

To our customers,

---

## Old Company Name in Catalogs and Other Documents

---

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

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

April 1<sup>st</sup>, 2010  
Renesas Electronics Corporation

Issued by: Renesas Electronics Corporation (<http://www.renesas.com>)

Send any inquiries to <http://www.renesas.com/inquiry>.

## Notice

1. All information included in this document is current as of the date this document is issued. Such information, however, is subject to change without any prior notice. Before purchasing or using any Renesas Electronics products listed herein, please confirm the latest product information with a Renesas Electronics sales office. Also, please pay regular and careful attention to additional and different information to be disclosed by Renesas Electronics such as that disclosed through our website.
2. Renesas Electronics does not assume any liability for infringement of patents, copyrights, or other intellectual property rights of third parties by or arising from the use of Renesas Electronics products or technical information described in this document. No license, express, implied or otherwise, is granted hereby under any patents, copyrights or other intellectual property rights of Renesas Electronics or others.
3. You should not alter, modify, copy, or otherwise misappropriate any Renesas Electronics product, whether in whole or in part.
4. Descriptions of circuits, software and other related information in this document are provided only to illustrate the operation of semiconductor products and application examples. You are fully responsible for the incorporation of these circuits, software, and information in the design of your equipment. Renesas Electronics assumes no responsibility for any losses incurred by you or third parties arising from the use of these circuits, software, or information.
5. When exporting the products or technology described in this document, you should comply with the applicable export control laws and regulations and follow the procedures required by such laws and regulations. You should not use Renesas Electronics products or the technology described in this document for any purpose relating to military applications or use by the military, including but not limited to the development of weapons of mass destruction. Renesas Electronics products and technology may not be used for or incorporated into any products or systems whose manufacture, use, or sale is prohibited under any applicable domestic or foreign laws or regulations.
6. Renesas Electronics has used reasonable care in preparing the information included in this document, but Renesas Electronics does not warrant that such information is error free. Renesas Electronics assumes no liability whatsoever for any damages incurred by you resulting from errors in or omissions from the information included herein.
7. Renesas Electronics products are classified according to the following three quality grades: “Standard”, “High Quality”, and “Specific”. The recommended applications for each Renesas Electronics product depends on the product’s quality grade, as indicated below. You must check the quality grade of each Renesas Electronics product before using it in a particular application. You may not use any Renesas Electronics product for any application categorized as “Specific” without the prior written consent of Renesas Electronics. Further, you may not use any Renesas Electronics product for any application for which it is not intended without the prior written consent of Renesas Electronics. Renesas Electronics shall not be in any way liable for any damages or losses incurred by you or third parties arising from the use of any Renesas Electronics product for an application categorized as “Specific” or for which the product is not intended where you have failed to obtain the prior written consent of Renesas Electronics. The quality grade of each Renesas Electronics product is “Standard” unless otherwise expressly specified in a Renesas Electronics data sheets or data books, etc.
  - “Standard”: Computers; office equipment; communications equipment; test and measurement equipment; audio and visual equipment; home electronic appliances; machine tools; personal electronic equipment; and industrial robots.
  - “High Quality”: Transportation equipment (automobiles, trains, ships, etc.); traffic control systems; anti-disaster systems; anti-crime systems; safety equipment; and medical equipment not specifically designed for life support.
  - “Specific”: Aircraft; aerospace equipment; submersible repeaters; nuclear reactor control systems; medical equipment or systems for life support (e.g. artificial life support devices or systems), surgical implantations, or healthcare intervention (e.g. excision, etc.), and any other applications or purposes that pose a direct threat to human life.
8. You should use the Renesas Electronics products described in this document within the range specified by Renesas Electronics, especially with respect to the maximum rating, operating supply voltage range, movement power voltage range, heat radiation characteristics, installation and other product characteristics. Renesas Electronics shall have no liability for malfunctions or damages arising out of the use of Renesas Electronics products beyond such specified ranges.
9. Although Renesas Electronics endeavors to improve the quality and reliability of its products, semiconductor products have specific characteristics such as the occurrence of failure at a certain rate and malfunctions under certain use conditions. Further, Renesas Electronics products are not subject to radiation resistance design. Please be sure to implement safety measures to guard them against the possibility of physical injury, and injury or damage caused by fire in the event of the failure of a Renesas Electronics product, such as safety design for hardware and software including but not limited to redundancy, fire control and malfunction prevention, appropriate treatment for aging degradation or any other appropriate measures. Because the evaluation of microcomputer software alone is very difficult, please evaluate the safety of the final products or system manufactured by you.
10. Please contact a Renesas Electronics sales office for details as to environmental matters such as the environmental compatibility of each Renesas Electronics product. Please use Renesas Electronics products in compliance with all applicable laws and regulations that regulate the inclusion or use of controlled substances, including without limitation, the EU RoHS Directive. Renesas Electronics assumes no liability for damages or losses occurring as a result of your noncompliance with applicable laws and regulations.
11. This document may not be reproduced or duplicated, in any form, in whole or in part, without prior written consent of Renesas Electronics.
12. Please contact a Renesas Electronics sales office if you have any questions regarding the information contained in this document or Renesas Electronics products, or if you have any other inquiries.

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

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

## **M16C/62**

### **Using the M16C/62 Power Saving Modes**

---

#### **1.0 Abstract**

This article discusses the various power saving modes of the M16C device. A short program is provided that can be run on the MSV3062 development board. Using an Amp meter connected to a special header, it is possible to measure only the current that the device uses. Thus the power consumption can be read, with the device in various clock modes and operating modes.

#### **2.0 Introduction**

Several power saving modes are critical, especially for battery-operated products. Products need to operate at full speed when required, as well as be able to go into the lowest power consuming sleep state. Intermediate states are also necessary for tasks not requiring full speed, thus conserving power.

#### **3.0 Power Conservation: Introduction**

Power conservation is accomplished three ways. The main clock can operate from the high speed or low speed crystal at various divided speeds (gears). WAIT mode is the next feature that allows power to be conserved, while still letting the peripherals function. STOP mode is the lowest power conserving mode, drawing the least amount of current, yet allowing the device to wake up and go back to full speed.

#### **4.0 M16C: Introduction**

The M16C family is designed from the beginning for low power consumption. These devices are single-chip microcomputers, built using the high-performance silicon gate CMOS process technology. They use sophisticated instructions featuring a high level of instruction efficiency. With 1M bytes of address space (/62 series), up to 16M bytes of address space (/80 series), and operation voltages from 2.7V to 5V, they are capable of executing from 16 MHz to 20 MHz. Other features include single-voltage Flash, a built-in multiplier, DMAC, 10 channels of A/D, and over 5 channels of serial I/O, making them ideal for industrial equipment, communications, and other high-speed processing applications requiring careful power management.

##### **4.1 M16C: WAIT Mode**

When a WAIT instruction is executed, the BCLK stops and the microcomputer enters WAIT mode. In this mode, oscillation continues but the BCLK and watchdog timer stop. Writing "1" to the WAIT peripheral function clock stop bit and executing a WAIT instruction stops the clock being supplied to the internal peripheral functions, allowing power dissipation to be reduced. However, peripheral function clock fC32 does not stop, so the peripherals using fC32 do not contribute to the power saving. When the MCU is running in low-speed or low

power dissipation mode, do not enter WAIT mode with this bit set to “1”. Table 1 shows the status of the ports in WAIT mode.

**Table 1 Port status during WAIT mode**

Pin		Memory expansion mode Microprocessor mode	Single-chip mode
Address bus, data bus, CS0, CS3, BHE		Retains status before WAIT mode	
RD, WR, WRL, WRH		“H”	
HLDA, BCLK		“H”	
ALE		“H”	
Port		Retains status before WAIT mode	Retains status before WAIT mode
CLK <sub>OUT</sub>	When fc selected	Valid only in single-chip mode	Does not stop
	When f <sub>8</sub> , f <sub>32</sub> selected	Valid only in single-chip mode	Does not stop when the WAIT peripheral function clock stop bit is “0”.  When the WAIT peripheral function clock stop bit is “1”, the status immediately prior to entering WAIT mode is maintained.

WAIT mode is cancelled by a hardware reset or an interrupt. If an interrupt is used to cancel WAIT mode, that interrupt must first have been enabled, and the priority level of the interrupts that are not used to cancel must have been changed to 0. If returning by an interrupt, the clock during which the WAIT instruction executed is set to BCLK by the microcomputer and the action is resumed from the interrupt routine. If only a hardware RESET or an NMI interrupt is used to cancel WAIT mode, change the priority level of all interrupts to 0, and then shift to WAIT mode.

#### 4.2 M16C: STOP Mode

Writing “1” to the all-clock stop control bit (bit 0 at address 000716) stops all oscillation and the microcomputer enters STOP mode. In STOP mode, the content of the internal RAM is retained provided that VCC remains above 2V.

Because the oscillation, BCLK, f<sub>1</sub> to f<sub>32</sub>, f<sub>1SIO2</sub> to f<sub>32SIO2</sub>, f<sub>C</sub>, f<sub>C32</sub>, and f<sub>AD</sub>, stops in STOP mode, peripheral functions such as the A-D converter and watchdog timer do not function. However, timer A and timer B operate provided that the event counter mode is set to an external pulse, and UART<sub>i</sub>(i = 0 to 2), SI/O<sub>3,4</sub> also function provided an external clock is selected. Table 2 shows the status of the ports in STOP mode.

**Table 2 Port status during STOP mode**

Pin		Memory expansion mode Microprocessor mode	Single-chip mode
Address bus, data bus, CS0, CS3, BHE		Retains status before STOP mode	
RD, WR, WRL, WRH		“H”	
HLDA, BCLK		“H”	
ALE		“H”	
Port		Retains status before STOP mode	Retains status before STOP mode
CLK <sub>OUT</sub>	When f <sub>c</sub> selected	Valid only in single-chip mode	“H”
	When f <sub>8</sub> , f <sub>32</sub> selected	Valid only in single-chip mode	Retains status before STOP mode

STOP mode is cancelled by a hardware reset or an interrupt. If an interrupt is to be used to cancel STOP mode, that interrupt must first have been enabled, and the priority level of the interrupt that is not used to cancel must have been changed to 0. If returning by an interrupt, that interrupt routine is executed. If only a hardware RESET or an NMI interrupt is used to cancel STOP mode, change the priority level of all interrupts to 0, and then shift to STOP mode.

When shifting from high-speed/medium-speed mode to STOP mode and at a reset, the main clock division select bit 0 (bit 6 at address 000616) is set to “1”. When shifting from low-speed/low power dissipation mode to STOP mode, the value before STOP mode is retained.

#### 4.3 M16C: Main Clock

The main clock is generated by the main clock oscillation circuit. After a reset, the clock is divided by 8 to the BCLK. The clock can be stopped using the main clock stop bit (bit 5 at address 000616). Stopping the clock, after switching the operating clock source of CPU to the sub-clock, reduces the power dissipation.

After the oscillation of the main clock oscillation circuit has stabilized, the drive capacity of the main clock oscillation circuit can be reduced using the XIN-XOUT drive capacity select bit (bit 5 at address 000716). Reducing the drive capacity of the main clock oscillation circuit reduces the power dissipation. This bit changes to “1” when shifting from high-speed/medium-speed mode to STOP mode and at a reset. When shifting from low-speed/low power dissipation mode to STOP mode, the value before STOP mode is retained.

#### 4.4 M16C: Sub-clock

The sub-clock is generated by the sub-clock oscillation circuit. No sub-clock is generated after a reset. After oscillation is started using the port XC select bit (bit 4 at address 000616), the sub-clock can be selected as the BCLK by using the system clock select bit (bit 7 at address 000616). However, be sure that the sub-clock oscillation has fully stabilized before switching.

After the oscillation of the sub-clock oscillation circuit has stabilized, the drive capacity of the sub-clock oscillation circuit can be reduced using the XCIN-XCOUT drive capacity select bit (bit 3 at address 000616). Reducing the

drive capacity of the sub-clock oscillation circuit reduces the power dissipation. This bit changes to “1” when shifting to STOP mode and at a reset.

When XCIN/XCOUT is used, set ports P86 and P87 as the input ports without pull-up. Also, when using the low drive option with both XIN and XCIN, take caution when designing your crystal circuit, especially when adding dampening resistors. The low drive option was designed to reduce the current consumption of the oscillator circuits, but if too much dampening resistance is introduced into the feedback loop, reliable oscillation of the crystal may not be possible.

#### **4.5 M16C: BCLK**

The BCLK is the clock that drives the CPU. The speed of this clock can be altered as a division of the main clock by 1, 2, 4, 8, or 16. After reset, the speed of BCLK is set to the main clock divided by 8 by default. The BCLK signal can be output from the BCLK pin by the BCLK output disable bit (bit 7 at address 000416) in the memory expansion and the microprocessor modes.

The main clock division select bit 0 (bit 6 at address 000616) changes to “1” when shifting from high-speed/medium-speed to STOP mode and at reset. When shifting from low-speed/low power dissipation mode to STOP mode, the value before STOP mode is retained.

#### **4.6 M16C: Peripheral Function Clock (f1, f8, f32, f1SIO2, f8SIO2, f32SIO2, fAD)**

The clock for the peripheral devices is derived from the main clock or by dividing it by 1, 8, or 32. The peripheral function clock is stopped by stopping the main clock or by setting the WAIT peripheral function clock stop bit (bit 2 at 000616) to “1” and then executing a WAIT instruction.

#### **4.7 M16C: fC32 Clock**

This clock is derived by dividing the sub-clock by 32. It is used for the timer A and timer B counts.

#### **4.8 M16C: fC Clock**

This clock has the same frequency as the sub-clock. It is used for the BCLK and for the watchdog timer.

#### **4.9 M16C: Clock Output Function**

In single-chip mode, the clock output function select bits (bits 0 and 1 at address 000616) enable f8, f32, or fc to be output from the P57/CLKOUT pin. When the WAIT peripheral function clock stop bit (bit 2 at address 000616) is set to “1”, the output of f8 and f32 stops when a WAIT instruction is executed.

#### **4.10 M16C: Status Transition of BCLK**

Power dissipation can be reduced and low-voltage operation achieved by changing the count source for BCLK. Table 3 shows the operating modes corresponding to the settings of system clock control registers 0 and 1.

**Table 3 Port status during STOP mode**

CM17	CM16	CM07	CM06	CM05	CM04	Operating mode of BCLK
0	1	0	0	0	Invalid	Division by 2 mode
1	0	0	0	0	Invalid	Division by 4 mode
Invalid	Invalid	0	1	0	Invalid	Division by 8 mode
0	1	0	0	0	Invalid	Division by 16 mode
1	0	0	0	0	Invalid	No-division mode
Invalid	Invalid	1	Invalid	0	1	Low-speed mode
Invalid	Invalid	1	Invalid	1	1	Low power dissipation mode

When reset, the device starts in division by 8 mode. The main clock division select bit 0 (bit 6 at address 000616) changes to “1” when shifting from high-speed/medium-speed to STOP mode and at a reset. When shifting from low-speed/low power dissipation mode to STOP mode, the value before STOP mode is retained.

The following are the operational modes of BCLK:

1. Division by 2 mode  
The main clock is divided by 2 to obtain the BCLK.
2. Division by 4 mode  
The main clock is divided by 4 to obtain the BCLK.
3. Division by 8 mode  
The main clock is divided by 8 to obtain the BCLK. When reset, the device starts operating from this mode. Before the user can go from this mode to no-division mode, division by 2 mode, or division by 4 mode, the main clock must be oscillating stably. When going to low-speed or lower power consumption mode, make sure the sub-clock is oscillating stably.
4. Division by 16 mode  
The main clock is divided by 16 to obtain the BCLK.
5. No-division mode  
The main clock is divided by 1 to obtain the BCLK.
6. Low-speed mode  
fC is used as the BCLK. Note that oscillation of both the main and sub-clocks must have stabilized before transferring from this mode to another or vice versa. At least 2 to 3 seconds are required after the sub-clock starts. Therefore, the program must be written to wait until this clock has stabilized immediately after powering up and after STOP mode is cancelled.
7. Low power dissipation mode  
fC is the BCLK and the main clock is stopped.

---

**Note:** Before the count source for BCLK can be changed from XIN to XCIN or vice versa, the clock to which the count source is going to be switched must be oscillating stably. Allow a wait time in software for the oscillation to stabilize before switching over the clock.

---

## 5.0 Power Control Modes

Power control is available in three modes. A description of each mode follows.

### 1. Normal operation mode

- High-speed mode

Divide-by-1 frequency of the main clock becomes the BCLK. The CPU operates with the BCLK. Each peripheral function operates according to its assigned clock.

- Medium-speed mode

Divide-by-2, divide-by-4, divide-by-8, or divide-by-16 frequency of the main clock becomes the BCLK. The CPU operates with the BCLK. Each peripheral function operates according to its assigned clock.

- Low-speed mode

fC becomes the BCLK. The CPU operates according to the fC clock. The fC clock is supplied by the sub-clock. Each peripheral function operates according to its assigned clock.

- Low power dissipation mode

The main clock operating in low-speed mode is stopped. The CPU operates according to the fC clock. The fC clock is supplied by the sub-clock. The only peripheral functions that operate are those with the sub-clock selected as the count source.

### 2. WAIT mode

The CPU operation is stopped. The oscillators do not stop.

### 3. STOP mode

All oscillators stop. The CPU and all built-in peripheral functions stop. This mode, among the three modes listed here, is the most effective in decreasing power consumption.

## 5.1 Power Control Tips

Listed here are the precautions to take when using power saving modes.

1. The processor will not switch to STOP mode when the NMI pin is at "L" level.
2. When returning from STOP mode by hardware reset, the RESET pin must be held at "L" level until main clock oscillation is stabilized.
3. When the MCU is running in low-speed or low power dissipation mode, do not enter WAIT mode with the WAIT peripheral function clock stop bit set to "1".
4. When switching to either WAIT mode or STOP mode, instructions occupying four bytes either from the WAIT



instruction or from the instruction that sets the every-clock stop bit to “1”, within the instruction queue, are prefetched and then the instruction stops. So put at least four NOPs in succession after either the WAIT instruction or the instruction that sets the every-clock stop bit to 1.

5. Before the count source for BCLK can be changed from Xin to Xcin or vice versa, the clock to which the count source is going to be switched must be oscillating stably. Allow a wait time in software for the oscillation to stabilize before switching over the clock.

The following are suggestions to reduce power consumption.

1. **Ports:** When entering WAIT or STOP mode, set nonused ports to input and stabilize the potential. The processor retains the state of each programmable I/O port even when it goes to WAIT mode or STOP mode. A current flows in active I/O ports. A pass current flows in input ports that float.
2. **A-D Converter:** A current always flows in the Vref pin. When entering WAIT mode or STOP mode, set the Vref connection bit to “0”, so that no current flows into the Vref pin.
3. **D-A Converter:** The processor retains the D-A state even when entering WAIT mode or STOP mode. Disable the output from the D-A converter; then configure I/O ports as specified above.
4. **Stopping peripheral functions:** In WAIT mode, stop nonused WAIT peripheral functions using the peripheral function clock stop bit.
5. **Switching the oscillation-driving capacity:** Set the driving capacity to “LOW” when oscillation is stable.
6. **External Clock:** When using an external clock input for the CPU clock, set the main clock stop bit to “1”. Setting the main clock stop bit to “1” causes the Xin-Xout pins to stop oscillating and the Xout pin to go to a high-level state and the power consumption goes down (when using an external clock input, the clock signal is input regardless of the content of the main clock stop bit).
7. **Protection Register:** The Processor mode registers (PM0 & PM1) and the Clock mode registers (CM0 & CM1) are protected by the Protection register. So be sure to remove protection before writing and return protection when finished.

## 6.0 Implementation: Hardware

The MSV3062 development board will be configured as follows:

1. JP13 (MVcc jumper) - Cut trace between pins on back of board.
2. Connect an Amp meter between pins.
3. Use the SW2 (INT1) button to exit STOP mode.

## 7.0 Implementation: Software

The low power program is written in C, and developed using the IAR embedded workbench. This program will cycle through these modes:

1. Xin / 1 → LED displays "1"
2. Xin / 2 → LED displays "2"
3. Xin / 4 → LED displays "3"
4. Xin / 8 → LED displays "4"
5. Xin / 16 → LED displays "5"
6. Xcin / 1 → LED displays "6"
7. WAIT mode → LED displays "7"
8. STOP mode → LED displays "0"

Each mode will run for about 5 seconds before switching to the next mode. Exit WAIT mode, by the 5-second timer interrupt, or by using the "SW2" button on the board (P8.3). Exit STOP mode only by pushing the "SW2" button.

Timer A0 cascades into Timer A1, which interrupts every 5 seconds.

The processor cycles through all modes in high drive, and then cycles through all modes in low drive and repeats.

Note that the port pins are configured appropriately before entering WAIT or STOP mode. The port pin configuration will change depending on your application.

## 8.0 Reference

**Renesas Technology Corporation Semiconductor Home Page**

<http://www.renesas.com>

**E-mail Support**

[support\\_apl@renesas.com](mailto:support_apl@renesas.com)

**Data Sheets**

- M16C/62 datasheets, 62aeds.pdf

## 9.0 Software Code

Here is the lowpower.c file, function `__low_level_init()`.

```

unsigned char __low_level_init (void) { // Safe Power-Up:- port usage, MCU mode &
                                        clock speed
    PUR2=0xFF; PUR1=0xFF; PUR0=0xFF;    // Pull-ups: P10,P9,P8,P7,P6,P5,P4,P3,P2,P1,P0
    POD=0;P1D=0;P2D=0;P3D=0;P4D=0;P5D=0; // Inputs (no external bus)
    #if defined(MINI)| defined(SKIT2)
    P6D=0x80;P7D=0;P8D=0;P10D=0;PCR.0=1; // Inputs (P0 NOT latched), except Txd1
    #else
    P6D=8;P7D=0;P8D=0;P10D=0;PCR.0=1;   // Inputs (P0 NOT latched), except Txd0
    #endif
    PRCR=0x07;                            // Unlock:- P9D, PM, CM
    P9D =0x00;                             // P9 inputs
    PM  =0x0040|(PM&~0x4708);              //P4=port,Wait,Bclk,NoMux,R/W, MUST preserve:- >15K&>192K & MCU-mode
    CM  =0x2000|(CM&~0xFFE0);              // cpu=X/1, Hi-drive, Preserve all-else eg P5.7
    PRCR=0x00;                             // Lock
    CSR =0x00;                             // One wait:-CS3,2,1,0, no-o/p:-CS3,2,1,0
    return 1;
}

```

## 9.1 Software Code

Here is the lowpower.c file, function `cpu_gear()`.

```

int cpu_gear(int gear) { // Read or Write CPU clock gear. NB No intelligence, assumes necessary clocks
                            running
    if (gear==~0000) {      // (-1) => Read current setting
        switch(CM & 0xC0C0) { // CM.15.14.7.6
            case 0x0000:      gear=1; break; // X/1
            case 0x4000:      gear=2; break; // X/2
            case 0x8000:      gear=4; break; // X/4
            case 0x0040: case 0x4040: case 0x8040: case 0xC040: gear=8; break; // X/8
            case 0xC000:      gear=16; break; // X/16
            case 0x0080: case 0x4080: case 0x8080: case 0xC080:
            case 0x00C0: case 0x40C0: case 0x80C0: case 0xC0C0: gear=32; break; // Xc/1
            default: ; }
    } else {                  // Otherwise assume: Write
                                new setting
        disable_interrupt(); PRCR.0=1; // Unlock
        switch(gear) {        // CM.15.14.7.6.8.4
            case 0: enable_interrupt();CM.8=1; NOP;NOP;NOP;NOP;NOP; break; // STOP (X=Off, Xc=Off)
            case 1: led_seg=0xFD; led_digit=0xCF; CM=(CM & 0x3FBF)|0x0000; CM.7=0; break; // X/1
            case 2: led_seg=0xFB; led_digit=0xA4; CM=(CM & 0x3FBF)|0x4000; CM.7=0; break; // X/2
            case 4: led_seg=0xF7; led_digit=0xB0; CM=(CM & 0x3FBF)|0x8000; CM.7=0; break; // X/4
            case 8: led_seg=0xEF; led_digit=0x99; CM=(CM & 0x3FBF)|0x0040; CM.7=0; break; // X/8
            case 16: led_seg=0xDF; led_digit=0x92; CM=(CM & 0x3FBF)|0xC000; CM.7=0; break; // X/16
            case 32: if (mode!=WAIT){led_seg=0xBF; led_digit=0x82;} CM.7=1; break; // Xc/1, display "6" if not
                                                                WAIT mode.
            default: ; } PRCR.0=0; enable_interrupt(); // relock
        } return gear;
    }
}

```

## 9.2 Software Code

Here is the `lowpower.c` file, function `main()`.

```

void main(void) {
    int     lps;                // low power state
    P0=0;P1=0;P2=0;P3=0;P4=0;P5=0;P6=0;P7=0;P8=0;P9=0;P10=0; // Force to low current outputs
    P0D=0xFF;P3D=0xFF;P4D=0xFF;P5D=0xFF; // Disable 87 x 50K pullups => too much
                                        // current!

#ifdef MINI | defined(SKIT2)
    P6D=0xBF;P7D=0xFF;PRCR.2=1;P9D=0xFF;PRCR.2=0;P10D=0xFF; // UART1 for debugger, P9D locked
#else
    P6D=0xCB;P7D=0xFF;PRCR.2=1;P9D=0xFF;PRCR.2=0;P10D=0xFF; // for MDECE30222 board
                                        // UART0 for debugger, P6.4&5 Buttons,
                                        // P9D locked
#endif

    P1D=0xFF; P1=0xFF; // P1: MSA0650 LEDs, off; MSA0654 LEDs(P1.0, .1), off
    P2D=0xFF; P2=0xFF; // P2: MSA0651 LEDs, off
    P8D      = 0x77 ; // P8.3: Button on MSA0654, P8.6=Xcout,P8.7=Xcin
    PUR2     = 0xFC; // P8: NO-pullups
    P5D.7    = 1; // P5.7: CLKout: f32 or fc
    P5D.3    = 1; // P5.3: BCLK output

//Setup Timer A0 in timer mode
    TABSR.0  = 0; // TA0: Stop
    TA0MR    = 0x80; // TA0: Timer mode, f=X/32
    UDF.0    = 0; // TA0: Count downwards
    TA0IC    = 0x05; // TA0: IRQ level 5 ie non-active
    NOP;NOP;TA0IC.3=0; // TA0: Clear any pending IRQ

//Setup Timer A1 in event counter mode
    TABSR.1  = 0; // TA1: Stop
    TA1MR    = 0x01; // TA1: Event mode
    TRGSR=0x02|(TRGSR&~0x03); // TA1: Count TA0 overflows
    UDF.1    = 0; // TA1: Count downwards
    TA1IC    = 0x07; // TA1: IRQ level 7 ie active
    TA1IC.3=0;NOP;NOP;NOP;NOP; // TA1: Clear any pending IRQ

    ADCON1.5 = 0; // ADC: Vref NOT connected
    INT1IC    = 6; // P8.3 INT1 interrupt level 6
    INT1IC.4  = 0; // P8.3 INT1 falling edge
    IFSR.1    = 0; // P8.3 INT1 two edges
    INT1IC.3=0; NOP;NOP;NOP;NOP; // INT1: Clear any pending IRQ
    write_ip1(5); // CPU: Allow interrupt levels >=5
    NOP;NOP;enable_interrupt(); // CPU
    highCurrent=CM.13; lps=0;cpu_gear(~0); // read current Low Power State
    dbgBrk(); // Script execution pauses here
    while(1) {
        if (CM.7) { UNLOCK();CM.5=0;CM.13=1;LOCK(); // Start X, high drive, wait ~250 microSec
            /*TABSR&=~3;TA0MR=0xC0;TA0=1;TA1=8; to=FALSE;TABSR|=3; while(!to);
            */ NOP;NOP;NOP;NOP;NOP;NOP;NOP;NOP; }
        else { UNLOCK(); CM.4=1;CM.3=1;LOCK(); // Start Xc, high drive, wait ~250 MILLI-Sec
            TABSR&=~3;TA0MR=0x80;TA0=5000;TA1=25;to=FALSE;TABSR|=3; while(!to); }
        switch(lps) { // Prepare low power request & resources for ~5 second pause
            case 1: gear= 2; nxt_lps=2; break;
            case 2: gear= 4; nxt_lps=3; break;
            case 3: gear= 8; nxt_lps=4; break;
            case 4: gear=16; nxt_lps=5; break;
            case 5: gear=32; nxt_lps=6; TA0MR=0xC0;TA0=100; break;
            case 6: gear=32; nxt_lps=7; TA0MR=0xC0;TA0=100; mode=WAIT;
                highCurrent=~highCurrent; led_seg=0xFC; led_digit=0xF8; break; // Push button to release WAIT mode
            case 7: gear= 0; nxt_lps=0; TA0MR=0xC0;TA0=100; mode=STOP;
        }
    }
}

```

```

    led_seg=0xFE; led_digit=0xC0; break; // Push button to release STOP mode
default:gear= 1; nxt_lps=1; P8.0=0;break; }
CPSRF.7=1; if (gear)cpu_gear(gear); UNLOCK(); // Start shutdown sequence f=X/?, unless STOP mode
if (highCurrent==1) { PM.15=0;PM.7=0;P1.0 = 0;P1.1=1; } // No Global-wait, Output BCLK,
display on LED1
else { P1.0=1; P1.1 = 0; PM.15=1;PM.7=1; CM.13=0;CM.3=1; } // Global-wait, No BCLK, X,Xc:Both
low-drive, display on LED2
if (gear==32) CM.5=1; else CM.4=0; // Stop: unused X or unused Xc
if (gear==32) CM=(CM&~2)|1; else CM|=3; LOCK(); // P5.7: output f32(f=X) or fc(f=Xc)
to=FALSE;TABS|=3; while(!to) { // Pause in each low power mode
    P0.7=CM.5;P0.6=~CM.4;P0=((P0&0x80)|(led_digit&0x7F)); // LEDs: P0=MSA0654, display a digit
    if (mode==STOP) cpu_gear(0); else // Stop(X,Xc) => RTI
    if (mode==WAIT) {UNLOCK();CM.2=1;LOCK(); wait_for_interrupt(); NOP;NOP;NOP;NOP;NOP;}
// CPU WAIT mode => RTI
} lps=nxt_lps; P0=0xFF;P1=0xFF;P2=P1; // Indicate ready for next low power state
}
}

```

### 9.3 Software Code

Here is the `lowpower.c` file, interrupt functions.

```

interrupt [21*4] void Int_TA0(void) { } // Cascaded, do NOT enable this interrupt
interrupt [22*4] void Int_TA1(void) { to=TRUE; } // Called every 2 seconds
interrupt [29*4] void Int_INT0(void) { }
interrupt [30*4] void Int_INT1(void) { P8.0=1;to=TRUE; nxt_lps=0; highCurrent=1; }

```

## Keep safety first in your circuit designs!

- Renesas Technology Corporation puts the maximum effort into making semiconductor products better and more reliable, but there is always the possibility that trouble may occur with them. Trouble with semiconductors may lead to personal injury, fire or property damage. Remember to give due consideration to safety when making your circuit designs, with appropriate measures such as (i) placement of substitutive, auxiliary circuits, (ii) use of nonflammable material or (iii) prevention against any malfunction or mishap.

## Notes regarding these materials

- These materials are intended as a reference to assist our customers in the selection of the Renesas Technology Corporation product best suited to the customer's application; they do not convey any license under any intellectual property rights, or any other rights, belonging to Renesas Technology Corporation or a third party.
- Renesas Technology Corporation assumes no responsibility for any damage, or infringement of any third-party's rights, originating in the use of any product data, diagrams, charts, programs, algorithms, or circuit application examples contained in these materials.
- All information contained in these materials, including product data, diagrams, charts, programs and algorithms represents information on products at the time of publication of these materials, and are subject to change by Renesas Technology Corporation without notice due to product improvements or other reasons. It is therefore recommended that customers contact Renesas Technology Corporation or an authorized Renesas Technology Corporation product distributor for the latest product information before purchasing a product listed herein.  
The information described here may contain technical inaccuracies or typographical errors. Renesas Technology Corporation assumes no responsibility for any damage, liability, or other loss arising from these inaccuracies or errors.  
Please also pay attention to information published by Renesas Technology Corporation by various means, including the Renesas Technology Corporation Semiconductor home page (<http://www.renesas.com>).
- When using any or all of the information contained in these materials, including product data, diagrams, charts, programs, and algorithms, please be sure to evaluate all information as a total system before making a final decision on the applicability of the information and products. Renesas Technology Corporation assumes no responsibility for any damage, liability or other loss resulting from the information contained herein.
- Renesas Technology Corporation semiconductors are not designed or manufactured for use in a device or system that is used under circumstances in which human life is potentially at stake. Please contact Renesas Technology Corporation or an authorized Renesas Technology Corporation product distributor when considering the use of a product contained herein for any specific purposes, such as apparatus or systems for transportation, vehicular, medical, aerospace, nuclear, or undersea repeater use.
- The prior written approval of Renesas Technology Corporation is necessary to reprint or reproduce in whole or in part these materials.
- If these products or technologies are subject to the Japanese export control restrictions, they must be exported under a license from the Japanese government and cannot be imported into a country other than the approved destination.  
Any diversion or reexport contrary to the export control laws and regulations of Japan and/or the country of destination is prohibited.
- Please contact Renesas Technology Corporation for further details on these materials or the products contained therein.