RX Family

Read/Write Operations in SDRAM Using the SDRAMC

Introduction

The SDRAM interface in the RX Family can be connected directly to an SDRAM up to 128 Mbytes (1024 Mbits) with a CAS latency of 1 to 3 cycles. This application note describes a method of using the RX65N Group and RX72M Group to read from and write to a 128 Mbit SDRAM (Micron).

Target Device

- RX Family MCU loaded with the SDRAM area controller

Contents

1. Specifications ........................................................................................................................................... 4
2. Operation Confirmation Conditions ........................................................................................................ 6
3. Reference Application Note .................................................................................................................. 7
4. Peripheral Function ............................................................................................................................... 8
4.1 Output Operation in the SDRAMC ....................................................................................................... 8
5. Hardware ................................................................................................................................................ 9
5.1 Hardware Configuration ...................................................................................................................... 9
5.2 Pins Used .......................................................................................................................................... 10
6. Software ................................................................................................................................................ 11
6.1 Operation Overview ........................................................................................................................... 11
6.1.1 Configuring the SDRAM Initialization Sequence .......................................................................... 11
6.1.2 Specifying the SDRAM Mode Register ...................................................................................... 13
6.1.3 Specifying the Auto-Refresh Cycle ............................................................................................ 14
6.1.4 Specifying the SDRAM Read/Write Timing ............................................................................... 15
6.2 File Composition ................................................................................................................................. 17
6.3 Option-Setting Memory ..................................................................................................................... 18
6.4 Constants .......................................................................................................................................... 19
6.5 Variables ........................................................................................................................................... 21
6.6 Functions .......................................................................................................................................... 21
6.7 Function Specifications ...................................................................................................................... 22
6.8 Flowcharts ........................................................................................................................................ 26
6.8.1 Sample Codes that Smart Configurator is not used .................................................................. 26
6.8.1.1 Main Processing .................................................................................................................... 26
6.8.1.2 SDRAMC Initialization ......................................................................................................... 27
6.8.1.3 Port Initialization ................................................................................................................... 31
6.8.2.4 Compare match event callback processing ................................................. 37
6.8.2.3 SDRAM Verification Error Processing ......................................................... 37
6.8.2.2 Port Initialization ....................................................................................... 36
6.8.2.1 Main Processing ......................................................................................... 35
6.8.2 Sample Codes that Smart Configurator is used ............................................... 35
6.8.1.6 SDRAM Refresh Control Register (SDRFCR) ............................................... 38
6.8.1.5 SDRAM Address Register (SDADR) .......................................................... 39
6.8.1.4 SDRAM Access Mode Register (SDAMOD) ................................................ 38
6.8.1.3 SDRAM Initialization Register (SDIR) ....................................................... 39
6.8.1.2 SDRAM Mode Register (SDCMOD) ......................................................... 38
6.8.1.1 BCLK (SDCLK) setting ............................................................................ 38
6.8.1 Concept of register setting in target device of SDRAM specification ............... 38
7.  Porting Sample Codes that Smart Configurator is not used to Other RX Family .... 41
7.1 Before Porting ................................................................................................. 41
7.2 Porting Procedure Flow .................................................................................. 41
7.3 Porting Procedure .......................................................................................... 42
7.3.1 Generating a Porting Destination Project .................................................... 42
7.3.2 Copying the Source Files of Porting Destination Initial Settings Example ........ 46
7.3.3 Copying the Source Files of the Application Note ......................................... 47
7.3.4 Setting Porting Destination Project .............................................................. 48
7.3.5 Changing Files ........................................................................................... 51
7.3.6 Setting r_sdram_api.c ................................................................................ 55
7.3.7 Setting r_sdram_api.h ............................................................................... 55
7.  Porting Sample Codes that Smart Configurator is used to Other RX Family .......... 57
7.1 Before Porting ................................................................................................. 57
7.2 Porting Procedure Flow .................................................................................. 57
7.3 Porting Procedure .......................................................................................... 58
7.3.1 Import sample code ................................................................................... 58
7.3.2 Change file name for sample code .............................................................. 61
7.3.3 MCU migration .......................................................................................... 62
7.3.4 Clock configuration .................................................................................... 66
7.3.5 Config_BSC(Buses) setting ........................................................................ 67
7.3.6 Pin configuration ......................................................................................... 70

RX Family Read/Write Operations in SDRAM Using the SDRAMC

6.8.1.4 Timer Initialization for Wait Time .............................................................. 32
6.8.1.5 Wait Processing Using the CMT ............................................................... 33
6.8.1.6 SDRAM Verification Error Processing ................................................... 34
6.8.2 Sample Codes that Smart Configurator is used ............................................... 35
6.8.2.1 Main Processing ....................................................................................... 35
6.8.2.2 Port Initialization ....................................................................................... 36
6.8.2.3 SDRAM Verification Error Processing ..................................................... 37
6.8.2.4 Compare match event callback processing ................................................. 37

7.  Concept of register setting in target device of SDRAM specification .................... 38
7.1 BCLK (SDCLK) setting .................................................................................. 38
7.2 SDC Control Register (SDCCR) ...................................................................... 38
7.3 SDC Mode Register (SDCMOD) ..................................................................... 38
7.4 SDRAM Access Mode Register (SDAMOD) .................................................... 38
7.5 SDRAM Refresh Control Register (SDRFCR) ................................................ 39
7.6 SDRAM Initialization Register (SDIR) ........................................................... 39
7.7 SDRAM Address Register (SDADR) ............................................................... 39
7.8 SDRAM Timing Register (SDTR) ................................................................... 40
7.9 SDRAM Mode Register (SDMOD) ................................................................. 40
8.  Porting Sample Codes that Smart Configurator is not used to Other RX Family ...... 41
8.1 Before Porting ................................................................................................. 41
8.2 Porting Procedure Flow .................................................................................. 41
8.3 Porting Procedure .......................................................................................... 42
8.3.1 Generating a Porting Destination Project .................................................... 42
8.3.2 Copying the Source Files of Porting Destination Initial Settings Example ........ 46
8.3.3 Copying the Source Files of the Application Note ......................................... 47
8.3.4 Setting Porting Destination Project .............................................................. 48
8.3.5 Changing Files ........................................................................................... 51
8.3.6 Setting r_sdram_api.c ................................................................................ 55
8.3.7 Setting r_sdram_api.h ............................................................................... 55
9.  Porting Sample Codes that Smart Configurator is used to Other RX Family .......... 57
9.1 Before Porting ................................................................................................. 57
9.2 Porting Procedure Flow .................................................................................. 57
9.3 Porting Procedure .......................................................................................... 58
9.3.1 Import sample code ................................................................................... 58
9.3.2 Change file name for sample code .............................................................. 61
9.3.3 MCU migration .......................................................................................... 62
9.3.4 Clock configuration .................................................................................... 66
9.3.5 Config_BSC(Buses) setting ........................................................................ 67
9.3.6 Pin configuration ......................................................................................... 70
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>9.3.7</td>
<td>Generate code</td>
<td>70</td>
</tr>
<tr>
<td>9.3.8</td>
<td>Changing Files</td>
<td>71</td>
</tr>
<tr>
<td>10.</td>
<td>Sample Code</td>
<td>72</td>
</tr>
<tr>
<td>11.</td>
<td>Reference Documents</td>
<td>72</td>
</tr>
<tr>
<td></td>
<td>Revision History</td>
<td>73</td>
</tr>
</tbody>
</table>
1. Specifications

The following processing is implemented in the sample code of this application note.

- The SDRAMC is used to read from and write to a 128 Mbit SDRAM (Micron MT48LC8M16A2P-6A: 2 M-word \( \times \) 16 bits \( \times \) 4 banks and Micron MT48LC4M32B2P-6A: 1 M-word \( \times \) 32 bits \( \times \) 4 banks).
- After a reset, the SDRAM is initialized, and data is written in word units to the 128 Mbit SDRAM area. After data has been written to all areas, the written values are read.
- When the read value matches the expected value, LED0 is turned on. If not, LED1 is turned on.

Table 1.1 lists the Peripheral Functions and Their Applications, and Table 1.2 lists the SDRAM (MT48LC8M16A2P-6A) Specifications, and Table 1.3 lists the SDRAM (MT48LC4M32B2P-6A) Specifications.

### Table 1.1 Peripheral Functions and Their Applications

<table>
<thead>
<tr>
<th>Peripheral Function</th>
<th>Application</th>
</tr>
</thead>
<tbody>
<tr>
<td>External bus</td>
<td>Connects to the SDRAM.</td>
</tr>
<tr>
<td>I/O ports</td>
<td>Turn on LEDs.</td>
</tr>
<tr>
<td>CMT0</td>
<td>Timer for wait time</td>
</tr>
</tbody>
</table>

### Table 1.2 SDRAM (MT48LC8M16A2P-6A) Specifications

<table>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Product</td>
<td>Micron MT48LC8M16A2P-6A</td>
</tr>
<tr>
<td>Configuration</td>
<td>2 M-word ( \times ) 16 bits ( \times ) 4 banks</td>
</tr>
<tr>
<td>Size</td>
<td>128 Mbits</td>
</tr>
<tr>
<td>Row addressing</td>
<td>A11 to A0</td>
</tr>
<tr>
<td>Column addressing</td>
<td>A8 to A0</td>
</tr>
<tr>
<td>Auto refresh cycle</td>
<td>4096 refresh cycles every 64 ms</td>
</tr>
<tr>
<td>CAS latency</td>
<td>2 or 3 cycles</td>
</tr>
<tr>
<td>Initial auto refresh</td>
<td>2 times</td>
</tr>
<tr>
<td>AUTO REFRESH period (tRFC)</td>
<td>60 ns (min.)</td>
</tr>
<tr>
<td>WRITE recovery time</td>
<td>22.67 ns (min.) (^{(1)})</td>
</tr>
<tr>
<td>PRECHARGE command period (tRP)</td>
<td>18 ns (min.)</td>
</tr>
<tr>
<td>ACTIVE-to-PRECHARGE command period (tRAS)</td>
<td>42 ns (min.)</td>
</tr>
<tr>
<td>ACTIVE-to-READ or WRITE delay (tRCD)</td>
<td>18 ns (min.)</td>
</tr>
</tbody>
</table>

Note 1: WRITE recovery time is defined as 1CLK+6ns. As SDCLK is set to 60Mhz, 1CLK is 16.67 ns.
<table>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Product</td>
<td>Micron MT48LC4M32B2P-6A</td>
</tr>
<tr>
<td>Configuration</td>
<td>1 M-word × 32 bits × 4 banks</td>
</tr>
<tr>
<td>Size</td>
<td>128 Mbits</td>
</tr>
<tr>
<td>Row addressing</td>
<td>A11 to A0</td>
</tr>
<tr>
<td>Column addressing</td>
<td>A7 to A0</td>
</tr>
<tr>
<td>Auto refresh cycle</td>
<td>4096 refresh cycles every 64 ms</td>
</tr>
<tr>
<td>CAS latency</td>
<td>2 or 3 cycles</td>
</tr>
<tr>
<td>Initial auto refresh</td>
<td>2 times</td>
</tr>
<tr>
<td>AUTO REFRESH period (tRFC)</td>
<td>60 ns (min.)</td>
</tr>
<tr>
<td>WRITE recovery time</td>
<td>19.5 ns (min.) (1)</td>
</tr>
<tr>
<td>PRECHARGE command period</td>
<td>18 ns (min.)</td>
</tr>
<tr>
<td>ACTIVE-to-PRECHARGE command period (tRAS)</td>
<td>42 ns (min.)</td>
</tr>
<tr>
<td>ACTIVE-to-READ or WRITE delay (tRCD)</td>
<td>18 ns (min.)</td>
</tr>
</tbody>
</table>

Note 1: WRITE recovery time is defined as 1CLK+7ns. As SDCLK is set to 80Mhz, 1CLK is 12.5 ns.
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>R5F565NEDDFC (RX65N Group)</td>
</tr>
<tr>
<td>Operating frequencies</td>
<td>- Main clock: 24 MHz</td>
</tr>
<tr>
<td></td>
<td>- PLL: 240 MHz (main clock divided by 1 and multiplied by 10)</td>
</tr>
<tr>
<td></td>
<td>- System clock (ICLK): 120 MHz (PLL divided by 2)</td>
</tr>
<tr>
<td></td>
<td>- Peripheral module clock A (PCLKA): 120 MHz (PLL divided by 2)</td>
</tr>
<tr>
<td></td>
<td>- Peripheral module clock B (PCLKB): 60 MHz (PLL divided by 4)</td>
</tr>
<tr>
<td></td>
<td>- Peripheral module clock C (PCLKC): 60 MHz (PLL divided by 4)</td>
</tr>
<tr>
<td></td>
<td>- Peripheral module clock D (PCLKD): 60 MHz (PLL divided by 4)</td>
</tr>
<tr>
<td></td>
<td>- SDRAM clock (SDCLK): 60 MHz (PLL divided by 4)</td>
</tr>
<tr>
<td>Operating voltage</td>
<td>3.3 V</td>
</tr>
<tr>
<td>Integrated development</td>
<td>Renesas Electronics Corporation</td>
</tr>
<tr>
<td>environment</td>
<td>e² studio 2020-04</td>
</tr>
<tr>
<td>C compiler</td>
<td>Renesas Electronics Corporation</td>
</tr>
<tr>
<td></td>
<td>C/C++ Compiler Package for RX Family V.3.02</td>
</tr>
<tr>
<td></td>
<td>Compile options</td>
</tr>
<tr>
<td></td>
<td>Default settings of integrated development environment</td>
</tr>
<tr>
<td>iodefine.h version</td>
<td>Version 2.30</td>
</tr>
<tr>
<td>Endian</td>
<td>Little endian</td>
</tr>
<tr>
<td>Operating mode</td>
<td>On-Chip ROM Enabled Extended Mode</td>
</tr>
<tr>
<td>Processor mode</td>
<td>Supervisor mode</td>
</tr>
<tr>
<td>Sample code version</td>
<td>Version 1.00</td>
</tr>
<tr>
<td>Board used</td>
<td>Renesas Starter Kit+ for RX65N-2MB (product part no.:RTK50565N2S80000BE)</td>
</tr>
<tr>
<td></td>
<td>(SDRAM: MT48LC8M16A2P-6A)</td>
</tr>
</tbody>
</table>
Table 2.2 Operation Confirmation Conditions

<table>
<thead>
<tr>
<th>Item</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>MCU used</td>
<td>R5F572MNDDBD (RX72M Group)</td>
</tr>
<tr>
<td>Operating frequencies</td>
<td>- Main clock: 24 MHz</td>
</tr>
<tr>
<td></td>
<td>- PLL: 240 MHz (main clock divided by 1 and multiplied by 10)</td>
</tr>
<tr>
<td></td>
<td>- System clock (ICLK): 240 MHz (PLL divided by 1)</td>
</tr>
<tr>
<td></td>
<td>- Peripheral module clock A (PCLKA): 120 MHz (PLL divided by 2)</td>
</tr>
<tr>
<td></td>
<td>- Peripheral module clock B (PCLKB): 60 MHz (PLL divided by 4)</td>
</tr>
<tr>
<td></td>
<td>- Peripheral module clock C (PCLKC): 60 MHz (PLL divided by 4)</td>
</tr>
<tr>
<td>Operating voltage</td>
<td>3.3 V</td>
</tr>
<tr>
<td>Integrated development</td>
<td>Renesas Electronics Corporation</td>
</tr>
<tr>
<td>environment</td>
<td>e² studio 2020-04</td>
</tr>
<tr>
<td>C compiler</td>
<td>Renesas Electronics Corporation</td>
</tr>
<tr>
<td></td>
<td>C/C++ Compiler Package for RX Family V.3.02</td>
</tr>
<tr>
<td>iodefine.h version</td>
<td>Version 1.00C</td>
</tr>
<tr>
<td>Endian</td>
<td>Little endian</td>
</tr>
<tr>
<td>Operating mode</td>
<td>On-Chip ROM Enabled Extended Mode</td>
</tr>
<tr>
<td>Processor mode</td>
<td>Supervisor mode</td>
</tr>
<tr>
<td>Sample code version</td>
<td>Version 1.00</td>
</tr>
<tr>
<td>Board used</td>
<td>Renesas Starter Kit+ for RX72M (product part no.: RTK5572MNDS10000BE)</td>
</tr>
<tr>
<td></td>
<td>(SDRAM: MT48LC4M32B2P-6A)</td>
</tr>
</tbody>
</table>

3. Reference Application Note

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

- RX65N Group, RX651 Group Initial Setting (R01AN3034)
- RX72M Group Initial Setting (R01AN4530)

However the latest version is always recommended. Visit the Renesas Electronics Corporation website to check and download the latest version.
4. **Peripheral Function**

This chapter provides supplementary information on the SDRAMC. Refer to the User’s Manual: Hardware for basic information.

4.1 **Output Operation in the SDRAMC**

With the RX65N SDRAMC, when an SDRAM command is issued, pin states associated with the SDRAM will be changed after a certain time of delay from the rising of SDCLK. The command is determined on the next rising edge of SDCLK. Refer to the Electrical Characteristics chapter in the User’s Manual: Hardware for details of the output delay time for each pin.

Figure 4.1 shows the Output on Pins Associated with the SDRAM and the Timing of Command Determination.

![Diagram of SDRAM command output and timing](image)

**Figure 4.1** Output on Pins Associated with the SDRAM and the Timing of Command Determination

The SDRAM command is output after a delay from the rising of SDCLK.

- **ACT**: Bank active command
- **RD**: Read command
- **PRA**: All bank precharge command
- **DSL**: Device deselect command

The SDRAM determines the command on the rising edge of SDCLK.
5. Hardware

5.1 Hardware Configuration

Figure 5.1 and Figure 5.2 shows the Connection Example.

![Connection Example Diagram]

**Note:**
1. SDRAM: MT48LC8M16A2P-6A (2 M-word × 16 bits × 4 banks)

**Figure 5.1 MT48LC8M16A2P-6A Connection Example**

![Connection Example Diagram]

**Note:**
1. SDRAM: MT48LC4M32B2P-6A (1 M-word × 32 bits × 4 banks)

**Figure 5.2 MT48LC4M32B2P-6A Connection Example**
5.2 Pins Used

Table 5.1 and Table 5.2 lists the Pins Used and Their Functions.

**Table 5.1 Pins Used and Their Functions (RX65N)**

<table>
<thead>
<tr>
<th>Pin Name</th>
<th>I/O</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>P73</td>
<td>Output</td>
<td>Outputs for LED0 (verification succeeded).</td>
</tr>
<tr>
<td>PG7</td>
<td>Output</td>
<td>Outputs for LED1 (verification error).</td>
</tr>
<tr>
<td>PA7 to PA0</td>
<td>Output</td>
<td>Outputs an address (A7 to A0).</td>
</tr>
<tr>
<td>PB6 to PB0</td>
<td>Output</td>
<td>Outputs an address (A14 to A8).</td>
</tr>
<tr>
<td>PD7 to PD0</td>
<td>I/O</td>
<td>Outputs data (D7 to D0).</td>
</tr>
<tr>
<td>PE7 to PE0</td>
<td>I/O</td>
<td>Outputs data (D15 to D8).</td>
</tr>
<tr>
<td>P70</td>
<td>Output</td>
<td>Outputs the SDCLK signal.</td>
</tr>
<tr>
<td>P61</td>
<td>Output</td>
<td>Outputs the SDCS# signal.</td>
</tr>
<tr>
<td>P62</td>
<td>Output</td>
<td>Outputs the RAS# signal.</td>
</tr>
<tr>
<td>P63</td>
<td>Output</td>
<td>Outputs the CAS# signal.</td>
</tr>
<tr>
<td>P64</td>
<td>Output</td>
<td>Outputs the WE# signal.</td>
</tr>
<tr>
<td>P65</td>
<td>Output</td>
<td>Outputs the CKE signal.</td>
</tr>
<tr>
<td>P66</td>
<td>Output</td>
<td>Outputs the DQM0 signal.</td>
</tr>
<tr>
<td>P67</td>
<td>Output</td>
<td>Outputs the DQM1 signal.</td>
</tr>
</tbody>
</table>

**Table 5.2 Pins Used and Their Functions (RX72M)**

<table>
<thead>
<tr>
<th>Pin Name</th>
<th>I/O</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>P42</td>
<td>Output</td>
<td>Outputs for LED0 (verification succeeded).</td>
</tr>
<tr>
<td>PH0</td>
<td>Output</td>
<td>Outputs for LED1 (verification error).</td>
</tr>
<tr>
<td>PA7 to PA2</td>
<td>Output</td>
<td>Outputs an address (A7 to A2).</td>
</tr>
<tr>
<td>PB7 to PB0</td>
<td>Output</td>
<td>Outputs an address (A15 to A8).</td>
</tr>
<tr>
<td>PD7 to PD0</td>
<td>I/O</td>
<td>Outputs data (D7 to D0).</td>
</tr>
<tr>
<td>PE7 to PE0</td>
<td>I/O</td>
<td>Outputs data (D15 to D8).</td>
</tr>
<tr>
<td>P97 to P90</td>
<td>I/O</td>
<td>Outputs data (D23 to D16).</td>
</tr>
<tr>
<td>PG7 to PG0</td>
<td>I/O</td>
<td>Outputs data (D31 to D24).</td>
</tr>
<tr>
<td>P70</td>
<td>Output</td>
<td>Outputs the SDCLK signal.</td>
</tr>
<tr>
<td>P61</td>
<td>Output</td>
<td>Outputs the SDCS# signal.</td>
</tr>
<tr>
<td>P62</td>
<td>Output</td>
<td>Outputs the RAS# signal.</td>
</tr>
<tr>
<td>P63</td>
<td>Output</td>
<td>Outputs the CAS# signal.</td>
</tr>
<tr>
<td>P64</td>
<td>Output</td>
<td>Outputs the WE# signal.</td>
</tr>
<tr>
<td>P65</td>
<td>Output</td>
<td>Outputs the CKE signal.</td>
</tr>
<tr>
<td>P66</td>
<td>Output</td>
<td>Outputs the DQM0 signal.</td>
</tr>
<tr>
<td>P67</td>
<td>Output</td>
<td>Outputs the DQM1 signal.</td>
</tr>
<tr>
<td>PA0</td>
<td>Output</td>
<td>Outputs the DQM2 signal.</td>
</tr>
<tr>
<td>PA1</td>
<td>Output</td>
<td>Outputs the DQM3 signal.</td>
</tr>
</tbody>
</table>
6. Software

6.1 Operation Overview

The initialization sequence, SDRAM mode register, auto refresh cycle, and SDRAM read/write timing are specified according to the SDRAM used.

This operation overview introduces an example of SDRAMC configuration with the Micron SDRAM (MT48LC8M16A2P-6A).

6.1.1 Configuring the SDRAM Initialization Sequence

After a reset, the SDRAM must be initialized before it can be used. Configure the initialization sequence considering AUTO REFRESH period (tRFC), number of times for initial auto refresh, PRECHARGE command period (tRP), and other settings according to the datasheet for the SDRAM.

Figure 6.1 shows the Timing of SDRAM (MT48LC8M16A2P-6A) Initialization and Table 6.1 lists the Example of the SDRAMC Initial Sequence Settings when Connecting to the SDRAM (MT48LC8M16A2P-6A).

![Figure 6.1 Timing of SDRAM (MT48LC8M16A2P-6A) Initialization](image)

- (1) After a reset, specifies pins associated with the SDRAM, sets the SYSCR0.EXBE bit to 1 (external bus is enabled) and starts outputting on the SDCLK pin. Then a high level signal is output from the CKE# pin. The CKE# pin is connected to GND through a resistor to drive the CKE# pin low after the power is turned on.

- (2) Waits for 100 $\mu$s or longer after the clock is started to output. The device deselect command is output during the wait time. Then specifies the initialization timing to the SDIR register and sets the SDICR.INIRQ bit to 1, then the all bank precharge command is output.

- (3) After the all bank precharge command is output, the auto-refresh command is output after the number of cycles specified by the SDIR.PRC[2:0] bits elapse. Set the value for the SDIR.PRC[2:0] bits to tRP or greater.

- (4) After the auto-refresh command is output, the wait time in cycles specified by the SDIR.ARFI[2:0] bits is inserted. Set the value for SDIR.ARFI[2:0] bits to tRFC or greater.

  When the number of initialization auto-refresh is set to 2 times or more by the SDIR.ARFC[3:0] bits, the auto-refresh command is output again.

- (5) After the auto-refresh command is output for the number of times specified by the SDIR.ARFC[3:0] bits, the initial sequence is complete.

Example of the SDRAMC Initial Sequence Settings when Connecting to the SDRAM (MT48LC8M16A2P-6A):

- PRA: All bank precharge command
- RFA: Auto-refresh command

Table 6.1 - Example of the SDRAMC Initial Sequence Settings when Connecting to the SDRAM (MT48LC8M16A2P-6A)
### Table 6.1  Example of the SDRAMC Initial Sequence Settings when Connecting to the SDRAM (MT48LC8M16A2P-6A)

<table>
<thead>
<tr>
<th>SDRAM Timing</th>
<th>Symbol</th>
<th>Setting Value</th>
<th>Setting in the SDRAMC with RX65N</th>
</tr>
</thead>
<tbody>
<tr>
<td>Wait time until the PRECHARGE command is input after SDCLK is input</td>
<td>—</td>
<td>100 μs</td>
<td>After starting to output SDCLK, waits for 100 μs by the software, and starts the initial sequence.</td>
</tr>
<tr>
<td>PRECHARGE command period</td>
<td>tRP</td>
<td>18 ns (min.)</td>
<td>SDIR.PRC[2:0] = 000b: 3 cycles (approx. 50 ns when SDCLK is 60 MHz)</td>
</tr>
<tr>
<td>AUTO REFRESH period</td>
<td>tRFC</td>
<td>60 ns (min.)</td>
<td>SDIR.ARF[3:0] = 0001b: 4 cycles (approx. 67 ns when SDCLK is 60 MHz)</td>
</tr>
<tr>
<td>Initial auto refresh</td>
<td>—</td>
<td>2 times</td>
<td>SDIR.ARFC[3:0] = 0010b: 2 times</td>
</tr>
</tbody>
</table>
6.1.2 Specifying the SDRAM Mode Register

After the SDRAM initialization, a mode has to be set for SDRAM. Set the SDRAM mode once after the initialization. When values are written to the SDRAM mode register (SDMOD), the mode register set command is output. For details on setting values to the SDRAM mode register, refer to the datasheet for the SDRAM.

Table 6.2 lists the SDRAM Mode Register of the SDRAM (MT48LC8M16A2P-6A) and Figure 6.2 shows the Timing Diagram of the Mode Register Set Command.

Table 6.2 SDRAM Mode Register of the SDRAM (MT48LC8M16A2P-6A)

<table>
<thead>
<tr>
<th>Bit</th>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
</table>
| b2 to b0| Burst Length            | Selection of a burst length
|         |                         | 000: 1
|         |                         | 001: 2
|         |                         | 010: 4
|         |                         | 011: 8
|         |                         | 111: Full page (only when b3 is 1) Do not set values other than above. |
| b3      | Burst Type              | Selection of a burst type
|         |                         | 0: Sequential
|         |                         | 1: Interleaved                                                             |
| b6 to b4| CAS Latency             | Selection of a CAS latency
|         |                         | 010: 2
|         |                         | 011: 3
|         |                         | Do not set values other than above.                                        |
| b8, b7  | Operating Mode          | 00: Standard operation
|         |                         | Do not set values other than above.                                        |
| b9      | Write Burst Mode        | Selection of a write burst mode
|         |                         | 0: Programmed burst length
|         |                         | 1: Single location access                                                  |
| b11, b10| Reserved                | Write 00b.                                                                 |

The burst length is 1 for the RX65N SDRAMC operation. If a value other than 1 is set as the burst length, the operation is not guaranteed.

The value set to the register in this application note is 230h (burst length: 1, CAS latency: 3 cycles).

Figure 6.2 Timing Diagram of the Mode Register Set Command
### 6.1.3 Specifying the Auto-Refresh Cycle

To retain data in the SDRAM, a refresh must be performed for the number of rows during the refresh period (tREF). Auto-refresh must be performed considering the refresh period (tREF), number of rows, AUTO REFRESH period (tRFC), and other settings according to the datasheet for the SDRAM.

Table 6.3 lists the AUTO REFRESH Timing for the SDRAM (MT48LC8M16A2P-6A) and Figure 6.3 shows the AUTO REFRESH Operating Timing.

**Table 6.3 AUTO REFRESH Timing for the SDRAM (MT48LC8M16A2P-6A)**

<table>
<thead>
<tr>
<th>SDRAM Timing</th>
<th>Symbol</th>
<th>Setting Value</th>
<th>Setting in the SDRAMC with RX65N</th>
</tr>
</thead>
<tbody>
<tr>
<td>Refresh period</td>
<td>tREF</td>
<td>64 ms</td>
<td>Used for calculating AUTO REFRESH cycle</td>
</tr>
<tr>
<td>Number of rows</td>
<td></td>
<td>4096</td>
<td>Used for calculating AUTO REFRESH cycle</td>
</tr>
<tr>
<td>Auto refresh cycle</td>
<td></td>
<td>15.625 μs (tREF ÷ number of rows)</td>
<td>SDRFCR.RFC[11:0] = 03A7h: 936 cycles (approx. 15.6 μs when SDCLK is 60 MHz)</td>
</tr>
<tr>
<td>AUTO REFRESH period</td>
<td>tRFC</td>
<td>60 ns (min.)</td>
<td>SDRFCR.REFW[3:0] = 0011b: 4 cycles (approx. 67 ns when SDCLK is 60 MHz)</td>
</tr>
</tbody>
</table>

![Diagram](image.png)

**Figure 6.3 AUTO REFRESH Operating Timing**

1. When the SDRFEN.RFEN bit is set to 1 (auto-refresh operation is enabled), the auto-refresh command is output.
2. After the auto-refresh command is output, the device deselect command is output until the number of cycles specified by the SDRFCR.REFW[3:0] bits elapse. Set a value for the SDRFCR.REFW[3:0] bits to tRFC or greater.
3. The auto-refresh command is output every number of cycles specified by the SDRFCR.RFC[11:0] bits. Set a value for the SDRFCR.RFC[11:0] bits to be the auto refresh cycle (tREF ÷ number of rows) or less.
6.1.4 Specifying the SDRAM Read/Write Timing

The SDRAM read/write timing is specified considering the SDRAM settings of CAS latency (CL), WRITE recovery time (tWR), PRECHARGE command period (tRP), ACTIVE-to-PRECHARGE command (tRAS), and ACTIVE-to-READ or WRITE delay (tRCD).

Table 6.4 lists the Read/Write Timing when Connecting to the SDRAM (MT48LC8M16A2P-6A), Figure 6.4 shows the Read timing, and Figure 6.5 shows the Write timing.

Table 6.4 Read/Write Timing when Connecting to the SDRAM (MT48LC8M16A2P-6A)

<table>
<thead>
<tr>
<th>SDRAM Timing</th>
<th>Symbol</th>
<th>Setting Value</th>
<th>Setting in the SDRAMC with RX6N</th>
</tr>
</thead>
<tbody>
<tr>
<td>CAS latency (2)</td>
<td>—</td>
<td>2 or 3 (1)</td>
<td>SDTR.CL[2:0] = 011b: 3 cycles</td>
</tr>
<tr>
<td>WRITE recovery time tWR</td>
<td>tWR</td>
<td>22.67 ns (min.)</td>
<td>SDTR.WR = 1: 2 cycles</td>
</tr>
<tr>
<td>PRECHARGE command period tRP</td>
<td>tRP</td>
<td>18 ns (min.)</td>
<td>SDTR.RP[2:0] = 001b: 2 cycle</td>
</tr>
<tr>
<td>ACTIVE-to-PRECHARGE command period (2)</td>
<td>tRAS</td>
<td>42 ns (min.)</td>
<td>SDTR.RAS[2:0] = 010b: 3 cycles</td>
</tr>
<tr>
<td>ACTIVE-to-READ or WRITE delay (2)</td>
<td>tRCD</td>
<td>18 ns (min.)</td>
<td>SDTR.RCD[1:0] = 01b: 2 cycle</td>
</tr>
</tbody>
</table>

Notes:
1. Select ‘3’ in the SDRAM mode setting.
2. Set a value for the ACTIVE-to-PRECHARGE command period to less than or equal to ACTIVE-to-READ or WRITE delay (SDTR.RCD[1:0] + SDTR.CL[2:0]).

![Figure 6.4 Read Timing when the SDRAMC with Setting in Table 6.4 is Used](image-url)
Figure 6.5  Write Timing when the SDRAMC with Setting in Table 6.4 is Used

<table>
<thead>
<tr>
<th>ACT</th>
<th>Bank active command</th>
</tr>
</thead>
<tbody>
<tr>
<td>WRI</td>
<td>Write command</td>
</tr>
<tr>
<td>PRA</td>
<td>All bank precharge command</td>
</tr>
<tr>
<td>DSL</td>
<td>Device deselect command</td>
</tr>
</tbody>
</table>

SDCLK  
SDRAM command  
Data bus  

Single write  

ACT  
WRI  
PRA  
DSL  

tRAS  
tRCD  
tWR  
tRP  

data

Figure 6.5  Write Timing when the SDRAMC with Setting in Table 6.4 is Used
6.2 File Composition

Table 6.5 lists the Files Used in the Sample Code. In the sample code of this Application Note, each of projects for RX65N and RX72 that Smart Configurator is used/not used is prepared.

Files generated by the integrated development environment are not included in this table. Files containing unmodified source code generated by the Code Generator function of Smart Configurator have been omitted.

<table>
<thead>
<tr>
<th>File Name</th>
<th>Outline</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>main.c</td>
<td>Main processing</td>
<td>Processing differs depending on whether Smart Configurator is used</td>
</tr>
<tr>
<td>r_init_stop_module.c</td>
<td>Stop processing for active peripheral functions after a reset</td>
<td>Only defined in projects that Smart Configurator is not used.</td>
</tr>
<tr>
<td>r_init_stop_module.h</td>
<td>Header file for r_init_stop_module.c</td>
<td>Only defined in projects that Smart Configurator is not used.</td>
</tr>
<tr>
<td>r_init_port_initialize.c</td>
<td>Nonexistent port initialization</td>
<td>Only defined in projects that Smart Configurator is not used.</td>
</tr>
<tr>
<td>r_init_port_initialize.h</td>
<td>Header file for r_init_non_existent_port.c</td>
<td>Only defined in projects that Smart Configurator is not used.</td>
</tr>
<tr>
<td>r_init_clock.c</td>
<td>Clock initialization</td>
<td>Only defined in projects that Smart Configurator is not used.</td>
</tr>
<tr>
<td>r_init_clock.h</td>
<td>Header file for r_init_clock.c</td>
<td>Only defined in projects that Smart Configurator is not used.</td>
</tr>
<tr>
<td>r_init_rom_cache.c</td>
<td>Initial ROM cache settings</td>
<td>Only defined in projects that Smart Configurator is not used and RX72M.</td>
</tr>
<tr>
<td>r_init_rom_cache.h</td>
<td>Header file of r_init_rom_cache.c</td>
<td>Only defined in projects that Smart Configurator is not used and RX72M.</td>
</tr>
<tr>
<td>r_cmt_wait.c</td>
<td>Wait processing using the CMT</td>
<td>Only defined in projects that Smart Configurator is not used.</td>
</tr>
<tr>
<td>r_cmt_wait.h</td>
<td>Header file for r_cmt_wait.c</td>
<td>Only defined in projects that Smart Configurator is not used.</td>
</tr>
<tr>
<td>r_sdram_api.c</td>
<td>SDRAM initialization.</td>
<td>Only defined in projects that Smart Configurator is not used.</td>
</tr>
<tr>
<td>r_sdram_api.h</td>
<td>Header file for r_sdram_api.c</td>
<td>Only defined in projects that Smart Configurator is not used.</td>
</tr>
</tbody>
</table>
### 6.3 Option-Setting Memory

Table 6.6 and Table 6.7 list the Option-Setting Memory Configured in the Sample Code. When necessary, set a value suited to the user system.

#### Table 6.6 RX65N Option-Setting Memory Configured in the Sample Code

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Address</th>
<th>Setting Value</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>OFS0</td>
<td>FE7F 5D07h to FE7F 5D04h</td>
<td>FFFF FFFFh</td>
<td>IWDT is halted after a reset. WDT is stopped after a reset.</td>
</tr>
<tr>
<td>OFS1</td>
<td>FE7F 5D08h to FE7F 5D08h</td>
<td>FFFF FFFFh</td>
<td>Voltage monitor 0 reset is disabled after a reset. HOCO oscillation is disabled after a reset.</td>
</tr>
<tr>
<td>MDE</td>
<td>FE7F 5D03h to FE7F 5D00h</td>
<td>FFFF FFFFh</td>
<td>Little endian</td>
</tr>
</tbody>
</table>

#### Table 6.7 RX72M Option-Setting Memory Configured in the Sample Code

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Address</th>
<th>Setting Value</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>OFS0</td>
<td>FE7F 5D07h to FE7F 5D04h</td>
<td>FFFF FFFFh</td>
<td>IWDT is halted after a reset. WDT is stopped after a reset.</td>
</tr>
<tr>
<td>OFS1</td>
<td>FE7F 5D08h to FE7F 5D08h</td>
<td>FFFF FFFFh</td>
<td>Voltage monitor 0 reset is disabled after a reset. HOCO oscillation is disabled after a reset.</td>
</tr>
<tr>
<td>MDE</td>
<td>FE7F 5D03h to FE7F 5D00h</td>
<td>FFFF FFFFh</td>
<td>Little endian Linear mode</td>
</tr>
</tbody>
</table>
### Constants

Table 6.8 and Table 6.9 lists the Constants Used in the Sample Code.

#### Table 6.8  Constants Used in the Sample Code (1/2)

<table>
<thead>
<tr>
<th>Constant Name</th>
<th>Setting Value</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>LED0_REG_PODR(3)</td>
<td>PORT7.PODR.BIT.B3&lt;sup&gt;(1)&lt;/sup&gt; PORT4.PODR.BIT.B2&lt;sup&gt;(2)&lt;/sup&gt;</td>
<td>LED0 output data store bit</td>
</tr>
<tr>
<td>LED0_REG_PDR(3)</td>
<td>PORT7.PDR.BIT.B3&lt;sup&gt;(1)&lt;/sup&gt; PORT4.PDR.BIT.B2&lt;sup&gt;(2)&lt;/sup&gt;</td>
<td>LED0 I/O select bit</td>
</tr>
<tr>
<td>LED0_REG_PMR(3)</td>
<td>PORT7.PMR.BIT.B3&lt;sup&gt;(1)&lt;/sup&gt; PORT4.PMR.BIT.B2&lt;sup&gt;(2)&lt;/sup&gt;</td>
<td>LED0 pin mode control bit</td>
</tr>
<tr>
<td>LED1_REG_PODR(3)</td>
<td>PORTG.PODR.BIT.B7&lt;sup&gt;(1)&lt;/sup&gt; PORTH.PODR.BIT.B0&lt;sup&gt;(2)&lt;/sup&gt;</td>
<td>LED1 output data store bit</td>
</tr>
<tr>
<td>LED1_REG_PDR(3)</td>
<td>PORTG.PDR.BIT.B7&lt;sup&gt;(1)&lt;/sup&gt; PORTH.PDR.BIT.B0&lt;sup&gt;(2)&lt;/sup&gt;</td>
<td>LED1 I/O select bit</td>
</tr>
<tr>
<td>LED1_REG_PMR(3)</td>
<td>PORTG.PMR.BIT.B7&lt;sup&gt;(1)&lt;/sup&gt; PORTH.PMR.BIT.B0&lt;sup&gt;(2)&lt;/sup&gt;</td>
<td>LED1 pin mode control bit</td>
</tr>
<tr>
<td>LED_ON(3)</td>
<td>0</td>
<td>LED output data: Turned on</td>
</tr>
<tr>
<td>LED_OFF(3)</td>
<td>1</td>
<td>LED output data: Turned off</td>
</tr>
<tr>
<td>SDRAM_TOP</td>
<td>(void*)0x080000000</td>
<td>Start address of the SDRAM area</td>
</tr>
<tr>
<td>SDRAM_END</td>
<td>(void*)0x090000000</td>
<td>End address of the SDRAM area</td>
</tr>
<tr>
<td>R_WT_CMT_CLOCK(3)</td>
<td>60000000L</td>
<td>CMT count source frequency (PCLK)</td>
</tr>
<tr>
<td>R_WT_CMT_DIVIDE&lt;sup&gt;(4)&lt;/sup&gt;</td>
<td>32L</td>
<td>Division ratio of the CMT count source</td>
</tr>
<tr>
<td>R_WT_BASE_US&lt;sup&gt;(3)&lt;/sup&gt;</td>
<td>1000000L</td>
<td>Calculated value for the wait time for 1 μs</td>
</tr>
<tr>
<td>R_WT_BASE_MS&lt;sup&gt;(3)&lt;/sup&gt;</td>
<td>1000L</td>
<td>Calculated value for the wait time for 1 ms</td>
</tr>
<tr>
<td>SDRAM_REG_MPC_PFAOE0&lt;sup&gt;(3)&lt;/sup&gt;</td>
<td>0x7F&lt;sup&gt;(1)&lt;/sup&gt; 0xFF&lt;sup&gt;(2)&lt;/sup&gt;</td>
<td>MPC.PFAOE0 register set value</td>
</tr>
<tr>
<td>SDRAM_REG_MPC_PFAOE1&lt;sup&gt;(3)&lt;/sup&gt;</td>
<td>0x00</td>
<td>MPC.PFAOE1 register set value</td>
</tr>
<tr>
<td>SDRAM_REG_MPC_PFBCR0&lt;sup&gt;(3)&lt;/sup&gt;</td>
<td>0x11&lt;sup&gt;(1)&lt;/sup&gt; 0x31&lt;sup&gt;(2)&lt;/sup&gt;</td>
<td>MPC.PFBCR0 register set value</td>
</tr>
<tr>
<td>SDRAM_REG_MPC_PFBCR1&lt;sup&gt;(3)&lt;/sup&gt;</td>
<td>0x00</td>
<td>MPC.PFBCR1 register set value</td>
</tr>
<tr>
<td>SDRAM_REG_MPC_PFBCR2&lt;sup&gt;(3)&lt;/sup&gt;</td>
<td>0x00</td>
<td>MPC.PFBCR2 register set value</td>
</tr>
<tr>
<td>SDRAM_REG_MPC_PFBCR3&lt;sup&gt;(3)&lt;/sup&gt;</td>
<td>0x00&lt;sup&gt;(1)&lt;/sup&gt; 0x40&lt;sup&gt;(2)&lt;/sup&gt;</td>
<td>MPC.PFBCR3 register set value</td>
</tr>
</tbody>
</table>

Note 1: RX65N sample code setting value.
Note 2: RX72M sample code setting value.
Note 3: Only defined in projects that Smart Configurator is not used.
Table 6.9 Constants Used in the Sample Code (2/2)

<table>
<thead>
<tr>
<th>Constant Name</th>
<th>Setting Value</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>SDRAM_REG_BSC_SDCCR</td>
<td>0x00(1) 0x10(2)</td>
<td>BSC.SDCCR register set value</td>
</tr>
<tr>
<td>SDRAM_REG_BSC_SDCMOD</td>
<td>0x00</td>
<td>BSC.SDCMOD register set value</td>
</tr>
<tr>
<td>SDRAM_REG_BSC_SDMOD</td>
<td>0x00</td>
<td>BSC.SDMOD register set value</td>
</tr>
<tr>
<td>SDRAM_REG_BSC_SDFCR</td>
<td>0x33A8(1) 0x44E0(2)</td>
<td>BSC.SDRFCR register set value</td>
</tr>
<tr>
<td>SDRAM_REG_BSC_SDIR</td>
<td>0x0021(1) 0x0022(2)</td>
<td>BSC.SDIR register set value</td>
</tr>
<tr>
<td>SDRAM_REG_BSC_SDADR</td>
<td>0x01(1) 0x00(2)</td>
<td>BSC.SDADR register set value</td>
</tr>
<tr>
<td>SDRAM_REG_BSC_SDTR</td>
<td>0x00021303(1) 0x00031303(2)</td>
<td>BSC.SDTR register set value</td>
</tr>
<tr>
<td>SDRAM_REG_BSC_SDMOD</td>
<td>0x0230</td>
<td>BSC.SDMOD register set value</td>
</tr>
<tr>
<td>LED0_PORT_PIN</td>
<td>GPIO_PORT_7_PIN_3(1) GPIO_PORT_4_PIN_2(2)</td>
<td>Enumerator of pin corresponding to LED0</td>
</tr>
<tr>
<td>LED1_PORT_PIN</td>
<td>GPIO_PORT_G_PIN_7(1) GPIO_PORT_H_PIN_0(2)</td>
<td>Enumerator of pin corresponding to LED1</td>
</tr>
</tbody>
</table>

Note 1: RX65N sample code setting value.
Note 2: RX72M sample code setting value.
Note 3: Only defined in projects that Smart Configurator is not used.
Note 4: Only defined in projects that Smart Configurator is used.
6.5 Variables
Table 6.10 lists the Global Variables.

Table 6.10 Global Variables

<table>
<thead>
<tr>
<th>Type</th>
<th>Variable Name</th>
<th>Description</th>
<th>Used by Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>static volatile uint8_t</td>
<td>s_g_cmt_event_flag</td>
<td>Compare match event occurrence flag</td>
<td>main</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>cmt_event_cb</td>
</tr>
</tbody>
</table>

Note: Only defined in projects that Smart Configurator is used.

6.6 Functions
Table 6.11 lists the Functions. Functions consisting of unmodified source code generated by the Code Generator function of Smart Configurator have been omitted.

Table 6.11 Functions

<table>
<thead>
<tr>
<th>Function Name</th>
<th>Outline</th>
</tr>
</thead>
<tbody>
<tr>
<td>main</td>
<td>Main processing</td>
</tr>
<tr>
<td>port_init</td>
<td>Port initialization</td>
</tr>
<tr>
<td>R_INIT_StopModule</td>
<td>Stop processing for active peripheral functions after a reset</td>
</tr>
<tr>
<td>R_INIT_Port_Initialize</td>
<td>Nonexistent port initialization</td>
</tr>
<tr>
<td>R_INIT_Clock</td>
<td>Clock initialization</td>
</tr>
<tr>
<td>R_INIT_ROM_Cache</td>
<td>Initial ROM cache settings</td>
</tr>
<tr>
<td>R_INIT_CMT_Wait</td>
<td>Timer initialization for wait time</td>
</tr>
<tr>
<td>R_CMT_Wait</td>
<td>Wait processing using the CMT</td>
</tr>
<tr>
<td>R_WAIT_US</td>
<td>Wait processing using the CMT (unit: μs) (1)</td>
</tr>
<tr>
<td>R_WAIT_MS</td>
<td>Wait processing using the CMT (unit: ms) (1)</td>
</tr>
<tr>
<td>R_SDRAM_Init</td>
<td>SDRAMC initialization</td>
</tr>
<tr>
<td>sdram_verify_err</td>
<td>SDRAM verification error processing</td>
</tr>
<tr>
<td>cmt_oneshot_cb</td>
<td>Compare match event callback processing.</td>
</tr>
</tbody>
</table>

Note 1: This function is a function-like macro.
Note 2: Only defined in projects that RX72M sample code.
Note 3: Only defined in projects that Smart Configurator is not used.
Note 4: Only defined in projects that Smart Configurator is used.
## Function Specifications

The following tables list the sample code function specifications.

<table>
<thead>
<tr>
<th>Function</th>
<th>Outline</th>
<th>Header</th>
<th>Declaration</th>
<th>Description</th>
<th>Arguments</th>
<th>Return Value</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>main</td>
<td>Main processing</td>
<td>None</td>
<td>void main(void)</td>
<td>After initialization, initializes the SDRAM and performs read/write operation in the SDRAM.</td>
<td>None</td>
<td>None</td>
<td>Processing differs depending on whether Smart Configurator is used.</td>
</tr>
<tr>
<td>port_init</td>
<td>Port initialization</td>
<td>None</td>
<td>static void port_init(void)</td>
<td>Initializes the ports.</td>
<td>None</td>
<td>None</td>
<td>Processing differs depending on whether Smart Configurator is used.</td>
</tr>
<tr>
<td>R_INIT_StopModule</td>
<td>Stop processing for active peripheral functions after a reset</td>
<td>r_init_stop_module.h</td>
<td>void R_INIT_StopModule(void)</td>
<td>Configures the setting to enter the module-stop state.</td>
<td>None</td>
<td>None</td>
<td>Transition to the module-stop state is not performed in the sample code. Refer to the RX65N Group, RX651 Group Initial Setting and RX72M Group Initial Setting application note for details on this function. This function is only defined in the projects that Smart Configurator are not used.</td>
</tr>
</tbody>
</table>
### R_INIT_Port_Initialize

<table>
<thead>
<tr>
<th>Outline</th>
<th>Nonexistent port initialization</th>
</tr>
</thead>
<tbody>
<tr>
<td>Header</td>
<td>r_init_port_initialize.h</td>
</tr>
<tr>
<td>Declaration</td>
<td>void R_INIT_Port_Initialize(void)</td>
</tr>
<tr>
<td>Description</td>
<td>Initializes port direction registers for ports that do not exist.</td>
</tr>
<tr>
<td>Arguments</td>
<td>None</td>
</tr>
<tr>
<td>Return Value</td>
<td>None</td>
</tr>
</tbody>
</table>
| Remarks                  | RX65N sample code the number of pins in the sample code is set for the 176-pin package (PIN_SIZE=176).
RX72M sample code the number of pins in the sample code is set for the 224-pin package (PIN_SIZE=224).
After this function is called, when writing in byte units to the PDR registers or PODR registers which have nonexistent ports, set the corresponding bits for nonexistent ports as follows: set the I/O select bits in the PDR registers to 1 and set the output data store bits in the PODR registers to 0.
Refer to the RX65N Group, RX651 Group Initial Setting and RX72M Group Initial Setting application note for details on this function.
This function is only defined in the projects that Smart Configurator are not used.

### R_INIT_Clock

<table>
<thead>
<tr>
<th>Outline</th>
<th>Clock initialization</th>
</tr>
</thead>
<tbody>
<tr>
<td>Header</td>
<td>r_init_clock.h</td>
</tr>
<tr>
<td>Declaration</td>
<td>void R_INIT_Clock(void)</td>
</tr>
<tr>
<td>Description</td>
<td>Initializes the clock.</td>
</tr>
<tr>
<td>Arguments</td>
<td>None</td>
</tr>
<tr>
<td>Return Value</td>
<td>None</td>
</tr>
</tbody>
</table>
| Remarks                  | The sample code selects processing which uses PLL as the system clock without using the sub-clock.
Refer to the RX65N Group, RX651 Group Initial Setting and RX72M Group Initial Setting application note for details on this function.
This function is only defined in the projects that Smart Configurator are not used.

### R_INIT_ROM_Cache

<table>
<thead>
<tr>
<th>Outline</th>
<th>Initial ROM cache settings</th>
</tr>
</thead>
<tbody>
<tr>
<td>Header</td>
<td>r_init_ROM_Cache.h</td>
</tr>
<tr>
<td>Declaration</td>
<td>void R_INIT_ROM_Cache(void)</td>
</tr>
<tr>
<td>Description</td>
<td>After specifying the non-cacheable areas, enables the ROM cache.</td>
</tr>
<tr>
<td>Arguments</td>
<td>None</td>
</tr>
<tr>
<td>Return Value</td>
<td>None</td>
</tr>
</tbody>
</table>
| Remarks                  | In the sample code, this function only makes it possible for the ROM cache to operate.
It is assumed that this function will be called while the ROM cache is in the disabled state after the system starts.
To specify non-cacheable areas after the ROM cache has been enabled, first disable the ROM cache and then call this function.
Refer to the RX65N Group, RX651 Group Initial Setting and RX72M Group Initial Setting application note for details on this function.
This function is only defined in the projects that Smart Configurator are not used and RX72M.
### R_SDRAM_Init

<table>
<thead>
<tr>
<th>Outline</th>
<th>SDRAMC initialization</th>
</tr>
</thead>
<tbody>
<tr>
<td>Header</td>
<td>None</td>
</tr>
<tr>
<td>Declaration</td>
<td>void R_SDRAM_Init(void)</td>
</tr>
<tr>
<td>Description</td>
<td>Initializes SDRAMC used.</td>
</tr>
<tr>
<td>Arguments</td>
<td>None</td>
</tr>
<tr>
<td>Return Value</td>
<td>None</td>
</tr>
<tr>
<td>Remarks</td>
<td>This function is only defined in the projects that Smart Configurator are not used.</td>
</tr>
</tbody>
</table>

### sdram_verify_err

<table>
<thead>
<tr>
<th>Outline</th>
<th>SDRAM verification error processing</th>
</tr>
</thead>
<tbody>
<tr>
<td>Header</td>
<td>None</td>
</tr>
<tr>
<td>Declaration</td>
<td>static void sdram_verify_err(void)</td>
</tr>
<tr>
<td>Description</td>
<td>When the SDRAM verification error occurs, turns on LED1 and executes loop processing.</td>
</tr>
<tr>
<td>Arguments</td>
<td>None</td>
</tr>
<tr>
<td>Return Value</td>
<td>None</td>
</tr>
<tr>
<td>Remarks</td>
<td>Processing differs depending on whether Smart Configurator is used.</td>
</tr>
</tbody>
</table>

### R_INIT_CMT_Wait

<table>
<thead>
<tr>
<th>Outline</th>
<th>Timer initialization for wait time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Header</td>
<td>r_cmt_wait.h</td>
</tr>
<tr>
<td>Declaration</td>
<td>void R_INIT_CMT_Wait (void)</td>
</tr>
<tr>
<td>Description</td>
<td>Initializes the timer (CMT0) for wait time.</td>
</tr>
<tr>
<td>Arguments</td>
<td>None</td>
</tr>
<tr>
<td>Return Value</td>
<td>None</td>
</tr>
<tr>
<td>Remarks</td>
<td>This function is only defined in the projects that Smart Configurator are not used.</td>
</tr>
</tbody>
</table>

### R_CMT_Wait

<table>
<thead>
<tr>
<th>Outline</th>
<th>Wait processing using the CMT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Header</td>
<td>r_cmt_wait.h</td>
</tr>
<tr>
<td>Declaration</td>
<td>void R_CMT_Wait (uint16_t cnt)</td>
</tr>
<tr>
<td>Description</td>
<td>Waits for the time specified by the argument.</td>
</tr>
<tr>
<td>Arguments</td>
<td>uint16_t cnt: Wait time</td>
</tr>
<tr>
<td>Return Value</td>
<td>None</td>
</tr>
<tr>
<td>Remarks</td>
<td>This function is used in the R_CMT_WAIT_US(t_us) or R_CMT_WAIT_MS(t_ms) function.</td>
</tr>
<tr>
<td></td>
<td>This function is only defined in the projects that Smart Configurator are not used.</td>
</tr>
</tbody>
</table>
### R_CMT_WAIT_US

<table>
<thead>
<tr>
<th>Outline</th>
<th>Wait processing using the CMT (unit: (\mu)s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Header</td>
<td>r_cmt_wait.h</td>
</tr>
<tr>
<td>Declaration</td>
<td>R_CMT_WAIT_US(t_us)</td>
</tr>
<tr>
<td>Description</td>
<td>Waits for the time ((\mu)s) specified by the argument.</td>
</tr>
<tr>
<td>Arguments</td>
<td>u_int16 t_us: Wait time ((\mu)s)</td>
</tr>
<tr>
<td>Return Value</td>
<td>None</td>
</tr>
<tr>
<td>Remarks</td>
<td>This function is a function-like macro.</td>
</tr>
<tr>
<td></td>
<td>#define R_CMT_WAIT_US(t_us) R_CMT.Wait(t_us * (R_WT_CMT_CLOCK / R_WT_BASE_US) / R_WT_CMT_DIVIDE)</td>
</tr>
<tr>
<td></td>
<td>This function is only defined in the projects that Smart Configurator are not used.</td>
</tr>
</tbody>
</table>

### R_CMT_WAIT_MS

<table>
<thead>
<tr>
<th>Outline</th>
<th>Wait processing using the CMT (unit: ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Header</td>
<td>r_cmt_wait.h</td>
</tr>
<tr>
<td>Declaration</td>
<td>R_CMT_WAIT_MS(t_ms)</td>
</tr>
<tr>
<td>Description</td>
<td>Waits for the time (ms) specified by the argument.</td>
</tr>
<tr>
<td>Arguments</td>
<td>u_int16 t_ms: Wait time (ms)</td>
</tr>
<tr>
<td>Return Value</td>
<td>None</td>
</tr>
<tr>
<td>Remarks</td>
<td>This function is a function-like macro.</td>
</tr>
<tr>
<td></td>
<td>#define R_CMT_WAIT_MS(t_ms) R_CMT.Wait(t_ms * (R_WT_CMT_CLOCK / R_WT_BASE_MS) / R_WT_CMT_DIVIDE)</td>
</tr>
<tr>
<td></td>
<td>This function is only defined in the projects that Smart Configurator are not used.</td>
</tr>
</tbody>
</table>

### cmt_event_cb

<table>
<thead>
<tr>
<th>Outline</th>
<th>Compare match event callback processing.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Header</td>
<td>None</td>
</tr>
<tr>
<td>Declaration</td>
<td>void cmt_oneshot_cb (void)</td>
</tr>
<tr>
<td>Description</td>
<td>Set 1 to s_g_cmt_event_flag when a compare match event occurs.</td>
</tr>
<tr>
<td>Arguments</td>
<td>CMT channel number</td>
</tr>
<tr>
<td>Return Value</td>
<td>None</td>
</tr>
<tr>
<td>Remarks</td>
<td>This function is only defined in the projects that Smart Configurator are not used.</td>
</tr>
</tbody>
</table>
### 6.8 Flowcharts

#### 6.8.1 Sample Codes that Smart Configurator is not used

#### 6.8.1.1 Main Processing

Figure 6.6 shows the main processing of projects that Smart Configurator is not used.

```
main
→ Disable maskable interrupts
  I flag ← 0
→ Stop processing for active peripheral functions after a reset
  R_INIT_StopModule()
→ Nonexistent port initialization
  R_INIT_Port_Initialize()
→ Clock initialization
  R_INIT_Clock()
→ Initial ROM cache settings
  R_INIT_ROM_Cache()
→ Port initialization
  port_init()
→ CMT initialization for wait time
  R_INIT_CMT_Wait()
→ SDRAMC initialization
  R_SDRAM_init()
→ Initialize data in the SDRAM
  Writes 0000h to the whole SDRAM area.
→ Initialize data to be written (0000h)
  Target address for write operation = Start address of the SDRAM
  Target address for write operation < End address of the SDRAM?
    Yes
    Write data to the target address for write operation in the SDRAM
    Increment the target address for write operation and update the data to be written
  No
  Initialize the data to be verified (0000h)
  Address for verification = Start address of the SDRAM
  Address for verification < Last address of the SDRAM?
    Yes
    Data in the address for verification = Data to be verified?
      Yes
      Error processing sdram_verify_err()
      Error processing
      Increment the address for verification and update the data to be verified
    No
    Turn on LED0
      + RX65N Sample code
      PORT7.PODR register
      B3 bit ← 0
      + RX72M Sample code
      PORT4.PODR register
      B2 bit ← 0
```

Figure 6.6 Main processing of projects that Smart Configurator is not used
6.8.1.2 SDRAMC Initialization

Figure 6.7 and Figure 6.8 show the RX65N SDRAMC Initialization. Figure 6.9 and Figure 6.10 show the RX72M SDRAMC Initialization.

R_SDRAM_init()

Disable register protection

PRCR register ← 0503h
PRC0 bit = 1: Enables writing to the registers related to the operation.
PRC1 bit = 1: Enables writing to the registers related to the operation.

Disabling SCLK and BCLK pin output

SCKCR register

PSTOP0 bit ← 1: SDCLK pin output is disabled. (Fixed high)
PSTOP1 bit ← 1: BCLK pin output is disabled. (Fixed high)

Specify the bus error monitoring function

BEREN register

IGAEN bit = 0: Illegal address access detection is disabled.
TOEN bit = 0: Bus timeout detection is disabled.

Specify the bus priority setting

BUSPRI register

Set pins used as bus pins to input ports

PORTA.PDR register ← 000h: Uses pins as A0 to A7.
PORTB.PDR register ← 000h: Uses pins as A8 to A14.
PORTD.PDR register ← 000h: Uses pins as D0 to D7.
PORT6.PDR register ← 000h: Uses pins as CKE, SDCS#, RAS#, CAS#, WE#, DQM0, and DQM1.
PORT7.PDR register

B0 bit ← 0: Uses as the SDCLK pin.

Figure 6.7 RX65N sample SDRAM Initialization (1/2)
Enable the external bus

Enable SDCLK pin output

Enable register protection

Wait for 100 µs after the SDCLK output
R_CMT_WAIT_US()

Configure the initial sequence

Start the initial sequence

Confirm the status register

Select the SDRAM bus width

Select SDRAM mode

Select the SDRAM timing

Specify the address multiplex selection

Specify the endian setting

Specify access mode

Specify the auto-refresh timing

Enable auto-refresh operation

Enable operation in the SDRAM address space.

SYSCR0 register ← 5A03h
ROME bit = 1: The on-chip ROM is enabled.
EXBE bit = 1: The external bus is enabled.

SCKCR register
PSTOP0 bit ← 0: SDCLK pin output is enabled.

PRC0 bit = 0: Disables writing to the registers related to the operation.
PRC1 bit = 0: Disables writing to the registers related to the operation.

WPDC[3:0] bits = 000b: Auto-refresh request interval setting: 936 cycles

Enable register protection

PRC0 bit = 0: Disables writing to the registers related to the operation.

Wait for 100 µs or longer after SDCLK is output.

SDIR register ← 0021h
AR[3:0] bits = 000b: Initial auto-refresh interval: 4 cycles
ARFC[3:0] bits = 0010b: Initial auto-refresh count: 2 times
PRC[2:0] bits = 000b: Initialization precharge cycle count: 3 cycles

SDICR register ← 01h
INIRQ bit = 1: Initialization sequence starts.

SDCCR register ← 00h
EXENB bit = 0: Disable the SDRAM operation
BSIZE[1:0] bits = 00b: A 16-bit bus space is selected.

SDMOD register ← 0230h
MR[14:0] bits = 0230h: Setting value for the mode register set command

SDTR register ← 0002 3103h
CL[2:0] bits = 011b: Column latency: 3 cycles
WR bit = 1: Write recovery interval: 2 cycles
RP[2:0] bits = 001b: Row precharge interval: 2 cycle
RCD[1:0] bits = 01b: Row column latency: 2 cycle
RAS[2:0] bits = 010b: Row active interval: 3 cycles

SDADR register ← 01h
MXC[1:0] bits = 01b: 9-bit shift

SDCMOD register ← 00h
EMODE bit = 0: Same endian as the one for the operating mode

SDAMOD register ← 00h
BE bit = 0: Continuous access is disabled.

SDFCR register ← 33A7h

SDFEN register ← 01h
RFEN bit = 1: Auto-refresh operation is enabled.

SDCCR register
EXENB bit ← 1: Operation is enabled in the SDRAM address space.

Figure 6.8 RX65N sample SDRAM Initialization (2/2)
Figure 6.9 RX72M sample SDRAM Initialization (1/2)
Enable the external bus

Enable SCLK pin output

Enable register protection

Wait for 100 µs after the SCLK output

Configure the initial sequence

Start the initial sequence

Confirm the status register

Select the SDRAM bus width

Select SDRAM mode

Select the SDRAM timing

Specify the address multiplex selection

Specify the endian setting

Specify access mode

Specify the auto-refresh timing

Enable auto-refresh operation

Enable operation in the SDRAM address space.

SYSCR0 register ← 5A03h
ROME bit = 1: The on-chip ROM is enabled.
EXBE bit = 1: The external bus is enabled.

SCKCR register
PSTOP0 bit ← 0: SCLK pin output is enabled.

PRCR register ← A500h
PRC0 bit = 0: Disables writing to the registers related to the operation.
PRC1 bit = 0: Disables writing to the registers related to the operation.

Waits for 100 µs or longer after SCLK is output.

SDIR register ← 0022h
ARFC[3:0] bits = 0010b: Initial auto-refresh count: 2 times
PRC[2:0] bits = 000b: Initialization precharge count: 3 cycles

SDICR register ← 01h
INIRQ bit = 1: Initialization sequence starts.

Confirms values of bits in the SDSR register are all 0.

SDCCR register ← 10h
EXENB bit = 0: Disable the SDRAM operation
BSIZE[1:0] bits = 01b: A 32-bit bus space is selected.

SDMOD register ← 0230h
MR[14:0] bits = 0230h: Setting value for the mode register set command

SDTR register ← 0003 3103h
CL[2:0] bits = 011b: Column latency: 3 cycles
WR bit = 1: Write recovery interval: 2 cycles
RP[2:0] bits = 001b: Row precharge interval: 2 cycle
RCD[1:0] bits = 01b: Row column latency: 2 cycle
RAS[2:0] bits = 011b: Row active interval: 4 cycles

SDADR register ← 00h
MXC[1:0] bits = 00b: 8-bit shift

SDCMOD register ← 00h
EMODE bit = 0: Same endian as the one for the operating mode

SDAMOD register ← 00h
BE bit = 0: Continuous access is disabled.

SDRFCR register ← 44E0h
RFC[11:0] bits = 4E0h: Auto-refresh request interval setting: 1249 cycles
REFW[3:0] bits = 0100b: Auto-refresh cycle: 5 cycles

SDRFEN register ← 01h
RFEN bit = 1: Auto-refresh operation is enabled.

Enable register protection

R_CMT_WAIT_US()

Specify the auto-refresh timing

Enable auto-refresh operation

Enable operation in the SDRAM address space.

return

Figure 6.10 RX72M sample SDRAM Initialization (2/2)
6.8.1.3 **Port Initialization**

Figure 6.11 shows the Port Initialize of RX65N projects that Smart Configurator is not used. Figure 6.12 shows the Port Initialize of RX72M projects that Smart Configurator is not used.

![Port Initialize RX65N](image1)

**Figure 6.11 Port Initialize of RX65N projects that Smart Configurator is not used**

![Port Initialize RX72M](image2)

**Figure 6.12 Port Initialize of RX72M projects that Smart Configurator is not used**
6.8.1.4 **Timer Initialization for Wait Time**

Figure 6.13 shows the Timer Initialization for Wait Time.

```plaintext
R_INIT_CMT_Wait()

Store the PRCR register value

<table>
<thead>
<tr>
<th>PRCR register ← A502h</th>
</tr>
</thead>
<tbody>
<tr>
<td>PRC1 = 1: Enables writing to the registers related to the operation.</td>
</tr>
</tbody>
</table>

Disable register protection

<table>
<thead>
<tr>
<th>MSTPORA register</th>
</tr>
</thead>
<tbody>
<tr>
<td>MSTPA15 bit ← 0: The module-stop state is canceled for CMT0 and CMT1.</td>
</tr>
</tbody>
</table>

Cancel the module-stop state

<table>
<thead>
<tr>
<th>CMSTR0 register</th>
</tr>
</thead>
<tbody>
<tr>
<td>STR0 bit ← 0: CMT0.CMCNT count is stopped.</td>
</tr>
</tbody>
</table>

Specify the CMT0 count source and enable the compare match interrupt

<table>
<thead>
<tr>
<th>CMCR register</th>
</tr>
</thead>
<tbody>
<tr>
<td>CKS[1:0] bits = 01b: PCLK/32</td>
</tr>
<tr>
<td>CMIE bit = 1: Compare match interrupt (CMI0) enabled</td>
</tr>
</tbody>
</table>

Stop the CMT0 count

<table>
<thead>
<tr>
<th>CMCNT register ← 0000h</th>
</tr>
</thead>
</table>

Specify the CMT0 count source and enable the compare match interrupt

<table>
<thead>
<tr>
<th>IR028 register</th>
</tr>
</thead>
<tbody>
<tr>
<td>IR flag ← 0: No CMT0.CMI0 interrupt request is generated.</td>
</tr>
</tbody>
</table>

Clear the CMT0 count

| Clear the CMT0 interrupt request |

Clear the CMT0 interrupt request

return
```

**Figure 6.13  Timer Initialization for Wait Time**
### 6.8.1.5 Wait Processing Using the CMT

Figure 6.14 shows the Wait Processing Using the CMT.

```plaintext
R_CMT_Wait()

Return

Stop the CMT0 count

Clear the CMT0 count

Is the wait time other than 0?

No

cnt <-- cnt - 1

Yes

Decrement the wait time

Set the wait time for CMT

Clear the CMT0 interrupt request

Start the CMT0 count

Wait for CMI0 interrupt request generation

Stop the CMT0 count

Clear the CMT0 interrupt request

CMSTR0 register

STR0 bit <-- 0: CMT0.CMCNT count is stopped.

CMCNT register <-- 0000h

Argument:

cnt: Wait time

IR028 register

IR flag <-- 0: No CMT0.CMI0 interrupt request is generated.

CMSTR0 register

STR0 bit <-- 1: CMT0.CMCNT count is started.

Read the IR028 register

IR bit: 0: No CMI0 interrupt request is generated.

1: CMI0 interrupt request is generated.

CMSTR0 register

STR0 bit <-- 0: CMT0.CMCNT count is stopped.

IR028 register

IR flag <-- 0: No CMT0.CMI0 interrupt request is generated.

CMCOR register <-- cnt

Clear the CMT0 interrupt request

IR flag <-- 0: No CMT0.CMI0 interrupt request is generated.

Figure 6.14  Wait Processing Using the CMT
```
6.8.1.6 **SDRAM Verification Error Processing**

Figure 6.15 shows the SDRAM Verification Error Processing.

Figure 6.15  SDRAM Verification Error Processing of projects that Smart Configurator is not used
6.8.2 Sample Codes that Smart Configurator is used

6.8.2.1 Main Processing

Figure 6.16 shows the Main processing of projects that Smart Configurator is used.

```
main

s_g_cmt_event_flag ← 0

s_g_cmt_event_flag = 0?

No

Yes

Port initialization
port_init()

SDRAM initialization
R_Config_BSC_init(SDRAM())

Initialize data in the SDRAM

Write 0000h to the whole SDRAM area.

Initialize data to be written (0000h)

Target address for write operation =
Start address of the SDRAM

Target address for write operation < End address of the SDRAM?

Yes

No

Write data to the target address for write operation in the SDRAM

Increment the target address for write operation and update the data to be written

Initialize the data to be verified (0000h)

Address for verification =
Start address of the SDRAM

Address for verification < Last address of the SDRAM?

Yes

No

Error processing
s dram_verify_err()

Increment the address for verification and update the data to be verified

Specify output data for LED0 port
R_GPIO_PinWrite()

• RX65N Sample code
PORT7.PODR register
B3 bit ← 0; LED0: Turned on

• RX72M Sample code
PORT4.PODR register
B2 bit ← 0; LED0: Turned on
```

Figure 6.16 Main processing of projects that Smart Configurator is used
6.8.2.2 Port Initialization

Figure 6.17 shows the Port Initialize of RX65N projects that Smart Configurator is used. Figure 6.18 shows the Port Initialize of RX72M projects that Smart Configurator is used. In the Sample Codes that Smart Configurator is used, the register is rewritten using the GPIO FIT function.

```c
port_init

Specify output data for LED0 port
R_GPIO_PinWrite()
PORT7.PODR register
B3 bit ← 1: LED0: Turned off

Specify output data for LED1 port
R_GPIO_PinWrite()
PORTG.PODR register
B7 bit ← 1: LED1: Turned off

Specify the LED0 port direction
R_GPIO_PinDirectionSet()
PORT7.PDR register
B3 bit ← 1: LED0: Output

Specify the LED1 port direction
R_GPIO_PinDirectionSet()
PORTG.PDR register
B7 bit ← 1: LED1: Output

Specify mode for LED0 port
R_GPIO_PinControl()
PORT7.PMR register
B3 bit ← 0: LED0: Uses as a general I/O pin.

Specify mode for LED1 port
R_GPIO_PinControl()
PORTG.PMR register
B7 bit ← 0: LED1: Uses as a general I/O pin.

return
```

**Figure 6.17 Port Initialize of RX65N projects that Smart Configurator is used**

```c
port_init

Specify output data for LED0 port
R_GPIO_PinWrite()
PORT4.PODR register
B2 bit ← 1: LED0: Turned off

Specify output data for LED1 port
R_GPIO_PinWrite()
PORTH.PODR register
B0 bit ← 1: LED1: Turned off

Specify the LED0 port direction
R_GPIO_PinDirectionSet()
PORT4.PDR register
B2 bit ← 1: LED0: Output

Specify the LED1 port direction
R_GPIO_PinDirectionSet()
PORTH.PDR register
B0 bit ← 1: LED1: Output

Specify mode for LED0 port
R_GPIO_PinControl()
PORT4.PMR register
B2 bit ← 0: LED0: Uses as a general I/O pin.

Specify mode for LED1 port
R_GPIO_PinControl()
PORTH.PMR register
B0 bit ← 0: LED1: Uses as a general I/O pin.

return
```

**Figure 6.18 Port Initialize of RX72M projects that Smart Configurator is used**
6.8.2.3 SDRAM Verification Error Processing

Figure 6.19 shows the SDRAM Verification Error Processing of projects that Smart Configurator is used. In the Sample Codes that Smart Configurator is used, the register is rewritten using the GPIO FIT function.

```
 RX65N sample code
PORTG.PODR register
B7 bit ← 0: LED1: Turned on
 RX72M sample code
PORTH.PODR register
B0 bit ← 0: LED1: Turned on
```

Figure 6.19  SDRAM Verification Error Processing of projects that Smart Configurator is used

6.8.2.4 Compare match event callback processing

Figure 6.20 shows the Compare match event callback processing.

```
cmt_event_cb()

Setting the compare match event occurrence flag
s_g_cmt_event_flag ← 1
return
```

Figure 6.20  Compare match event callback processing
7. Concept of register setting in target device of SDRAM specification

This article explains the concept of register setting in the target device of the SDRAM specification using MT48LC4M32B2P-6A as an example.

For the settings of the pins used in SDRAMC, check the User’s Manual: Hardware of the device used, hardware, "List of Output Enable Settings" in the I/O Ports, or "How to Set the External Bus Interface" in the Multi-Function Pin Controller (MPC).

7.1 BCLK (SDCLK) setting

The clock supplied to the CLK pin of SDRAM must be longer than the time specified in AC characteristics in the SDRAM data sheet.

In case of MT48LC4M32B2P-6A, CL = 3, it is necessary to supply the clock of 7ns or more (within 142MHz) per cycle. Also, since the maximum frequency that can be output from the SDCLK pin differs depending on the microcomputer used, check each User’s Manual: Hardware.

As an example, if the RX72M and the clock source are 240MHz, the specifications can be met if the BCK[3:0] of the SCKCR register is set to a division of 3 or more and "ICLK≥BCLK".

7.2 SDC Control Register (SDCCR)

Operation enable of the SDRAM address space (EXENB bit) and SDRAM Bus Width Select (BSIZE bit) can be performed.

In the case of MT48LC4M32B2P-6A, since it can be accessed with a 32-bit width, setting BSIZE[1:0] to "01b" (A 32-bit bus space is selected) can meet the specifications. To enable of the SDRAM address space, set EXENB to "1" (Operation is enabled) after the SDRAMC related register settings are completed.

7.3 SDC Mode Register (SDCMOD)

You can select the Endian Mode select (EMODE bit).

In the case of MT48LC4M32B2P-6A, if the device operation mode is little endian, the endian of the SDRAM address space is the same as the endian of the operation mode. In this case, it can be operated by setting EMODE to "0" (Endian of SDRAM address space is the same as the endian of operating mode.).

7.4 SDRAM Access Mode Register (SDAMOD)

You can select the Continuous Access Enable (EB bit).

In the case of MT48LC4M32B2P-6A, continuous access is possible, so you can operate it by setting BE to "1" (Continuous access is enabled) after setting access using EXDMAC.
7.5 **SDRAM Refresh Control Register (SDRFCR)**

You can set the Auto-Refresh Request Interval Setting (RFC bit) and the Auto-Refresh Cycle/ Self-Refresh Clearing Cycle Count Setting (REFW bit).

The auto refresh request interval can be calculated from the following formula.

\[
\text{Refresh cycle / number of row addresses}
\]

In the case of MT48LC4M32B2P-6A, the refresh cycle is 64 ms and the number of row addresses is “4096”, so the auto refresh request interval is 15.625 μs. The auto refresh request must be made within this time, the auto refresh request interval must be set by the RFC bit. As an example, if SDCLK is 60MHz, set RFC[11:0] to “3A7h” (936 cycles) or less to 15.6μs or less, which can meet the specifications.

Set the auto-refresh request interval setting to tRFC or more. In case of MT48LC4M32B2P-6A, tRFC is 60ns. As an example, if SDCLK is 60MHz, setting REFW[3:0] to “0011b” (4 cycles) or more makes it 67 ns or more, which satisfies the specifications.

7.6 **SDRAM Initialization Register (SDIR)**

You can set the Initialization Auto-Refresh Interval (ARFI bit), Initialization Auto-Refresh Count (ARFC bit), and Initialization Precharge Cycle Count (PRC bit).

Set the initialization auto-refresh interval to tRFC or more. In case of MT48LC4M32B2P-6A, tRFC is 60ns. As an example, if SDCLK is 60MHz, setting ARFI[3:0] to “0001b” (4 cycles) or more makes it 67 ns or more, which satisfies the specifications.

For initialization auto-refresh, execute the number of times specified in the data sheet. In the case of MT48LC4M32B2P-6A, it is twice or more, so by setting ARFC[3:0] to “0010b” (twice) or more. The specifications can be met more than twice.

Set the initialization precharge cycle count to tRP or more. In case of MT48LC4M32B2P-6A, tRP is 18ns. As an example, when SDCLK is 60MHz, PRC[3:0] is set to “0000b” (3 cycles) or more and it becomes 50ns or more, which can meet the specifications.

7.7 **SDRAM Address Register (SDADR)**

You can select the Address Multiplex Select (MXC bit). Match the shift amount of address multiplex with the width of column addressing. The MT48LC4M32B2P-6A has an 8-bit width, so the specifications can be met by setting MXC [1:0] to “00b” (8-bit shift).
7.8 SDRAM Timing Register (SDTR)

The following settings can be made in this register.

- SDRAMC Column Latency (CL bit)
- Write Recovery Interval(WR bit)
- Row Precharge Interval(RP bit)
- Row Column Latency(RCD bit)
- Row Active Interval(RAS bit)

Set the SDRAMC column latency setting so that it is the same as the column latency set in SDRAM. As an example, if the SDRAM column latency setting is “3”, CL[2:0] can be set to "011b" (3 cycles) to meet the specifications.

The write recovery interval setting should be set to tWR or more. For MT48LC4M32B2P-6A, when SDCLK is 60MHz, tWR is 23.67ns. As an example, if SDCLK is 60MHz, setting the WR bit to “1” (2 cycles) makes it 33ns, which satisfies the specifications.

Set the row precharge interval to tRP or more. In case of MT48LC4M32B2P-6A, tRP is 18ns. As an example, if SDCLK is 60MHz, setting RP[2:0] to "001b" (2 cycles) or more will result in 33ns or more, which satisfies the specifications.

Set the row column latency setting to be tRCD or more. For MT48LC4M32B2P-6A, tRCD is 18ns. As an example, if SDCLK is 60MHz, setting RCD[1:0] to "01b" (2 cycles) or more will result in 33ns or more, which can meet the specifications.

Set the row active interval to tRAS or more. In case of MT48LC4M32B2P-6A, tRAS is 42ns. As an example, if SDCLK is 60MHz, setting [2:0] to “010b” (3 cycles) or more will make it 50 ns or more, which can meet the specifications.

7.9 SDRAM Mode Register (SDMOD)

You can write the setting to the mode register of SDRAM. In case of MT48LC4M32B2P-6A, if you want to set column latency to “3” and burst length to “1”, by setting "0x0230" in the SDRAM mode register, you can make the expected settings in SDRAM.
8. Porting Sample Codes that Smart Configurator is not used to Other RX Family

Sample codes included in this application note can be ported to other RX Family loaded with the exception vector table and software configurable interrupts. This section shows an example of porting sample code that RX65N and does not use the Smart Configurator to the RX72M (Renesas Starter Kit+ for RX72M).

8.1 Before Porting

Confirm the following specifications before porting sample codes. If there is a difference in the specifications, the method described in this section may not be used. After making sure of the specifications, use this application.

- The SDRAMC specification of the porting source and porting destination
- The CMT specification of the porting source and porting destination

8.2 Porting Procedure Flow

![Porting Procedure Flow Diagram]
8.3 Porting Procedure

8.3.1 Generating a Porting Destination Project

Start e² studio and create a new project.

1) Generating a porting destination project

   1-1) Start e² studio and click [File].
   1-2) Click [C/C++ Project] of [New] to start the New C/C++ Project wizard.
1-3) Click [Renesas RX].
1-4) Click [Renesas CC-RX C/C++ Executable Project].
1-5) Click [Next >].
1-6) Enter the project name.
1-7) Click [Next >].
1-8) Change [Target Device:] to [R5F572MNDxBD].
   (When porting to another RX Family, change to the porting destination RX Family.)
1-9) Select the emulator to be used.
1-10) Click [Finish].

1-11) Delete [<Project name>.c] in the generated project.
8.3.2 Copying the Source Files of Porting Destination Initial Settings Example

Copy the source files of the initial settings example application note of the porting destination RX Family to the newly generated project.

1) Downloading the initial settings example application note
   1-1) From the Renesas Electronics website, download [RX72M Group Initial Settings (R01AN4530)].
       (When porting to another RX Family, download the initial settings example application note corresponding to the porting destination RX Family.)
   1-2) Extract the downloaded zip file to the desired folder.

2) Copying the source files of the initial settings example application note to the project
   2-1) Use Explorer to open the extracted folder and copy all files from [r01an4530_rx72m] -> [r01an4530_src] to the generated project.
8.3.3 Copying the Source Files of the Application Note

Copy the source files of the application to the generated project.

1) Copy \[r\_cmt\_wait.c\], \[r\_cmt\_wait.h\], \[r\_sdram\_api.c\], and \[r\_sdram\_api.h\] from \[r01an5441\_rx65n\_sdram\] -> \[r01an5441\_src\] of this application to the project.
8.3.4 Setting Porting Destination Project

Change the build settings of the generated project.

1) Adding the final address section of the RAM
   1-1) Right-click the generated project and click [Properties].
1-2) Click [C/C++ Build] -> [Settings].
1-3) Click [Tool Settings] -> [Linker] -> [Section].
1-4) Click […] at the right end of [Section].
1-6) Click [OK].

1-7) Click [Apply and Close].
8.3.5 Changing Files
Change each source file copied in order to run the sample code of the application.

1) Changing the path to the include file.
   1-1) The include file path of the source file differs depending on the initial settings example; review and change the include file path according to the porting destination project.

2) Changing [main.c]
   2-1) To [main.c], add the include path to [r_sdram_api.h] [r_cmt_wait.h].

   Include <System includes>, "Project Include
#include <machine.h>
#include <stdint.h>
#include "iodefine.h"

#include "r_init_clock.h"
#include "r_init_port_initialization"
#include "r_init_rcm_cache.h"
#include "r_init_stop_module.h"
#include "r_sdram_api.h"
#include "r_cmt_wait.h"

2-2) Add the following define under [#include "r_cmt_wait"] in [main.c] and change the define related to LED0 and LED1 to the port used by LED0 and LED1. This time, LED0 is set to [P42] and LED1 is set to [PH0] to match LED0 and LED1 of Renesas Starter Kit+ for RX72M.

   /******************************************************************************
    * Macro definitions
   *****************************************************************************/
   /* **** LEDs **** */
   /* ==== LED0 (SDRAM verify OK) ==== */
   #define LED0_REG_PODR PORT4.PODR.BIT.B2   /* LED0 Output data store bit */
   #define LED0_REG_PDR PORT4.PDR.BIT.B2    /* LED0 I/O select bit */
   #define LED0_REG_PMR PORT4.PMR.BIT.B2    /* LED0 Pin mode control bit */
   /* ==== LED1 (SDRAM verify error) ==== */
   #define LED1_REG_PODR PORTH.PODR.BIT.B0   /* LED1 Output data store bit */
   #define LED1_REG_PDR PORTH.PDR.BIT.B0    /* LED1 I/O select bit */
   #define LED1_REG_PMR PORTH.PMR.BIT.B0    /* LED1 Pin mode control bit */
   /* ==== LEDs output data ==== */
   #define LED_ON          (0)                 /* LED on */
   #define LED_OFF         (1)                 /* LED off */
   /* **** SDRAM address **** */
   #define SDRAM_TOP   (void*)(0x08000000)             /* SDRAM top address 0x0800 0000 */
   #define SDRAM_END   (void*)(0x09000000)             /* SDRAM end address 0x0900 0000 */

2-3) Define [SDRAM TOP] and [SDRAM END] following to [#define LED_OFF (1)] in [main.c], based on the connected SDRAM and the specified External address space. This time, [SDRAM_TOP] is set to [(void*)(0x08000000)] and [SDRAM_END] is set to [(void*)(0x09000000)] to match the specifications of this application note.

   /* **** SDRAM address **** */
   #define SDRAM_TOP   (void*)(0x08000000)             /* SDRAM top address 0x0800 0000 */
   #define SDRAM_END   (void*)(0x09000000)             /* SDRAM end address 0x0900 0000 */
2-4) Add prototype function declarations for the [port_init] function and [sdram_verify_err] function in [main.c].

```c
void main(void);
static void port_init(void);
static void sdram_verify_err(void);
```

2-5) Define the [*sp_sdram_adr], [s_sdram_data], and [s_sdram_cmp] variables in the [main] function of [main.c] with a type that matches the data bus width of the connected SDRAM. This time, it is defined in [uint32_t] to match the specifications of the SDRAM installed in Renesas Starter Kit+ for RX72M.

```c
uint32_t *sp_sdram_adr; /* address pointer (SDRAM) */
uint32_t *s_sdram_data; /* sdram write back (SDRAM) */
uint32_t *s_sdram_cmp; /* compare data */
```

2-6) In the [main] function of [main.c], call the [port_init] function, [R_INIT_CMT_Wait] function, and [R_SDRAM_Init] function before while statement. Also, add the processing to change the [External bus clock (BCLK) select bit] according to the specifications of the SDRAM connected. This time, it does not need to be added because it matches the specifications of the SDRAM installed in the Renesas Starter Kit+ for RX72M.

```c
R_INIT_ROM_Cache();
R_INIT_CMT_Wait();
R_SDRAM_Init();
```
2-7) Add the following SDRAM access processing and LED0 lighting processing under the [main] function and [R_SDRAM_Init] function of [main.c]. This process is the same as that used in the application note.

```c
/****************************************************************************
 * Memory access (SDRAM)
 ****************************************************************************/

/* ---- Data initialize for SDRAM ---- */
for(sp_sdram_adr = SDRAM_TOP; sp_sdram_adr < SDRAM_END; sp_sdram_adr++)
{
    *sp_sdram_adr = 0x00000000;          /* Initialize data */
}

/* ---- Write data for SDRAM ---- */
s_sdram_data = 0x00000000;            /* SDRAM write data initialize */
for(sp_sdram_adr = SDRAM_TOP; sp_sdram_adr < SDRAM_END; sp_sdram_adr++)
{
    *sp_sdram_adr = s_sdram_data++;    /* Write increment data */
}

/* ---- Verify SDRAM data ---- */
s_sdram_cmp_data = 0x00000000;            /* SDRAM verify data initialize */
for(sp_sdram_adr = SDRAM_TOP; sp_sdram_adr<SDRAM_END; sp_sdram_adr++)
{
    /* ---- Verify error check ---- */
    if(s_sdram_cmp_data != (*sp_sdram_adr))
    {
        /* ---- Verify error ---- */
        sdram_verify_err();
    }
    s_sdram_cmp_data++;               /* Verify data increment */
}

/* LED0 ON (SDRAM verify OK) */
LED0_REG_PODR = LED_ON;
```
2-8) Define the following [port_init] function and [sdram_verify_err] function entities in [main.c].

```c
static void port_init(void)
{

    /* ---- Initialize LEDs ---- */

    /* Set port output data - LEDs OFF */
    LED0_REG_PODR = LED_OFF;
    LED1_REG_PODR = LED_OFF;

    /* Set port direction - Output */
    LED0_REG_PDR = 1;
    LED1_REG_PDR = 1;

    /* Set port mode - Use pin as general I/O port */
    LED0_REG_PMR = 0;
    LED1_REG_PMR = 0;
}

static void sdram_verify_err(void)
{

    /* LED1 ON (SDRAM verify error) */
    LED1_REG_PODR = LED_ON;

    while (1)
    {
    
    }
}
```
8.3.6 Setting r_sdram_api.c
Change the port direction of the pin used for SDRAM connection to input. This time, to match the specifications of the SDRAM installed in the Renesas Starter Kit+ for RX72M, PORT9 and PORTG corresponding to D16 to D31 are set as input ports.

```
/* --- Set up pins to be used as input port pins. --- */
PORTA.PDR.BYTE = 0x00;
PORTB.PDR.BYTE = 0x00;
PORTD.PDR.BYTE = 0x00;
PORTF.PDR.BYTE = 0x00;
PORT9.PDR.BYTE = 0x00;  /* PORT9(D16-D23) input */
PORTG.PDR.BYTE = 0x00;  /* PORTG(D24-D31) input */
PORT6.PDR.BYTE = 0x00;  /* SDCS#, RAS#,CAS#, WE#, ... */
PORT7.PDR.BIT.0 = 0;    /* SDCLK input */
```

8.3.7 Setting r_sdram_api.h
Change [r_sdram_api.h] in accordance with the porting destination environment.

The setting value when porting to RX72M (Renesas Starter Kit+ for RX72M) is shown below. When porting to another RX Family, change to the setting value corresponding to the porting destination environment. Also, see 7. Concept of register setting in target device of SDRAM specification.

1) Setting Address Output Enable Register
Change the settings of [SDRAM_REG_MPC_PFAOE0] (Address Output Enable Register 0) and [SDRAM_REG_MPC_PFAOE1] (Address Output Enable Register 1) for the specifications of the connected SDRAM. This time, [SDRAM_REG_MPC_PFAOE0] is changed to “0xFF” to match the specifications of the SDRAM installed in the Renesas Starter Kit+ for RX72M. No need to change [SDRAM_REG_MPC_PFAOE1].

```
#define SDRAM_REG_MPC_PFAOE0 (0xFF)  /* Address Output Enable Register 0 */
#define SDRAM_REG_MPC_PFAOE1 (0x00)  /* Address Output Enable Register 1 */
```

2) Setting External Bus Control Register
Change the settings of [SDRAM_REG_MPC_PFBCR0] (External Bus Control Register 0), [SDRAM_REG_MPC_PFBCR1] (External Bus Control Register 1), [SDRAM_REG_MPC_PFBCR2] (External Bus Control Register 2) and [SDRAM_REG_MPC_PFBCR3] (External Bus Control Register 3) for the specifications of the connected SDRAM. This time, [SDRAM_REG_MPC_PFBCR0] is changed to “0x31” and [SDRAM_REG_MPC_PFBCR3] is changed to “0x40” to match the specifications of the SDRAM installed in the Renesas Starter Kit+ for RX72M. No need to change [SDRAM_REG_MPC_PFBCR1] and [SDRAM_REG_MPC_PFBCR2].

```
#define SDRAM_REG_MPC_PFBCR0 (0x31)  /* External Bus Control Register 0 set value */
#define SDRAM_REG_MPC_PFBCR1 (0x00)  /* External Bus Control Register 1 set value */
#define SDRAM_REG_MPC_PFBCR2 (0x00)  /* External Bus Control Register 2 set value */
#define SDRAM_REG_MPC_PFBCR3 (0x40)  /* External Bus Control Register 3 set value */
```

3) Setting SDC Control Register
Change the setting of [SDRAM_REG_BSC_SDCCR] (SDC Control Register) for the specifications of the connected SDRAM. This time, it is changed to “0x10” to match the specifications of the SDRAM installed in the Renesas Starter Kit+ for RX72M.

```
#define SDRAM_REG_BSC_SDCCR (0x10)  /* Control Register set value */
```
4) Setting SDC Mode Register
Change the setting of [SDRAM_REG_BSC_SDCMOD] (SDC Mode Register) for the specifications of the connected SDRAM. This time, we have not changed it to meet the specifications of the SDRAM embedded with the Renesas Starter Kit + for RX72M.

5) SDRAM Access Mode Register
Change the setting of [SDRAM_REG_BSC_SDAMOD] (SDRAM Access Mode Register) for the specifications of the connected SDRAM. This time, we have not changed it to meet the specifications of the SDRAM embedded with the Renesas Starter Kit + for RX72M.

6) Setting SDRAM Refresh Control Register
Change the setting of [SDRAM_REG_BSC_SDRFCR] (SDRAM Refresh Control Register) for the specifications of the connected SDRAM. This time, it is changed to "0x44E0" to match the specifications of the SDRAM installed in the Renesas Starter Kit+ for RX72M.

```c
#define SDRAM_REG_BSC_SDRFCR (0x44E0)
/* SDRAM Refresh Control Register set value */
```

7) Setting SDRAM Initialization Register
Change the setting of [SDRAM_REG_BSC_SDIR] (SDRAM Initialization Register) for the specifications of the connected SDRAM. This time, it is changed to "0x0022" to match the specifications of the SDRAM installed in the Renesas Starter Kit+ for RX72M.

```c
#define SDRAM_REG_BSC_SDIR (0x0022)
/* SDRAM Initialization Register set value */
```

8) Setting SDRAM Address Register
Change the setting of [SDRAM_REG_BSC_SDADR] (SDRAM address register) for the specifications of the connected SDRAM. This time, it is changed to "0x00" to match the specifications of the SDRAM installed in the Renesas Starter Kit+ for RX72M.

```c
#define SDRAM_REG_BSC_SDADR (0x00)
/* SDRAM Address Register set */
```

9) Setting SDRAM Timing Register
Change the setting of [SDRAM_REG_BSC_SDTR] (SDRAM Timing Register) for the specifications of the connected SDRAM. This time, it is changed to "0x00031303" to match the specifications of the SDRAM installed in the Renesas Starter Kit+ for RX72M.

```c
#define SDRAM_REG_BSC_SDTR (0x00031303)
/* SDRAM Timing Register set value */
```

10) Setting SDRAM Mode Register
Change the setting of [SDRAM_REG_BSC_SDMOD] (SDRAM Mode Register) for the specifications of the connected SDRAM. This time, we have not changed it to meet the specifications of the SDRAM embedded with the Renesas Starter Kit + for RX72M.
9. Porting Sample Codes that Smart Configurator is used to Other RX Family

Sample codes included in this application note can be ported to other RX Family loaded with the exception vector table and software configurable interrupts. This section shows an example of porting sample code that RX65N and does used the Smart Configurator to the RX72M (Renesas Starter Kit+ for RX72M).

9.1 Before Porting

Confirm the following specifications before porting sample codes. If there is a difference in the specifications, the method described in this section may not be used. After making sure of the specifications, use this application.

- The SDRAMC specification of the porting source and porting destination
- The CMT specification of the porting source and porting destination

9.2 Porting Procedure Flow

![Porting Procedure Flow Diagram]
9.3 Porting Procedure

9.3.1 Import sample code

Import the sample code of this application

1) Start e² studio and click [File].
2) Click [Import]
3) Click [Rename & Import Existing C/C++ Project into Workspace].
4) Click [Next >].
5) Enter the project name.
6) Click [Select root directory:]
7) Click [Browse] and Select the folder that contains the downloaded sample code.
8) Click [r01an5441_rx65n_sdram_SC].
9) Click [Finish].
9.3.2 Change file name for sample code

Change the file name of the imported sample code.

1) Right-click [r01an5441_rx65n_sDRAM_UseSC.scfg]
2) click [Rename] and enter the changed project name.
### 9.3.3 MCU migration

Perform MCU migration of the imported sample code.

1. Right-click [Project name]
2. Click [Change Device]
3) Change [Target Device:] to [R5F572MNDxBD].
   (When porting to another RX Family, change to the porting destination RX Family.)
4) Click [Next >]

Select the porting destination RX Family.
5) Confirm the message displayed in [Discovered Issues]
6) Click [Next >]
7) Confirm items to be changed
8) Click [Finish]
9.3.4 Clock configuration

Use Smart Configurator to change the clock settings.

1) Open [<project name>.scfg] in the created project and click [Clocks] on the tab at the bottom.

2) Change [SCKCR(BCK[3:0])] according to the specifications of the connected SDRAM and check the SDRAM clock (SDCLK). This time, to match the specifications of the SDRAM installed in the Renesas Starter Kit+ for RX72M and the specifications of the application note, the [SCKCR(ICLK[3:0])] setting is set to divide by 1, [SCKCR(BCK[3:0])] setting is set to divide by 3.

![Clock configuration diagram](image-url)
9.3.5 Config_BSC(Buses) setting
Change [Config_BSC(Buses)] in accordance with the porting destination environment.

The setting value when porting to RX72M (Renesas Starter Kit+ for RX72M) is shown below. When porting to another RX Family, change to the setting value corresponding to the porting destination environment. Also, see 7. Concept of register setting in target device of SDRAM specification.

1) External bus area setting

   Click [Config_BSC] from the components to display the setting screen. Confirm that the displayed setting screen is the [General setting] tab. To use SDRAM, check [Use SDCS] in [External bus area settings].

2) External bus area setting

   Change the setting of [Address output pin setting] on the [General setting] tab according to the specifications of the connected SDRAM. This time, to match the specifications of the SDRAM installed in the Renesas Starter Kit+ for RX72M, we have changed the settings from [A7-A0, BC0#, DQM2, DQM3] and [A8] to [A15].

   - Read access after write access: ☑ ☑
   - Write access after write access: ☐ ☐

   Address output pin setting:
   - [A7-A0, BC0#, DQM2, DQM3]: ☑
   - [A0]: ☑
   - [A9]: ☑
   - [A10]: ☑
   - [A11]: ☑
   - [A12]: ☑
   - [A13]: ☑
   - [A14]: ☑
   - [A15]: ☑
   - [A16]: ☐
   - [A17]: ☐
   - [A18]: ☐
   - [A19]: ☐
   - [A20]: ☐
   - [A21]: ☐
   - [A22]: ☐
   - [A23]: ☐

Settings for External Address Buses A0 to A7:
Set PA0 to PA7.

Settings for External Address Buses A16 to A2:
- (Option 1) Set PC0 to PC7.
- (Option 2) Set PC0, PC1, P71, P72, P74, and PC P90 to P97.

Select the check box.
3) Area setting

Click the [SDCS] tab and change the settings for [Area setting] according to the specifications of the connected SDRAM. This time, [Bus width] is changed to [32 bits] and [Address multiplex] is changed to [8-bit shift] to match the specifications of the SDRAM installed in the Renesas Starter Kit+ for RX72M.

4) Auto refresh setting

Change the contents of [Auto refresh setting] on the [SDCS] tab according to the specifications of the connected SDRAM. This time, [Initialization auto-refresh count] is changed to “2” to match the specifications of the SDRAM installed in the Renesas Starter Kit+ for RX72M.
5) Bus timing setting

Change the setting of [Bus timing setting] on the [SDCS] tab according to the specifications of the connected SDRAM. This time, we changed the settings as shown below to match the specifications of the SDRAM installed in the Renesas Starter Kit+ for RX72.

- **The [Auto-refresh request interval] should be [Refresh cycle / number of row addresses] or less.** In case of MT48LC4M32B2P-6A, Auto-refresh request interval is 15625ns. This time, it is set to 15612.5ns by setting 1249 cycles.

- Set the [Auto-refresh cycle/ self-refresh clearing cycle count] setting to tRFC or more. In case of MT48LC4M32B2P-6A, tRFC is 60ns. This time, it is set to 62.5ns by setting 5 cycles.

- **Set the [Initialization auto-refresh interval] to tRFC or more.** In case of MT48LC4M32B2P-6A, tRFC is 60ns. This time, it is set to 62.5ns by setting 5 cycles.

- **Set the [Initialization precharge cycle count] to tRP or more.** In case of MT48LC4M32B2P-6A, tRP is 18ns. This time, it is set to 25ns by setting 2 cycles.

- Set the [SDRAMC column latency] setting so that it is the same as the column latency set in SDRAM. This time, it is set to same as SDRAM by setting 3 cycles.

- **The [Write recovery interval] setting should be set to tWR or more.** For MT48LC4M32B2P-6A, when SDCLK is 60MHz, tWR is 23.67ns. This time, it is set to 25ns by setting 2 cycles.

- Set the [Row precharge interval] to tRP or more. In case of MT48LC4M32B2P-6A, tRP is 18ns. This time, it is set to 25ns by setting 2 cycles.

- **Set the [Row column latency] setting to be tRCD or more.** For MT48LC4M32B2P-6A, tRCD is 18ns. This time, it is set to 25ns by setting 2 cycles.

- **Set the [Row active Interval] to TRAS or more.** In case of MT48LC4M32B2P-6A, TRAS is 42ns. This time, it is set to 50ns by setting 4 cycles.
9.3.6 Pin configuration
Use Smart Configurator to set the pins to be used.

1) Open [<project name>.scfg] in the created project and click [Pins] on the tab at the bottom.

2) Click [Buses] of the hardware resource.

3) Change the pin settings according to the specifications of the connected SDRAM. This time, the terminal number of [D1] is changed to [C7], and the terminal number of [D5] is changed to [D9] to match the specifications of the SDRAM installed in the Renesas Starter Kit+ for RX72M.

9.3.7 Generate code
Click the [Generate code] button to generate the code.
9.3.8 Changing Files
Change [main.c] copied in order to run the sample code of the application.

1) Setting Address Output Enable Register
   Change the settings of [LED0_PORT_PIN] and [LED1_PORT_PIN] according to the LED pin used.
   This time, [LED0_PORT_PIN] is set to [GPIO_PORT_4_PIN_2] and [LED1_PORT_PIN] is set to [GPIO_PORT_H_PIN_0] to match LED0 and LED1 of Renesas Starter Kit+ for RX72M.

   #define LED0_PORT_PIN   GPIO_PORT_4_PIN_2       /* LED0 Use  nor+ */
   #define LED1_PORT_PIN   GPIO_PORT_H_PIN_0      /* LED1 Use  Set */

2) Define the [*sp_sdram_adr], [s_sdram_data], and [s_sdram_data] variables of the [main] function with a type that matches the data bus width of the connected SDRAM. This time, it is defined in [uint32_t] to match the specifications of the SDRAM installed in Renesas Starter Kit+ for RX72M.

   volatile static uint32_t *sp_sdram_adr; /* addresses pointer(SDRAM) */
   volatile static uint32_t s_sdram_data;   /* Set  write data (SDRAM) */
   volatile static uint32_t s_sdram_cmp_data; /* compare data(SDRAM) */
10. Sample Code
Sample code can be downloaded from the Renesas Electronics website.

11. Reference Documents
User’s Manual: Hardware
   RX65N Group, RX651 Group User’s Manual: Hardware (R01UH0590)
   RX72M Group User’s Manual: Hardware (R01UH0804)
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.

User’s Manual: Development Tools
   RX Family Compiler CC-RX User’s Manual (R20UT3248)
   The latest version can be downloaded from the Renesas Electronics website.
# Revision History

<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>Jun. 30, 2020</td>
<td>—</td>
<td>—</td>
<td>First edition issued</td>
</tr>
</tbody>
</table>
General Precautions in the Handling of Microprocessing Unit and Microcontroller Unit Products

The following usage notes are applicable to all Microprocessing unit and Microcontroller unit products from Renesas. For detailed usage notes on the products covered by this document, refer to the relevant sections of the document as well as any technical updates that have been issued for the products.

1. Precaution against Electrostatic Discharge (ESD)
   A strong electrical field, when exposed to a CMOS device, can cause destruction of the gate oxide and ultimately degrade the device operation. Steps must be taken to stop the generation of static electricity as much as possible, and quickly dissipate it when it occurs. Environmental control must be adequate. When it is dry, a humidifier should be used. This is recommended to avoid using insulators that can easily build up static electricity. Semiconductor devices must be stored and transported in an anti-static container, static shielding bag or conductive material. All test and measurement tools including work benches and floors must be grounded. The operator must also be grounded using a wrist strap. Semiconductor devices must not be touched with bare hands. Similar precautions must be taken for printed circuit boards with mounted semiconductor devices.

2. Processing at power-on
   The state of the product is undefined at the time 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 time 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 time 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 time when power is supplied until the power reaches the level at which resetting is specified.

3. Input of signal during power-off state
   Do not input signals or an I/O pull-up power supply while the device is powered off. The current injection that results from input of such a signal or I/O pull-up power supply may cause malfunction and the abnormal current that passes in the device at this time may cause degradation of internal elements. Follow the guideline for input signal during power-off state as described in your product documentation.

4. Handling of unused pins
   Handle unused pins in accordance 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 the 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.

5. Clock signals
   After applying a reset, only release the reset line after the operating clock signal becomes stable. When switching the clock signal during program execution, wait until the target clock signal is 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. Additionally, 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.

6. Voltage application waveform at input pin
   Waveform distortion due to input noise or a reflected wave may cause malfunction. If the input of the CMOS device stays in the area between \( V_{IL} \) (Max.) and \( V_{IH} \) (Min.) due to noise, for example, the device may malfunction. Take care to prevent chattering noise from entering the device when the input level is fixed, and also in the transition period when the input level passes through the area between \( V_{IL} \) (Max.) and \( V_{IH} \) (Min.).

7. Prohibition of access to reserved addresses
   Access to reserved addresses is prohibited. The reserved addresses are provided for possible future expansion of functions. Do not access these addresses as the correct operation of the LSI is not guaranteed.

8. Differences between products
   Before changing from one product to another, for example to a product with a different part number, confirm that the change will not lead to problems. The characteristics of a microprocessing unit or microcontroller unit products in the same group but having a different part number might differ in terms of internal memory capacity, layout pattern, and other factors, which can affect the ranges of electrical characteristics, such as characteristic values, operating margins, immunity to noise, and amount of radiated noise. When changing to a product with a different part number, implement a system-evaluation test for the given product.
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 or any other use of the circuits, software, and information in the design of your product or system. Renesas Electronics disclaims any and all liability for any losses and damages incurred by you or third parties arising from the use of these circuits, software, or information.

2. Renesas Electronics hereby expressly disclaims any warranties against and liability for infringement or any other claims involving 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, including but not limited to, the product data, drawings, charts, programs, algorithms, and application examples.

3. 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 shall not alter, modify, copy, or reverse engineer any Renesas Electronics product, whether in whole or in part. Renesas Electronics disclaims any and all liability for any losses or damages incurred by you or third parties arising from such alteration, modification, copying or reverse engineering.

5. Renesas Electronics products are classified according to the following two quality grades: “Standard” and “High Quality”. The intended 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 electronic appliances; machine tools; personal electronic equipment; industrial robots; etc.

   *High Quality*: Transportation equipment (automobiles, trains, ships, etc.); traffic control (traffic lights); large-scale communication equipment; key financial terminal systems; safety control equipment; etc.

Unless expressly designated as a high reliability product or a product for harsh environments in a Renesas Electronics data sheet or other Renesas Electronics document, Renesas Electronics products are not intended or 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 damage (space system; undersea repeaters; nuclear power control systems; aircraft control systems; key plant systems; military equipment; etc.). Renesas Electronics disclaims any and all liability for any damages or losses incurred by you or any third parties arising from the use of any Renesas Electronics product that is inconsistent with any Renesas Electronics data sheet, user’s manual or other Renesas Electronics document.

6. When using Renesas Electronics products, refer to the latest product information (data sheets, user’s manuals, application notes, “General Notes for Handling and Using Semiconductor Devices” in the reliability handbook, etc.), and ensure that usage conditions are within the ranges specified by Renesas Electronics with respect to maximum ratings, operating power supply voltage range, heat dissipation characteristics, installation, etc. Renesas Electronics disclaims any and all liability for any malfunctions, failure or accident arising out of the use of Renesas Electronics products outside of such specified ranges.

7. Although Renesas Electronics endeavors to improve the quality and reliability of Renesas Electronics products, semiconductor products have specific characteristics, such as the occurrence of failure at a certain rate and malfunctions under certain use conditions. Unless designated as a high reliability product or a product for harsh environments in a Renesas Electronics data sheet or other Renesas Electronics document, Renesas Electronics products are subject to radiation resistance design. You are responsible for implementing safety measures to guard against the possibility of bodily injury, injury or damage caused by fire, and/or danger to the public in the event of a failure or malfunction of Renesas Electronics products, 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 and impractical, you are responsible for carefully and sufficiently investigating applicable laws and regulations that regulate the inclusion or use of controlled substances, including without limitation, the RoHS Directive, and using Renesas Electronics products in compliance with all these applicable laws and regulations. Renesas Electronics disclaims any and all liability for damages or losses occurring as a result of your noncompliance with applicable laws and regulations.

8. You shall not alter, modify, copy, or reverse engineer any Renesas Electronics product, whether in whole or in part. Renesas Electronics disclaims any and all liability for any losses and damages incurred by you or third parties arising from such alteration, modification, copying or reverse engineering.

9. Renesas Electronics products and technologies shall 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 shall comply with any applicable export control laws and regulations promulgated and administered by the governments of any countries asserting jurisdiction over the parties or transactions.

10. It is the responsibility of the buyer or distributor of Renesas Electronics products, or any other party who distributes, disposes of, or otherwise sells or transfers the product to a third party, to notify such third party in advance of the contents and conditions set forth in this document.

11. This document shall not be reprinted, 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.

(Note1) "Renesas Electronics” as used in this document means Renesas Electronics Corporation and also includes its directly or indirectly controlled subsidiaries.

(Note2) "Renesas Electronics product(s)” means any product developed or manufactured by or for Renesas Electronics.

(Rv.4.0-1  November 2017)

Corporate Headquarters

TOYOSU FORESIA, 3-2-24 Toyosu,
Koto-ku, Tokyo 135-0061, Japan
www.renesas.com

Contact information

For further information on a product, technology, the most up-to-date version of a document, or your nearest sales office, please visit:
www.renesas.com/contact/

Trademarks

Renesas and the Renesas logo are trademarks of Renesas Electronics Corporation. All trademarks and registered trademarks are the property of their respective owners.

© 2020 Renesas Electronics Corporation. All rights reserved.