M16C/26

Power Modes

1.0 Abstract

The following article discusses the various power modes of the M30262 MCU. A sample program is available that runs on the MSV30262 SKP board for testing and evaluation. Using an ammeter connected to the JP1 header, one can measure the amount of current that the MCU consumes in various modes.

2.0 Introduction

The Renesas M30262 is a 16-bit MCU based on the M16C/60 series CPU core. The MCU features include up to 64KB of Flash ROM, 2KB of RAM, and 4KB of virtual EEPROM. The peripheral set includes 10-bit A/D, UARTs, Timers, DMA, and GPIO. Having several power modes are critical, especially for battery operated products. Depending on the application, products may need to operate at full speed operation at certain times and go into the lowest power-consuming mode at other times.

3.0 Operating modes

Power conservation can be accomplished in several ways and this varies from one application to another. For applications that need to use Normal Operation mode, intermediate modes (e.g. medium speed, low speed, etc.) are available for reducing power consumption. Another way is to use WAIT mode, which allows better power conservation while still allowing the use of peripherals. Because it has the lowest power consumption, STOP mode provides the best power conservation and yet, still allows the device to wake up and go back to desired operation.

3.1 Normal Operation Mode

In Normal Operation Mode, M30262 operates using the main clock or sub-clock. After reset, the MCU is in Normal Operation mode with the main clock driving BCLK, the clock that drives the CPU. While in operation, BCLK can be switched from the main clock to the sub-clock to reduce power consumption. The operational modes under Normal Operation mode are shown in Table 1.

Table 1 Operational modes in Normal Operation Mode

Mode	BCLK	Peripheral Clock
High Speed	**f1**	Assigned clock (main clock derivatives or **fc**)
Medium Speed	**f2, f4, f8, f16**	Assigned clock (main clock derivatives or **fc**)
Low Speed	**fc**	Assigned clock (main clock derivatives or **fc**)
Low Power Dissipation	**fc**	**fc** (only peripherals that can run with **fc**)* *

Note:

* BCLK is discussed in section 4.0.

** In Low Power Dissipation, the main clock is stopped and so only peripherals that can operate with the subclock fc will continue running.

3.2 WAIT mode

The M30262 goes into WAIT mode when a WAIT instruction is executed. In this mode, BCLK, the clock that drives the CPU, and the watchdog timer is stopped. However, peripherals may continue to operate. Writing a "1" to the WAIT peripheral function clock stop bit of System Clock Control Register 0 (bit 2 at address 0006_{16}) before executing a WAIT instruction stops the clock being supplied to the internal peripheral functions, allowing more power conservation. However, if peripheral function clock fc32 is used is NOT stopped and so will not contribute to power conservation. Table 2 shows the status of the ports in WAIT mode.

Note: When using the low-speed or low power dissipation mode, do NOT enter WAIT mode with the WAIT peripheral function clock stop bit set to "1".

Table 2 Port status during WAIT mode

Pin		Single chip mode
Port		Retains status before WAIT mode
CLK_{OUT}	When fc selected	Does not stop
	When f_8, f_{32} selected	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.

Hardware reset or an interrupt can be used to exit from WAIT mode. If an interrupt is used to exit from WAIT mode, the interrupt must be enabled (and all other maskable interrupts disabled) before executing the wait instruction. If returning by an interrupt, the MCU will resume operation from the interrupt routine with the same BCLK as the BCLK when the WAIT instruction was executed. If hardware RESET or an NMI (non-maskable interrupt – a hardware interrupt) is used to exit from WAIT mode, disable interrupts prior to executing the wait instruction.

3.3 STOP mode

In STOP mode, BCLK (f1 to f32, fC, and fC32) and peripheral clocks (f1SIO2 to f32SIO2, and fAD) are stopped including the watchdog timer with the following exceptions:

- Timers A and B that operate in Event Counter mode with an external pulse input
- UARTi (i = 0 to 2) that uses external clock inputs

In STOP mode, the content of the internal RAM is retained as long as VCC remains above 2V. Table 3 shows the status of the ports in STOP mode.

Writing a "1" to the ***all-clock stop control bit*** of System Clock Control Register 1 (bit 0 at address 0007_{16}) stops all oscillation and the M30262 enters STOP mode. Hardware reset or interrupt can be used to exit from STOP mode, similar to WAIT mode. If an interrupt is to be used to exit from STOP mode, the interrupt must be enabled before entering STOP mode. If returning by an interrupt, program resumes from the interrupt routine. If hardware RESET or an NMI interrupt is used to exit from STOP mode, ensure that all maskable interrupts are disabled before shifting to STOP mode.

Table 3 Port status during STOP mode

Pin		**Single chip mode**
Port		Retains status before STOP mode
CLK_{OUT}	When fc selected	"H"
	When f_8, f_{32} selected	Retains status before STOP mode

4.0 M30262 Clocks

4.1 BCLK

BCLK is the clock that drives the CPU. It can be driven by the main clock or sub-clock. When using the main clock, aside from full speed (main clock frequency), BCLK can be set to intermediate frequencies, which are derivatives of the main clock that may contribute in reducing MCU power consumption.

4.1.1 Main Clock

The main clock is generated by the main clock oscillation circuit. After a reset, the clock is divided by 8, f8, to the BCLK. To reduce power consumption, the clock can be stopped using the main clock stop bit (bit 5 at address 0006_{16}) after switching BCLK to the sub-clock.

After the oscillation of the main clock oscillation circuit has stabilized, reducing the drive capacity of the main clock oscillation circuit using the ***X_{IN}-X_{OUT} drive capacity select bit*** of System Clock Control Register 1 (bit 5 at address 0007_{16}) can also reduce power consumption.

4.1.2 Sub-clock, fc

The sub-clock is generated by the sub-clock oscillation circuit. After reset, the sub-clock is disabled by default and the pins function as GPIO's. To start oscillation, set ***Port X_C select bit*** of System Clock Control Register 0 (bit 4 at address 0006_{16}) to 1. To switch BCLK to the sub-clock, set the ***System clock select bit*** of System Clock Control Register 0 (bit 7 at address 0006_{16}). However, ensure that the sub-clock oscillation has stabilized before switching.

After the oscillation of the sub-clock oscillation circuit has stabilized, reducing the drive capacity of the main clock oscillation circuit using the ***X_{CIN}-X_{COUT} drive capacity select bit*** of System Clock Control Register 0 (bit 3 at address 0006_{16}) can also reduce power consumption. This bit changes to "1" when shifting to STOP mode and at a reset.

The watchdog timer also uses the sub-clock. A derivative of the sub-clock is fc32, which is sub-clock frequency divided by 32.

Note: Before changing BCLK from X_{IN} to X_{CIN} or vice versa, clock oscillation must be stable. Allow a timeout period in software for oscillation to stabilize before switching the clock. Refer to the oscillator/crystal datasheet for details.

4.1.3 Status Transition of BCLK

As mentioned earlier, power dissipation can be reduced and low-voltage operation achieved by changing the count source for BCLK. Table 4 shows the operating modes corresponding to the settings of system clock control registers 0 and 1.

Table 4 BCLK operating modes

CM17	CM16	CM07	CM06	CM05	CM04	Operating mode of BCLK*	Clock Speed
0	1	0	0	0	Invalid	Division by 2 mode, f_2	Main Clock div by 2
1	0	Invalid	0	0	Invalid	Division by 4 mode, f_4	Main Clock div by 4
Invalid	Invalid	0	1	0	Invalid	Division by 8 mode (default after reset), f_8	Main Clock div by 8
1	1	0	0	0	Invalid	Division by 16 mode, f_{16}	Main Clock div by 16
0	0	0	0	0	Invalid	No-division mode	Main Clock
Invalid	Invalid	1	Invalid	0	1	Low-speed mode	Sub-clock
Invalid	Invalid	1	Invalid	1	1	Low power dissipation mode	Sub-clock

Note: The ***Main clock division select bit 0***, CM06, 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.

4.2 Peripheral Function Clock (f_1, f_8, f_{32}, f_{1SIO2}, f_{8SIO2}, f_{32SIO2}, f_{AD})

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 0006_{16}) to "1" and then executing a WAIT instruction.

5.0 Technical Notes in Power Modes

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

1. The MCU will not switch to STOP mode if the NMI pin is at "L" level (if using the NMI pin).
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 WAIT or STOP mode, due to the pre-fetch queue, instructions occupying four bytes either from the WAIT instruction or from the instruction that sets the ***All clock stop control bit*** to "1" are already fetched. So add at least four NOPs in succession after either the WAIT instruction or the instruction that sets the ***All clock stop control bit*** to 1.
5. 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.

The following are suggestions to reduce power consumption.

1. **Ports:** Set unused ports to outputs and set them to 0.
2. **A-D Converter:** Disconnect Vref, by setting the *Vref connect bit*, *VCUT*, to "0", if the ADC is unused or before going into WAIT or STOP mode as current always flows in the Vref pin.
3. **Oscillation Drive Capacity:** Set the driving capacity to "LOW" after oscillation stabilizes.
4. **External Clock:** When using an external clock as main clock, set the main clock stop bit to "1". Note, however, that the BCLK will only be at divided by 8 mode. 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).

6.0 Reference

Renesas Technology Corporation Semiconductor Home Page

http://www.renesas.com

E-mail Support

support_apl@renesas.com

Data Sheets

- M16C/26 datasheets, M30262eds.pdf

User's Manual

- M16C/20/60 C Language Programming Manual, 6020c.pdf
- M16C/20/60 Software Manual, 6020software.pdf
- Writing interrupt handlers in C for the M16C Application Note
- MSV30262-SKP or MSV-Mini26-SKP Quick start guide
- MSV30262-SKP or MSV-Mini26-SKP Users Manual
- MDECE30262 or MSV-Mini26-SKP Schematic

7.0 Software Code

The multi power mode program is written in C and compiled using the KNC30 compiler. The program starts initially at Xin / 8 mode and operation can be verified based on the user LEDs. From this mode, any of the following modes can be selected:

- Xin / 1 → press sw2 and Green is ON but Red & Yellow are OFF
- Xin / 2 → press sw3 and Yellow is ON but Red & Green are OFF
- Xin / 4 → press sw4 and Yellow & Green are ON but Red is OFF
- Xin / 8 → no switch is pressed and all user LEDs are ON
- Xin / 16 → press both sw2 & sw3 and Red is ON but Green & Yellow are OFF
- Xcin / 1 → press sw2 & sw3 & sw4 at the same time and Red & Yellow are ON but Green is OFF
- WAIT mode → press both sw3 & sw4 and Red & Green are ON but Yellow is OFF
- STOP mode → press sw2 & sw4 and all user LED's are OFF

```
/*****************************************************************************
*
*       File Name: main.c
*
*       Content:   This program operates a firefly MCU chip on MSV30262 board at
*                  various power modes. The default mode is given by
*                  main clock (Xin) divided by 8 mode and is reflected by lighting
*                   up all three LEDs (Red, Yellow and Green) simultaneously.
*                  Seven other modes are demonstrated by pressing switches sw2, sw3,
*                  sw4, sw2&sw3, sw3&4, sw2&sw3&sw4 and sw2&sw4.
*                  Respective power modes are:
*                  Xin, Xin/2, Xin/4, Xin/16, WAIT, Xcin and STOP. The light-up
*                  of the RED-YELLOW-GREEN led set signifies "1", "2", "3", "4",
*                  "5", "6" and "0" corresponding to the 7 modes mentioned abve.
*                  The default (Xin/8) mode shows a "7" by
*                  lighting up all three LEDs and the STOP mode has all three LEDs
*                  turned OFF. To exit the wait or the stop mode, press any of the
*                  three switches sw2, sw3 or sw4 and exit to default (Xin/8) mode
*                  when all three LEDs light up.
*
*       Date:  11-21-2002
*       This program was written to run on the MDECE30262 Board for MSV30262-SKP.
*
*       Copyright 2003 Renesas Technology America, Inc.
*       All rights reserved
*
*****************************************************************************
*       $Log:$
*****************************************************************************/
#include "sfr262.h"            /* M16C/26 special function register definitions*/
#pragma INTERRUPT KeyInput_ISR /* Interrupt Service Routine for Key Input */
                               /* and invoked by pressing any of the three */
                               /* switches sw2, sw3 or sw4 */

/* LEDs */
#define red_led                p7_0
#define yellow_led     p7_1
#define green_led      p7_2

/* SWITCHES */
#define sw2            p10_5
#define sw3            p10_6
#define sw4            p10_7

#define UNLOCK_CM_REG  prc0 = 1; /* unlock clock mode registers cm0 and cm1 */
#define LOCK_CM_REG    prc0 = 0; /* lock clock mode registers cm0 and cm1 */

#define ON             0       /* turns LED ON */
#define OFF            1       /* turns LED OFF */
```

```
void mcu_init(void);            /* routine for initializing mcu */
void enter_wait(void);          /* routine for invoking wait mode */
void enter_stop(void);          /* routine for invoking stop mode */
void KeyInput_ISR(void);        /* Interrupt Service routine for Key Input */
void generate_subClock(void);   /* Starts MCU sub clock */
void led_RYG(int,int,int);      /* controls LED operation */
void config_ports_adc(void);    /* configures unused ports and ADC for low power */

/* declare operating mode state variables */
int wait_mode;                  /* turns to 1 on entering wait mode */
int stop_mode;                  /* turns to 1 on entering stop mode */
int default_mode;               /* default Xin/8 mode after reset */
int i;                          /* a delay counting variable */

/*****************************************************************************
Name:           main
Parameters:     None
Returns:        None
Description:    main program loop and initialization
*****************************************************************************/
main() {
        mcu_init();             /* initialize mcu board */

        kupic = 4;              /* set Key Input Interrut priority level = 4 */

        _asm( "fset i" );       /* enable interrupts */

        /******************************** PROGRAM LOOP ********************************/
        while(1){
                /****************** Xin/8 mode ***************************************/
                /* No switch is pressed */
                /* MCU in default Xin/8 mode */
                if(sw2!= 0 && sw3!= 0 && sw4!= 0 && wait_mode==0 && stop_mode==0 && default_mode==1)
                {
                        /* turn on all LEDs of the RYG LED set */
                        led_RYG( ON, ON, ON);
                }

                /*********************** Xin mode ************************************/
                /* Press only sw2 */
                /* invoke sub-clock */
                /* run CPU at BCLK = Xin */
                else if (sw2 == 0 && sw3 != 0 && sw4 != 0 && wait_mode == 0 && stop_mode == 0)
                {
                                default_mode = 0;       /* no longer in default mode */
                                UNLOCK_CM_REG           /* un-lock cm0 and cm1 */
                                /* generate sub-clock */
                                generate_subClock();
```

```
                /* assign BCLK = Xin */
                cm07  = 0;      /* select system clock Xin/Xout */
                cm06  = 0;      /* enable cm16 and cm17  */
                cm16  = 0;      /* select no division clock mode */
                cm17  = 0;      /* select no division clock mode */
                LOCK_CM_REG      /* lock the system clock control register */

          /* turn ON only green LED of the R-Y-G LED set to display "1" */
                led_RYG( OFF, OFF, ON);
        }

/************************ Xin/2 mode ************************************/
/* Press only sw3 */
/* invoke sub-clock */
/* run CPU at BCLK = Xin/2 */
else if (sw2 != 0 && sw3 == 0 && sw4 != 0 && wait_mode == 0 && stop_mode == 0)
        {
                default_mode = 0;       /* no longer in default mode */
                UNLOCK_CM_REG           /* un-lock cm0 and cm1 */
                /* generate sub-clock */
                generate_subClock();

                /* assign BCLK = Xin/2 */
                cm07  = 0;              /* select system clock Xin/Xout */
                cm06  = 0;              /* enable cm16 and cm17  */
                cm16  = 1;              /* select division by 2 */
                cm17  = 0;              /* clock mode */
                LOCK_CM_REG     /* lock the system clock control register */

          /* turn ON only yellow LED of the R-Y-G LED set to display "2" */
                led_RYG( OFF, ON, OFF);
        }

/***************************** Xin/4 mode ********************************/
/* Press only sw4 */
/* invoke sub-clock */
/* run CPU at BCLK = Xin/4 */
else if (sw2 != 0 && sw3 != 0 && sw4 == 0 && wait_mode == 0 && stop_mode == 0)
        {
        default_mode = 0;       /* no longer in default mode */
                UNLOCK_CM_REG           /* un-lock cm0 and cm1 */

                /* generate sub-clock */
                generate_subClock();
```

```
                        /* assign BCLK = Xin/2 */
                        cm07  = 0;              /* select system clock Xin/Xout */
                        cm06  = 0;              /* enable cm16 and cm17  */
                        cm16  = 0;              /* select division by 2 */
                        cm17  = 1;              /* clock mode */
                        LOCK_CM_REG     /* lock the system clock control register */

                    /* turn ON yellow and green LEDs of the R-Y-G LED set to display "3"
*/
                        led_RYG( OFF, ON, ON);
                }

        /***************************** Xin/16 mode *******************************/
        /* Press both switches sw2 and sw3. */
        /* invoke sub-clock */
        /* run CPU at BCLK = Xin/16 */
        else if (sw2 == 0 && sw3 == 0 && sw4 != 0 && wait_mode == 0 && stop_mode == 0)
                {
                        default_mode = 0;       /* no longer in default mode */
                        UNLOCK_CM_REG           /* un-lock cm0 and cm1 */

                        /* generate sub-clock */
                        generate_subClock();

                        /* assign BCLK = Xin/2 */
                        cm07  = 0;              /* select system clock Xin/Xout */
                        cm06  = 0;              /* enable cm16 and cm17  */
                        cm16  = 1;              /* select division by 16 */
                        cm17  = 1;              /* clock mode */

                        LOCK_CM_REG     /* lock the system clock control register */

                    /* turn ON only red LED of the R-Y-G LED set to display "4" */
                                led_RYG( ON, OFF, OFF);
        /* allow delay time for switches to dis-engage (i.e., open). This delay takes*/
        /* care of the slight discrepancy that may occur while attempting to release */
        /* multiple switches simultaneously */
                for (i=0; i<0xffff; i++)                /* loop for sometime */
                        ;                       /* do nothing */

                }
```

```
                /***************************** Xcin mode ******************************/
                /* Press all switches sw2, sw3 and sw4. */
                /* invoke sub-clock */
                /* run CPU at BCLK = Xcin and enter Low-speed mode */
                /* NOTE: KD30 debugger may not respond properly at this low frequency */
                else if (sw2 == 0 && sw3 == 0 && sw4 == 0 && wait_mode == 0 && stop_mode == 0)
                        {
                                default_mode = 0; /* no longer in default mode */
                        UNLOCK_CM_REG     /* un-lock cm0 and cm1 */
                                cm03 = 1;          /* set Xcin/Xcout drive capacity to High */

                                /* generate sub-clock */
                                generate_subClock();

                                /* assign BCLK = Xcin */
                                cm07  = 1;        /* select system clock Xcin/Xcout */
                                LOCK_CM_REG      /* lock the system clock control register */

                          /* turn ON all LEDs of the R-Y-G LED set to display "6" */
                                        led_RYG( ON, ON, OFF);

                /* allow delay time for switches to dis-engage (i.e., open). This delay takes*/
                /* care of the slight discrepancy that may occur while attempting to release */
                /* multiple switches simultaneously */
                for (i=0; i<0x0fff; i++)           /* loop for sometime */
                                ;                          /* do nothing */
                        }

                /***************************** WAIT mode *********************************/
                /* Press both switches sw3 and sw4 to enter wait mode */
                else if (sw2 != 0 && sw3 == 0 && sw4 == 0 && wait_mode == 0 && stop_mode == 0)
                {
                                        enter_wait();
                }

                /***************************** STOP mode *********************************/
                /* Press both switches sw2 and sw4 to enter stop mode */
                else if (sw2 == 0 && sw3 !=0 && sw4 == 0 && wait_mode == 0 && stop_mode == 0)
                {
                        enter_stop();
                }
        }
}

/*****************************************************************************
Name:           mcu_init
Parameters:     None
Returns:                None
Description:    Initialization routine for the different MCU peripherals.
*****************************************************************************/
```

```
void mcu_init(void) {

        /* Mode state initialization */
        wait_mode = 0;                  /* Not in wait mode */
        stop_mode = 0;                  /* Not in stop mode */
        default_mode = 1;               /* start with default Xin/8 mode */

        /* LED initialization */
        pd7_0 = 1;              /* set LED ports to outputs (connected to LEDs) */
        pd7_1 = 1;
        pd7_2 = 1;

        /* configure port pins for Key Input interrupts */
        pd10_5 = 0;     /* set Key Input port pins of switches -2, -3 and -4 to inputs */
        pd10_6 = 0;
        pd10_7 = 0;
        pd10_4 = 1;     /* set the remaining (unused) Ket Input port pin to output for avoiding*/
                        /* undesirable (low) input signal that may cause unwanted interference
*/
        return;
}

/************************************************************************
Name:           enter_wait
Parameters:     None
Returns:                None
Description:    enters wait mode. stops BCLK and peripheral functional clock.
                Xin and Xcin remain active.
************************************************************************/
void enter_wait( void ){ /* enter wait mode; BCLK absent; Xin/Xcin present*/

        UNLOCK_CM_REG           /* unlock cm0 (and cm1) */
        cm02 = 1;               /* stop peripheral functional clock */
        LOCK_CM_REG             /* lock cm0 (and cm1) */
        config_ports_adc();     /* configure unused ports and ADC for low power */

        /* turn ON red and green LEDs in the R-Y-G LED set to display "5" */
        led_RYG( ON, OFF, ON);

        /* invoke wait mode */
        wait_mode = 1;          /* set wait mode variable just before entering wait mode */
        default_mode = 0;       /* no longer in default mode */
        _asm ( "wait" );        /* invoke wait mode */
}

/************************************************************************
Name:           enter_stop
Parameters:     None
Returns:                None
Description:    enters stop mode. All clocks stopped.
************************************************************************/
void enter_stop( void ){                /* enter stop mode: all clocks stopped */

        config_ports_adc();             /* configure unused ports and ADC for low power */
```

```
        /* turn OFF all LEDs in the R-Y-G LED set to display "0" */
        led_RYG( OFF, OFF, OFF);

        /* invoke wait mode */
        stop_mode = 1;          /* set stop mode variable just before entering stop mode */
        default_mode = 0;       /*no longer in default mode */

        UNLOCK_CM_REG           /* unlock cm0 and cm1 */
        cm10 = 1;               /* stop all clocks */
        LOCK_CM_REG             /* lock cm0 and cm1 */
}

/************************************************************************
Name:           KeyInput_ISR
Parameters:     None
Returns:                None
Description:    Key Input Interrupt Service Routine. Returns to Xin/8 mode from
                WAIT or STOP mode at the press of any of the three switches
                sw2, sw3 or sw4.
************************************************************************/
void KeyInput_ISR(void) { /* Return from wait or stop mode to medium speed (Xin/8) mode */
                        /* on pressing any of the three switches: sw2, sw3 or sw4  */

        if( wait_mode == 1 || stop_mode == 1 ) {       /* check if in wait_mode or in stop mode
*/
                _asm( "fset i" );                      /* enable interrupt for ROM monitor */

                /* set clock for default restart medium speed (div by 8) mode */
                UNLOCK_CM_REG           /* unlock cm0 (and cm1) */
                cm07 = 0;               /* select Xin clock */
                cm06 = 1;               /* select Xin/8 clock frequency */
                LOCK_CM_REG             /* lock cm0 (and cm1) */

                /* update wait/stop state variables */
                wait_mode = 0;          /* exited wait mode or */
                stop_mode = 0;          /* exited stop mode */
                default_mode = 1;       /* enter default Xin/8 mode */

/* allow delay time for switches to dis-engage (i.e., open). This delay takes*/
/* care of the slight discrepancy that may occur while attempting to release */
/* multiple switches simultaneously */
        for (i=0; i<0x0fff; i++)                /* loop for sometime */
                        ;                       /* do nothing */
        }
}
/************************************************************************
Name:           generate_subClock
Parameters:     None
Returns:                None
Description:    Starts up sub clock Xcin.
************************************************************************/
void generate_subClock(void){
```

```
                pd8_6 = 0;              /* set p8_6 as input to use Xcout */
                pd8_7 = 0;              /* set p8_7 as input to use Xcin  */
                pu21  = 0;              /* no pull-up in p8_6 and p8_7    */
                cm04  = 1;              /* Xcin/Xcout generation enabled */
                return;
}
/************************************************************************
Name:           led_RYG
Parameters:     int, int, int
Returns:                None
Description:    operates Red, Yellow and Green LEDs
************************************************************************/
void led_RYG( int RED, int YELLOW, int GREEN){
        red_led = RED;
        yellow_led = YELLOW;
        green_led = GREEN;
        return;
}
/************************************************************************
Name:           config_ports_adc
Parameters:     None
Returns:                None
Description:    Configures unused ports and AD converter for low power at WAIT or
                STOP mode. Sets unused output ports (P1, P6 and P9) to output and
                disconnects Vref of unused ADC.
************************************************************************/
void config_ports_adc(void){ /* configurations for low power */

        /* set unused ports to inputs */
        pd6 = 0x00;             /* p6_0 - p6_7 configured as input */
        pd1 = 0x00;             /* p1_5 - p1_7 configured as input */
        prc2= 1;                /* unlock pd9 protect register */
        pd9 = 0x00;             /* p9_1 - p9_4 configured as input */
        prc2= 0;                /* lock pd9 protect register */

        /* turn on relevant pull-ups */
        pu03 = 1;               /* p1_5 - p1_7 pulled high */
        pu14 = 1;               /* p6_0  - p6_3 pulled high */
        pu15 = 1;               /* p6_4  - p6_7 pulled high */
        pu22 = 1;               /* p9_0  - p9_3 pulled high */

        /* disconnect  unused ADC Vref */
        vcut = 0;               /* ADC Vref disconnected */
}
```

In order for this program to run properly, the key input interrupt vector needs to point to the interrupt function, KeyInput_ISR. Insert the function label "_KeyInput_ISR" into the interrupt vector table at vector 23 as shown below.

Key Input Interrupt vector assignment in Sect30_26.inc:

```
;-----------------------------------------------------------------
; variable vector section
;-----------------------------------------------------------------
        .section        vector                  ; variable vector table
        .org    VECTOR_ADR
        .lword  dummy_int               ; vector 0 (BRK)
        .org    (VECTOR_ADR +16)
                        :
                        :
        .glb    _KeyInput_ISR
        .lword  _KeyInput_ISR           ; Key-on wakeup (for user)

        .lword  dummy_int               ; AD Convert (for user)
        .lword  dummy_int               ; UART2 transmit/NACK, with iic mode NACK is selected.
                        :
                        :
```

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 rising 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.