1. Abstract

This document describes the method for rewriting the data flash in EW1 mode.

2. Introduction

The application example described in this document applies to the following microcomputers (MCUs).

- MCUs: M16C/63 Group
  - M16C/64 Group
  - M16C/64A Group
  - M16C/64C Group
  - M16C/65 Group (Only in a product with program ROM 1 that is 512 KB or less)
  - M16C/65C Group
  - M16C/6C Group
  - M16C/5LD Group
  - M16C/56D Group
  - M16C/5L Group
  - M16C/56 Group
  - M16C/5M Group
  - M16C/57 Group

The sample program in this application note can be used with other R8C Family MCUs which have the same special function registers (SFRs) as the above groups. Check the manual for any modifications to functions. Careful evaluation is recommended before using this application note.
3. Application Example

This application note describes an example method for rewriting the flash memory in EW1 mode.

3.1 CPU Rewrite Mode

In CPU rewrite mode, the flash memory can be rewritten when the CPU executes software commands. CPU rewrite mode consists of erase-write mode 0 (EW0 mode) and erase-write mode 1 (EW1 mode).

Table 3.1 shows the difference between EW0 Mode and EW1 Mode.

<table>
<thead>
<tr>
<th>Item</th>
<th>EW0 Mode</th>
<th>EW1 Mode</th>
</tr>
</thead>
<tbody>
<tr>
<td>Operating modes</td>
<td>• Single-chip mode</td>
<td>Single-chip mode</td>
</tr>
<tr>
<td></td>
<td>• Memory expansion mode</td>
<td></td>
</tr>
<tr>
<td>Rewrite control program allocatable areas</td>
<td>• Program ROM 1</td>
<td>• Program ROM 1</td>
</tr>
<tr>
<td></td>
<td>• Program ROM 2</td>
<td>• Program ROM 2</td>
</tr>
<tr>
<td></td>
<td>• External area</td>
<td></td>
</tr>
<tr>
<td>Rewrite control program executable areas</td>
<td>The rewrite control program must be transferred to an area other than the flash memory (e.g., RAM) before being executed.</td>
<td>The rewrite control program can be executed in program ROM 1 and program ROM 2.</td>
</tr>
<tr>
<td>Rewritable areas</td>
<td>• Program ROM 1</td>
<td>• Program ROM 1</td>
</tr>
<tr>
<td></td>
<td>• Program ROM 2</td>
<td>• Program ROM 2</td>
</tr>
<tr>
<td></td>
<td>• Data flash</td>
<td>• Data flash</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Excluding blocks with the rewrite control program</td>
</tr>
<tr>
<td>Software command restrictions</td>
<td>None</td>
<td></td>
</tr>
<tr>
<td>Mode after program or erase</td>
<td>Read status register mode</td>
<td>Read array mode</td>
</tr>
<tr>
<td>State during auto write and auto erase</td>
<td>Hold state is not maintained.</td>
<td>Hold state is maintained (I/O ports maintains the state before the command execution).</td>
</tr>
<tr>
<td>Flash memory status detection</td>
<td>• Read bits FMR00, FMR06, and FMR07 in the FMR0 register.</td>
<td>Read bits FMR00, FMR06, and FMR07 in the FMR0 register.</td>
</tr>
<tr>
<td></td>
<td>• Execute the read status register command, and then read bits SR7, SR5, and SR4 in the status register.</td>
<td></td>
</tr>
</tbody>
</table>
3.2 EW1 Mode

EW1 mode is selected by setting the FMR60 bit in the FMR6 register to 1 after setting the FMR01 bit in the FMR0 register to 1.

The FMR0 register indicates whether a program or erase operation is completed. This status register cannot be read while in EW1 mode.

When a program or erase operation is initiated, the CPU halts all program execution until the operation is completed.

Figure 3.1 shows the Entering and Exiting EW1 Mode.

---

**Figure 3.1 Entering and Exiting EW1 Mode**

1. The frequency of CPU clock that can be used in CPU rewrite mode (EW0, EW1 mode) is different according to the product. Confirm the manual about details. And, set the PM17 bit in the PM1 register to 1 (one wait inserted).

2. Set the CPU clock 1 MHz or more in the M16C/64 and M16C/65 Groups (countermeasure described in technical update TN-16C-A175A/E).
3.3 Program Command

The program command is used to write two words (4 bytes) of data to the flash memory. By writing xx41h in the first bus cycle and data to the write address in the second and third bus cycles, auto-program operation (data program and verify) is started. Set the end of the write address to 0h, 4h, 8h, or Ch. The FMR00 bit in the FMR0 register indicates whether the auto-program operation has been completed. The FMR00 bit is 0 (busy) during the auto-program operation, and 1 (ready) after the auto-program operation is completed. Do not execute other commands while the FMR00 bit is 0. After the auto-program operation is completed, the FMR06 bit in the FMR0 register indicates whether or not the auto-program operation has been completed as expected. Do not rewrite the addresses already programmed. Figure 3.2 shows the Program Command.

The lock bit protects individual blocks from being programmed inadvertently. In EW1 mode, do not execute this command on a block to which the rewrite control program is allocated.

![Figure 3.2 Program Command](image-url)
3.4 Block Erase Command

By writing xx20h in the first bus cycle and xxD0h to the highest-order even address of a block in the second bus cycle, an auto-erase operation (erase and verify) is started on the specified block.

The FMR00 bit in the FMR0 register indicates whether the auto-erase operation has been completed.

The FMR00 bit is 0 (busy) during the auto-erase operation, and 1 (ready) when the auto-erase operation is completed. Do not execute other commands while the FMR00 bit is 0.

After the auto erase operation is completed, the FMR07 bit in the FMR0 register indicates whether or not the auto erase operation has been completed as expected.

Figure 3.3 shows the Flow Chart of the Block Erase Command Programming.

The lock bit protects individual blocks from being erased inadvertently.

In EW1 mode, do not execute this command on the block to which the rewrite control program is allocated.

---

Figure 3.3 Block Erase Command

Start

- Write command code xx20h to any address in the block

- Write xxD0h to high-order address in the block.

- FMR00 = 1 ?
  - No
  - Yes

- Full status check

- Block erase command completed
3.5 Clear Status Register Command

The clear status register command is used to clear the status register.
By writing the command code xx50h, bits FMR07 and FMR06 in the FMR0 register (SR5 and SR4 in the status register) become 00b.

Figure 3.4 Clear Status Register Command
4. Description of Reference Program

4.1 Write Data to the Data Flash Area

This application note assumes that one record is 64 bytes. These records are divided into two blocks (A and B) wherein the block A has 0E000h to 0EFFFh and the block B has 0F000h to 0FFFFh, and each blocks contains 16 records, 0 to 63.

Figure 4.1 shows the Relationship between Data Flash and Records.

When writing data, write in record units starting from record 0 in block A. After writing to record 63, erase all contents (block erase) of block B. When writing the next data, start from record 0 in block B. In the same way, after writing to record 63 in block B, erase all content in block A. When writing the next data, start writing from record 0 in block A.

4.2 Error Processing

This application note does not include any error processing when accessing data flash. Perform error processing if an error occurs.
### 4.3 Function Tables

#### Declaration

<table>
<thead>
<tr>
<th>Declaration</th>
<th>void write_record_init(void)</th>
</tr>
</thead>
</table>

#### Outline

Initialize write record

#### Argument

<table>
<thead>
<tr>
<th>Argument name</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td>None</td>
</tr>
</tbody>
</table>

#### Variable (global)

<table>
<thead>
<tr>
<th>Variable name</th>
<th>Content</th>
</tr>
</thead>
<tbody>
<tr>
<td>unsigned short write_record</td>
<td>Initial setting</td>
</tr>
<tr>
<td>unsigned char block_select</td>
<td>Initial setting</td>
</tr>
</tbody>
</table>

#### Returned value

<table>
<thead>
<tr>
<th>Type</th>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td>None</td>
<td>None</td>
</tr>
</tbody>
</table>

#### Function

Clear the data flash area, and initialize the block used (block_select) and the write record number (write_record).

---

#### Declaration

<table>
<thead>
<tr>
<th>Declaration</th>
<th>unsigned char flash_write(unsigned short *data)</th>
</tr>
</thead>
</table>

#### Outline

Data write control

#### Argument

<table>
<thead>
<tr>
<th>Argument name</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>unsigned short *data</td>
<td>Table starting address of write data</td>
</tr>
</tbody>
</table>

#### Variable (global)

<table>
<thead>
<tr>
<th>Variable name</th>
<th>Content</th>
</tr>
</thead>
<tbody>
<tr>
<td>unsigned short write_record</td>
<td>Referring/Setting</td>
</tr>
<tr>
<td>unsigned char block_select</td>
<td>Referring/Setting</td>
</tr>
</tbody>
</table>

#### Returned value

<table>
<thead>
<tr>
<th>Type</th>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>none</td>
<td>COMPLETE</td>
<td>Completed successfully (0x00)</td>
</tr>
<tr>
<td></td>
<td>DATA_PROGRAM_ERR</td>
<td>Data write error (0x01)</td>
</tr>
<tr>
<td></td>
<td>ERASE_ERR</td>
<td>Data erase error (0x02)</td>
</tr>
</tbody>
</table>

#### Function

Write the record data before updating the write record number (write_record). When writing data to the last record (record 63), erase unused blocks and clear the record write information (writing_info) to change used block. When a write failure or an erase failure occurs, the returned value becomes DATA_PROGRAM_ERR + ERASE_ERR (0x03)

---

#### Declaration

<table>
<thead>
<tr>
<th>Declaration</th>
<th>unsigned char block_erase(unsigned short *ers_addr)</th>
</tr>
</thead>
</table>

#### Outline

Block erase

#### Argument

<table>
<thead>
<tr>
<th>Argument name</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>unsigned short *ers_addr</td>
<td>Table starting address of erase block</td>
</tr>
</tbody>
</table>

#### Variable (global)

<table>
<thead>
<tr>
<th>Variable name</th>
<th>Content</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td>None</td>
</tr>
</tbody>
</table>

#### Returned value

<table>
<thead>
<tr>
<th>Type</th>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>none</td>
<td>COMPLETE</td>
<td>Completed successfully</td>
</tr>
<tr>
<td></td>
<td>ERASE_ERR</td>
<td>Erase error</td>
</tr>
</tbody>
</table>

#### Function

Erase the specified block in EW1 mode.
### Rewriting Flash Memory (EW1 Mode)

**Declaration**

```c
unsigned char data_write(unsigned short *write_data)
```

**Outline**

**Argument**

- **Argument name**: | **Meaning**: Table starting address of write data
- `unsigned short *write_data`

**Variable (global)**

- **Variable name**: | **Content**: Referring
- `unsigned short write_record`
- `unsigned char block_select`

**Returned value**

- **Type**: | **Value**: | **Meaning**: 
- `unsigned char` | `COMPLETE` | Completed successfully
- `DATA_PROGRAM_ERR` | Writing error

**Function**

Write data to the write record (write_record) of the block used (block_select) in EW1 mode.

---

**Declaration**

```c
void make_data(unsigned short *data)
```

**Outline**

**Creating write data**

**Argument**

- **Argument name**: | **Meaning**: Table starting address of write data
- `unsigned short *data`

**Variable (global)**

- **Variable name**: | **Content**: None
- None

**Returned value**

- **Type**: | **Value**: | **Meaning**: None
- None | None | None

**Function**

Create write record data for data flash. As dummy data, values from 0000h to 001FH are generated in this application note.
### Declaration
void cpu_slow(void)

### Outline
System clock slow down

### Argument
<table>
<thead>
<tr>
<th>Argument name</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td>None</td>
</tr>
</tbody>
</table>

### Variable (global)
<table>
<thead>
<tr>
<th>Variable name</th>
<th>Content</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td>None</td>
</tr>
</tbody>
</table>

### Returned value
<table>
<thead>
<tr>
<th>Type</th>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td>None</td>
<td>None</td>
</tr>
</tbody>
</table>

### Function
Set the CPU clock divisor to CM06 = 0, CM17 to CM16 = 01b (divide-by-2), PM17 = 1 (one wait inserted).

---

### Declaration
void cpu_fast(void)

### Outline
System clock speed up

### Argument
<table>
<thead>
<tr>
<th>Argument name</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td>None</td>
</tr>
</tbody>
</table>

### Variable (global)
<table>
<thead>
<tr>
<th>Variable name</th>
<th>Content</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td>None</td>
</tr>
</tbody>
</table>

### Returned value
<table>
<thead>
<tr>
<th>Type</th>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td>None</td>
<td>None</td>
</tr>
</tbody>
</table>

### Function
Set the CPU clock divisor to CM06 = 0, CM17 to CM16 = 01b (not divided), PM17 = 0 (no wait inserted).

---

### Declaration
void command_write(unsigned short *addr, unsigned short *data)

### Outline
Program command issue

### Argument
<table>
<thead>
<tr>
<th>Argument name</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>unsigned short *addr</td>
<td>Table starting address of data flash for writing data</td>
</tr>
<tr>
<td>unsigned short *data</td>
<td>Table starting address of write data</td>
</tr>
</tbody>
</table>

### Variable (global)
<table>
<thead>
<tr>
<th>Variable name</th>
<th>Content</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td>None</td>
</tr>
</tbody>
</table>

### Returned value
<table>
<thead>
<tr>
<th>Type</th>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td>None</td>
<td>None</td>
</tr>
</tbody>
</table>

### Function
Issue program command and write data to data flash.
### 4.4 Flowcharts

#### 4.4.1 Main Function

```
main()
asm("fclr I")
CPU clock setting
mcu_init()
Initial setting for write record
write_record_init()

Create write data.
make_data(write_data)

Data write control
flash_write(write_data)

Data write error occurs ?
Yes
Data write error processing (1)
No

Erase error occurs ?
Yes
Erase error processing (1)
No
```

Note:
1. This application note does not include any error processing when accessing data flash. Perform error processing if an error occurs.
4.4.2 CPU Initialization

```
mcu_init(void)
prcr ← 0x03
pm0 ← 0x00
pm1 ← 0x09
cm2 ← 0x00

prcr ← 0x00

cm1 ← 0xA0
cm0 ← 0x08

cm1 ← 0x60

Main clock: Divide-by-4

Main clock: Divide-by-2

Main clock: Not divided

Set protect.

return
```

4.4.3 Write Record Initialization

```
write_record_init(void)

Block erase
block_erase(BLOCK_A)

Initialize data flash.

Block erase
block_erase(BLOCK_B)

Initialize data flash.

Set write record (write_record) to 0.

Set used block as block A.

return
```
### 4.4.4 System Clock Slow Down

```c
void cpu_slow()
{
    prcr = 0x03; // Release protect.
    cm1 = 0x60; // Main clock: Divide-by-2
    cm0 = 0x08;
    pm17 = 1; // Set wait: 1 wait inserted
    prcr = 0x00; // Set protect.
    return;
}
```

### 4.4.5 System Clock Speed Up

```c
void cpu_fast()
{
    prcr = 0x03; // Release protect.
    cm1 = 0x20; // Main clock: Not divided
    cm0 = 0x08;
    pm17 = 0; // Set wait: No wait inserted
    prcr = 0x00; // Set protect.
    return;
}
```
4.4.6 Data Write Control

```
flash_write(unsigned short *data)
  Argument
  *data: start address of the data written to the record

  initialize the result (write/erase result) to COMPLETE.

  data_write(data)

  data programming

  Program result

  Program failed

  result ← result | DATA_PROGRAM_ERR

  Program success

  result ← result | DATA_PROGRAM_ERR

  Write record number+1.

  Add 1 to the write record number.

  Is there blank record in block used ?

  Yes

  Block used is block A ?

  Yes

  block_erase(BLOCK_A)

  Block erase

  Block erase result

  Erase result

  Erase failed

  result ← result | ERASE_ERR

  Erase successful

  result ← result | ERASE_ERR

  Initialize the write record number to 0.

  Block used ← Block B

  No (write record number ≥ number of records per one block)

  No (block used B)

  Block used is block A ?

  No

  block_erase(BLOCK_B)

  Block erase

  Erase result

  Erase failed

  result ← result | ERASE_ERR

  Erase successful

  result ← result | ERASE_ERR

  Initialize the write record number to 0.

  Block used ← Block A

  return result
```
### 4.4.7 Block Erase

```c
block_erase(unsigned short *ers_addr)
asm("pushc flg")
asm("lor 1")
Set the CPU clock.
cpu_slow
fmr0 ← 0x01
fmr0 ← 0x03
fmr1 ← 0x82
fmr6 ← 0x03
fmr1 ← 0x80
*ers_addr ← 0x0020
*ers_addr ← 0x00D0

Auto-erase operation completed ?
No (FMR00 is 0)
Yes (FMR00 is 1)
Erase result
Erase failed (FMR07 is 1)
Erase successful (FMR07 is 0)
erase_result ← COMPLETE
erase_result ← ERASE_ERR

Set the CPU clock.
cpu_fast
asm("popc flg")

return erase_result
```

Argument
*ers_addr: High order address of erase block

Store the flag register.
Disables interrupts

Set the CPU clock to less than 10 MHz\(^{1}\) and set the PM17 bit to 1 (one wait inserted).

When setting the FMR01 bit to 1 (CPU rewrite mode enabled), write 1 immediately after writing 0.
Make sure no DMA transfers occur between writing 0 and 1.

Enable writing to the FMR6 register.

Set to EW1 mode.

Disable writing to the FMR6 register.

Issue the block erase command.
Write 0x0020 in the first bus cycle. Write in word units.
Issue the clear status command.

Erase result

*ers_addr ← 0x50
Issue clear status command.

Erase successful (FMR07 is 0)

*ers_addr ← 0x00D0
Write 0x00D0 in the second bus cycle. Write in word units.

Erase failed (FMR07 is 1)

Set the FMR01 bit to 0 (CPU rewrite mode disabled).

Set the FMR01 bit to 0 (CPU rewrite mode disabled).

Return the CPU clock to its original setting and set the PM17 bit to 0 (no wait inserted).

Restore the flag register.

Note:
1. The frequency of CPU clock that can be used in CPU rewrite mode (EW0, EW1 mode) is different according to the product. Confirm the manual about details.
4.4.8 Record Write

```c
data_write(unsigned short *write_data)
program_result ← COMPLETE
asm("pushc flg")
Which block used?

Block A used
Set the block A starting address to WA
(write_addr).
Calculate WA (write_addr).
asm("fclr I")
Set the CPU clock.
cpu_slow
fmr0 ← 0x01
fmr0 ← 0x03
fmr1 ← 0x82
fmr6 ← 0x03
fmr1 ← 0x80
i ← 0
Record write completed?

Yes (i ≥ RECORD_SIZE)

Issue program command
command_write

Auto-program operation completed?

No (FMR00 bit is 0)
Yes (FMR00 bit is 1)
Program result
Program successful (FMR06 bit is 0)
i ← i + 2
write_addr ← 0x50
write_result ← DATA_PROGRAM_ERR
asm("pushc flg")
return program_result

fmr0 ← 0x01
Set the CPU clock
cpu_fast
asm("popc flg")
return program_result

Program failed (FMR06 bit is 1)

Set the block B starting address to WA
(write_addr).
Block B used

calculation method
Write address + write record (write_record) x record size (32 words)
Disable interrupts.

Set the CPU clock to less than 10 MHz[1] and set the PM17 bit to 1 (one wait inserted). When setting the FMR01 bit to 1 (CPU rewrite mode enabled), write 1 immediately after writing 0. Do not generate an interrupt between writing 0 and writing 1.

Disable writing to the FMR6 register.
Set to EW1 mode.

Disable writing to the FMR6 register.
Initialize loop counter i.

Note:
1. The frequency of CPU clock that can be used in CPU rewrite mode (EW0, EW1 mode) is different according to the product. Confirm the manual about details.
4.4.9 Issue Program Command

Argument
*addr: Starting address of data flash for writing data.
<data: Starting address of data for writing record.

Disable interrupts.

Issue the program command.
Write 0x0041 in the first bus cycle (write in word units).

Issue the program command.
Write data in the second bus cycle (16 low-order bits) and the third bus cycle (16 high-order bits).
Write in word units.

Restore the flag register.

```c
command_write
(unsigned short *addr,unsigned short *data)

asm("pushc flg")
asm("fclr I")
*addr ← 0x0041
*addr ← *data
*addr ← (data + 1)
asm("popc flg")
return
```

*addr: Starting address of data flash for writing data.
*data: Starting address of data for writing record.

Store the flag register.
5. **Sample Code**

Sample code can be downloaded from the Renesas Electronics website.

6. **Reference Documents**

- M16C/63 Group User’s Manual: Hardware Rev.1.00
- M16C/64 Group User’s Manual: Hardware Rev.1.05
- M16C/64A Group User’s Manual: Hardware Rev.1.10
- M16C/64C Group User’s Manual: Hardware Rev.0.10
- M16C/65 Group User’s Manual: Hardware Rev.1.10
- M16C/65C Group User’s Manual: Hardware Rev.0.10
- M16C/6C Group User’s Manual: Hardware Rev.1.00
- M16C/5LD Group, M16C/56D Group User’s Manual: Hardware Rev.1.10
- M16C/5L Group, M16C/56 Group User’s Manual: Hardware Rev.1.00
- M16C/5M Group, M16C/57 Group User’s Manual: Hardware Rev.1.01

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.

7. **Website and Support**

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

Inquiries
http://www.renesas.com/inquiry
## Revision History

<table>
<thead>
<tr>
<th>Rev.</th>
<th>Date</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.00</td>
<td>Oct. 30, 2009</td>
<td>First edition issued</td>
</tr>
<tr>
<td>1.01</td>
<td>Dec. 28, 2010</td>
<td>Add M16C/64, M16C/64C, M16C/65C, M16C/5LD, M16C/56D, M16C/5L, M16C/56, M16C/5M, M16C/57</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Corrected note 1 of figure 3.1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&quot;4.4.7 Block Erase&quot; was added note 1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&quot;4.4.8 Record Write&quot; was added note 1</td>
</tr>
</tbody>
</table>
General Precautions in the Handling of MPU/MCU Products

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

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

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

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

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

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

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

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

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

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

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

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

7. Renesas Electronics products are classified according to the following three quality grades: “Standard,” “High Quality,” and “Specific.” The recommended applications for each Renesas Electronics product depend on the product’s quality grade, as indicated below. You must check the quality grade of each Renesas Electronics product before using it in a particular application. You may not use any Renesas Electronics product for any application categorized as “Specific” without the prior written consent of Renesas Electronics. Further, you may not use any Renesas Electronics product for any application for which it is not intended without the prior written consent of Renesas Electronics. Renesas Electronics shall not be in any way liable for any damages incurred by you or third parties arising from the use of any Renesas Electronics product for an application categorized as “Specific” or for which the product is not intended where you have failed to obtain the prior written consent of Renesas Electronics.

8. The quality grade of each Renesas Electronics product is “Standard” unless otherwise expressly specified in a Renesas Electronics data sheets or data books, etc.

9. “Standard” — Computer; office equipment; communications equipment; test and measurement equipment; audio and visual equipment; home electrical appliances; machine tools; personal electronic equipment; and industrial robots.

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

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

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

Notice

Renesas Electronics Corporation

SALES OFFICES

Renesas Electronics America Inc.
2980 Scott Boulevard Santa Clara, CA 95050-2544, U.S.A.
Tel: +1-408-588-6000, Fax: +1-408-588-6130

Renesas Electronics Canada Limited
1117 Richardson Road North York, Ontario L3Y 9C3, Canada
Tel: +1-905-898-3441, Fax: +1-905-898-3200

Renesas Electronics Corporation
41-1, Yagiyama, Hachioji, Tokyo 192-0031, Japan
Tel: +81-42-568-5000, Fax: +81-42-568-5001

Renesas Electronics Corporation
9-8-11, Akasaka, Minato-ku, Tokyo 107-8032, Japan
Tel: +81-3-3589-3500, Fax: +81-3-3589-3501

Renesas Electronics Corporation
5-5-21, Akasaka, Minato-ku, Tokyo 107-0061, Japan
Tel: +81-3-3528-8000, Fax: +81-3-3528-8001

Renesas Electronics Corporation

Colophon 1.0

© 2010 Renesas Electronics Corporation. All rights reserved.