To our customers,

Old Company Name in Catalogs and Other Documents

On April 1st, 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 1st, 2010
Renesas Electronics Corporation

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

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

RENESAS

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.

Application Note

V850ES/Jx3

Sample Program (Watchdog Timer 2 (WDT2))

Reset Generated by Infinite Loop Detection

This document summarizes the operations of the sample program and describes how to use the sample program and how to set and use watchdog timer 2. In the sample program, an interrupt is generated by detecting the falling edge of the switch input. If no WDT2 overflow occurs, LED1 blinks in a cycle of approximately 55 ms while SW1 is turned on. After SW1 is turned off, LED1 blinks in a cycle of approximately 120 ms. If a WDT2 overflow occurs, a reset signal is generated by WDT2. After the reset is released, LED2 is turned on and LED1 blinks in a cycle of approximately 120 ms.

Target devices

V850ES/JG3 microcontroller

V850ES/JJ3 microcontroller

CONTENTS

CHAPTER 1 OVERVIEW 3
1.1 Initial Settings 4
1.2 Operation of Watchdog Timer 2 (WDT2) 5
1.3 Main processing 5
1.4 Interrupt Servicing 6
1.5 Example of the watchdog timer 2 (WDT2) overflow generation 7

CHAPTER 2 CIRCUIT DIAGRAM 8
2.1 Circuit Diagram 8
2.2 Peripheral Hardware 8

CHAPTER 3 SOFTWARE 9
3.1 File Configuration 9
3.2 On-Chip Peripheral Functions Used 10
3.3 Initial Settings and Operation Overview 10
3.4 Flowchart 12
3.5 Differences Between V850ES/JG3-L and V850ES/JF3-L 14
3.6 Security ID 14
3.7 Caution of this sample program operation by using MINICUBE2 14

CHAPTER 4 SETTING REGISTERS 15
4.1 Settings of Watchdog Timer 2 (WDT2) 16
4.2 Checking Detection of WDT2 Reset 22

CHAPTER 5 RELATED DOCUMENTS 23

APPENDIX A PROGRAM LIST 24

Document No. U20020EJ1V0AN00
Date Published September 2009 NS

CHAPTER 1 OVERVIEW

This sample program describes the usage of watchdog timer 2 (WDT2).

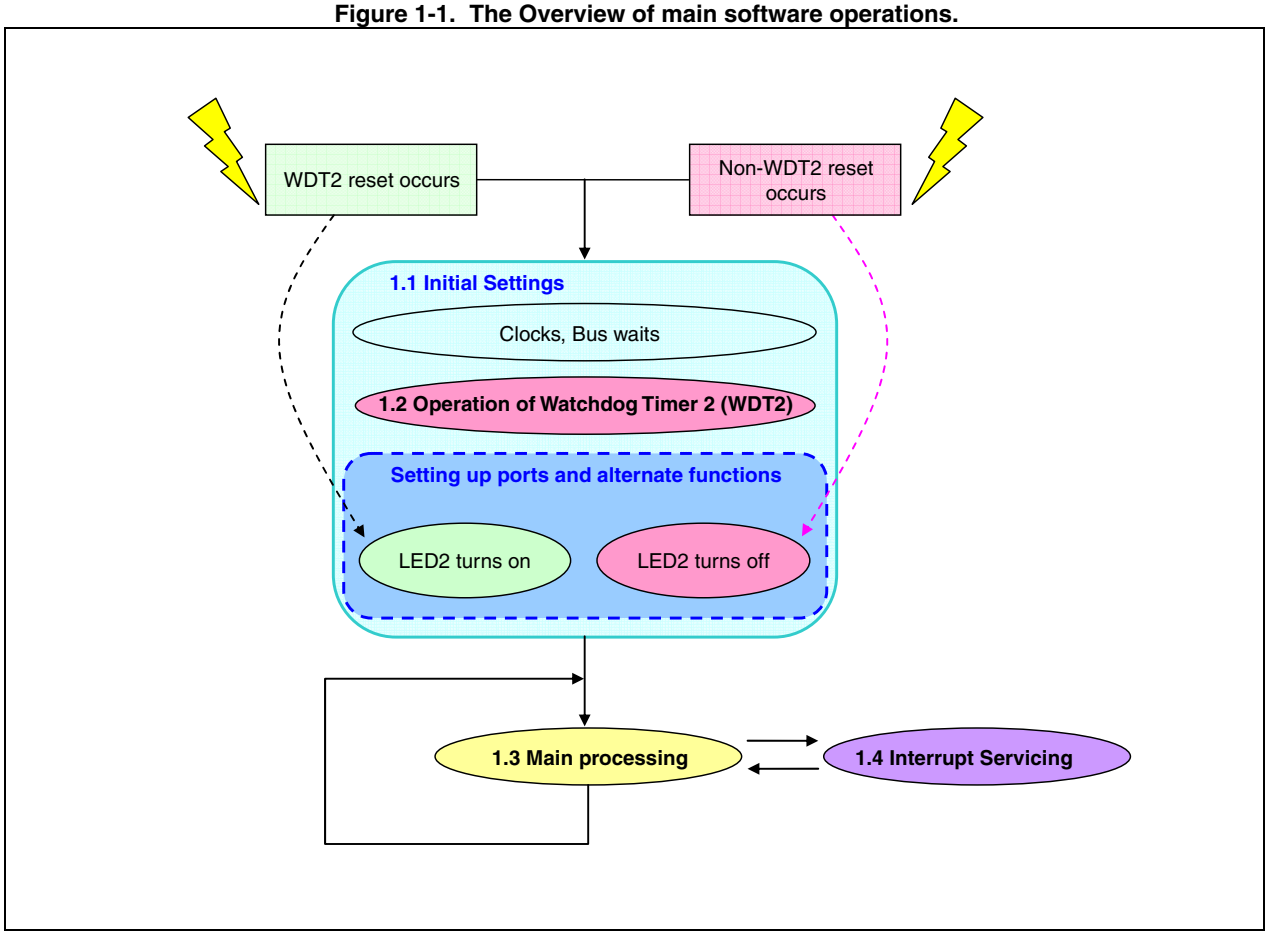
The overflow time of WDT2 is specified as 125 ms based on the subclock. When an overflow occurs, an internal reset signal (WDT2RES) is generated.

After initial setup is complete, an interrupt is generated and serviced when the falling edge of the switch input is detected. LED1 blinks and the WDT2 counter is cleared in a cycle of 120 ms while the switch is not being pressed. LED1 blinks and the WDT2 counter is cleared in a cycle of 55 ms while the switch is being pressed. When a reset is generated by WDT2, the initial settings specify that LED2 turns on.

The settings for peripheral functions that remain stopped after reset is released and that are not used in this sample program have not been specified.

The main software operations are shown below.

Figure 1-1. The Overview of main software operations.



1.1 Initial Settings

<Settings of on-chip peripheral functions>

- Setting wait operations <wait: 2> for bus access to on-chip peripheral I/O registers
- Setting on-chip debug mode register to normal operation mode
- Stopping the internal oscillator
- Setting watchdog timer 2 mode <reset mode>, and setting the use of the subclock
- Setting internal system clock
- Setting the system clock to 32 MHz by multiplying the input clock by 8 using the PLL

<Pin settings>

- Setting unused pins
- Setting external interrupt pins (edge specification, priority specification, unmasking)
- Setting LED output pins

<Settings of 16-bit interval timer M (TMM0)>

- Selecting the count clock (f_{XX}/512)
- Setting the TMM0 count (120 ms)
- Masking interrupts from timer M
- Enabling operation of TMM0

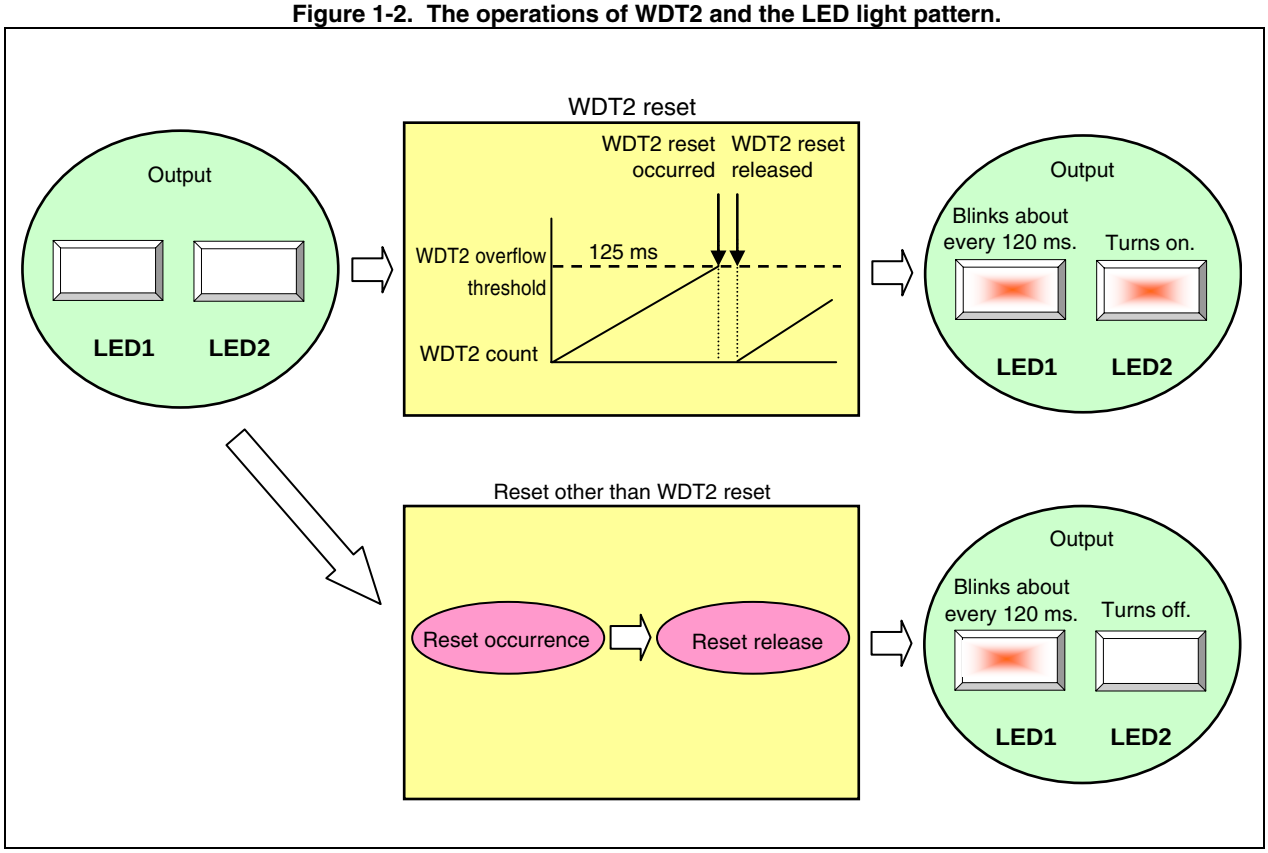
1.2 Operation of Watchdog Timer 2 (WDT2)

Watchdog timer 2 (WDT2) sets to counting by operation clocks, and when counter is overflowed, the watchdog timer 2 generates an internal reset signal (WDT2RES) or interrupt servicing [Note].

In this sample program, watchdog timer 2 generates an internal reset signal (WDT2RES) when an overflow occurs. In particular, when a WDT2 reset occurs, LED2 turns on and LED1 blinks about every 120 ms. When a reset other than a WDT2 reset occurs, LED2 turns off and LED1 blinks about every 120 ms.

Note Watchdog timer 2 (WDT2) automatically starts in the reset mode following reset release. When watchdog timer 2 is not used, either stop its operation before reset is executed via this function, or clear watchdog timer 2 once and stop it within the next interval time.

Figure 1-2. The operations of WDT2 and the LED light pattern.



1.3 Main processing

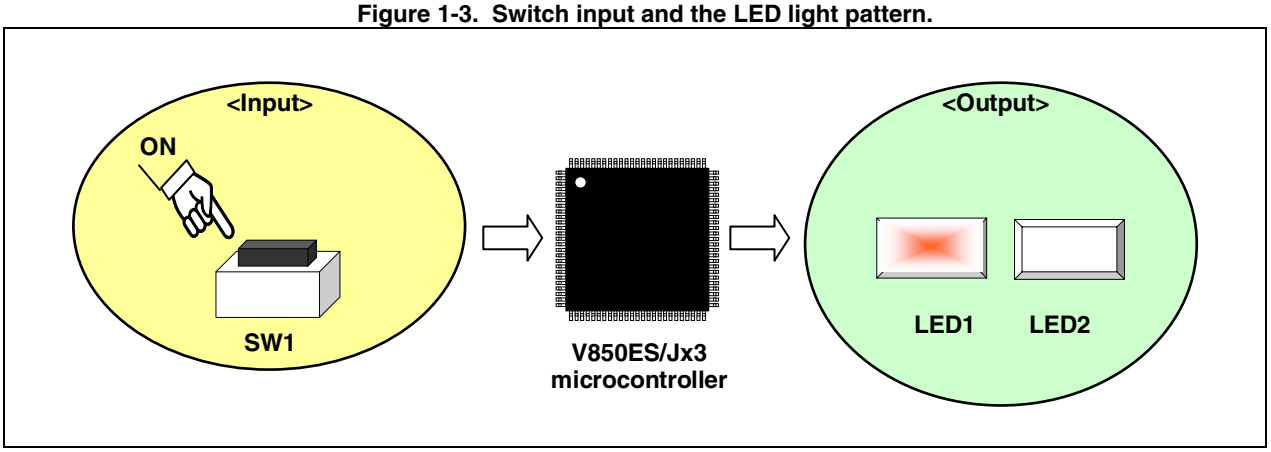
- Enabling interrupts by using the EI instruction
- Executing an infinite loop (LED1 blinks about every 120 ms while the system is waiting for an interrupt generated by switch input.)

1.4 Interrupt Servicing

Interrupts are serviced by detecting the falling edge of the INTP0 pin, caused by switch input. In interrupt servicing, the LED1 blinking cycle is changed by confirming that the switch is on, after about 10 ms have elapsed after the falling edge of the INTP0 pin was detected.

The switch being off, after about 10 ms have elapsed after the falling edge of the INTP0 pin was detected, is identified as chattering noise and the LED1 blinking cycle is not changed.

Figure 1-3. Switch input and the LED light pattern.



No switch input → LED1 blinks about every 120 ms
Switch input → LED1 blinks about every 55 ms

Caution See each product user's manual (V850ES/Jx3) for cautions when using the device.

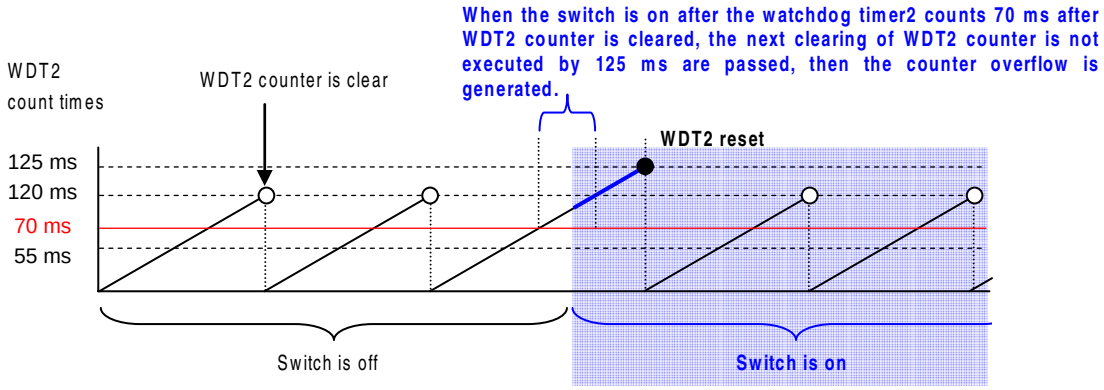
[Column] What is chattering?
Chattering is a phenomenon that an electric signal alternates between being on and off when a connection flip-flops mechanically immediately after a switch is switched.

1.5 Example of the watchdog timer 2 (WDT2) overflow generation

In this sample program, the timing example of the internal reset occurrence when watchdog timer 2 (WDT2) is overflowed is showed the following.

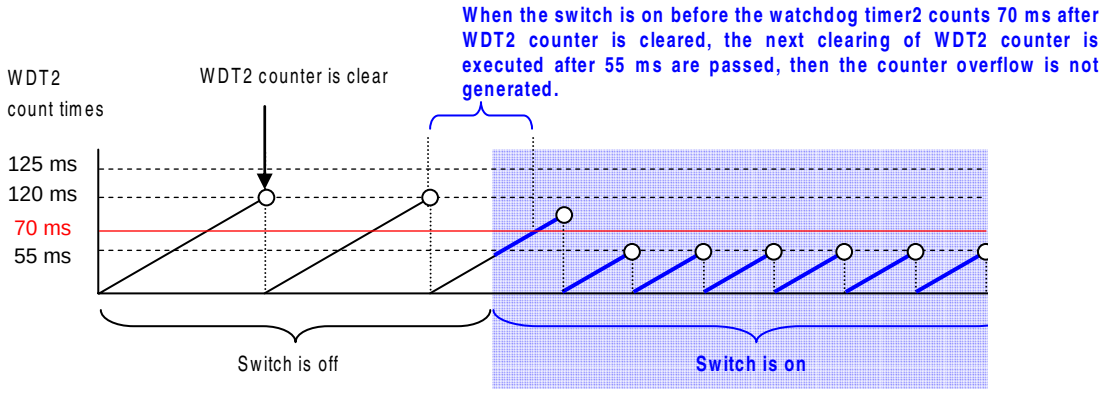
Figure 1-4. The timing example of WDT2 overflows is generated.

(i) The timing example of the overflow is generated.



Caution As this timing example, when the switch is kept on input before WDT2 reset was generated, LED1 blinks about 120 ms cycles as same as switch is off because of not detecting the falling edge of the INTP0 after WDT2 reset released.

(ii) The timing example of the overflow is not generated.



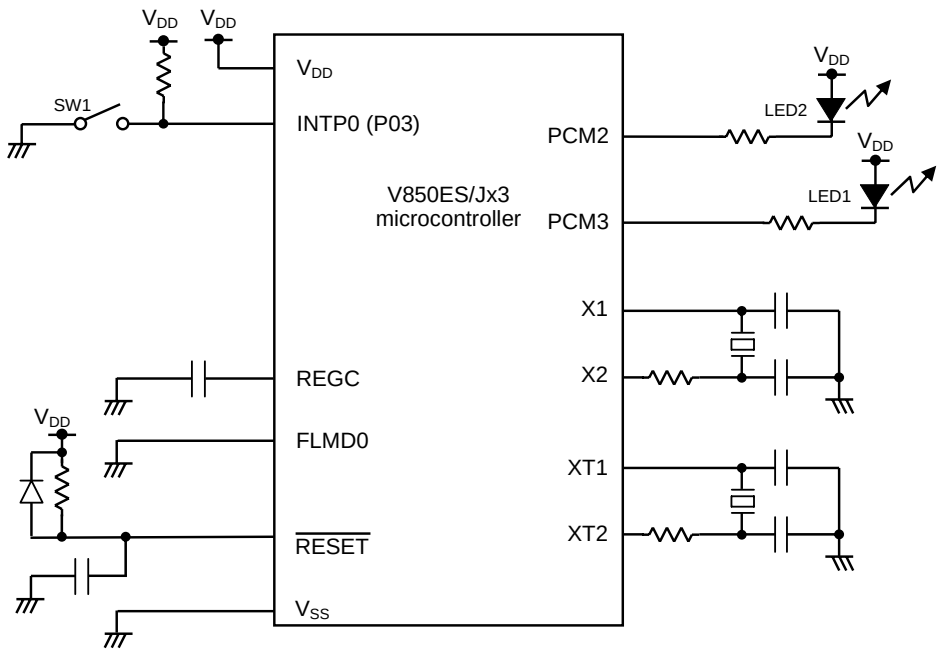
CHAPTER 2 CIRCUIT DIAGRAM

This chapter describes a circuit diagram and the peripheral hardware to be used in this sample program.

2.1 Circuit Diagram

The circuit diagram is shown below.

Figure 2-1. Circuit Diagram



Cautions 1. Connect the EV_{DD}, AV_{REF0}, and AV_{REF1} pins directly to V_{DD}.

2. Connect the EV_{SS} and AV_{SS} pins directly to GND.

3. Connect the FLMD0 pin to GND in normal operation mode.

4. Connect REGC to GND via a capacitor (recommended value: 4.7 μF).

5. Leave all unused ports open, because they will be handled as output ports.

2.2 Peripheral Hardware

The peripheral hardware to be used is shown below.

(1) Switch (SW1)

This switch is used as an interrupt input to control the lighting of LED1.

(2) LEDs (LED1, LED2)

LED1 is used as outputs corresponding to switch inputs.

When a reset signal is generated by watchdog timer 2, LED2 turns on following reset release.

CHAPTER 3 SOFTWARE

This chapter describes the file configuration of the compressed files to be downloaded, on-chip peripheral functions of the microcontroller to be used, and the initial settings and an operation overview of the sample program. A flowchart is also shown.

3.1 File Configuration

The following table shows the file configuration of the compressed files to be downloaded.

File Name (Tree Structure)	Description	Compressed (*.zip) Files Included	
		ZIP	PM 32
conf ─ crtE.s	Startup routine file[Note 1]	–	●
├ AppNote_WDT2.dir	Link directive file[Note 2]	●	●
├ AppNote_WDT2.prj	Project file for integrated development environment PM+	–	●
└ AppNote_WDT2.prw	Workspace file for integrated development environment PM+	–	●
src ─ main.c	C language source file including descriptions of hardware initialization processing and main processing of microcontroller	●	●
└ minicube2.s	Source file for reserving area for MINICUBE2	●	●

Notes 1. This is the startup file copied when “Copy sample for use (C)” is selected when “Specify startup file” is selected when creating a new workspace. (If the default installation path is used, the startup file will be a copy of C:\Program Files\NEC Electronics Tools\PM+*Version used*\lib850\r32\crtE.s.)

2. This is the link directive file automatically generated when “Copy sample for use (C)” is selected and “Memory usage: Internal memory only (I)” is checked when “Specify link directive file” is selected when creating a new workspace, and to which a segment for MINICUBE2 is added. (If the default installation path is used, C:\Program Files\NEC Electronics Tools\PM+*Version used*\bin\w_data\V850_i.dat is used as the reference file.)

Remark ZIP : Only the source file is included.

PM 32 : The files to be used with integrated development environment PM+ are included.

3.2 On-Chip Peripheral Functions Used

The following on-chip peripheral functions of the microcontroller are used in this sample program.

- Watchdog timer 2 (used to generate an overflow): WDT2
- 16-bit interval timer M (used to generate the LED blinking cycle): TMM0
- External interrupt input (for switch input): INTP0 (SW1)
- Output ports (for lighting LEDs): PCM2 (LED2), PCM3 (LED1)

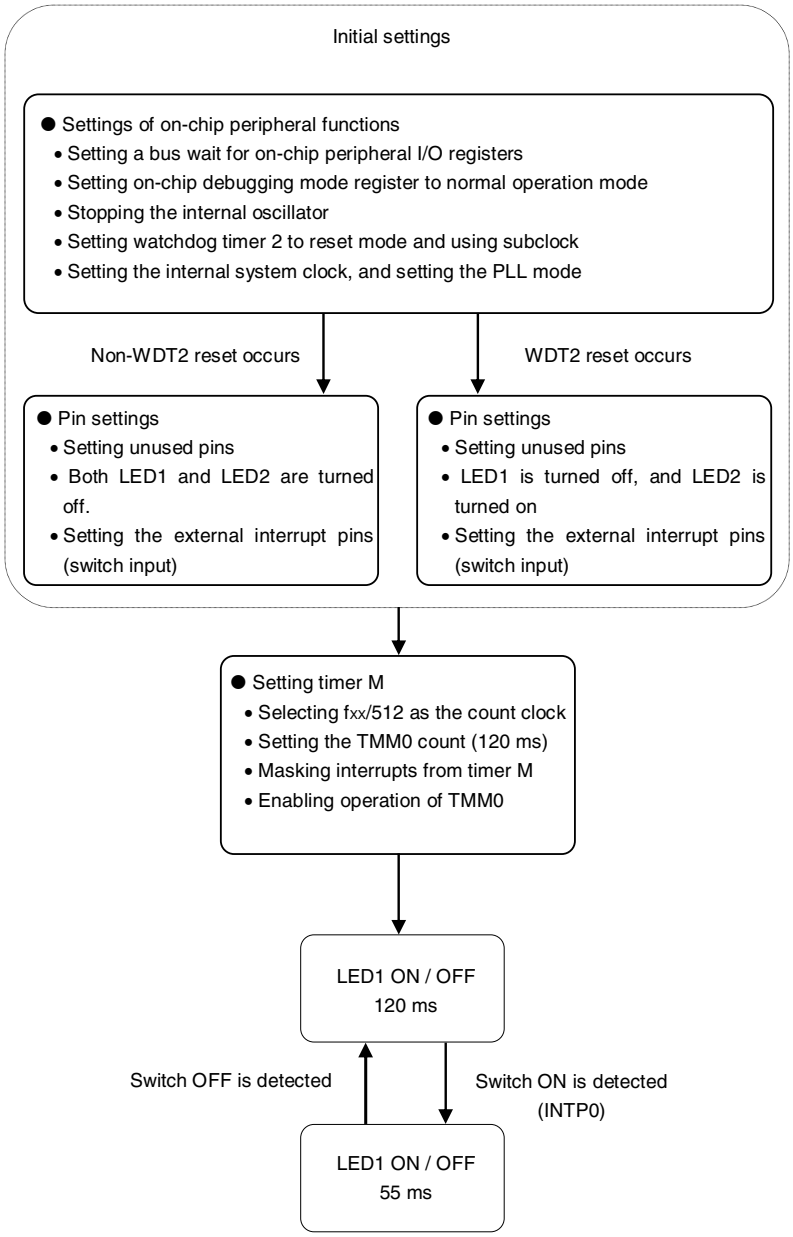
3.3 Initial Settings and Operation Overview

In this sample program, the selection of the clock frequency, setting of WDT2, setting of the I/O ports and external interrupt pins, setting of a TMM count clock and setting of interrupts are performed in the initial settings.

After initial setup is complete, LED1 blinks about every 120 ms, and interrupts are generated and serviced upon detection of the falling edge of the switch input (SW1). If no WDT2 overflow occurs during interrupt servicing, LED1 blinks about every 55 ms while SW1 is turned on. When SW1 is turned off, LED1 blinks about every 120 ms. If a WDT2 overflow occurs during interrupt servicing, watchdog timer 2 generates a reset signal. After the reset is released, LED2 turns on and LED1 blinks about every 120 ms.

The details are described in the state transition diagram shown below.

Figure 3-1 The state transition diagram



3.4 Flowchart

A flowchart for the sample program is shown below.

Figure 3-2 Flowchart (1/2)

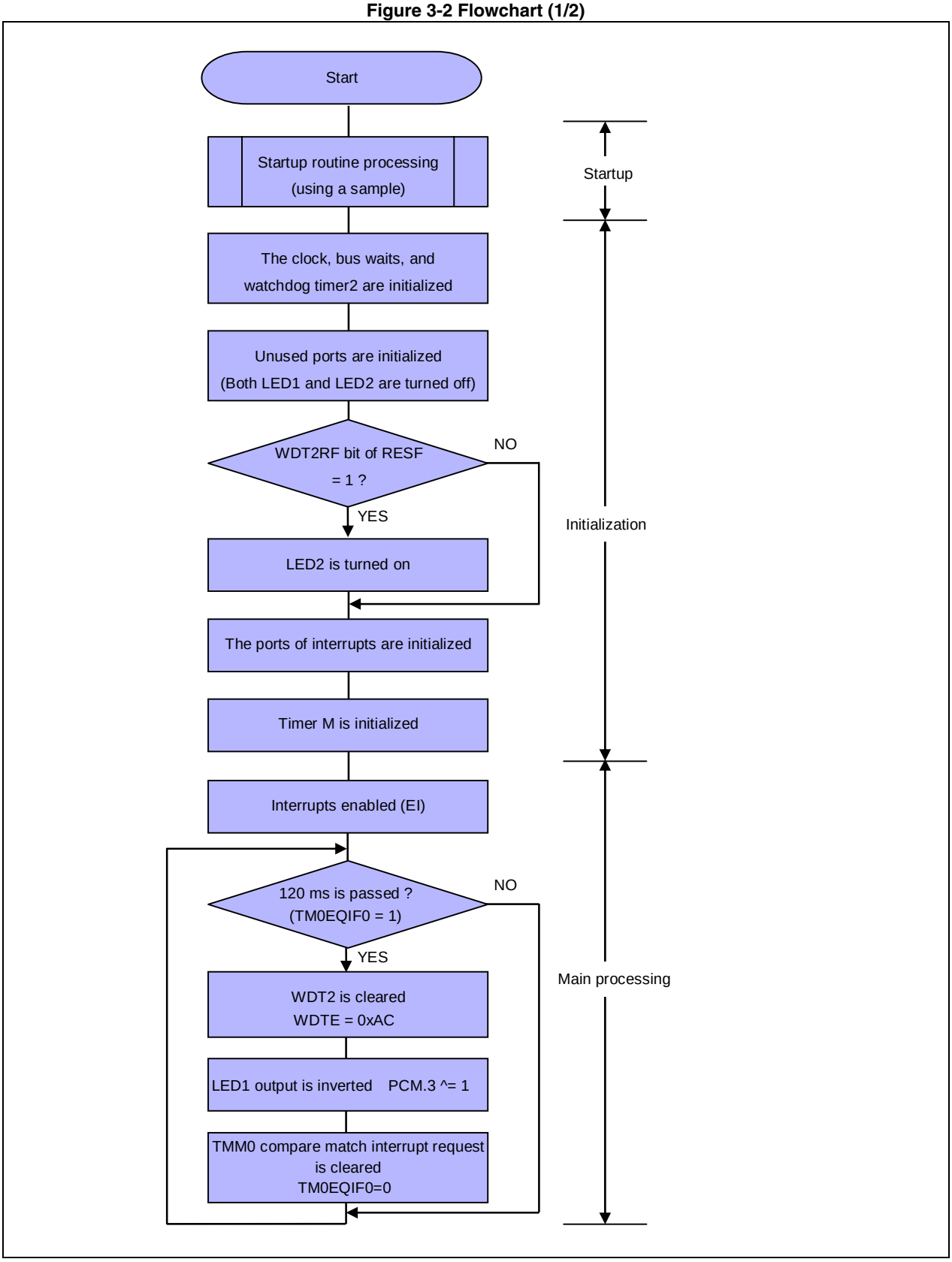
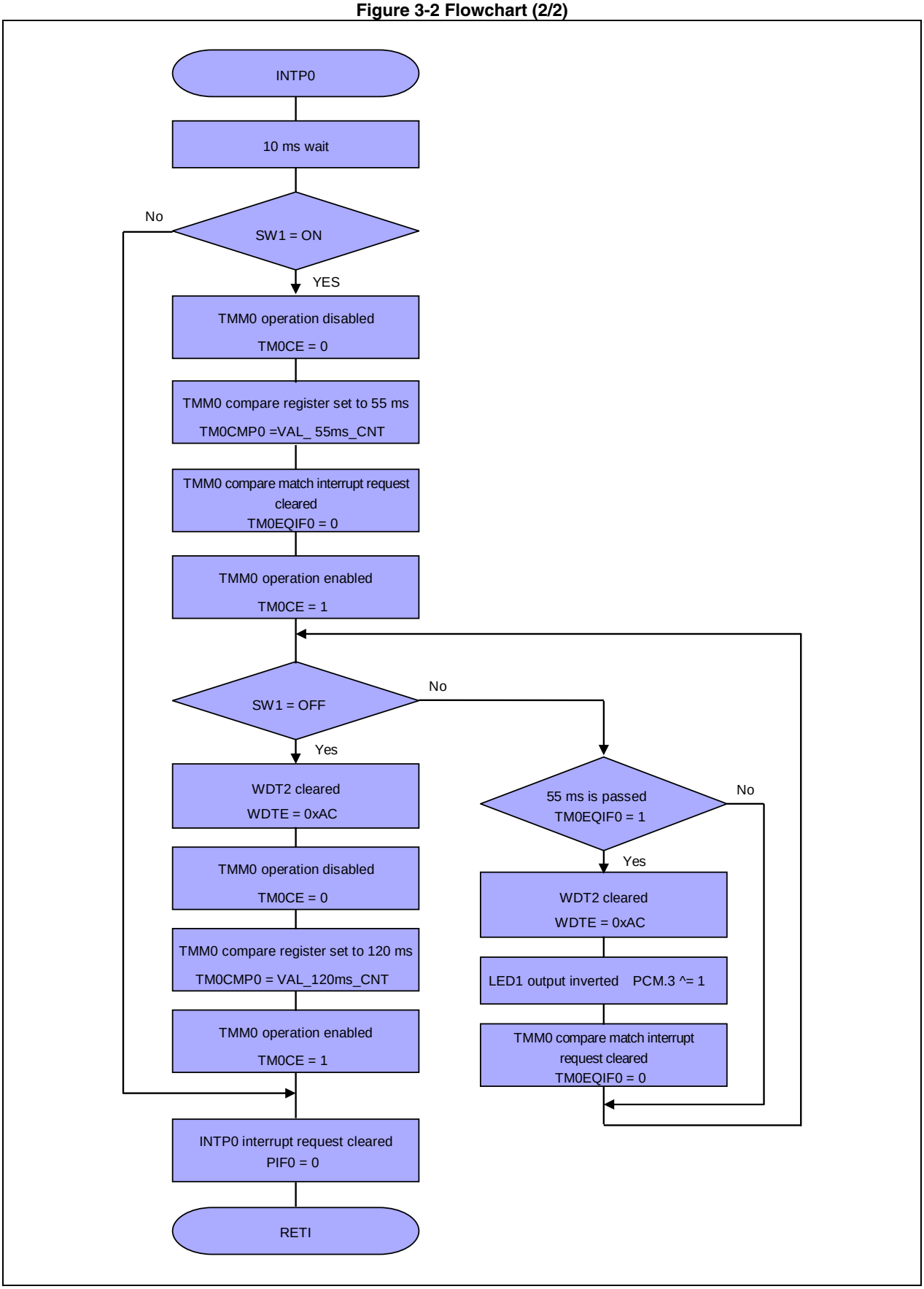


Figure 3-2 Flowchart (2/2)



[Column] Contents of the startup routine

The startup routine is a routine that is executed before executing the main function after reset of the V850 is released. Basically, the startup routine executes initialization so that the program written in C language can start operating.

Specifically, the following are performed.

- Securing the argument area of the main function
- Securing the stack area
- Setting the RESET handler when reset is issued
- Setting the text pointer (tp)
- Setting the global pointer (gp)
- Setting the stack pointer (sp)
- Setting the element pointer (ep)
- Setting mask values to the mask registers (r20 and r21)
- Clearing the sbss and bss areas to 0
- Setting the CTBP value for the prologue epilogue runtime library of the function
- Setting r6 and r7 as arguments of the main function
- Branching to the main function

3.5 Differences Between V850ES/JJ3 and V850ES/JG3

The V850ES/JJ3 is the V850ES/JG3 with its functions, such as I/Os, timer/counters, and serial interfaces, expanded.

In this sample program, the initialization range in I/O initialization differs.

See **APPENDIX A PROGRAM LIST** for details of the sample program.

3.6 Security ID

The content of the flash memory can be protected from unauthorized reading by using a 10-byte ID code for authorization when executing on-chip debugging using an on-chip debug emulator.

For details of ID security, see the **V850ES/Jx3 Sample Program (Interrupt) External Interrupt Generated by Switch Input Application Note**.

3.7 Caution of this sample program operation by using MINICUBE2

This sample program is executed by on-chip debug emulation, then watchdog timer2 (WDT2) is not operated. Because the monitoring program for debugging stops the watchdog timer 2 when debugger is started. Thus it is impossible that checking the operation of this sample program by on-chip debug using MINICUBE2.

For a detailed explanation of how to execute a program when using MINICUBE2, see the **QB-MINI2 On-Chip Debug Emulator with Programming Function User's Manual** and the **ID850QB Ver. 3.40 Integrated Debugger Operation User's Manual**.

CHAPTER 4 SETTING REGISTERS

This chapter describes the watchdog timer 2 (WDT2) settings.

For other initial settings, refer to the **V850ES/Jx3 Sample Program (Initial Settings) LED Lighting Switch Control Application Note**. For interrupt, refer to the **V850ES/Jx3 Sample Program (Interrupt) External Interrupt Generated by Switch Input Application Note**.

Among the peripheral functions that are stopped after reset is released, those that are not used in this sample program are not set.

For how to set registers, see each product user's manual.

- V850ES/JJ3 32-bit Single-Chip Microcontroller
  Hardware User's Manual
- V850ES/JG3 32-bit Single-Chip Microcontroller
  Hardware User's Manual

See the following user's manuals for details of extended descriptions in C languages.

- CA850 C Compiler Package C Language User's Manual

4.1 Settings of Watchdog Timer 2 (WDT2)

Watchdog timer 2 operates in the following two modes:

- A mode in which WDT2 is used as a reset trigger (see **[Example 1]**)
- A mode in which WDT2 is used as a non-maskable interrupt [Note] trigger (see **[Example 2]**)

Note A non-maskable interrupt request signal is acknowledged even when interrupt are disabled (DI) by the CPU. A non-maskable interrupt is not subject to priority control and takes precedence over the other interrupt request signals.

Watchdog timer 2 is mainly controlled by the following two registers:

- Watchdog timer mode register 2 (WDTM2)
- Watchdog timer enable register (WDTE)

4.1.1 Watchdog timer mode register 2 (WDTM2)

Watchdog timer mode register 2 (WDTM2) is used to set the operation mode, overflow time and operating clock of watchdog timer 2.

WDTM2 can be read and written in 8-bit units. This register can be read any number of times, but it can be written only once following reset release.

Reset sets this register to 0x67.

Caution Accessing the WDTM2 register is prohibited in the following statuses.

- **When the CPU is operating on the subclock and main clock oscillation is stopped.**
- **When the CPU is operating on the internal oscillation clock.**

Figure 4-1. Format of WDTM2 Register

Watchdog timer mode register 2 (WDTM2)

Address: 0xFFFF6D0

7	6	5	4	3	2	1	0
0	WDM21	WDM20	WDCS24	WDCS23	WDCS22	WDCS21	WDCS20

WDM21	WDM20	Selection of watchdog timer 2 operation mode
0	0	Operation stopped
0	1	Non-maskable interrupt request mode (WDT2 generates the INTWDT2 signal)
1	× [Note]	Reset mode (WDT2 generates the WDT2RES signal)

WDCS24	WDCS23	WDCS22	WDCS21	WDCS20	Selection of overflow time and operation clocks
1	× [Note]	0	1	1	$2^{12}/f_{XT}$ (125 ms), f_{XT}

Note Either 0 or 1

Remark
- The red values in the table indicate the values set in the sample program (WDTM2 = 0x53).
- f_{XT} = 32.768 kHz

[Column] When the operation clock of watchdog timer 2 sets to subclock

Usually WDTM2 register are set in initial stage of program because the watchdog timer 2 automatically starts in the reset mode following reset release. Although subclock is set as the operation clock of watchdog timer 2, then watchdog timer 2 may start to operation before the subclock is not already stable. To avoid the starting of watchdog timer2 operated by not stable subclock, the wait as same as appropriate oscillation stabilization time should set before WDTM2 register are set to using the subclock. Additionally WDT2 is already operating in this wait time, and the count is cleared aptly before the overflow is generated.

4.1.2 Watchdog timer enable register (WDTE)

Writing 0xAC to the WDTE register clears the counter of watchdog timer 2 and causes the counter to start counting up again.

This register can be read and written in 8-bit units. Writing WDTE using a 1-bit memory manipulation instruction will cause an overflow to occur.

Reset sets this register to 0x9A.

Figure 4-2. Format of WDTE Register

Watchdog timer enable register (WDTE)

Address: 0xFFFFF6D1

7	6	5	4	3	2	1	0

Cautions 1. Writing a value other than 0xAC to the WDTE register will trigger forcible generation of the overflow signal.

2. To deliberately trigger generation of the overflow signal, write a value other than 0xAC to the WDTE register once, or write to the WDTM2 register twice. Note, however, that if operation of watchdog timer 2 has been stopped, writing a value other than 0xAC to the WDTE register once or writing to the WDTM2 register twice will not trigger generation of the overflow signal.

3. The value read from the WDTE register is 0x9A (which differs from the value written—0xAC).

[Example 1] Using the detection of a watchdog timer 2 overflow as a trigger to generate a reset
(Same usages as sample program)

The following settings show that operation clock sets to subclock and the reset generates 125 ms later after the count started. And to avoid occurrence of reset, the counter is cleared before 125 ms passed. These settings enable that CPU operation reset to initial state when the counter clear is not executed by any problem of CPU operation.

- Setup procedure
  <1> Set WDTM2 to 0x53 (in the program example, this sets reset mode, an operating clock of f_{XT} and overflow time is 125 ms).
  <2> Set WDTE to 0xAC before the overflow detection time elapses (clearing the watchdog timer 2 count value) to stop the occurrence of a reset.

- Program example

```
Initialization processing for other than WDT2 is omitted here

/* Using detection of WDT2 overflow as a reset trigger */
  WDTM2 = 0x53;                    /* Starts watchdog timer 2 operation   */  } <1>

  if( The condition is true everytime before the overflow of WDT2 )
  {
    WDTE = 0xAC;                    /* Clears WDT2 count                   */  } <2>
  }
```

[Example 2] Using the detection of a watchdog timer 2 overflow as a trigger to generate an interrupt

The following settings show that operation clock sets to subclock and the interrupt (INTWDT2) generates and servicing 125 ms later after the count started. And to avoid occurrence or interrupt, the counter is cleared before 125 ms passed.

- Setup procedure

<1> Specifying INTWDT2 interrupt handler and prototype-declaration of the interrupt functions.

<2> Set WDTM2 to 0x33 (in the program example, this sets non-maskable interrupt, an operating clock of f_{XT} and overflow time is 125 ms).

<3> Set WDTE to 0xAC before the overflow detection time elapses (clearing the watchdog timer 2 count value) to stop the occurrence of a interrupt.

<4> Define the INTWDT2 interrupt function.

- Program example 1

```
#pragma interrupt INTWDT2 f_int_intwdt2      /* specifying interrupt handler */   <1>
static void f_int_intwdt2( void );           /* INTWDT2 interrupt function   */

      Initialization processing for the other  is omitted here


/* Using detection of WDT2 overflow as a interrupt trigger */
  WDTM2 = 0x33;                        /* Starts watchdog timer 2 operation  */   <2>

  if(  The condition is true everytime  before the overflow of WDT2  )
  {
      WDTE = 0xAC;                         /* Clears WDT2 count         */          <3>
  }

__interrupt
void f_int_intwdt2( void )
{
        The processing of interrupt servicing                                      <4>
}
```

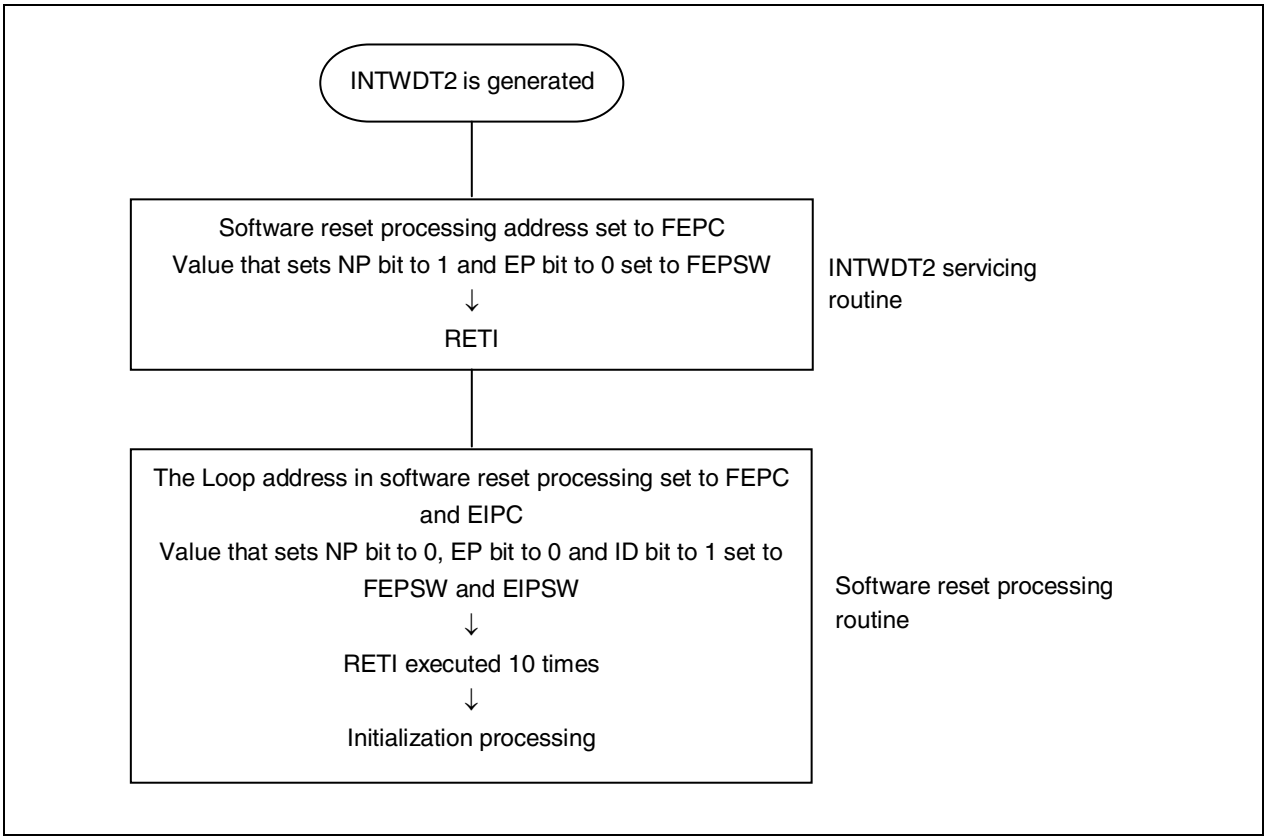
The overflow of watchdog timer 2 is likely to cause the CPU to enter an infinite processing loop. It is therefore recommended to set watchdog timer 2 to the reset mode (by setting the WDM21 bit to 1) so that the CPU is reset when WDT2 overflows.

If you choose to set watchdog timer 2 to non-maskable interrupt request mode (by setting the WDM21 bit to 0 and the WDM20 bit to 1), however, be sure to stop the system after the system error processing has been completed by the INTWDT2 interrupt servicing routine. The system cannot be returned to the main processing routine by using the RETI instruction after the servicing of a non-maskable interrupt generated by the INTWDT2

signal has finished. To return the system to the main processing routine following the servicing of an INTWDT2 interrupt, execute the software reset processing shown below.

Note that in this software reset processing, registers that can be set only once following the release of reset (such as the WDTM2 register) cannot be set again. These registers must be initialized by executing a hardware reset by means such as inputting a signal to the reset pin.

Figure 4-3. Software reset processing



• Example of program 2

```
#*********************************************#
#       Interrupt initialization processing   #
#*********************************************#
        .global _initint

_initint:
        mov     initloop1,r6   -- Sets the processing routine address
        ldsr    r6,2           -- Set to FEPC
        mov     0xa0,r6        -- NP=1, EP=0
        ldsr    r6,3           -- Set to FEPSW
        reti                   -- Processing restored from NMI and returned to initloop1



initloop1:
        mov     0x20,r6        -- PSW.NP set to 0 by next RETI instruction
        ldsr    r6,3           -- Set to FEPSW
        ldsr    r6,1           -- Set to EIPSW
        mov     initloop2,r6   -- Sets processing routine address 2
        ldsr    r6,2           -- Set to FEPC
        ldsr    r6,0           -- Set to EIPC
        mov     0x0b,r6        -- Count initial value (10 times)
        reti

initloop2:
        sub     1,r6           -- Loop count (-1)
        bz      initend        -- End of interrupt initialization
        reti

initend:
```

4.2 Checking Detection of WDT2 Reset

When a WDT2 reset occurs, the WDT2RF bit of the reset source flag register (RESF) is set. On the other hand, when a reset is generated by an input to the $\overline{\text{RESET}}$ pin, the WDT2RF bit is cleared. Therefore, by checking the WDT2RF bit after the reset is released, it is possible to ascertain whether the reset source was an WDT2 reset or the other reset source.

4.2.1 Reset source flag register (RESF)

The RESF register stores information on which reset signal—the reset signal from which source—generated a reset.

This register can be read or written in 8-bit or 1-bit units.

Note, however, that the RESF register can only be written using a combination of specific sequences.

A reset generated by an input to the $\overline{\text{RESET}}$ pin sets this register to 0x00. A reset generated by any other source, such as watchdog timer 2 (WDT2), the low-voltage detector (LVI), or the clock monitor (CLM), sets the flag of the corresponding source (WDT2RF, CLMRF, LVIRF bits); the other source flags hold their previous values.

Figure 4-4. Format of RESF Register

Reset source flag register (RESF)
Address: 0xFFFFF888

7	6	5	4	3	2	1	0
0	0	0	WDT2RF	0	0	CLMRF	LVIRF

WDT2RF	Occurrence of reset signal from WDT2
0	Did not occur
1	Occurred

CLMRF	Occurrence of reset signal from CLM
0	Did not occur
1	Occurred

LVIRF	Occurrence of reset signal from LVI
0	Did not occur
1	Occurred

Cautions1. Only 0 can be written to each bit. If writing 0 conflicts with the flag being set (due to the occurrence of a reset), flag setting takes precedence.

2. If watchdog timer 2 (WDT2), the low-voltage detector (LVI), and the clock monitor (CLM) are being used at the same time, the relevant reset source flag must be cleared after checking the reset source .

Remark The blue values indicate the bits to be checked in the sample program.

[Clearing the reset source flag]
As mentioned in Note 2 on the previous page, there are cases when the reset source flag has to be cleared after checking the reset source. In this sample program, however, the reset source flag does not have to be cleared because only the watchdog timer 2 (WDT2) is used.

CHAPTER 5 RELATED DOCUMENTS

Document	document number
V850ES/JJ3 Hardware User's Manual	U18376E
V850ES/JG3 Hardware User's Manual	U18708E
V850ES 32-Bit Microprocessor Core for Architecture	U15943E
PM+ Ver. 6.30 User's Manual	U18416E
CA850 Ver. 3.20 C Compiler Package Operation	U18512E
CA850 Ver. 3.20 C Compiler Package C Language	U18513E
CA850 Ver.3.20 C Compiler Package for Link Directives	U18515E
QB-MINI2 User's Manual	U18371E
ID850QB Ver.3.40 Integrated Debugger for Operation	U18604E

Document Search URL http://www.necel.com/search/en/index.html#doc

APPENDIX A PROGRAM LIST

The V850ES/JJ3 microcontroller source program is shown below as a program list example.

● minicube2.s

```
#-----------------------------------------------------------------------------
#
#    NEC Electronics     V850ES/Jx3 microcontroller
#
#-----------------------------------------------------------------------------
#    V850ES/JJ3 JG3 sample program
#-----------------------------------------------------------------------------
#    Reset Generation When Infinite Loop Detected
#-----------------------------------------------------------------------------
#[History]
#    2009.09.--   Released
#-----------------------------------------------------------------------------
#[Overview]
#    This sample program secures the resources required when using MINICUBE2.
#       (Example of using MINICUBE2 via CSIB0)
#-----------------------------------------------------------------------------

     -- Securing a 2 KB space as the monitor ROM section
     .section "MonitorROM", const
     .space 0x800, 0xff

     -- Securing an interrupt vector for debugging
     .section "DBG0"
     .space 4, 0xff

     -- Securing a reception interrupt vector for serial communication
     .section "INTCB0R"
     .space 4, 0xff

     -- Securing a 16-byte space as the monitor RAM section
     .section "MonitorRAM", bss
     .lcomm monitorramsym, 16, 4
```

● AppNote_LED.dir

```
#   Sample link directive file (not use RTOS/use internal memory only)
#
#   Copyright (C) NEC Electronics Corporation 2002
#   All rights reserved by NEC Electronics Corporation.
#
#   This is a sample file.
#   NEC Electronics assumes no responsibility for any losses incurred by customers or
#   third parties arising from the use of this file.
#
#   Generated      : PM+ V6.31  [ 9 Jul 2007]
#   Sample Version : E1.00b [12 Jun 2002]
#   Device         : uPD70F3746 (C:\Program Files\NEC Electronics Tools\DEV\DF3746.800)
#   Internal RAM   : 0x3ff0000 - 0x3ffefff
#
#   NOTICE:
#        Allocation of SCONST, CONST and TEXT depends on the user program.
#
#        If interrupt handler(s) are specified in the user program then
#        the interrupt handler(s) are allocated from address 0 and
#        SCONST, CONST and TEXT are allocated after the interrupt handler(s).

SCONST  : !LOAD ?R {
        .sconst         = $PROGBITS     ?A .sconst;
};

CONST   : !LOAD ?R {
        .const          = $PROGBITS     ?A .const;
};

TEXT    : !LOAD ?RX {
        .pro_epi_runtime = $PROGBITS    ?AX .pro_epi_runtime;
        .text            = $PROGBITS    ?AX .text;
};

### For MINICUBE2 ###
MROMSEG : !LOAD ?R V0x0ff800{
         MonitorROM     = $PROGBITS ?A MonitorROM;
};
```

Address values vary depending on the product internal ROM size.

This is an example of the product that's internal ROM size is 1024KB.

Difference from the default link directive file(additional code).

A reserved area for MINICUBE2 is secured.

```
SIDATA  : !LOAD ?RW V0x3ff0000 {
        .tidata.byte    = $PROGBITS     ?AW .tidata.byte;
        .tibss.byte     = $NOBITS       ?AW .tibss.byte;
        .tidata.word    = $PROGBITS     ?AW .tidata.word;
        .tibss.word     = $NOBITS       ?AW .tibss.word;
        .tidata         = $PROGBITS     ?AW .tidata;
        .tibss          = $NOBITS       ?AW .tibss;
        .sidata         = $PROGBITS     ?AW .sidata;
        .sibss          = $NOBITS       ?AW .sibss;
};

DATA    : !LOAD ?RW V0x3ff0100 {
        .data           = $PROGBITS     ?AW  .data;
        .sdata          = $PROGBITS     ?AWG .sdata;
        .sbss           = $NOBITS       ?AWG .sbss;
        .bss            = $NOBITS       ?AW  .bss;
};


### For MINICUBE2 ###
MRAMSEG : !LOAD ?RW V0x03ffeff0{
         MonitorRAM     = $NOBITS ?AW MonitorRAM;
};


__tp_TEXT @ %TP_SYMBOL;
__gp_DATA @ %GP_SYMBOL &__tp_TEXT{DATA};
__ep_DATA @ %EP_SYMBOL;
```

Difference from the default link directive file(additional code).

A reserved area for MINICUBE2 is secured.

● main.c

```
/*-----------------------------------------------------------------------------*/
/*
/*  NEC Electronics     V850ES/Jx3 microcontroller
/*
/*-----------------------------------------------------------------------------*/
/*  V850ES/JJ3 sample program
/*-----------------------------------------------------------------------------*/
/*  Reset Generation When Infinite Loop Detected
/*-----------------------------------------------------------------------------*/
/*[History]
/*  2009.09.--  Released
/*-----------------------------------------------------------------------------*/
/*[Overview]
/*  This sample program presents an example of using the watchdog timer 2 (WDT2).
/*  The WDT2 overflow time is set to 125 ms and an internal reset signal
/*  (WDT2RES) is generated when an overflow occurs.
/*  After initial setup is complete, interrupts are generated and serviced upon
/*  detection of the falling edge of the switch input. While the switch is being
/*  pressed, LED1 blinks in a cycle of 55 ms and the count of WDT2 is cleared.
/*  When a reset is generated by WDT2, the initial settings specify that LED2
/*  turns on.
/*
/*
/*  Among the peripheral functions that are stopped after reset is released,
/*  those that are not used in this sample program are not set.
/*
/*
/* <Main setting contents>
/*  • Using pragma directives to enable setting the interrupt handler and
/*    describing peripheral I/O register names
/*  • Defining a wait adjustment value of 10 ms for chattering
/*  • Defining the LED1 blinking time to be set to TMM0 (120 ms, 55 ms)
/*  • Performing prototype definitions
/*  • Setting a bus wait for on-chip peripheral I/O registers, starting the WDT2
      operation, and setting the clock
/*  • Initializing unused ports
/*  • Initializing external interrupt ports (falling edge) and LED output ports
/*  • The TMM0 compare match interrupt request flag is monitored and when an
/*    interrupt request occurs, the WDT2 count is cleared, the output of LED1 is
/*    inverted, and the TMM0 compare match interrupt request flag is cleared.
/* <Interrupt servicing>
/*  • The LED1 blinking cycle is set to 55 ms.
/*  • The TMM0 compare match interrupt request flag is monitored and when an
      interrupt request occurs, the WDT2 count is cleared, the output of LED1 is
      inverted, and the TMM0 compare match interrupt request flag is cleared.
```

```
/*  • When the switch is off, the WDT2 count is cleared, and the LED1 blinking
/*    cycle is set to 120 ms.
/*   (Chattering elimination time during switch input: 10 ms)
/*
/*
/*[I/O port settings]
/*
/* Input port      : P03(INTP0)
/* Output ports    : PCM2, PCM3
/* Unused ports    : P00 to P02, P04 to P06, P10 and P11, P30 to P39,P40 to P42,
/*                   P50 to P55, P60 to P615, P70 to P715, P80 and P81, P90 to P915,
/*                   PCD0 to PCD3, PCM0 and PCM1, PCM4 and PCM5, PCS0 to PCS7,
/*                   PCT0 to PCT7, PDH0 to PDH7, PDL0 to PDL15
/*                   *Preset all unused ports as output ports (low-level output).
/*
/*-------------------------------------------------------------------------------*/



/*---------------------------*/
/*   pragma directives       */
/*---------------------------*/
#pragma ioreg                        /* Enables describing to peripheral I/O
                                        registers.                                */
#pragma interrupt INTP0 f_int_intp0 /* Specifies the interrupt handler.           */



/*---------------------------*/
/* Constant definitions      */
/*---------------------------*/
#define LIMIT_10ms_WAIT     (40280) /* Defines the constant for a 10 ms wait adjustment. */
#define LIMIT_250ms_WAIT    (11360) /* Defines the constant for a 250 ms wait adjustment.*/
#define VAL_55ms_CNT        (3437)  /* LED1 blinking cycle (55 ms)                       */
#define VAL_120ms_CNT       (7499)  /* LED1 blinking cycle (120 ms)                      */
/*---------------------------*/
/*  Prototype definitions    */
/*---------------------------*/
static void f_init( void );              /* Initialization function                   */
static void f_init_clk_bus_wdt2( void ); /* Clock bus WDT2 initialization function    */
static void f_init_port_func( void );    /* Port/alternate-function initialization
                                            function                                  */
static void f_init_int_tmm( void );      /* TMM0 initialization function              */
```

```
/**********************************/
/*      Main module               */
/**********************************/
void main( void )
{
        f_init();                       /* Executes initialization.                        */

        __EI();                         /* Enables interrupts.                             */

        while( 1 )                      /* Main loop (infinite loop)                       */
        {
                if ( TM0EQIF0 == 1 )    /* Is there an INTTM0EQ0 interrupt request signal? */
                {
                        WDTE = 0xAC;    /* Clears the WDT2 count.                          */
                        PCM.3 ^= 1;     /* Inverts the LED1 output.                        */
                        TM0EQIF0 = 0;   /* Clears the TMM0 compare match interrupt request.*/
                }
        }

        return;
}


/*----------------------------*/
/*  Initialization module    */
/*----------------------------*/
static void f_init( void )
{
        f_init_clk_bus_wdt2();          /* Sets a bus wait for on-chip peripheral I/O
                                           registers, stops WDT2, and sets the clock.      */

        f_init_port_func();             /* Sets the port/alternate function.               */

        f_init_int_tmm();               /* Sets the TMM0 timer.                            */

        return;
}
```

```
/*-------------------------------------*/
/* Initializing clock/bus wait/WDT2     */
/*-------------------------------------*/
static void f_init_clk_bus_wdt2( void )
{
       unsigned int loop_wait;        /* for loop counter                        */

       VSWC  = 0x11;                  /* Sets a bus wait for on-chip
                                         peripheral I/O registers.               */

                                      /* Specifies normal operation mode for OCDM. */
#pragma asm
       st.b   r0, PRCMD
       st.b   r0, OCDM
#pragma endasm
```

Note that accessing a specific register must be described using an assembler.

```
       RSTOP  = 1;                    /* Stops the internal oscillator.          */

/* The wait processing in the case of subclock stabilization time is 250 ms.      */
/* Caution: TO be exact, the oscillation stabilization time setting should be
            based on the evaluation of oscillation */
for( loop_wait = 0 ; loop_wait < LIMIT_250ms_WAIT ; loop_wait++ )
{
       WDTE   = 0xAC;                 /* Clears the WDT2 count.                  */
}

       WDTM2  = 0x53;                 /* Selects the operation mode of WDT2      */

       PLLON = 0;                     /* stops the PLL */
                                      /* PLL multiplication is set to 8 */
#pragma asm
    push r10
    mov 0x0B, r10
    st.b r10, PRCMD
    st.b r10, CKC
    pop r10
#pragma endasm
```

Note that accessing a specific register must be described using an assembler.

```
       PLLON = 1;                     /* Enable PLL operation          */

       while( LOCK );                 /* Wait for PLL stabled       */

       SELPLL = 1;                    /* Set PLL mode */
```

```
/* Setting the PCC register */
/* Sets to not divide the clock. */

    #pragma asm
       push r10
       mov 0x80, r10
       st.b r10, PRCMD
       st.b r10, PCC
       pop r10
    #pragma endasm

      return;
}


/*----------------------------------------*/
/*  Setting the port/alternate function  */
/*----------------------------------------*/
static void f_init_port_func( void )
{
    P0     = 0x00;                  /* Sets P00 to P06 to output low level.          */
    PM0    = 0x88;
    PFC0   = 0x00;                  /* Sets the P03 pin as INTP0 input */
    PMC0   = 0x08;

    P1     = 0x00;                  /* Sets P10 and P11 to output low level.         */
    PM1    = 0xFC;

    P3     = 0x0000;                /* Sets P30 to P39 to output low level.          */
    PM3    = 0xFC00;
    PMC3   = 0x0000;

    P4     = 0x00;                  /* Sets P40 to P42 to output low level.         */
    PM4    = 0xF8;
#if(0) /* To use P4 as CSIB0 when using MINICUBE2,                  */
       /* P4 is not initialized as an unused pin (QB-V850ESJJ3-TB)  */
    PMC4   = 0x00;
#endif

    P5     = 0x00;                  /* Sets P50 to P55 to output low level.          */
    PM5    = 0xC0;
    PMC5   = 0x00;

    P6     = 0x0000;                /* Sets P60 to P615 to output low level.         */
    PM6    = 0x0000;
    PMC6   = 0x0000;
```

Caution must be exercised because access to a special register must be described in assembly language.

With V850ES/JG3, the setting value is 0x8B

With V850ES/JG3, these are not set because the registers do not exist.

```
P7H    = 0x00;                    /* Sets P70 to P715 to output low level.  */
P7L    = 0x00;
PM7H   = 0x00;
PM7L   = 0x00;
```

With V850ES/JG3, the setting value is 0xF0.

```
P8     = 0x00;                    /* Sets P80 and P81 to output low level. */
PM8    = 0xFC;
PMC8   = 00;
```

With V850ES/JG3, these are not set because the registers do not exist.

```
P9     = 0x0000;                  /* Sets P90 to P915 to output low level. */
PM9    = 0x0000;
PMC9   = 0x0000;

PCD    = 0x00;                    /* Sets PCD0 to PCD3 to output low level. */
PMCD   = 0xF0;
```

With V850ES/JG3, these are not set because the registers do not exist.

```
PCM    = 0x0C;                    /* Sets PCM0,PCM1, PCM4 and PCM5 to output low
                                     level and values to turn off the LEDs to
                                     PCM2 and PCM3 */

 if( RESF.4 == 1)
{
    PCM = 0x08;                   /* Sets that LED2 is turned on when the reset
                                      source is WDT2 */
}

PMCM   = 0xC0;
PMCCM  = 0x00;
```

With V850ES/JG3, the setting value is 0xF0.

```
PCS    = 0x00;                    /* Sets PCS0 to PCS7 to output low level.    */
PMCS   = 0x00;
```

With V850ES/JG3, these are not set because the registers do not exist.

```
PCT    = 0x00;                    /* Sets PCT0 to PCT7 to output low level.    */
PMCT   = 0x00;
PMCCT  = 0x00;
```

With V850ES/JG3, the setting value is 0xAC.

```
PDH    = 0x00;                    /* Sets PDH0 to PDH7 to output low level.    */
PMDH   = 0x00;
PMCDH  = 0x00;
```

With V850ES/JG3, the setting value is 0xC0.

```
PDL    = 0x0000;                  /* Sets PDL0 to PDL15 to output low level. */
PMDL   = 0x0000;
PMCDL  = 0x0000;
```

```
    /* Setting the interrupt function */
    INTF0 = 0x08;                       /* Specifies the falling edge of INTP0.    */
    INTR0 = 0x00;                       /* ↓                                       */
    PIC0  = 0x07;                       /* Sets the priority of INTP0 to level 7
                                           and unmasks INTP0.                      */

    return;
}


/*---------------------------*/
/*     Setting timer M       */
/*---------------------------*/
static void f_init_int_tmm( void )
{
        TM0CTL0 = 0x04;           /* Disables TMM0 operation.                      */
                                  /* Count clock = fxx/512                         */
        TM0CMP0 = VAL_120ms_CNT;  /* Sets TMM0 count.                              */
        TM0EQMK0 = 1;             /* Masks timer M interrupts.                     */
        TM0CE = 1;                /* Enables TMM0 operation.                       */

        return;
}



/*******************************/
/*     Interrupt module       */
/*******************************/
__interrupt
void f_int_intp0( void )
{
       unsigned int loop_wait;                /* for loop counter                */

       /* 10 ms wait to eliminate chattering */
       for( loop_wait = 0 ; loop_wait < LIMIT_10ms_WAIT ; loop_wait++ )
       {
               __nop();
       }

       if( ( P0 & 0x08 ) == 0x00 )            /* Identifies that SW1 has been
                                                 pressed after the wait.         */
       {
               /* SW ON        */
               TM0CE = 0;                     /* Stops the count operation.      */
               TM0CMP0 = VAL_55ms_CNT;        /* Sets the LED1 blinking cycle to
                                                 55 ms.                          */
               TM0EQIF0 = 0;                  /* Clears the TMM0 compare match
                                                 interrupt request.              */
               TM0CE = 1;                     /* Starts the count operation.     */
```

```
            while ( ( P0 & 0x08 ) == 0x00 )
            {
                    if ( TM0EQIF0 == 1 )    /* Is there an interrupt request
                                               signal?                        */
                    {
                            WDTE = 0xAC;    /* Clears the WDT2 count.         */
                            PCM.3 ^= 1;     /* Inverts the LED1 output.       */
                            TM0EQIF0 = 0;   /* Clears the TMM0 compare match
                                               interrupt request.             */
                    }
            }
            /* SW on -> off                 */
            WDTE = 0xAC;                    /* Clears the WDT2 counter.       */
            TM0CE = 0;                      /* Stops the count operation.     */
            TM0CMP0 = VAL_120ms_CNT;        /* Sets the LED1 blinking cycle to
                                               120 ms.                        */
            TM0CE = 1;                      /* Starts the count operation.    */
    }

    PIF0 = 0;
                                            /* Failsafe: Multiple requests
                                               Cleared.                       */

    return;                                 /* Processing moves to reti,
                                               depending on the _interrupt
                                               modifier.                      */
}
```

Published by: NEC Electronics Corporation (http://www.necel.com/)
Contact: http://www.necel.com/support/