7086 has 8 kinds of operation modes (modes 0–7). Which mode is selected depends on how the FWE pin and the mode pins (MD1, MD0) are set.

Modes 0–2 are external extension modes in which external memory and peripheral devices can be accessed.

Modes 2, 5 and 6 permit an external address space to be set up from one area to another by the bus controller after program execution started.

Mode 3 is a single-chip startup extension mode in which the access to external memory or peripheral device can be switched over when program execution starts.

Modes 4–7 respectively are boot mode, user boot mode and user program mode in which the flash memory can be written to and erased.

Be sure that the FWE pin and the mode pins (MD1, MD0) do not change state while the LSI is in operation.

For details, see the hardware manual.

<table>
<thead>
<tr>
<th>Pin Setting</th>
<th>Bus Width of CS0 Space</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mode No.</td>
<td>FWE</td>
</tr>
<tr>
<td>Mode 0</td>
<td>0</td>
</tr>
<tr>
<td>Mode 1</td>
<td>0</td>
</tr>
<tr>
<td>Mode 2</td>
<td>0</td>
</tr>
<tr>
<td>Mode 3</td>
<td>0</td>
</tr>
<tr>
<td>Mode 4*</td>
<td>1</td>
</tr>
<tr>
<td>Mode 5*</td>
<td>1</td>
</tr>
<tr>
<td>Mode 6*</td>
<td>1</td>
</tr>
<tr>
<td>Mode 7*</td>
<td>1</td>
</tr>
</tbody>
</table>
1.3 On-board Programming Mode

There are three on-board programming modes: boot mode, user program mode and user boot mode.

<table>
<thead>
<tr>
<th>Item</th>
<th>Boot Mode</th>
<th>User Program Mode</th>
<th>User Boot Mode</th>
</tr>
</thead>
<tbody>
<tr>
<td>Operating mode</td>
<td>Mode 4</td>
<td>Mode 6</td>
<td>Mode 5</td>
</tr>
<tr>
<td>Function</td>
<td>This mode is a program mode that uses an on-chip SCI interface. The user area and user boot area can be programmed. This mode can automatically adjust the bit rate between the host and the LSI. All areas in the user area and user boot area are erased first.</td>
<td>The user area can be programmed by using a desired interface.</td>
<td>The user boot program of a desired interface can be created and the user area can be programmed.</td>
</tr>
<tr>
<td>Control program</td>
<td>Boot area (On-chip boot program)</td>
<td>User area (User-created user program)</td>
<td>User boot area (User-created user boot program)</td>
</tr>
<tr>
<td>Programming/erasing enable area</td>
<td>User area</td>
<td>User area</td>
<td>User area</td>
</tr>
<tr>
<td>All erase</td>
<td>✓ (Automatic)</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Block division erase</td>
<td>✓*1</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Program-data transfer</td>
<td>From the host via the SCI</td>
<td>From a desired device via RAM</td>
<td>From a desired device via RAM</td>
</tr>
<tr>
<td>Reset start</td>
<td>On-chip boot program storage area (Boot area)</td>
<td>User area</td>
<td>User boot area*2</td>
</tr>
<tr>
<td>Transition to user program mode</td>
<td>Changing mode setting and reset</td>
<td>Changing the FLSHE bit setting</td>
<td>Changing mode setting and reset</td>
</tr>
</tbody>
</table>

Notes:
1. All-erase is performed. After that, the specified block can be erased.
2. Firstly, the activation is made from the embedded program storage area. After the flash memory related registers are checked, the reset vector is fetched from the user boot area.

The user boot area can be written to and erased in only boot mode. In boot mode, the user area and user boot area are once erased in their entirety. After that, the user area or user boot area can be written to by issuing a command, but their contents cannot be read out up until this state.

The programming modes may be used in various ways. For example, you might want to write to the user boot area only and then rewrite the user area in user boot mode, or rewrite the user area only because the user boot mode is unused.

In user boot mode, a boot operation of any desired interface can be realized by setting up the mode pins differently than in user program mode.
2. Functionality of the Flash Development Toolkit

The Flash Development Toolkit is an on-board flash programming tool for Renesas flash microcomputers, featuring a highly functional, easy to use graphical user interface. The Flash Development Toolkit, when used in combination with the High-performance Embedded Workshop from Renesas, provides the developers of built-in software using Renesas flash microcomputers with an integrated development environment. Furthermore, the Flash Development Toolkit may also be used as an editor of general-purpose S record format or hexadecimal files.

2.1 Main Facilities

• Connection with a device: Connects a device to the Flash Development Toolkit interface.
• Disconnection of a device: Disconnects a device from the Flash Development Toolkit interface.
• Block erase: Erases a specific block or the entire block of a device’s flash memory from the Block Erase dialog box it opens.
• Blank check: Checks whether the flash part of the target device is blank or not.
• Upload: Uploads data from the target device.
• Target file download: Downloads the file active in a hexadecimal editor.
• Flash checksum: Returns a checksum of flash memory data.
• Flash area specification: Sets a flash area subject to non-programming operations (e.g., upload and blank check).
• Other: The Flash Development Toolkit has an easy to operate simple interface mode and a basic simple interface mode.

For details, see the “Renesas Flash Development Toolkit 4.01 User’s Manual.”
3. Method for Using the Flash Development Toolkit in Each Mode

For a write to the user boot area in user boot mode or a write to the user area in user program mode to be executed, there must be the load module file for user program mode (shared with user boot mode) written in the area concerned.

In this document, we first explain how to write the load module file for user program mode to the user boot area or user area in boot mode, and then explain how the user boot mode and user program mode works. The procedure is shown below.

Connecting the E8a emulator

Setting up the Flash Development Toolkit

For user boot mode

Boot mode 1
(Write to the user boot area)

User boot mode

For user program mode

Boot mode 2
(Write to the user area)

User program mode
3.1 Connecting the E8a Emulator

The E8a emulator, connected between the host computer and user system, has the facility to write the user application program to or erase the program written in a flash microcomputer’s internal flash memory on the user system (on-board) by using the Flash Development Toolkit.

The relationship between the pin numbers and pin names of the E8a emulator and the pin settings of the Flash Development Toolkit is shown below.

<table>
<thead>
<tr>
<th>Pin No.</th>
<th>E8a Pin Name</th>
<th>Pin Setting of Flash Development Toolkit</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>io0 / CLK</td>
<td>D</td>
</tr>
<tr>
<td>2</td>
<td>GND</td>
<td>-</td>
</tr>
<tr>
<td>3</td>
<td>io1</td>
<td>C</td>
</tr>
<tr>
<td>4</td>
<td>io2</td>
<td>A</td>
</tr>
<tr>
<td>5</td>
<td>RxD ( User side: TxD)</td>
<td>-</td>
</tr>
<tr>
<td>6</td>
<td>io3</td>
<td>E</td>
</tr>
<tr>
<td>7</td>
<td>io4/SIO</td>
<td>B</td>
</tr>
<tr>
<td>8</td>
<td>UVcc</td>
<td>-</td>
</tr>
<tr>
<td>9</td>
<td>UVcc2</td>
<td>-</td>
</tr>
<tr>
<td>10</td>
<td>io6</td>
<td>F</td>
</tr>
<tr>
<td>11</td>
<td>TxD ( User side: RxD)</td>
<td>-</td>
</tr>
<tr>
<td>12</td>
<td>GND</td>
<td>-</td>
</tr>
<tr>
<td>13</td>
<td>/RES</td>
<td>-</td>
</tr>
<tr>
<td>14</td>
<td>GND</td>
<td>-</td>
</tr>
</tbody>
</table>

[Note] Always be sure that the pins 2, 8, 12, 13 and 14 are connected.
An example of a connection between the E8a emulator and the SH7068 is shown below. The pullup and pulldown resistance values are given for reference purposes only. These values need to be evaluated in your system before they are actually used.

```
* 1: Made by Sumitomo 3M Ltd.
* 2: ○○ shows plated parts
* 3: Open-collector buffer
* 4: According to the operating mode, change it to be pulled down.
```
In this application note, we use the CPU board “HSB70865F” made by Hokuto Denshi Co., Ltd. as the SH7086 user system. For details, refer to the URL of Hokuto Denshi Co., which is given below.
http://www.hokutodenshi.co.jp/

The relationship between the pin numbers and the pin names of the HSB70865F is shown below.

<table>
<thead>
<tr>
<th>Pin No.</th>
<th>Pin name</th>
<th>Pin No.</th>
<th>Pin name</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>RES</td>
<td>2</td>
<td>GND</td>
</tr>
<tr>
<td>3</td>
<td>RWE</td>
<td>4</td>
<td>GND</td>
</tr>
<tr>
<td>5</td>
<td>MD0</td>
<td>6</td>
<td>GND</td>
</tr>
<tr>
<td>7</td>
<td>MD1</td>
<td>8</td>
<td>GND</td>
</tr>
<tr>
<td>9</td>
<td>I/O0</td>
<td>10</td>
<td>GND</td>
</tr>
<tr>
<td>11</td>
<td>I/O1</td>
<td>12</td>
<td>GND</td>
</tr>
<tr>
<td>13</td>
<td>I/O2</td>
<td>14</td>
<td>GND</td>
</tr>
<tr>
<td>15</td>
<td>TXD</td>
<td>16</td>
<td>GND</td>
</tr>
<tr>
<td>17</td>
<td>RXD</td>
<td>18</td>
<td>Vcc</td>
</tr>
<tr>
<td>19</td>
<td>SCK</td>
<td>20</td>
<td>Vcc</td>
</tr>
</tbody>
</table>
Since the E8a emulator and the HSB70865F connect to external devices via 14 pins and 20 pins, respectively, the 14 ↔ 20-pin conversion connector “FDM-E8a” made by Hokuto Denshi is used in this document. For details, refer to the URL of Hokuto Denshi Co., which is given below.
http://www.hokutodenshi.co.jp/
The pin conversion table of the FDM-E8a is shown below.

<table>
<thead>
<tr>
<th>20 pins (HSB70865F pin names)</th>
<th>14 pins (FDT pin settings)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 pin (RES)</td>
<td>13 pin</td>
</tr>
<tr>
<td>2.4,6,8,10,12,14,16pin (GND)</td>
<td>2,12,14pin</td>
</tr>
<tr>
<td>15 pin (TX)</td>
<td>5 pin</td>
</tr>
<tr>
<td>17 pin (RX)</td>
<td>11 pin</td>
</tr>
<tr>
<td>18 pin (Vcc)</td>
<td>8 pin</td>
</tr>
<tr>
<td>20 pin (Vcc)</td>
<td>8 pin</td>
</tr>
<tr>
<td>3pin (FWE)</td>
<td>3pin (C)</td>
</tr>
<tr>
<td>5pin (MD0)</td>
<td>4pin (A)</td>
</tr>
<tr>
<td>7pin (MD1)</td>
<td>6pin (E)</td>
</tr>
<tr>
<td>9pin (V/CC)</td>
<td>7pin (B)</td>
</tr>
<tr>
<td>11 pin (V/Cl)</td>
<td>10pin (F)</td>
</tr>
<tr>
<td>13 pin (V/Cl)</td>
<td>-</td>
</tr>
<tr>
<td>19 pin (SCK)</td>
<td>1pin (D)</td>
</tr>
</tbody>
</table>

To use the Flash Development Toolkit with the SH7086, it is necessary to set up the FWE pin and the MD1 and MD0 pins.
In this application note, we set pin C (FWE), pin E (MD1) and pin A (MD0) in the pin setting phase of the Flash Development Toolkit.
3.2 Setting Up the Flash Development Toolkit

Before writing a program to the flash memory, first set up the Flash Development Toolkit.

(1) Start the Flash Development Toolkit.
From All Programs, choose “Flash Development Toolkit 4.01.”

(2) The Welcome! screen of the Flash Development Toolkit will be displayed.
Select the “Create New Project Workspace” radio button and click OK.
From the next time on, because the previously selected device and port information are retained, select “Open Recently Used Project Workspace” and click OK.
(3) Set up a new project workspace.
First, specify a workspace name and a project name.
Here, use the same name to specify a workspace name and a project name.
Choose “Browse,” and from the ensuing list select the location in which you want to save the workspace.
When you’ve completed the dialog box, click OK.
(4) Select the target device.
Select “SH/7086F (Generic)” and click Next.
(5) Select a communication port.  
From the pulldown menu, choose “E8aDirect” and click Next.

(6) Set up power supply.  
Simply click OK leaving the Power Supply check box unselected.
(7) Set up the E8a pins for boot mode.
Set pin C (FWE) to 1 and pin E (MD1) and pin A (MD0) to 0, respectively and then click OK.

<table>
<thead>
<tr>
<th>Pin Setting</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mode No.</td>
</tr>
<tr>
<td>Mode 4</td>
</tr>
</tbody>
</table>

When a connection is complete, click OK.

.WARNING: Incorrect settings could damage your hardware.

FLASH Development Toolkit

FDT will now attempt to connect to your generic device.
Please ensure the board is connected, powered and in Boot mode.
(8) Check the device for confirmation.

Select the E8a and click OK.

Device confirmation is complete. Click OK.
Set up the device.
For Input Clock, enter the clock frequency used in the board in MHz units.
Enter 10.00 MHz.
Set 8 for Main Clock Multiply Factor and 4 for Peripheral Clock Multiply Factor, and then click Next.

The input clock here refers to the clock frequency that is directly fed to the microcomputer. Enter the frequency of the quartz crystal resonator or ceramic resonator connected to the user system. The input clock frequency and the operating clock frequency (PLL output) differ.
(10) Set up the type of connection. From the pulldown menu, set the baud rate. Select “625000” and click Next.

(11) Select write options. Select “Automatic” for Protection Level and “Advanced” for Output Message Level, and then click Next.
(12) Set the E8a pins as required when the device restarts in reset mode. Since there is no need to set here, simply click Finish.
(13) The SH7086 board will be connected to the Flash Development Toolkit in boot mode. At this time, the user boot area and user area have had their contents erased.
(14) Disconnect the device.
Choose “Disconnect from Device” from the Device menu.
The device will be disconnected.
(15) Save the workspace and quit. Choose “Exit” from the File menu.

Click Yes.

The Flash Development Toolkit will be closed. The workspace for the Flash Development Toolkit will be saved as 7086.WAS file.
3.3 Boot Mode 1 (Write to the User Boot Area)

Write the load module file for user program mode to the user boot area in boot mode. The program written to the user boot area is the 7086F.mot file (S type file). Here, use the saved workspace file (7086.AWS) to start. This program already has the bit rate in it corrected according to the clock frequency. For details about bit rate correction, see paragraph (1), “Setting the bit rate (GenTest.h)” in Section 6.1, “Header File.”

(1) Launch the Flash Development Toolkit. From All Programs, choose “Flash Development Toolkit 4.01.”

(2) The Welcome! screen of the Flash Development Toolkit will be displayed. Select the “Open Recently Used Project Workspace” radio button and then the project workspace file “7086.AWS,” and click OK.
The 7086 project will be displayed.

The Flash Development Toolkit can also be launched directly by opening the project workspace file “7086.AWS” (by double-clicking on it).
(3) Choose “Connect to Device” from the Device menu.

Simply click OK leaving the Power Supply check box unselected.
Select “E8a” and click OK.

The device will be connected.
(4) Select the file to write.
Choose "Add File" from the Project menu.
(5) In the Add File dialog box, select the “7086F.mot” file and click Add.

The “7086F.mot” file will be added to the project.
(6) To verify that the user boot area and user area have no data written in each, perform a blank check. Choose “Blank Check” from the Device menu.
The result of a blank check will be displayed. It shows that the user boot area and user area have no data written in each.
(7) Verify a checksum value of the user boot area and user area when these areas have no data written in each.

Choose “Flash Checksum” from the Device menu.
The result of a checksum will be displayed.
(8) Set the user boot area as a preparatory step before writing to that area.
Right-click the 7086F.mot file, and from the popup menu choose “User Boot Flash” to set the user boot area for the download destination.
(9) Right-click the 7086F.mot file again, and from the popup menu choose “Download [User Boot Flash]” to download the 7086F.mot file into the user boot area.
The program has been downloaded into the user boot area.
(10) To verify that the program has been written to the user boot area, perform a blank check and a checksum.

Choose “Blank Check” from the Device menu.
Choose “Flash Checksum” from the Device menu.
The results of a blank check and a checksum will be displayed.

The result shows that the program has been written to the user boot area.
(11) Choose “Disconnect from Device” from the Device menu to disconnect the device.
(12) Delete a file.
Choose “Delete Files” from the Project menu.
(13) The Delete Project Files dialog box will be displayed. Click the Delete All button.

Click OK.
The files will be deleted.
(14) Delete a folder.
Right-click a folder, and from the popup menu choose “Delete Folder.”
The folder will be deleted.
3.4 Boot Mode 2 (Write to the User Area)

Write the load module file for user program mode to the user area in boot mode. The program written to the user area is the same file that was used in Section 3.3, “Boot Mode 1 (Write to the User Boot Area).”

(1) Launch Flash Development Toolkit 4.01, open the project workspace file “7086.AWS,” and connect to the device.
(2) Select the file to write.
Choose “Add File” from the Project menu.

In the Add File dialog box, select the “7086F.mot” file and click Add.
The “7086F.mot” file will be added to the project.
(3) Before writing the 7086F.mot file to the user area, perform a blank check and a checksum. Choose “Blank Check” from the Device menu. Choose “Flash Checksum” from the Device menu. The results of a blank check and a checksum will be displayed.
(4) Right-click the 7086F.mot file, and from the popup menu choose “Download [User Area]” to download the 7086F.mot file into the user area. The default is “Download [User Area].”
The program has been downloaded into the user area.
(5) To verify that the program has been written to the user area, perform a blank check and a checksum. Choose “Blank Check” from the Device menu. Choose “Flash Checksum” from the Device menu. The results of a blank check and a checksum will be displayed.

The result shows that the program has been written to the user area.
3.5 User Boot Mode

In user boot mode, it is possible to write to and erase the user area. This mode does not allow writing to and erasing the user boot area. Therefore, it is necessary that the load module file for user program mode be written in the user boot area beforehand.

(1) Launch Flash Development Toolkit 4.01, open the project workspace file “7086.AWS,” and write the 7086F.mot file to the user boot area in boot mode.

![Flash Development Toolkit](image)

Writing image to device... [0x00000000 - 0x0000007F]
Writing image to device... [0x00000100 - 0x000001FF]
Data programmed at the following positions:
- 0x00000000 - 0x0000007F, Length: 0x00000080
- 0x00001000 - 0x00001FFF, Length: 0x00000100

4.13 X programmed in 1 seconds
Image successfully written to device
(2) Choose “Disconnect from Device” from the Device menu to disconnect the device.
(3) Choose “Project Setting” from the Device menu.
The project setting window will be displayed.
(4) Set up the user boot mode. Select the Device tab of the project setup window and double-click the “Interface Direct Connection” line.
Click Next.

The Flash Development Toolkit supports connection through the standard PC Serial port and the USB port. Use this page to select your desired communications port. All settings may be changed after the project is created.

Select port: FooDirect

Select an interface type to connect to the target device with. Normally this will be "Direct Connection" or simply left blank.

Select Interface:

Click Next.

Please enter the specific device options based on:

[R5F708E] using [Protocol C]

Select the external clock or the internal clock:

Enter the CPU crystal frequency for the selected device:

Enter the clock mode for the selected device:

Select the multiplier for the Main clock frequency (CKM):

Select the multiplier for the Peripheral clock frequency (CKP):

Click Next.
In the Select Connection column, select “USER Program Mode” and then click Next.
Set up the E8a pins for user boot mode. Set pin C (FWE) and pin A (MD0) to 1 and pin E (MD1) to 0, respectively and then click OK.
Click the Finish button.
The user boot mode has been set up.
(5) Choose “Connect to Device” from the Device menu to connect the device.
(6) Add the program to write to the user area. Here, write the test_mot file to the user area.
(7) Before writing the test_mot file to the user area, perform a blank check and a checksum. Choose “Blank Check” from the Device menu. Choose “Flash Checksum” from the Device menu. The results of a blank check and a checksum will be displayed.

![Flash Development Toolkit](image-url)
(8) Write to the user area in user boot mode.
Right-click the test_mot file, and from the popup menu choose “Download [User Area]” to download the test_mot file into the user area.
The program has been downloaded into the user area.
(9) To verify that the program has been written to the user area, perform a blank check and a checksum. Choose “Blank Check” from the Device menu. Choose “Flash Checksum” from the Device menu. The results of a blank check and a checksum will be displayed.

The result shows that the program has been written to the user area.
3.6 User Program Mode

In user program mode, it is possible to write to and erase the user area. Here, write the test.mot file to the user area the same way as in Section 3.5, "User Boot Mode." This mode does not allow writing to and erasing the user boot area. Therefore, it is necessary that the load module file for user program mode be written in the user area beforehand.

(1) Launch Flash Development Toolkit 4.01, open the project workspace file “7086.AWS,” and write the 7086F.mot file to the user area in boot mode. After writing to the user area, disconnect the device and then choose “Project Setting” from the Device menu to display the project setting window.
(2) Set up the user program mode. Select the Device tab of the project setup window and double-click the “Interface Direct Connection” line.
Click Next.

Click Next.

Click Next.
In the Select Connection column, select “USER Program Mode” and then click Next.
Set up the E8a pins for user program mode.
Here, set for mode 7.
Set pin C (FWE), pin E (MD1) and pin A (MD0) to 1, and then click OK.

<table>
<thead>
<tr>
<th>Mode No.</th>
<th>FWE</th>
<th>MD1</th>
<th>MD0</th>
<th>Mode Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mode 6</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>User programming</td>
</tr>
<tr>
<td>Mode 7</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>mode</td>
</tr>
</tbody>
</table>

![Pin Settings](image)
Click the Finish button.
The user program mode has been set up.
(3) Choose “Connect to Device” from the Device menu to connect the device.
(4) Before writing the test_mot file to the user area, perform a blank check and a checksum. Choose “Blank Check” from the Device menu. Choose “Flash Checksum” from the Device menu. The results of a blank check and a checksum will be displayed.
(5) Write to the user area in user program mode.
Right-click the test_mot file, and from the popup menu choose “Download [User Area]” to download the test_mot file into the user area.
The program has been downloaded into the user area.
To verify that the program has been written to the user area, perform a blank check and a checksum. Choose “Blank Check” from the Device menu. Choose “Flash Checksum” from the Device menu. The results of a blank check and a checksum will be displayed.

The result shows that the program has been written to the user area.
4. Processes of the Flash Development Toolkit

The Flash Development Toolkit has three modes in which it connects to the device: Boot mode, user boot mode and user program mode. In each mode, it is possible to specify a continuance of execution from the preceding session. Normally, use a new connection process. The codes expressed in hexadecimal are the command codes of the Flash Development Toolkit. For details, see Section 2.3, “Flash Memory,” of the hardware manual.

<table>
<thead>
<tr>
<th>Mode</th>
<th>New Connection Processing</th>
<th>Continuation of the Execution from a Previous Session</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boot mode</td>
<td>Baud rate adjustment</td>
<td>H'27 (Programming unit inquiry)</td>
</tr>
<tr>
<td></td>
<td>H'13 (Device selection)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>H'11 (Clock mode selection)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>H'3F (New baud rate setting)</td>
<td></td>
</tr>
<tr>
<td>User boot mode</td>
<td>H'27 (Programming unit inquiry)</td>
<td></td>
</tr>
<tr>
<td>User program mode</td>
<td>H'13 (Device selection)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>H'11 (Clock mode selection)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>H'3F (New baud rate setting)</td>
<td></td>
</tr>
</tbody>
</table>
5. Files for User Program Mode
This section describes the files for user program mode.

5.1 Source File List
The source files are listed below.

<table>
<thead>
<tr>
<th>File</th>
<th>File Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Baud rate</td>
<td>BaudRate.src</td>
<td>BRR calculation assembly language file</td>
</tr>
<tr>
<td>Command function</td>
<td>CmdFunc.c</td>
<td>Command processing source file</td>
</tr>
<tr>
<td>Command function header</td>
<td>CmdFunc.h</td>
<td>Command function definition file</td>
</tr>
<tr>
<td>Command header</td>
<td>commands.h</td>
<td>Command code definition file</td>
</tr>
<tr>
<td>Device information header</td>
<td>Deviceinfo.h</td>
<td>Device information definition file</td>
</tr>
<tr>
<td>Erase function</td>
<td>FDTErase.c</td>
<td>Erase function source file</td>
</tr>
<tr>
<td>Main function</td>
<td>FDTUMain.c</td>
<td>Main kernel function source file</td>
</tr>
<tr>
<td>Main function header</td>
<td>FDTUMain.h</td>
<td>Main kernel function definition file</td>
</tr>
<tr>
<td>Programming function</td>
<td>FDTWrite.c</td>
<td>Programming function source file</td>
</tr>
<tr>
<td>Test function</td>
<td>GenTest.c</td>
<td>User program mode test function source file</td>
</tr>
<tr>
<td>Test function header</td>
<td>GenTest.h</td>
<td>User program mode test definition file</td>
</tr>
<tr>
<td>Library header</td>
<td>io78055.h</td>
<td>Peripheral module register definition file</td>
</tr>
<tr>
<td>Library header</td>
<td>KAlg.h</td>
<td>Programming and erasing library definition file</td>
</tr>
<tr>
<td>Device header</td>
<td>KDevice.h</td>
<td>Device information definition file</td>
</tr>
<tr>
<td>Structure header</td>
<td>KStruct.h</td>
<td>Structure definition file</td>
</tr>
<tr>
<td>Type header</td>
<td>KTypes.h</td>
<td>Type definition file</td>
</tr>
<tr>
<td>RAM address definition</td>
<td>rom2ram.src</td>
<td>RAM address definition file</td>
</tr>
<tr>
<td>Start function</td>
<td>Start006.src</td>
<td>Start function assembly language file</td>
</tr>
<tr>
<td>Micro function</td>
<td>uGenu.c</td>
<td>Micro kernel function source file</td>
</tr>
<tr>
<td>Micro function header</td>
<td>uGenu.h</td>
<td>Micro kernel definition file</td>
</tr>
</tbody>
</table>
### 5.2 Module List

The modules are listed below.

<table>
<thead>
<tr>
<th>File</th>
<th>Module</th>
<th>Module Name</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>CmdFunc.c</td>
<td>Reference function</td>
<td>ReferFunc</td>
<td>Reference function</td>
</tr>
<tr>
<td></td>
<td>Device selection</td>
<td>SelectDevice</td>
<td>Selects a device</td>
</tr>
<tr>
<td></td>
<td>Clock mode selection</td>
<td>SelectClockMode</td>
<td>Selects a clock mode</td>
</tr>
<tr>
<td></td>
<td>New baud rate setting</td>
<td>SetNewBaudRate</td>
<td>Sets a new baud rate</td>
</tr>
<tr>
<td></td>
<td>Program status</td>
<td>RequestBootPigSts</td>
<td>Program status</td>
</tr>
<tr>
<td></td>
<td>Sum check</td>
<td>SumCheck</td>
<td>Sum check</td>
</tr>
<tr>
<td></td>
<td>ACK transmission</td>
<td>SendAck</td>
<td>Sends ACK</td>
</tr>
<tr>
<td></td>
<td>Blank check</td>
<td>CheckBlank</td>
<td>Checks the blank status</td>
</tr>
<tr>
<td></td>
<td>Memory read</td>
<td>ReadMemory</td>
<td>Reads memory</td>
</tr>
<tr>
<td></td>
<td>Command read</td>
<td>GetCmdData</td>
<td>Reads a command</td>
</tr>
<tr>
<td></td>
<td>FDT_erase.c</td>
<td>EraseFLASH</td>
<td>Erases flash memory</td>
</tr>
<tr>
<td></td>
<td>Erase data reception</td>
<td>GetEraseData</td>
<td>Receives erase data</td>
</tr>
<tr>
<td></td>
<td>Erase initial setting</td>
<td>EraseInit</td>
<td>Performs erase initial setting</td>
</tr>
<tr>
<td></td>
<td>Erasing start</td>
<td>EraseStart</td>
<td>Starts erasing operation</td>
</tr>
<tr>
<td></td>
<td>FDT_Main.c</td>
<td>RAMMain</td>
<td>RAM main processing</td>
</tr>
<tr>
<td></td>
<td>Command processing</td>
<td>ProcessCommand</td>
<td>Processes commands</td>
</tr>
<tr>
<td></td>
<td>Library transfer</td>
<td>LIBTrans</td>
<td>Transfers a library</td>
</tr>
<tr>
<td></td>
<td>SCO bit setting</td>
<td>SocBitSet</td>
<td>Sets the SCO bit</td>
</tr>
<tr>
<td></td>
<td>User boot area selection</td>
<td>UserBootSelect</td>
<td>Selects the user boot area</td>
</tr>
<tr>
<td></td>
<td>User area selection</td>
<td>UserMatSelect</td>
<td>Selects the user area</td>
</tr>
<tr>
<td></td>
<td>FDT_Write.c</td>
<td>WriteFLASH</td>
<td>Programs flash memory</td>
</tr>
<tr>
<td></td>
<td>Programming data reception</td>
<td>GetWriteData</td>
<td>Receives programming data</td>
</tr>
<tr>
<td></td>
<td>Programming initial setting</td>
<td>WriteInit</td>
<td>Performs programming initial setting</td>
</tr>
<tr>
<td></td>
<td>Programming start</td>
<td>WriteStart</td>
<td>Starts programming</td>
</tr>
<tr>
<td></td>
<td>GenTest.c</td>
<td>main</td>
<td>Tests main processing</td>
</tr>
<tr>
<td></td>
<td>SCI initial setting</td>
<td>InitSCI</td>
<td>Performs SCI initial setting</td>
</tr>
<tr>
<td></td>
<td>Reception</td>
<td>Get</td>
<td>Reception</td>
</tr>
<tr>
<td></td>
<td>Transmission</td>
<td>Put</td>
<td>Transmission</td>
</tr>
<tr>
<td></td>
<td>Str7086.src</td>
<td>startup</td>
<td>Sets and starts the stack pointer</td>
</tr>
<tr>
<td></td>
<td>Ugenu.c</td>
<td>ROMMain</td>
<td>ROM main processing</td>
</tr>
<tr>
<td></td>
<td>Command function</td>
<td>CmdFunc</td>
<td>Receives and controls commands</td>
</tr>
<tr>
<td></td>
<td>Transfer start</td>
<td>TransStart</td>
<td>Starts transferring a program</td>
</tr>
<tr>
<td></td>
<td>Copy</td>
<td>RamCopy</td>
<td>Copies a program into RAM</td>
</tr>
</tbody>
</table>
5.3 Hierarchical Module Structure

The hierarchical structure of the modules is shown below.

```
RESET_VECTOR Reset vector
|---startup Start
|   |---main Main processing
|   |   |---InitSCI SCI initial setting
|   |   |---RomMain ROM main processing
|   |   |---TransStart Transfer start
|   |   |   |---RamCopy Copy
|   |   |---CmdFunc Command function
|   |   |   |---Get Reception
|   |   |---SendAck ACK transmission
|   |   |---Put Transmission
|   |   |---RefFunc Reference function
|   |   |---Put Transmission
|   |---GetCmdData Command read
|   |---Get Reception
|---SelectDevice Device selection
|   |---SendAck ACK transmission
|   |---ErrorCode Error code macro
|   |---Put Transmission
|---SelectClockMode Clock mode selection
|   |---SendAck ACK transmission
|   |---ErrorCode Error code macro
|   |---Put Transmission
|   |---SetNewBaudRate New bit rate setting
|   |   |---ErrorCode Error code macro
|   |---Put Transmission
|   |---Calc_brr BRR calculation
|   |---SendAcki ACK transmission
|   |---Get Reception
|---RequestBootPrjigSts Program status
|   |---Put Transmission
|   |---Put Transmission
|---RamMain RAM main processing
```

(To be continued)
(Continued)

- RamMain  RAM main processing
  - ProcessCommand  Command processing
    - Get  Reception
    - RequestBootProgSts  Program status
    - SumCheck  Sum check
      - UserBootSelect  User boot area selection
        - nop  NOP macro
      - UserMatSelect  User area selection
        - nop  NOP macro
    - Put  Transmission
    - LibTrans  Library transfer
      - SocBitSet  SCO bit setting
        - nop  NOP macro
    - SendAck  ACK transmission
    - EraseFLASH  Flash erasing
      - EraseInit  Erase initial setting
      - UserMatSelect  User area selection
      - INIT_ADDR  Initial setting entry address
      - ErrorCode  Error code macro
    - Put  Transmission
    - Get  Reception
    - RequestBootProgSts  Program status
    - GetEraseData  Erase data reception
    - Get  Reception
    - ErrorCode  Error code macro
    - Put  Transmission
    - EraseStart  Erasing start
      - WRITE_ERASE_ADDR  Programming/erasing entry address
    - SendAck  ACK transmission
    - WriteFLASH  Flash programming
      - WriteInit  Programming initial setting
      - UserMatSelect  User area selection
      - INIT_ADDR  Initial setting entry address
      - ErrorCode  Error code macro
    - Put  Transmission
    - Get  Reception
    - RequestBootProgSts  Program status
    - GetWriteData  Programming data reception
    - Get  Reception
    - ErrorCode  Error code macro
    - Put  Transmission
    - WriteStart  Programming start
      - WRITE_ERASE_ADDR  Programming/erasing entry address
    - SendAck  ACK transmission
    - GetCmdData  Command read
    - ReadMemory  Memory read

(To be continued)
(Continued)

- ReadMemory  Memory read
- UserBootSelect  User boot area selection
- UserMemSelect  User area selection
- ErrorCode  Error code macro
- Put  Transmission
- CheckBlank  Blank check
- UserBootSelect  User boot area selection
- UserMemSelect  User area selection
- ErrorCode  Error code macro
- Put  Transmission
- SendAck  ACK transmission
- Put  Transmission
5.4 Program Flow

This section describes a program flow referring to the hierarchical module structure.

(1) Program processing flow

The processing flow of the program is shown below. In user program mode, the bit rate synchronization and user area erase process normally executed in boot mode are not performed. Therefore, the program and data written into the flash memory can be saved.
(2) Main process (main)
The flow of the main process is shown below.
1. Branch from the reset vector to start (startup).
2. Start (startup) sets the stack pointer and calls the main process (main).
3. The main process (main) calls SCI initialization (InitSCI) and branches to ROM main process (RomMain).
4. The ROM main process (RomMain) transfers RAM main process to the RAM and then accepts and processes commands and sets specification.
When setting is complete, it branches to RAM main process (RamMain) in the RAM.
5. The RAM main process (RamMain) processes received commands to execute the processing listed below.
- Write/erase library transfer (LibTrans)
- Erase flash memory (EraseFLASH)
- Write to flash memory (WriteFLASH)
- User boot area/user area memory read (ReadMemory)
- User boot area/user area sum check (SumCheck)
- User boot area/user area blank check (CheckBlank)
[Note] The ROM main process (RomMain) is also referred to as the micro kernel. It operates in the ROM.
The RAM main process (RamMain) is also referred to as the main kernel. It operates in the RAM.

(3) ROM main process (RomMain)
The flow of the ROM main process (RomMain) is shown below.
1. In transfer start (TransStart), transfer a program from the ROM where it is stored to the RAM.
   This is to enable library transfers and write/erases to be processed in the RAM.
2. In command function (CmdFunc), process commands and set selection corresponding to inquiries.
3. Inquiries are handled in the reference function (ReferFunc) and program status (RequestBootPrgSts), with the following commands supported.
   - Support device inquiry
   - Clock mode inquiry
   - Multiply factor inquiry
   - Operating clock frequency inquiry
   - User boot area information inquiry
   - User area information inquiry
   - Erase block information inquiry
   - Write size inquiry
   - Boot program status inquiry
4. Selection settings are made in the following modules.
   - Device selection (SelectDevice): Selects device code
   - Clock mode selection (SelectClockMode): Notifies selected clock mode
   - New baud rate setting (SetNewBaudRate): Selects new baud rate
5. When inquiry selection is complete, branch to the RAM main process (RamMain) that was transferred to the RAM.
(4) RAM main process (RamMain)

The flow of the RAM main process (RamMain) is shown below.

1. In command process (ProcessCommand), process the commands. The processed commands are listed below.
   Since the files for user program mode are run in user programming mode, selecting a write to the user boot area and erasing blocks of the user boot area cannot be performed.
   - User area write selection
   - 128 byte write
   - Erase selection
   - Block erase
   - Memory read
   - User boot area sum check
   - User area sum check
   - User boot area blank check
   - User area blank check
   - Boot program status inquiry

2. In the user area write select command, transfer a write library by library transfer (LibTrans) and then branch to flash write (WriteFLASH).

3. In flash write (WriteFLASH), set a clock frequency by write initialization (WriteInit).
   Next, read in a command and if it's a 128 byte write, receive the write data by write data reception (GetWriteData) and write it to the flash memory by write start (EraseStart).
   If the 128-byte data is the one at the end of write address, set an end of write code in the data to write and at the address to which to write, and then terminate a write (actually by calling write start (EraseStart)) to complete the write process.

4. In the erase select command, transfer an erase library by library transfer (LibTrans) and then branch to flash erase (EraseFLASH).

5. In flash erase (EraseFLASH), set a clock frequency by erase initialization (EraseInit).
   Next, read in a command and if it's a block erase, receive the erase data by erase data reception (GetEraseData) and erase a specified block by erase start (EraseStart).
   If the received erase data is an end of erase data, terminate the erase process.

6. In the memory read command, the address from which to read is specified by command read (GetCmdData). In memory read (ReadMemory), read memory contents from the user boot area and user area.

7. In the user boot area sum check and user area sum check commands, check the user boot area and user area for data integrity by sum check (SumCheck).

8. In the user boot area blank check and user area blank check commands, check the user boot area and user area for blank blocks by blank check (CheckBlank)

9. In the boot program status inquiry command, transmit the processing status of boot by program status (RequestBootPrgSts).
6. Sources of the Files for User Program Mode
The main sources of the files for user program mode are shown below.

6.1 Header Files
The files for user program mode use the header files listed below.

(1) Bit rate setting (GenTest.h)
Sets a bit rate.

```c
/* RATE:9600/bps CLOCK:10MHz Main:x2 Peripheral:x2 */
#define MA_BRR_SCI 0x40 /* Bit rate register channel 1 */
```

The user program mode is connected at 9,600 bps. For this reason, it is necessary that the bit rate register (BRR) value of the CSI module be set according to the operating clock frequency used. Here, since the clock frequency is 20 MHz (10 MHz × 2), MA_BRR_SCI is set to 64 (0x40) to obtain a bit rate of 9,600 bps. The relationship between the operating clock frequency and the set value of the BRR register when the bit rate = 9,600 bps is shown below.

<table>
<thead>
<tr>
<th>Clock frequency ω (MHz)</th>
<th>BRR setting</th>
<th>Error (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>32</td>
<td>-1.36</td>
</tr>
<tr>
<td>12</td>
<td>38</td>
<td>0.16</td>
</tr>
<tr>
<td>14</td>
<td>45</td>
<td>-0.93</td>
</tr>
<tr>
<td>16</td>
<td>51</td>
<td>0.16</td>
</tr>
<tr>
<td>18</td>
<td>58</td>
<td>-0.59</td>
</tr>
<tr>
<td>20</td>
<td>64</td>
<td>0.16</td>
</tr>
<tr>
<td>22</td>
<td>71</td>
<td>-0.54</td>
</tr>
<tr>
<td>24</td>
<td>77</td>
<td>0.16</td>
</tr>
<tr>
<td>26</td>
<td>84</td>
<td>-0.43</td>
</tr>
<tr>
<td>28</td>
<td>90</td>
<td>0.16</td>
</tr>
<tr>
<td>30</td>
<td>97</td>
<td>-0.35</td>
</tr>
<tr>
<td>32</td>
<td>103</td>
<td>0.16</td>
</tr>
<tr>
<td>34</td>
<td>110</td>
<td>-0.29</td>
</tr>
<tr>
<td>36</td>
<td>116</td>
<td>0.16</td>
</tr>
<tr>
<td>38</td>
<td>123</td>
<td>-0.24</td>
</tr>
<tr>
<td>40</td>
<td>129</td>
<td>0.16</td>
</tr>
</tbody>
</table>

After setting the value of MA_BRR_SCI appropriately to suit the board’s operating clock frequency, build the source in the HEW to create an S type file of program.
(2) IO register definition (io7086.h)
Defines the SCI module and ROM related registers and bits and the PFC register.

```c
/** SH/7086F Internal I/O Include File */
/****************************************************************************/
/** SCI 
/**************************************************************************/
/** CHANNEL 1 
**************************************************************************/
#define SCI_SMR  (*(volatile unsigned char *)0xFFFFC080)
#define SCI_BRR  (*(volatile unsigned char *)0xFFFFC082)
#define SCI_SCR  (*(volatile unsigned char *)0xFFFFC084)
#define SCI_TDR  (*(volatile unsigned char *)0xFFFFC086)
#define SCI_SSR  (*(volatile unsigned char *)0xFFFFC088)
#define SCI_RDR  (*(volatile unsigned char *)0xFFFFC08A)
#define TE_RE   (unsigned char)(TE | RE)
#define TE     (unsigned char)0x20
#define RE     (unsigned char)0x10
#define TDRE   (unsigned char)0x80
#define RDRF   (unsigned char)0x40
#define RDRF_ERR_CLR (unsigned char)0x87
#define TEND   (unsigned char)0x04

#define FCCS  (*(volatile unsigned char *)0xFFFFCC00)
#define FPCS  (*(volatile unsigned char *)0xFFFFCC01)
#define FECS  (*(volatile unsigned char *)0xFFFFCC02)
#define FKEY  (*(volatile unsigned char *)0xFFFFCC04)
#define FMATS  (*(volatile unsigned char *)0xFFFFCC05)
#define FTDAR  (*(volatile unsigned char *)0xFFFFCC06)

#define PACRL2   (*(volatile unsigned short *)0xFFFFD114)
#define PA4MD0   (unsigned short)0x0001
#define PACRL1   (*(volatile unsigned short *)0xFFFFD116)
#define PA3MD0   (unsigned short)0x0001
```

Flash Development Toolkit
Method for Using the User Program Mode (SH7086 Application)
```c
#define FRQCR  (*(volatile unsigned short *)0xFFFFE800)
#define STBCR3  (*(volatile unsigned char *)0xFFFFE806)
#define MSTP12  (unsigned char)0x10
```
(3) Macro definitions (FDTUMain.h, KAlg.h)
Defines the labels used in the program.

- FDTUMain.h

```c
/* DEFINE */
enum {
    FmatsUserBootMat = 0xaa,
    FmatsUserMat = 0x00,
    WriteMode = 0x01,
    EraseMode = 0x01,
    FkeyEnable = 0xA5
};
```

- KAlg.h

```c
/* DEFINES */
#define LOOP_END      1
#define bufSize      0x80
#define BLOCK_NO_ERROR     0x09
#define ERASE_END      0xFF
#define WRITE_END      0xFFFFFFFF
#define ADDRESS_ERROR    0x03
#define WRITE_ERASE_ENABLE    0x5A
```
6.2 Main Process and ROM Main Process

(1) Hierarchical module structure
The hierarchical structure of the main process and ROM main process modules is shown below.

```
RESET_VECTOR  Reset vector
|  __________ Start
|     |  ____________ Main processing
|     |     |  ________ InitSCI  SCI initial setting
|     |     |  ________ RomMain  ROM main processing
|     |     |  ________ TransStart  Transfer start
|     |     |  ________ CmdFunc  Command function
|     |     |  ________ RamMain  RAM main processing
```

Branch from the reset vector to start, set the stack pointer (Strt7086.src) and branch to the main process (GenTest.c, main).
In the main process, initialize the SCI (GetTest.c, InitSCI) to make it capable of transmission/reception, and branch to ROM main process (Ugenu.c, RomMain).
In the ROM main process, transfer RAM main process, etc. to the RAM (Ugenu.c, TransStart) and process the commands (Ugenu.c, CmdFunc). At end of data, branch to RAM main process (FDTUMain.c, RamMain).
The main process and the ROM main process are executed in the ROM.

(2) Reset vector (GenTest.c, GenTest.h)
The reset vector is shown below.

- GenTest.c

```c
/*Declare the vector table*/
#pragma section _VECT
const DWORD RESET_VECTOR = (DWORD)RESET_JMP_ADDRESS;
#pragma section
```

- GenTest.h

```c
#define RESET_JMP_ADDRESS 0x1000
```

(3) Transfer start (Ugenu.c, rom2ram.src)
In the transfer of RAM main process, etc., transfer the modules listed below from ROM to RAM according to the transfer table (rom2ram.src). The sections use the ROM option.

<table>
<thead>
<tr>
<th>Section</th>
<th>Module</th>
</tr>
</thead>
<tbody>
<tr>
<td>P_RAM_SCI</td>
<td>Get, Put (GenTest.c)</td>
</tr>
<tr>
<td>P_RAM_MAIN</td>
<td>RomMain and others (FDTUMain.c)</td>
</tr>
<tr>
<td>P_RAM_CMD</td>
<td>RequestBootProSts and others (CmdFunc.c)</td>
</tr>
<tr>
<td>P_RAM_WRITE</td>
<td>WriteFLASH and others (FDTWrite.c)</td>
</tr>
<tr>
<td>P_RAM_ERASE</td>
<td>EraseFLASH (FDTERase.c)</td>
</tr>
</tbody>
</table>

(4) Command functions (Ugenu.c, commands.h, CmdFunc.c, DeviceInfo.h)
The command function (CmdFunc) processes inquiries and the set commands. The commands are macro-defined (commands.h), and the processes (CmdFunc.c) corresponding to the respective commands are executed. For inquiry commands, the processes (CmdFunc.c, ReferFunc) that output the responses (DeviceInfo.h) corresponding to the commands are executed.
6.3 RAM Main Process

The RAM main process involves transferring a library, erasing the flash memory, and writing to the flash memory. These processes are executed in the RAM.

(1) Library transfer (FDTUMain.c)

- LibTrans
  If commandID is prepareERASE (0x48), set FECS to EraseMode (0x01) and select the erase library. Otherwise (prepareUserAreaWrite, 0x43), set FPCS to WriteMode (0x01) and select the write library. Set FKEY to FkeyEnable (0xA5) to select Transfer and then set the SCO bit.

```c
/*
 */
/*/ LibTrans Function /*

void LibTrans(BYTE commandID)
{
    if (commandID == prepareErase){
        FECS = EraseMode;
    } else{
        FPCS = WriteMode;
    }

    FKEY = FkeyEnable;
    ScoBitSet();
}
```
• ScoBitSet
Set the library transfer destination address in the FTDAR register, set the VBR register to 0x84000000, and set the SCO bit of the FCCS register to 1. There must be 4 or more of NOP instructions after the SCO bit setting.
To determine whether an error occurred during a transfer, write 0xFF to the first byte of library transfer destination address before performing a transfer and check that the address value is 0x00 after the transfer.

```c
/*
   // ScoBitSet Function //
   */

BYTE ScoBitSet(void)
{
    volatile BYTE i;
    WORD work;
    void **save_vbr;

    /* Transmission error check initialization */
    *((volatile BYTE *)TRANS_RAM_ADDR) = 0xFF;

    FTDAR = FTDAR_VALUE;
    save_vbr = (void **)get_vbr();
    set_vbr((void **)0x84000000);
    work = FRQCR;
    FRQCR = 0x36DB;
    FCCS |= 0x01;  /* SCO interruption */

    /* for(i=0; i < 2; i++);*/
    nop();
    nop();
    nop();
    nop();

    set_vbr(save_vbr);
    FRQCR = work;

    /* Transmission error check */
    if(0x00 == *((volatile BYTE *)TRANS_RAM_ADDR)) {
        return(NORMAL); /* Transmission normal end */
    }

    return(ABNORMAL);  /* Transmission error */
}
```

TRANS_RAM_ADDR and FTDAR_VALUE are defined in KDevice.h as follows:

```c
/* SCO define */
#define TRANS_RAM_ADDR 0xFFFF9000
#define FTDAR_VALUE   0x00 /* RAMTOP+0Kb */
```
(2) Area selection (FDTUMain.c)
To select the user boot area or the user area, set FmatsuUserBootMat (0xaa) or FmatsUserMat (0x00) in the FMATS register. There must be 2 or more of NOP instructions after the setting.

```c
/*
   // UserBootSelect Function //
   ///////////////////////////////
*/
void UserBootSelect(void)
{
    volatile BYTE i;

    FMATS = FmatsUserBootMat;
    for(i=0; i < 1; i++); /*
        nop();
        nop();
    */
}

/*
   // UserMatSelect Function //
   ///////////////////////////////
*/
void UserMatSelect(void)
{
    volatile BYTE i;

    FMATS = FmatsUserMat;
    for(i=0; i < 1; i++);
    /*
        nop();
        nop();
    */
}
(3) Flash memory erase (FDTErase.c)

• EraseInit
Select the user area and after specifying the operating clock frequency, initialize the erase library. For
the operating clock frequency, the operating clock frequency specified by FDT is transmitted to the
device in new bit rate selection. The library initialization uses this operating clock frequency.

    /*
     // EraseInit Function //
     //////////////////////////
     */
    BYTE EraseInit(void)
    {
        InitPtr ERASE_INIT = (InitPtr)INIT_ADDR;
        FKEY = WRITE_ERASE_ENABLE;
        return ((*ERASE_INIT)(Frequency,0));
    }

• EraseStart
After specifying the block number to be erased, call the erase library. The block numbers are received
from the Flash Development Toolkit. For details, refer to the sources of the files for user program mode.

    /*
     // EraseStart Function //
     //////////////////////////
     */
    BYTE EraseStart(BYTE blk_no)
    {
        ErasePtr ERASE_BLOCK = (ErasePtr)WRITE_ERASE_ADDR;
        return ((*ERASE_BLOCK)(blk_no));
    }

INIT_ADDR and WRITE_ERASE_ADDR defined in KDevice.h as follows:

    #define TRANS_RAM_ADDR  0xFFFF9000
    #define INIT_ADDR   (TRANS_RAM_ADDR+32)
    #define WRITE_ERASE_ADDR (TRANS_RAM_ADDR+16)
(4) Flash memory write (FDTWrite.c)

• WriteInit
Select the user area and after specifying the operating clock frequency, initialize the write library.

/*
   // WriteInit Function //
   //*************************
*/
BYTE WriteInit(void)
{
    InitPtr WRITE_INIT = (InitPtr)INIT_ADDR;

    FKEY = WRITE_ERASE_ENABLE;
    return ((*WRITE_INIT)(Frequency,0));
}

• WriteStart
After specifying the address where the data to write is stored and the address to which to write, call the write library.
The write data and the write destination address are received from the Flash Development Toolkit. For details, refer to the sources of the files for user program mode.

/*
   // WriteStart Function //
   //*************************/
BYTE WriteStart(BYTE *data, DWORD adr)
{
    WritePtr WRITE_DATA = (WritePtr)WRITE_ERASE_ADDR;

    return ((*WRITE_DATA)((BYTE *)data, (BYTE *)adr));
}
• Execution of a write termination process (WriteFLASH)
This is part of the flash memory write termination process. For details, refer to the sources of the files for user program mode.
In write data reception (GetWriteData), receive the address where the data to write is stored and the address to which to write. If the write destination address is WRITE_END (0xFFFFFFFF), execute a write termination process.
Read out the write library.

```c
/* Acquisition of command data */
if (GetWriteData(pData, &pAddress, add_sum)){
    return;
}
if (pAddress != WRITE_END){
    /* A setup of boot status */
    BootStatus = MODE_WRITE_RUN;

    /* Program start */
    if (ErrorStatus = WriteStart(pData, pAddress)){
```
7. Programming Guide
This section explains how to write a program using the flash microcomputer's standard boot program. A sample program and the precautions to take are described. For details, see the hardware manual.

7.1 Functional Outline
The microcomputer's standard boot program is comprised of a transfer library, erase library and a write library. The functionality of the boot program is outlined below.
- Transfers the write library and erase library to a specified area of the RAM
- In initialization, specifies the operating clock frequency
- Specifies a block number to erase a block
- Specifies the data to write and the address to which to write before performing a write
- Selects the user boot area or user area
7.2 Control Registers and Control Bits

The following shows the library transfer and user boot area related control registers and control bits.

1. Selecting the functionality
   Use the FKEY register to select transfer and write/erase. Set the FKEY register to H’A5 to transfer the write/erase library, or H’5A to execute the write/erase process.

<table>
<thead>
<tr>
<th>State</th>
<th>Value</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>Transfer enabled</td>
<td>H’A5</td>
<td>Can transfer a library.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Can write a value to the SCO bit.</td>
</tr>
<tr>
<td>Programming/erasing enabled</td>
<td>H’5A</td>
<td>Can program or erase flash memory.</td>
</tr>
</tbody>
</table>

2. Starting a library download
   To transfer a library, set the SCO bit (FCCS register bit 0) to 1.

<table>
<thead>
<tr>
<th>State</th>
<th>Value</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>Source program copy disabled</td>
<td>0</td>
<td>Does not download a library to RAM.</td>
</tr>
<tr>
<td>Source program copy enabled</td>
<td>1</td>
<td>Issues a request to download a library to RAM.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>H’A5 must be written to FKEY and execution in on-chip RAM must be in progress.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>The SCO bit is cleared to 0 when downloading is completed.</td>
</tr>
</tbody>
</table>

3. Selecting a library
   To select a library, set the corresponding bit in the FPCS or FECS register to 1.

<table>
<thead>
<tr>
<th>Program to Be Transferred</th>
<th>Register</th>
<th>Bit Name</th>
<th>Bit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Programming program</td>
<td>FPCS</td>
<td>PPVs bit</td>
<td>Bit 0</td>
</tr>
<tr>
<td>Erasing program</td>
<td>FECS</td>
<td>EPvB bit</td>
<td>Bit 0</td>
</tr>
</tbody>
</table>

4. Selecting the user boot area
   To select the user boot area, set the FMATS register to H’AA.

<table>
<thead>
<tr>
<th>State</th>
<th>Value</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>User area selection</td>
<td>Other than HAA</td>
<td>Selects the user area.</td>
</tr>
<tr>
<td>User boot area selection</td>
<td>H’AA</td>
<td>Selects the user boot area.</td>
</tr>
</tbody>
</table>

   [Note] Selections switchable in only the RAM.

5. Selecting the destination of transfer
   Use the FTDAR register to set the RAM address to which a library is transferred. Unless address settings are correct, bit 7 of the FTDAR register is set to 1.

<table>
<thead>
<tr>
<th>Transfer Destination Address</th>
<th>Setting</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>RAM start address + 20 Kbytes</td>
<td>H’00</td>
<td>Sets the start address to download a program to H’FF9000.</td>
</tr>
<tr>
<td>RAM start address + 24 Kbytes</td>
<td>H’01</td>
<td>Sets the start address to download a program to H’FFA000.</td>
</tr>
<tr>
<td>RAM start address + 28 Kbytes</td>
<td>H’02</td>
<td>Sets the start address to download a program to H’FFB000.</td>
</tr>
<tr>
<td>RAM start address + 32 Kbytes</td>
<td>H’03</td>
<td>Sets the start address to download a program to H’FFC000.</td>
</tr>
</tbody>
</table>
7.3 Using the Libraries

This section explains how to use the libraries.

(1) Transfer

Follow the procedure below to perform a transfer.

1. Select the write library or erase library to transfer. For the write library, set the PPVS bit (bit 0) of the FPCS register to 1.
   For the erase library, set the EPVB bit (bit 1) of the FPCS register to 1.
2. In the FTDAR register, specify the transfer destination address in the RAM.
3. Set the FKEY register to H'A5 to enable a transfer.
4. To allow for the transfer result to be checked, set the first byte of the transfer destination address in the RAM to H'FF.
5. Set the VBR register to H'84000000.
6. Set the SCO bit (bit 0) of the FCCS register to 1. Insert 4 NOP instructions after the bit manipulation instruction.
7. A return value will have been set in the first byte of RAM, so check that the value is H'00.

(2) Erase

Follow the procedure below to perform an erase.

1. Call the erase initialization entry (transfer destination + 32 bytes) and set the operating clock frequency (R4).
   The processing result is set in the R0 register.
2. Set the FKEY register to H'5A to enable an erase/write.
3. Select the user boot area or the user area using the FMATS register. For the user boot area, set H'AA in this register; for the user area, set any value other than H'AA, e.g., H'00. Insert 2 NOP instructions after the FMATS setting.
4. Set an erase block number in the R4 register and call the erase entry (transfer destination + 16 bytes).
5. The processing result is set in the R4 register.

(3) Write

Follow the procedure below to perform a write.

1. Call the write initialization entry (transfer destination + 32 bytes) and set the operating clock frequency (R4).
   The processing result is set in the R0 register.
2. Set the FKEY register to H'5A to enable an erase/write.
3. Select the user boot area or the user area using the FMATS register. For the user boot area, set H'AA in this register; for the user area, set any value other than H'AA, e.g., H'00. Insert 2 NOP instructions after the FMATS setting.
4. Set the address of the write data in the R4 register and the write destination address in the R5 register, and then call the write entry (transfer destination + 16 bytes).
5. The processing result is set in the R4 register.
6. Call the write entry.
7.4 List of Module Facilities

There are three libraries: Transfer library, erase library and write library. The functionality of the respective modules are listed below.

<table>
<thead>
<tr>
<th>Library</th>
<th>Module Name</th>
<th>Entry</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>Transfer</td>
<td>Transfer start</td>
<td>Setting the SCO bit to 1</td>
<td>Transfers the program corresponding to the specified program type and program code.</td>
</tr>
<tr>
<td>Erasing</td>
<td>Erase initial setting</td>
<td>(Transfer destination + 32 bytes)</td>
<td>Calculates the erasing wait time using the specified operating frequency.</td>
</tr>
<tr>
<td></td>
<td>Block erasing</td>
<td>(Transfer destination + 16 bytes)</td>
<td>Erases the specified block.</td>
</tr>
<tr>
<td>Programming</td>
<td>Programming initial setting</td>
<td>(Transfer destination + 32 bytes)</td>
<td>Calculates the programming wait time using the specified operating frequency.</td>
</tr>
<tr>
<td></td>
<td>Programming</td>
<td>(Transfer destination + 16 bytes)</td>
<td>Programs the specified data in the specified programming destination address.</td>
</tr>
</tbody>
</table>
7.5 Module Specifications
Specifications of the library modules are shown below for reference.
For details, see the hardware manual.

(1) Transfer start

<table>
<thead>
<tr>
<th>Name</th>
<th>Transfer start</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>None</td>
</tr>
<tr>
<td>Library is transferred by setting the SCO bit of FCCS register to 1.</td>
<td></td>
</tr>
<tr>
<td>Functionality</td>
<td>Program transfer</td>
</tr>
<tr>
<td>Parameter</td>
<td>None</td>
</tr>
<tr>
<td>Input</td>
<td>For the write library, the PPVS bit (bit 0) of the FPCS register is set to 1.</td>
</tr>
<tr>
<td></td>
<td>For the erase library, the EPVB bit (bit 0) of the FECS register is set to 1.</td>
</tr>
<tr>
<td></td>
<td>The transfer destination address of RAM is specified in the FTDAR register.</td>
</tr>
<tr>
<td></td>
<td>The FKEY register is set to H'A5.</td>
</tr>
<tr>
<td></td>
<td>The first byte of transfer destination address in RAM is set to H'FF.</td>
</tr>
<tr>
<td></td>
<td>The VBR register is set to H'84000000.</td>
</tr>
<tr>
<td>Return value</td>
<td>None</td>
</tr>
<tr>
<td>Output</td>
<td>TDER bit (bit 7) of FTDAR register: Parameter check flag</td>
</tr>
<tr>
<td></td>
<td>Terminated normally: 0</td>
</tr>
<tr>
<td></td>
<td>FTDAR register value abnormal: 1 (download is aborted)</td>
</tr>
<tr>
<td></td>
<td>First byte of transfer destination address in RAM: Processing result</td>
</tr>
<tr>
<td></td>
<td>Terminated normally: H'00</td>
</tr>
<tr>
<td></td>
<td>FKEY register value abnormal: H'03</td>
</tr>
<tr>
<td></td>
<td>Multiple selections error: H'05</td>
</tr>
<tr>
<td>Processing</td>
<td>Determines the library to transfer from the PPVS bit of FPCS register and the EPVB bit of FECS register</td>
</tr>
<tr>
<td></td>
<td>If library selection is abnormal, returns from the module after setting the processing result</td>
</tr>
<tr>
<td></td>
<td>Unless FKEY is H'A5, returns from the module after setting the TDER bit to 1</td>
</tr>
<tr>
<td></td>
<td>Transfers the library to the RAM address specified by FTDAR</td>
</tr>
<tr>
<td></td>
<td>Clears the SCO bit to 0</td>
</tr>
<tr>
<td></td>
<td>Return to the instruction next to the one that set the SCO bit.</td>
</tr>
</tbody>
</table>

(2) Erase initialization

<table>
<thead>
<tr>
<th>Name</th>
<th>Erase initial setting</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>typedef BYTE (*IntPtr)/WORD);</td>
</tr>
<tr>
<td>Function</td>
<td>Performs erase initial setting.</td>
</tr>
<tr>
<td>Argument</td>
<td>WORD: Operating frequency</td>
</tr>
<tr>
<td>Return Value</td>
<td>Processing result</td>
</tr>
<tr>
<td></td>
<td>Normal termination: H'00</td>
</tr>
<tr>
<td></td>
<td>Operating frequency error: H'03</td>
</tr>
<tr>
<td>Processing</td>
<td>Calculates the erasing wait time using the operating frequency.</td>
</tr>
</tbody>
</table>
(3) Block erase

<table>
<thead>
<tr>
<th>Name</th>
<th>Block erasing</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>typedef BYTE (*ErasePtr)(BYTE);</td>
</tr>
<tr>
<td>Function</td>
<td>Erases a block.</td>
</tr>
<tr>
<td>Argument</td>
<td>BYTE: Erase block number</td>
</tr>
<tr>
<td>Return Value</td>
<td>Processing result</td>
</tr>
<tr>
<td></td>
<td>Normal termination: H'00</td>
</tr>
<tr>
<td></td>
<td>Erase block number error: H'09</td>
</tr>
<tr>
<td></td>
<td>FKEY error: H'11</td>
</tr>
<tr>
<td></td>
<td>Erasing error: H'21</td>
</tr>
<tr>
<td></td>
<td>Error protection: H'41</td>
</tr>
<tr>
<td>Processing</td>
<td>Checks FWE, FKEY, and block number.</td>
</tr>
<tr>
<td></td>
<td>If an error occurs, sets an error code and returns control.</td>
</tr>
<tr>
<td></td>
<td>Obtains the address using the block number.</td>
</tr>
<tr>
<td></td>
<td>Erases the address corresponding to the block.</td>
</tr>
<tr>
<td></td>
<td>If an erasing error occurs, sets an error code and returns control.</td>
</tr>
<tr>
<td></td>
<td>Returns control at normal termination.</td>
</tr>
</tbody>
</table>

(4) Write initialization

<table>
<thead>
<tr>
<th>Name</th>
<th>Programming initial setting</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>typedef BYTE (*InitPrt)(WORD);</td>
</tr>
<tr>
<td>Function</td>
<td>Performs programming initial setting.</td>
</tr>
<tr>
<td>Argument</td>
<td>WORD: Operating frequency</td>
</tr>
<tr>
<td>Return Value</td>
<td>Processing result</td>
</tr>
<tr>
<td></td>
<td>Normal termination: H'00</td>
</tr>
<tr>
<td></td>
<td>Operating frequency error: H'03</td>
</tr>
<tr>
<td>Processing</td>
<td>Calculates the programming wait time using the operating frequency.</td>
</tr>
</tbody>
</table>

(5) Write

<table>
<thead>
<tr>
<th>Name</th>
<th>Programming</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>typedef BYTE (*WritePrt)(BYTE *, BYTE *);</td>
</tr>
<tr>
<td>Function</td>
<td>Performs programming.</td>
</tr>
<tr>
<td>Arguments</td>
<td>BYTE * (first argument): Programming data storage address</td>
</tr>
<tr>
<td></td>
<td>BYTE * (second argument): Programming destination address</td>
</tr>
<tr>
<td>Return Value</td>
<td>Processing result</td>
</tr>
<tr>
<td></td>
<td>Normal termination: H'00</td>
</tr>
<tr>
<td></td>
<td>Programming data address error: H'03</td>
</tr>
<tr>
<td></td>
<td>Programming address error: H'05</td>
</tr>
<tr>
<td></td>
<td>FKEY error: H'11</td>
</tr>
<tr>
<td></td>
<td>Programming error: H'21</td>
</tr>
<tr>
<td></td>
<td>Error protection: H'41</td>
</tr>
<tr>
<td>Processing</td>
<td>Checks FWE, FKEY, and programming addresses. If an error occurs, sets an error code and returns control.</td>
</tr>
<tr>
<td></td>
<td>Verifies and programs data.</td>
</tr>
<tr>
<td></td>
<td>Verifies the programmed data. When there is no error, returns control.</td>
</tr>
<tr>
<td></td>
<td>If there is an error, reprograms data.</td>
</tr>
<tr>
<td></td>
<td>If the programming count is exceeded, returns control with a programming count error.</td>
</tr>
<tr>
<td></td>
<td>When programming terminates normally, returns control.</td>
</tr>
</tbody>
</table>
Home Page and Where to Contact for Support

Renesas Technology home page
http://www.renesas.com/
Where to contact
http://www.renesas.com/inquiry

Revision Record

<table>
<thead>
<tr>
<th>Rev.</th>
<th>Issue date</th>
<th>Contents of revision</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.00</td>
<td>2009.02.13</td>
<td>—</td>
</tr>
<tr>
<td></td>
<td></td>
<td>First edition issued</td>
</tr>
</tbody>
</table>


Notes regarding these materials

1. This document is provided for reference purposes only so that Renesas customers may select the appropriate Renesas products for their use. Renesas neither makes warranties or representations with respect to the accuracy or completeness of the information contained in this document nor grants any license to any intellectual property rights or other rights of Renesas or any third party with respect to the information in this document.

2. Renesas shall have no liability for damages or infringement of any intellectual property or other rights arising out of the use of any information in this document, including, but not limited to, product data, diagrams, charts, programs, algorithms, and application circuit examples.

3. You should not use the products or the technology described in this document for the purpose of military applications such as the development of weapons of mass destruction or for the purpose of any other military use. When exporting the products or technology described herein, you should follow the applicable export control laws and regulations, and procedures required by such laws and regulations.

4. All information included in this document such as product data, diagrams, charts, programs, algorithms, and application circuit examples, is current as of the date this document is issued. Such information, however, is subject to change without any prior notice. Before purchasing or using any Renesas products listed in this document, please confirm the latest product information with a Renesas sales office. Also, please pay regular attention to additional and different information to be disclosed by Renesas such as that disclosed through our website. (http://www.renesas.com)

5. Renesas has used reasonable care in compiling the information included in this document, but Renesas assumes no liability whatsoever for any damages incurred as a result of errors or omissions in the information included in this document.

6. When using or otherwise relying on the information in this document, you should evaluate the information in light of the total system before deciding about the applicability of such information to the intended application. Renesas makes no representations, warranties or guaranties regarding the suitability of its products for any particular application and specifically disclaims any liability arising out of the application and use of the information in this document or Renesas products.

7. With the exception of products specified by Renesas as suitable for automobile applications, Renesas products are not designed, manufactured or tested for applications or otherwise in systems the failure or malfunction of which may cause a direct threat to human life or create a risk of human injury or which require especially high quality and reliability such as safety systems, or equipment or systems for transportation and traffic, healthcare, combustion control, aerospace and aeronautics, nuclear power, or undersea communication systems. If the use of the products for such purposes, please contact a Renesas sales office beforehand. Renesas shall have no liability for damages arising out of the uses set forth above.

8. Notwithstanding the preceding paragraph, you should not use Renesas products for the purposes listed below:
   (1) artificial life support devices or systems
   (2) surgical implantations
   (3) healthcare intervention (e.g., excision, administration of medication, etc.)
   (4) any other purposes that pose a direct threat to human life

Renesas shall have no liability for damages arising out of the uses set forth in the above and purchasers who elect to use Renesas products in any of the foregoing applications shall indemnify and hold harmless Renesas Technology Corp., its affiliated companies and their officers, directors, and employees against any and all damages arising out of such applications.

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

10. Although Renesas endeavors to improve the quality and reliability of its products, IC products have specific characteristics such as the occurrence of failure at a certain rate and malfunctions under certain use conditions. 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 product, such as safety design for hardware and software including but not limited to redundancy, fire control and malfunction prevention, appropriate treatment for aging degradation or any other applicable measures. Among others, since the evaluation of microcomputer software alone is very difficult, please evaluate the safety of the final products or system manufactured by you.

11. In case Renesas products listed in this document are detached from the products to which the Renesas products are attached or affixed, the risk of accident such as swallowing by infants and small children is very high. You should implement safety measures so that Renesas products may not be easily detached from your products. Renesas shall have no liability for damages arising out of such detachment.

12. This document may not be reproduced or duplicated, in any form, in whole or in part, without prior written approval from Renesas.

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

© 2009. Renesas Technology Corp., All rights reserved.