1. **Abstract**

   This application note describes the setting method for low current consumption in the M16C/63 Group.

2. **Introduction**

   The application example described in this document applies to the following microcomputer (MCU):
   
   - MCU: M16C/63 Group

   This application note can be used with other M16C Family MCUs which have the same special function registers (SFRs) as the above group. Check the user's manual for any modifications to functions. Careful evaluation is recommended before using the program described in this application note.
3. **Outline**

The amount of current consumption correlates with the number of operating clocks and frequency. If there are fewer operating clocks and a lower frequency, current consumption will be low. Sub clock division and f1 provide disabled function are added for the low consumption system in the M16C/63 Group.

This section introduces a method to reduce current consumption.

3.1 **Clock Operating Modes**

3.1.1 **Normal Operating Mode**

In normal operating mode, the CPU clock and the peripheral function clocks are both supplied, thus the CPU and the peripheral functions are operating. Power control is exercised by controlling the CPU clock frequency. The higher the CPU clock frequency, the higher the processing capability. The lower the CPU clock frequency, the lower the power consumption in the chip. If unnecessary oscillators are stopped, current consumption is further reduced.

3.1.2 **Wait Mode**

In wait mode, the CPU clock stops, then the CPU, watchdog timer, and NMI/SD digital filter also stop as they are operated by the CPU clock. However, if the CSPRO bit in the CSPR register is 1 (count source protection enabled), the watchdog timer remains active. Since the clock oscillator does not stop, peripheral functions which are provided peripheral function clocks keep operating.

**Peripheral Function Clock Stop Function:**
When the CM02 bit in the CM0 register is 1 (peripheral function clock f1 stops in wait mode), the f1 clock stops while in wait mode, then current consumption is reduced. With the exception of f1, peripheral clocks (i.e. fOCO40M, fOCO-F, fOCO-S, fC, and fC32) are not stopped by the CM02 bit.

3.1.3 **Stop Mode**

In stop mode, all oscillators stop. Therefore the CPU clock and peripheral function clocks stop, then the CPU and peripheral functions using these clocks stop operating. The least amount of current is consumed in this mode. If the voltage applied to pins VCC1 and VCC2 is VRAM or greater, the contents of the internal RAM are retained. When applying 1.8 V or less to pins VCC1 and VCC2, make sure VCC1 ≥ VCC2 ≥ VRAM.

The peripheral functions activated by external signals keep operating.
3.2 Power Control in Flash Memory

3.2.1 Stopping Flash Memory

When the flash memory is stopped, current consumption is reduced. While the flash memory is stopped, execute a program in any area other than the flash memory. Figure 3.1 shows the Stopping and Restarting the Flash Memory.

![Figure 3.1 Stopping and Restarting the Flash Memory](image)

1. Set the FMSTP bit to 1 after the FMR01 bit is set to 1.
2. Wait until the target clock stabilizes before switching the clock source of the CPU clock.
3. Add tps wait time by a program. Do not access the flash memory during this wait time.
3.2.2 Reading the Flash Memory

Current consumption while reading the flash memory can be reduced by setting bits FMR22 and FMR23 in the FMR2 register.

3.2.2.1 Slow Read Mode

Slow read mode can be used when f(BCLK) is slower than or equal to f(SLOW_R) and the PM17 bit in the PM1 register is 1 (1 wait). When the sub clock or 125 kHz on-chip oscillator clock is used as the clock source of the CPU clock, a wait is not necessary. (Technical update number: TN-16C-A179A/E).

Figure 3.2 shows Setting and Canceling Slow Read Mode.

![Figure 3.2 Setting and Canceling Slow Read Mode](image)

Do not set the FMR22 bit in the FMR2 register to 1 (slow read mode enabled) when the FMR01 bit in the FMR0 register is 1 (CPU rewrite mode enabled).
3.2.2.2 Low Current Consumption Read Mode

Low current consumption read mode can be used when the CM07 bit in the CM0 register is 1 (sub clock used as CPU clock). Figure 3.3 shows Setting and Canceling Low Current Consumption Read Mode.

![Diagram of Low Current Consumption Read Mode]

- Set the CM07 bit to 1 (sub clock).
- Set the CM05 bit in the CM0 register to 1 (main clock off), and the FRA00 bit in the FRA0 register to 0 (40 MHz on-chip oscillator off).
- Set the FMR22 bit in the FMR2 register to 0, and then to 1 (slow read mode enabled).
- Set the FMR23 bit in the FMR2 register to 0, and then to 1 (low current consumption read mode enabled).
- Processing in low current consumption read mode:
  - Write 0 to the FMR22 bit.
  - Write 0 to the FMR23 bit.
- End

**Note:**
1. Do not rewrite bits FMR22 and FMR23 simultaneously.

**Figure 3.3 Setting and Canceling Low Current Consumption Read Mode**

Do not enter wait mode from low current consumption read mode. To enter wait mode from this mode, set the FMR23 bit in the FMR2 register to 0 (low current consumption read mode disabled).

Do not enter wait mode from CPU rewrite mode. To enter wait mode from this mode, set the FMR01 bit in the FMR0 register to 0 (CPU rewrite mode disabled), and disable DMA transfer.

Do not enter stop mode from low current consumption read mode. To enter stop mode from this mode, set the FMR23 bit in the FMR2 register to 0 (low current consumption read mode disabled).

Do not enter stop mode from CPU rewrite mode. To enter stop mode from this mode, set the FMR01 bit in the FMR0 register to 0 (CPU rewrite mode disabled), and disable DMA transfer.

Enter low current consumption read mode through slow read mode.

When the FMR23 bit in the FMR2 register is 1 (low current consumption read mode enabled), do not set the FMSTP bit to 1 (flash memory stopped). Also, when the FMSTP bit is 1, do not set the FMR23 bit to 1. When the FMR01 bit in the FMR0 register to 1 (CPU rewrite mode enabled), do not set the FMR23 bit in the FMR2 register to 1 (low current consumption read mode enabled).
3.3  Reducing Current Consumption
To reduce current consumption, refer to the descriptions below when designing a system or writing a program.

3.3.1  Ports
The MCU retains the state of each I/O port even when it enters wait mode or stop mode. A current flows in the active output ports. A shoot-through current flows to the input ports in the high-impedance state. When entering wait mode or stop mode, first set unused ports to input and stabilize the potential.

3.3.2  A/D Converter
When not performing A/D conversion, set the PUMPON bit in the ADCON1 register to 0 (voltage multiplier off) and the ADSTBY bit in the ADCON1 register to 0 (A/D operation stop). Also, before entering wait mode or stop mode, fix analog pins to the stabilized potential.

3.3.3  D/A Converter
When not performing D/A conversion, set the DAiE bit in the DACON register to 0 (output disabled) and the DAi register to 00h (i = 0, 1).

3.3.4  Stopping Peripheral Functions
Use the PCLKSTP1 register to disable providing f1 to the peripheral functions not using f1. Use the CM02 bit in the CM0 register to stop unnecessary peripheral functions while in wait mode.

3.3.5  Switching the Oscillation-Driving Capacity
Set the driving capacity to low when oscillation is stable.
4. Application Example

4.1 Settings in the Sample Code

The sample code configures the settings below from the settings for low current consumption described in 3. Outline.
Table 4.1 lists the Settings of Sample Code.

Table 4.1 Settings of Sample Code

<table>
<thead>
<tr>
<th>Item</th>
<th>When in Low Current Consumption Mode</th>
<th>When in Normal Operating Mode</th>
</tr>
</thead>
<tbody>
<tr>
<td>Program location</td>
<td>RAM</td>
<td>ROM</td>
</tr>
<tr>
<td>CPU clock</td>
<td>Sub clock divided-by-2</td>
<td>Main clock (no division)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(low drive capacity)</td>
</tr>
<tr>
<td>Main clock</td>
<td></td>
<td>Oscillating</td>
</tr>
<tr>
<td>125 kHz on-chip oscillator</td>
<td>Stopped</td>
<td>Stopped (oscillating only after reset)</td>
</tr>
<tr>
<td>40 MHz on-chip oscillator</td>
<td></td>
<td>Stopped</td>
</tr>
<tr>
<td>Sub clock</td>
<td>Oscillating</td>
<td>Oscillating</td>
</tr>
<tr>
<td>Peripheral clocks (real-time clock, timers, UART, remote control signal receiver, A/D converter, SIO, pulse width modulator, multi-master I2C)</td>
<td>f1 provision disabled</td>
<td></td>
</tr>
<tr>
<td>Flash memory operation</td>
<td>Stopped</td>
<td>Operating</td>
</tr>
</tbody>
</table>

Settings for the other peripheral functions are as follows:
• A/D converter: A/D operation stopped, voltage multiplier off
• D/A converter: D/A0 output disabled, D/A1 output disabled
• Unused ports: Input mode (pull-down (1))

Note:
1. Pull-down is performed externally in the sample code.
4.2 Sample Code Operation

The following shows the sample code operation.

1. Initialize the CPU and peripheral functions.
2. Wait until the INTO interrupt occurs.
3. Transfer the program used in the RAM.
4. Set the start address of the relocatable vector table for the RAM in the INTB register.
5. Jump to the RAM, and execute the program.
   1. Set the sub clock as the CPU clock divide-by-2.
   2. Stop the main clock and on-chip oscillator.
   3. Enable CPU rewrite mode.
   4. Stop the flash memory, and set to low current consumption status.
   5. Wait until the INTO interrupt occurs.
   6. Enable flash memory operation. (1)
   7. Disable CPU rewrite mode.
   8. Wait until the flash memory circuit becomes stable.
   9. Set the main clock (no division) as the CPU clock after the main clock oscillates.
6. Set the start address of relocatable vector table for the ROM in the INTB register, and return to (2).

Note:
1. When enabling flash memory operation (FMSTP bit is set to 0), wait for tps or longer after the flash memory stops (FMSTP bit is set to 1).
   tps: Wait time until the flash memory circuit stabilizes.

Figure 4.1 shows the Operation Outline.
## 4.3 Function Tables

<table>
<thead>
<tr>
<th>Declaration</th>
<th>void main(void)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Outline</td>
<td>Main function</td>
</tr>
<tr>
<td>Argument</td>
<td>None</td>
</tr>
<tr>
<td>Variable (global)</td>
<td>None</td>
</tr>
<tr>
<td>Returned value</td>
<td>None</td>
</tr>
<tr>
<td><strong>Function</strong></td>
<td>When the INT0 interrupt occurs, the program written in the internal flash memory is transferred to the RAM. Then the transferred program on the RAM is executed.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Declaration</th>
<th>void mcu_init(void)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Outline</td>
<td>CPU initialization</td>
</tr>
<tr>
<td>Argument</td>
<td>None</td>
</tr>
<tr>
<td>Variable (global)</td>
<td>None</td>
</tr>
<tr>
<td>Returned value</td>
<td>None</td>
</tr>
<tr>
<td><strong>Function</strong></td>
<td>Set the main clock (no division) as the CPU clock.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Declaration</th>
<th>void peripheral_init(void)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Outline</td>
<td>Peripheral low current consumption setting</td>
</tr>
<tr>
<td>Argument</td>
<td>None</td>
</tr>
<tr>
<td>Variable (global)</td>
<td>None</td>
</tr>
<tr>
<td>Returned value</td>
<td>None</td>
</tr>
<tr>
<td><strong>Function</strong></td>
<td>Stop providing f1 to the peripheral functions not using f1. Set XIN-XOUT drive capacity to low, the sub clock to divided-by-2, and oscillate the sub clock. Set ports, the A/D converter and the D/A converter.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Declaration</th>
<th>void stopping_flash_memory(void)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Outline</td>
<td>Flash memory stop/operate function</td>
</tr>
<tr>
<td>Argument</td>
<td>None</td>
</tr>
<tr>
<td><strong>Variable (global)</strong></td>
<td>Type</td>
</tr>
<tr>
<td></td>
<td>unsigned char</td>
</tr>
<tr>
<td>Returned value</td>
<td>None</td>
</tr>
<tr>
<td><strong>Function</strong></td>
<td>Enable CPU rewrite mode, disable the flash memory operation, and set the mode to low current consumption mode. Enable the flash memory operation after the INT1 interrupt occurs.</td>
</tr>
</tbody>
</table>
### Declaration void cpu_slow(void)

#### Outline
System clock slow function

#### Argument
None

#### Variable (global)
None

#### Returned value
None

#### Function
- Set the sub clock as the CPU clock.
- Stop the main clock and on-chip oscillator.

### Declaration void cpu_fast(void)

#### Outline
System clock fast function

#### Argument
None

#### Variable (global)
None

#### Returned value
None

#### Function
- Oscillate the main clock by setting the CM05 bit in the CM0 register.
- After the main clock oscillation has been confirmed by the CM23 bit in the CM2 register 10 times continuously, set the main clock (no division) as the CPU clock.

### Declaration void send_to_ram(void)

#### Outline
Send to RAM function

#### Argument
None

#### Variable (global)
None

#### Returned value
None

#### Function
Transfer the flash memory stop/operate function, system clock slow function, and system clock fast function to the RAM.

### Declaration void send_to_ram_vector(void)

#### Outline
Send to RAM vector function

#### Argument
None

#### Variable (global)
None

#### Returned value
None

#### Function
Transfer an interrupt handler used in the RAM.

### Declaration void renewal_of_ram_vector_t(void)

#### Outline
Renewal of RAM vector table function

#### Argument
None

#### Variable (global)
None

#### Returned value
None

#### Function
Create the relocatable vector table used in the RAM.
### Declaration

```
void asm_smovf(void_far * _source, void _near * _dest, unsigned int _size)
```

### Outline

RAM transfer function

### Argument

<table>
<thead>
<tr>
<th>Argument name</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>void_far * _source</td>
<td>Transferred source address (program)</td>
</tr>
<tr>
<td>void _near * _dest</td>
<td>Transferred destination address (RAM area)</td>
</tr>
<tr>
<td>unsigned int _size</td>
<td>Transfer size</td>
</tr>
</tbody>
</table>

### Variable (global)
None

### Returned value
None

### Function

Transfer the specified area to the RAM area.

---

### Declaration

```
void ram_int_dummy(void)
```

### Outline

RAM interrupt dummy function

### Argument
None

### Variable (global)
None

### Returned value
None

### Function

Dummy function for the RAM. Add a processing if needed.

---

### Declaration

```
void ram_int_int1(void)
```

### Outline

RAM INT1 interrupt function

### Argument
None

### Variable (global)

<table>
<thead>
<tr>
<th>Type</th>
<th>Variable name</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>unsigned char</td>
<td>flg_wait_int1_int</td>
<td>For checking an INT1 interrupt occurrence</td>
</tr>
</tbody>
</table>

### Returned value
None

### Function

Invert port P0_0.
### 4.4 Flowcharts

#### 4.4.1 Main Processing

Start

- I flag ← 0
- mcu_init()
- CPU initialization
- peripheral_init()
- Peripheral low current consumption setting

int0ic ← 0x00

- INT0 interrupt occurred?
  - Yes (ir_int0ic is 1)
    - send_to_ram()
    - Send to RAM function
    - send_to_ram_vector()
    - Send to RAM vector function
    - renewal_of_ram_vector_t()
    - Renewal of RAM vector table function
    - asm("PUSHC INTBH");
    - asm("PUSHC INTBL");
    - Set the start address of relocatable vector table for RAM in the INTB register.
    - p = (void (*)(void))ram_p;
    - (*p());
    - asm("POPC INTBL");
    - asm("POPC INTBH");

  - No (ir_int0ic is 0)
    - Clear INT0 interrupt request bit.
    - Transfer the interrupt handler used in the RAM.
    - Create the relocatable vector table used in the RAM.
    - Store the INTB register value.
    - Execute the flash memory stop/operate function in the RAM.
    - Restore the INTB register value.

- CPU clock: main clock with no division (20 MHz)
- Set the following: XIN-XOUT drive capacity to low, f1 provide disabled, sub clock to divided-by-2, sub clock to oscillate, ports, A/D converter, and D/A converter.
4.4.2 Peripheral Low Current Consumption Setting

```
void peripheral_init(void)
{
  prcr ← 01h
  pckstp1 ← 7Fh
  cm15 ← 0
  scm0 ← 01h
  cm04 ← 1
  cm03 ← 0

  Wait until sub clock oscillation stabilizes
  prcr ← 00h

  Ports initialization
  pumpon ← 0
  adstby ← 0
  da0e ← 0
  da1e ← 0
  da0 ← 00h
  da1 ← 00h

  return
}
```

Enable write access to registers CM0, CM1, CM2, PCLKR, SCM0, PCLKSTP1, and FRA0.

Stop providing f1 to the peripheral functions not using f1.

Set XIN-XOUT drive capacity to low.

Set sub clock to divided-by-2.

Set XCIN-XCOUT oscillation function.

Start oscillating XCIN clock.

Wait for approximately 1 second.

Disable write access to registers CM0, CM1, CM2, PCLKR, SCM0, PCLKSTP1, and FRA0.

Initialize unused ports.

Set voltage multiplier off.

Stop A/D operation.

Disable D/A0 output.

Disable D/A1 output.

Set D/A0 output value to 0.

Set D/A1 output value to 0.
### 4.4.3 Flash Memory Stop/Operate Function

```c
void stopping_flash_memory(void)
{
    cpu_slow();  // System clock slow function

    fmr01 ← 0;  // Enable CPU rewrite mode
    fmr01 ← 1;  // Disable flash memory operation
    fnstp ← 1;  // Set INTT interrupt check flag

    int1ic ← 01h;  // INTT interrupt priority level 1
    asm("fset i");  // Interrupt enabled
    asm("fclr i");  // Interrupt disabled

    fmr01 ← 0;  // Enable CPU rewrite mode
    fnstp ← 0;  // Disable flash memory operation

    cpu_fast();  // System clock fast function
    return;
}
```

- **Set the sub clock as the CPU clock.**
  - Main clock stopped, on-chip oscillator stopped.
- **Enable CPU rewrite mode**
- **Disable flash memory operation**
- **Set INTT interrupt check flag**
- **INTT interrupt priority level 1**
- **Interrupt enabled**
  - **Interrupt handler**
    - `ram_int_int1`
    - Execute the interrupt handler transferred to the RAM
  - **Interrupt disabled**
    - **INTT interrupt disabled**
    - **Flash memory operation**
    - **CPU rewrite mode disabled**
  - **Wait until the flash memory circuit stabilizes 50 µs or longer**
  - **Main clock oscillating**
    - Set the main clock (no division) as the CPU clock
    - `cpu_fast()`
4.4.4 System Clock Slow Function

```c
void cpu_slow(void)
{
  prcr ← 03h
  cm07 ← 1
  cm05 ← 1
  cm14 ← 1
  fra00 ← 0
  pm17 ← 1
  prcr ← 00h
  return
}
```

Enable write access to registers CM0, CM1, CM2, PCLKR, and FRA0.
Enable write access to registers PM0, PM1, PM2, TB2SC, INVC0, and INVC1.
Set the sub clock as the CPU clock.
Stop the main clock.
Stop 125 kHz on-chip oscillator.
Stop 40 MHz on-chip oscillator.
Set wait state (1 wait).
Disable write access to registers CM0, CM1, CM2, PCLKR, and FRA0.
Disable write access to registers PM0, PM1, PM2, TB2SC, INVC0, and INVC1.

4.4.5 System Clock Fast Function

```c
void cpu_fast(void)
{
  cm05 ← 0
  cm15 ← 0
  cm21 ← 0
  cm07 ← 0
  cm06 ← 0
  cm1 ← cm1 & 3Fh
  pm17 ← 0
  prcr ← 00h
  return
}
```

Enable write access to registers CM0, CM1, CM2, PCLKR, SCM0, PCLKSTP1, and FRA0.
Enable write access to registers PM0, PM1, PM2, TB2SC, INVC0, and INVC1.
Start oscillating main clock.
Main clock oscillation confirmed 10 times continuously?
Yes (ocs_count ≥ 10)
Main clock oscillates?
Yes (CM23 is 0)
ocs_count++
Increment the counter.
No (CM23 is 1)
ocs_count ← 0
Initialize the counter.
No (ocs_count < 10)
Main clock oscillates?
Wait for 5 ms
Yes
Set the XIN-XOUT drive capacity to low.
Set the main clock as the CPU clock.
Set main clock to no division.
Set to no wait state.
Disable write access to registers CM0, CM1, CM2, PCLKR, SCM0, PCLKSTP1, and FRA0.
Disable write access to registers PM0, PM1, PM2, TB2SC, INVC0, and INVC1.
### 4.4.6 Send to RAM Function

```c
void send_to_ram(void)
{
    size ← (unsigned long)send_to_ram - (unsigned long)stopping_flash_memory + 1

    asm_smovf((void far *)stopping_flash_memory,
               (void near *)ram_p, ((unsigned int)size/2))

    return
}
```

Calculate sizes of the flash memory stop/start function, system clock slow function, and system clock fast function that are transferred to the RAM.

Transfer the flash memory stop/start function, system clock slow function, and system clock fast function to the RAM.

### 4.4.7 Send to RAM Vector Function

```c
void send_to_ram_vector(void)
{
    size ← (unsigned long)dummy - (unsigned long)ram_int_dummy + 1

    asm_smovf((void far *)ram_int_dummy,
               (void near *)ram_vector,
               ((unsigned int)size/2))

    return
}
```

Calculate size of an interrupt handler for RAM that is transferred to the RAM.

Transfer the interrupt handler for RAM to the RAM.

### 4.4.8 Renewal of RAM Vector Table Function

```c
void renewal_of_ram_vector_t(void)
{
    i ← 0

    Yes (i ≥ 64)
    Completed creating relocatable vector table for RAM?
    No (i < 64)
    ram_vector_table[i] ← offset_table_for_ram[i] + (unsigned long)&ram_vector[0]
    i++

    return
}
```

Initialize the counter.

Write the address of the interrupt handler for RAM in the relocatable vector table for RAM.

Increment the counter.
4.4.9 RAM Transfer Function

void asm_smovf(void _far * _source, void _near * _dest, unsigned int _size)

- Store registers R1 and A0.
- Transfer lower 16 bits in the source address (R0) to the A0 register.
- Transfer upper 4 bits in the source address (R2) to the R1 register.
- Transfer upper 4 bits in the source address (R1L) to the R1H register.
- Transfer to the RAM area.
- Restore registers R1 and A0.

4.4.10 RAM INT1 Interrupt Function

void ram_int_int1(void)

- Invert the port P0_0 output.
- Set the port P0_0 to an output port.
- Clear the flag for checking the INTT interrupt.
5. **Sample Code**
Sample code can be downloaded from the Renesas Electronics website.

6. **Reference Documents**
M16C/63 Group User’s Manual: Hardware Rev. 2.00
The latest version can be downloaded from the Renesas Electronics website.

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

C Compiler Manual
M16C Series/R8C Series C Compiler Package V.5.45
C Compiler User’s Manual Rev. 2.00
The latest version can be downloaded from the Renesas Electronics website.

**Website and Support**
Renesas Electronics website
http://www.renesas.com/

Inquiries
http://www.renesas.com/inquiry
<table>
<thead>
<tr>
<th>Rev.</th>
<th>Date</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.00</td>
<td>Aug. 31, 2010</td>
<td>First edition issued</td>
</tr>
<tr>
<td>1.01</td>
<td>Oct 29, 2010</td>
<td>修改Flash Memory的停止和重启方式。修订了第3图。</td>
</tr>
<tr>
<td></td>
<td></td>
<td>第4图：取消慢速读模式的设置和取消的说明。</td>
</tr>
<tr>
<td></td>
<td></td>
<td>第5图：取消低电流消耗读模式的设置和取消的说明。</td>
</tr>
<tr>
<td>1.02</td>
<td>Dec. 28, 2011</td>
<td>修订程序规格。</td>
</tr>
<tr>
<td></td>
<td></td>
<td>删除慢速读/等待功能。</td>
</tr>
<tr>
<td></td>
<td></td>
<td>添加并放置在慢速读/等待功能原来的位置。</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3.1.2等待模式：更改部分描述。</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3.1.3停止模式：修改第一句话。</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3.1.2等待模式：更改部分描述。</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3.1.3停止模式：修改第一句话。</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3.2.2.1慢速读模式：添加第二句话并在图3.2下方。</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3.2.2.2低电流消耗读模式：重写描述。</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3.2.2.2低电流消耗读模式：重写描述。</td>
</tr>
<tr>
<td></td>
<td></td>
<td>4.1设置在示例代码中：更改包括表格4.1。</td>
</tr>
<tr>
<td></td>
<td></td>
<td>4.2示例代码操作：更改包括图4.1。</td>
</tr>
<tr>
<td></td>
<td></td>
<td>4.3功能表：更改mcu_init()的Outline。</td>
</tr>
<tr>
<td></td>
<td></td>
<td>更改Peripheral_init()的Outline和Function。</td>
</tr>
<tr>
<td></td>
<td></td>
<td>更改ram_int_int1()。</td>
</tr>
<tr>
<td></td>
<td></td>
<td>4.4.1主处理：更改Peripheral_init()的说明。</td>
</tr>
<tr>
<td></td>
<td></td>
<td>4.4.10 RAM INT1中断功能：添加最后一个过程，flg_wait_int1_int ← 0。</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 any prior notice. Before purchasing or using any Renesas Electronics products listed herein, please confirm the latest product information with a Renesas Electronics sales office. Also, please pay regular and careful attention to additional and different information to be disclosed by Renesas Electronics such that disclosed through our websites.

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

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

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

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

6. Renesas Electronics has used reasonable care in preparing the information included in this document, but Renesas Electronics does not warrant that such information is error free. Renesas Electronics assumes no liability whatsoever for any damages incurred by you resulting from errors 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 or losses incurred by you or third parties arising from the use of any Renesas Electronics product for an application categorized as "Specific" or for which the product is not intended where you have failed to obtain the prior written consent of Renesas Electronics.

The quality grade of each Renesas Electronics product is "Standard" unless otherwise expressly specified in a Renesas Electronics data sheet or data books, etc.

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

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

"Specific": Aircraft; aerospace equipment; submarine repeaters; nuclear reactor control systems; medical equipment or systems for life support (e.g. artificial life support devices or systems); surgical implantation, or other health products that pose a direct threat to human life.

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

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

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

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

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

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

(Note 2) “Renesas Electronics products” means any product developed or manufactured by or for Renesas Electronics.